ORTAÖGRETİM

BİLGİSAYAR BİLİMİ

KUR-2 (öğretmen ders notu)

KUR-2 (öğretmen ders notu)

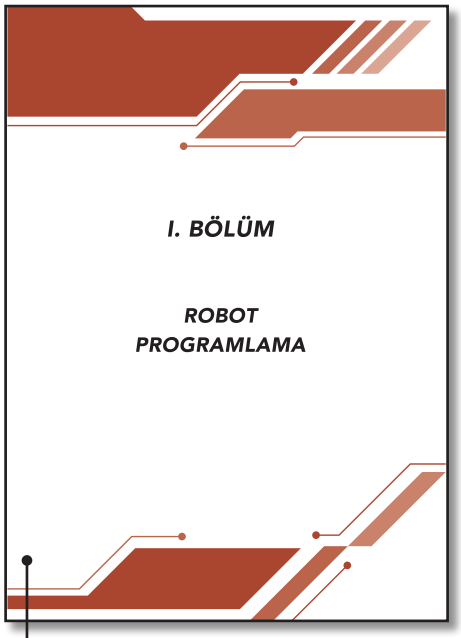
İÇİNDEKİLER

I. BÖLÜM

II. BÖLÜM

III. BÖLÜM

ORGANİZASYON ŞEMASI

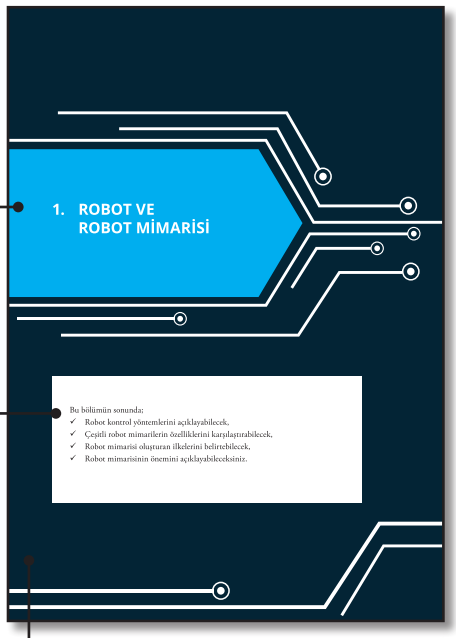


BÖLÜM KAPAKLARI

Kitap üç ana bölümden oluşmaktadır. Bu bölümler görselde yer alan kapaklar ile başlamaktadır.

ÜNİTE ADI VE NUMARASI

Ünitenin adı ve numarasını gösterir.



EDİNİMLER

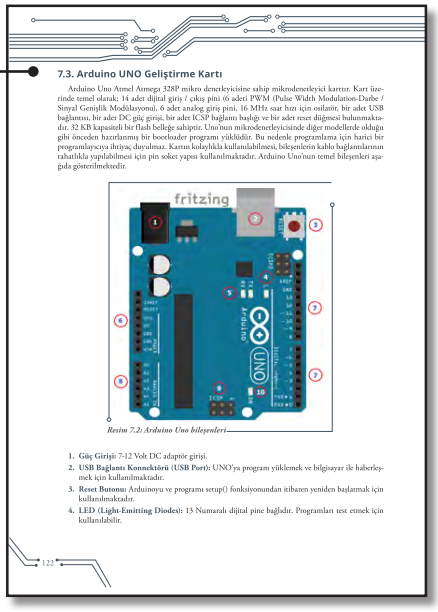
Ünite sonunda elde edilecek bilgi ve becerilerin listesi yer almaktadır.

ÜNİTE KAPAKLARI

Ünitenin içeriğini ifade eden bir görsel ile her üniteye özgü renk şablonu yer almaktadır.

KONU BAŞLIKLARI

İçindekiler bölümündeki sırayla listelenen konuların başlıkları yer almaktadır.


7.3. Arduino UNO Geliştirme Kartı

Arduino Uno Atmel Atmega 328P mikro denetleyicisine sahip mikrodenetleyici karttır. Kart üzerinde temel olarak; 14 adet dijital giriş / çıkış pini (6 adet PWM (Pulse Width Modulation-Darbe / Sinyal Genişlik Modülasyonu), 6 adet analog giriş pini, 16 MHz saat hızı için osilatör, bir adet USB bağlantısı, bir adet DC güç girişi, bir adet ICSP bağlantı başlığı ve bir ader reset düğmesi bulunmaktadır. 32 KB kapasiteli bir flaslı belleğe sahiptir. Uno'nun mikrodenetleyicisinde diğer modellerde olduğu gibi önceden hazırlanmış bir bootloader programı yüklüdür. Bu nedenle programlama için harici bir programlayıcıya ihtiyaç duyulmaz. Kartın kolaylıkla kullanılabilmesi, bileşenlerin kablo bağlantılarının rahatlıkla yapılabilmesi için pin soket yapısı kullanılmaktadır. Arduino Uno'nun temel bileşenleri aşağıda gösterilmektedir.

Resim 7.2: Arduino Uno bileşenleri

1. **Güç Girişi:** 7-12 Volt DC adaptör girişi.
2. **USB Bağlantı Konnektörü (USB Port):** UNO'ya program yüklemek ve bilgisayar ile haberleşmek için kullanılmaktadır.
3. **Reset Butonu:** Arduinoyu ve programı setup() fonksiyonundan itibaren yeniden başlatmak için kullanılmaktadır.
4. **LED (Light-Emitting Diodes):** 13 Numaralı dijital pine bağlıdır. Programları test etmek için kullanılabilir.

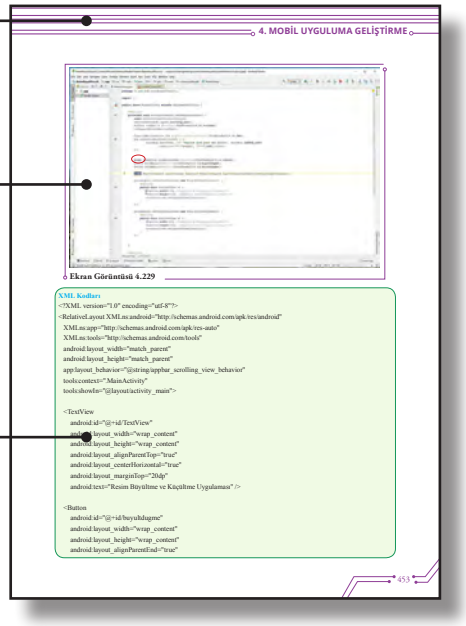
122

ÜNİTE BANTLARI

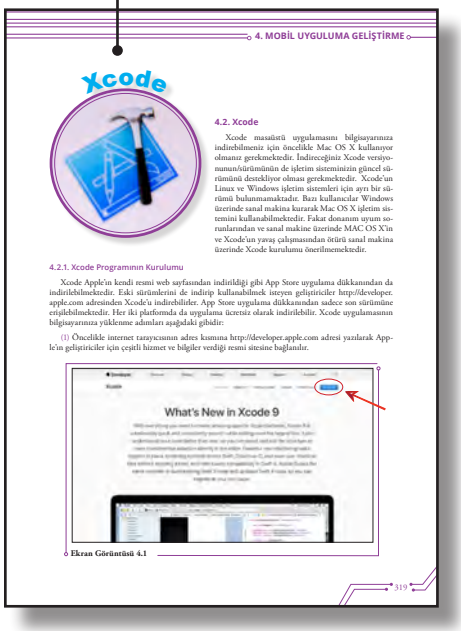
Her ünitede bu bölüm kendine özgü renge sahiptir.

EKRAN GÖRÜNTÜLERİ

UYGULAMA KODLARI



III. Bölümde Xcode programı ile yapılan uygulamaların yeraldığı bölümdür.



III. Bölümde Android Stüdyo programı ile yapılan uygulamaların yeraldığı bölümdür.

I. BÖLÜM

ROBOT PROGRAMLAMA

1. ROBOT VE ROBOT MİMARİSİ

Bu bölümün sonunda,

- ✓ Robot kontrol yöntemlerini açıklayabilecek,
- ✓ Çeşitli robot mimarilerin özelliklerini karşılaştırabilecek,
- ✓ Robot mimarisi oluşturan ilkelerini belirtebilecek,
- ✓ Robot mimarisinin önemini açıklayabileceksiniz.

1.1. Robot ve Robot Mimarisi

Robotlar, kendi kendine (otonom) veya önceden programlanmış görevleri yerine getirebilen elektromekanik araçlardır. Bunu yapabilmeleri için çevrelerini algılayabilmeleri, bilgi alabilmeleri ve bu bilgileri işleyerek tepkide bulunmaları, genellikle anlamlı bir amaç için kullanabilmeleri gerekmektedir. Bu açıdan değerlendirdiğimiz de robotun; işlem yapma, işlemin sonucunu belirleme ve karar verme yeteneği bulunmalıdır. Bu özelliklerin bulunduğu elektromekanik bir araç robot özelliğini kazanmaktadır. Robotlar bunları yaparken doğrudan bir operatörün kontrolünde çalışabildikleri gibi bağımsız olarak bir bilgisayar programının kontrolünde de çalışabilirler. Kontrol için farklı sistem ve yöntemler bir arada etkileşimli olarak kullanılabilmektedir. Robotları kontrol etmek için kullanılan sistem ve yöntemler temelde robot mimarilerini oluşturmaktadır.

1.2. Robot Kontrol Yöntemleri

Robotun hangi durumda ne yapacağına, ne tepki göstereceğine karar verme işlemine robot kontrolü adı verilmektedir. Robot kontrol sistemleri farklı araç ve programlardan oluşmaktadır. Aynı şekilde kontrol sistemleri için farklı kontrol yöntemleri kullanılmaktadır. Kullanılan kontrol yöntemleri şunlardır:

1. **Tepkisel (Reactive) Kontrol:** Etki tepki prensibiyle çalışan kontrol yöntemidir. Bu kontrol yöntemi uyaran-cevap ikililerinden oluşan kurallar içerir. Bu kontrol yöntemi "algılama" ve "hareket etme" modelini taban almıştır. Daha önce yapılan işlemleri hafızada tutmadığı gibi belirli bir hafızası da yoktur. Ne yapacağını düşünmediği için çok hızlıdır. Tepkisel kontrolü robotlar öğrenemez (kurallarını değiştirmez) ve ileriye yönelik plan yapamaz.

2. **Bilinçli (Deliberative) Kontrol:** Önce ayrıntılı olarak düşünen, sonra bu düşünce sonucuna göre hareket eden kontrol yöntemidir. Bu kontrol yöntemi "algılama", "planlama" ve "hareket etme" modelini taban almıştır. Planlama-araştırma gerektiği için ve araştırma da zaman aldığından bu kontrol yöntemi yavaştır. Bilinç kontrolü robotlarda düşünme ve hareket etme peş peşe gerçekleştirilir.

3. **Karma (Hibrit) Kontrol:** Düşünme ve hareket işleminin paralel olarak yürütüldüğü kontrol yöntemidir. Tepkisel ve bilinçli kontrol yöntemlerinin birleşmesinden oluşmaktadır.

4. **Davranışsal (Behavioral) Kontrol:** Karma kontrole alternatif olarak sunulmuştur. Tepkisel ve bilinçli hareket özelliklerine sahiptir.

Robot mimarileri bu kontrol yöntemlerinin uygulanmasındaki farklı görüş ve tartışmalardan ortaya çıkmış, belirli dönemlerde belirli mimarilere sahip robotlar üretilmiştir. Robotik anlama (Sense-algılama), planlama (Plan) ve hareket etme (Act-eylem) arasındaki ilişkiler ve algılayıcılar tarafından üretilen duyusal verilerin robotik sistem tarafından işlenmesindeki ve değerlendirilmesindeki farklar çeşitli mimarilerin ortaya çıkmasına neden olmuştur.

1.3. Robot Mimarisinde İlkeler

Robot mimarisinde uzun bir süre boyunca yaygın olarak kabul edilen ilkeller **sense, plan** ve **act** arasındaki ilişkilere dayalı olarak açıklanmıştır. **Sense,** algılayıcılardan bilgi almayı ve diğer bileşen-

ler için "çıktı" üretmeyi sağlamaktadır. **Plan,** algılayıcılardan veya diğer işlevsel bileşenlerden alınan tüm bilgileri kullanarak, gerçekleştirecek görevler üretmeyi, hareket planı yapmayı sağlar. **Act,** görevleri yerine getiren işlevsel bileşenlerin, hareket biçimini sağlar. Bu ilkelere dayalı olarak geliştirilen Hiyerarşik (Deliberative Kontrol) Mimari, Tepkisel (Reactive Kontrol) Mimari ve Karma (Hibrit Kontrol) Mimari yaygın olarak robot tasarımlarında kullanılır.

Hiyerarşik mimaride; "algılama", "planlama" ve "hareket etme" peş peşe gelen bir süreç olup herhangi bir robotik eylem için çevre algılanmalı, buna dayalı yapılacaklar planlanmalı ve bundan sonra harekete geçilmelidir. Her adımda robot, sonraki hamlesini planlamalıdır. Bilgiler ardışık olarak işlendiği için bileşenlerinin herhangi birindeki başarısızlık bütün sistemi etkilemektedir. Bu tür mimarilerin en önemli dezavantajı performanslarının düşüklüğüdür. Bu model robotun çalışmakta olduğu çevrenin değişmediği sabit durumlar örneğin endüstriyel ortamlar oldukça uygundur.

ROBOT İLKELERİ	GİRİŞ	ÇIKIŞ
ALGILAMA	Sensör verileri	Alınan bilgi
PLANLAMA	Bilgi (Algılanan ve/veya bilişsel)	Direktifler
HAREKET ETME	Direktifler	Hareket komutları

Şekil 1.1: Hiyerarşik mimari

Tepkisel mimaride; "algılama" ve "hareket etme" eş zamanlı olarak gerçekleştirilir. Algılamaya karşılık hareket üretilmektedir. Burada bir planlama süreci bulunmamaktadır. Aşağıdaki şekilde tepkisel mimaride ilkelerin ilişkileri gösterilmiştir.

ROBOT İLKELERİ	GİRİŞ	ÇIKIŞ
ALGILAMA	Sensör verileri	Alınan bilgi
PLANLAMA		
HAREKET ETME	Direktifler	Hareket komutları

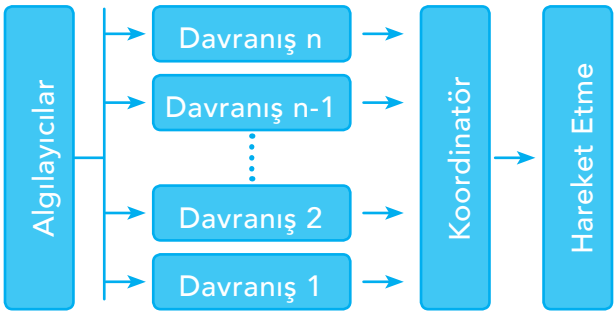
Şekil 1.2: Tepkisel mimari

Karma mimaride, 'algılama' ve 'hareket etme' eş zamanlı olarak gerçekleştirilirken planlama da yapılmaktadır. Aşağıdaki şekilde karma mimaride ilkelerin ilişkileri gösterilmiştir.

ROBOT İLKELERİ	GİRİŞ	ÇIKIŞ
PLANLAMA	Bilgi (Algılanan ve/veya bilişsel)	Direktifler
ALGILAMA HAREKET ETME	Sensör verileri	Hareket komutları

Şekil 1.3: Karma mimari

Geçmiş yıllarda robot mimarileri arasındaki temel fark daha planlamacı veya daha fazla tepkisel olup olmadığına dayanırdı. Daha sonra davranışsal mimariler öne çıkmıştır. Davranışsal mimaride; robotun çevresiyle ilgili durumlar için programlanmasına gerek yoktur. Çevresiyle ilgili bütün bilgiler algılayıcıları aracılığıyla kendisine ulaşmaktadır. Algılayıcılarından elde ettiği bu bilgileri yavaş yavaş yakın çevresindeki değişikliklere göre hareketlerini düzeltmek için kullanmaktadır. Bu yöntemle robot karşılaşabileceği her türlü durumla başa çıkabilecek bir tepkisel davranış sağlamaktadır. Aşağıdaki şekilde davranışsal mimaride ilkelerin ilişkileri gösterilmiştir.



Şekil 1.4: Davranışsal mimari

Günümüzde ise Olasılıksal (Probabilistic) Robotik olarak da adlandırılan istatistiksel robotik alan; robotların öngörülemeyen, belirsizlik içeren ortam ve olaylara maruz kaldığı durumlarda istenilen robotik kontrol ve davranışları yapmasını sağlar. Daha önce karşılaşmadığı ortamlarda etkin bir şekilde çalışabilen robotların geliştirilmesini amaçlanmaktadır. Bu nedenle bir robotun, tanımlanan istatistiksel fonksiyonlara dayalı belirli bir hareket ya da eylemi için, en olabilecek sonuçları geliştirmesi ve sonra da bunlardan en uygun olanı uygulayabilmesi gerekmektedir.

Robot mimarisini bir örnek üzerinde anlamaya çalışalım. Robot yönetim sistemi (pilot), robot görüş (vizyon) sistemi ve robot yönlendirme (navigasyon) sistemi olmak üzere üç sistem tarafından kontrol edilen bir robot düşünelim. Aslında bu üç sistem, robotumuzu kontrol etmek için kullandığımız mimariyi oluşturmaktadır.

Robot yönetim sistemi; robotun yönetimini sağlayan, örneğin bir engele çarpmayı önlemek için robotun hareket yönünü değiştiren, robotu hedef bölgeye doğru götürmek için kullanılan sistemdir. Robot görüş sistemi; belirli bir alanındaki bilinen yer işaretlerini, engelleri tanıması veya yenilerini bulması için kullanılan sistemdir. Robot yönlendirme sistemi ise robotun yerini ve hareket yönünü belirlemek için kullanılan sistemdir. Robot kontrolünü sağlamada bu üç sistem işbirliği içinde hareket etmek zorundadır. Örneğin navigasyon sistemiyle robot; hedefe doğru ilerlerken, bulunduğu ortamın belirli bir

alanındaki bilinen yer işaretlerini tanımak veya yenilerini bulmak için vizyon sistemine, aynı zamanda hedef bölgeye doğru ulaşmak için de pilot sistemine ihtiyaç duymaktadır. Pilot sistem bir engeli önlemek için robotun hareket yönünü değiştirmelidir. Üstelik pilot, önünde bir engel bulunup bulunmadığını kontrol etmek için kameraya ihtiyaç duyabilir. Aynı zamanda navigasyon sisteminin, bilinen yer işaretlerini tanıyarak robotun yerini belirlemek için arkasına bakması da gerekebilir. Bu nedenle, farklı sistemler arasındaki bu etkileşimleri sağlamak için bazı koordinasyon mekanizmalarına ihtiyaç duyulmaktadır. Kullanılacak mekanizma ve robotik sistemin mevcut kaynakları, bu etkileşimlerden elde edilen birleşimle birlikte robotun hedefine ulaşmasını sağlamak zorundadır. Bu nedenle robot mimarileri her zaman birer paradigma olarak ele alınmıştır.

1.4. Düşünelim/Tartışalım

Yer yer bataklıkların ve ağaçların bulunduğu bir araziyi geçmek zorunda olan bir robotumuzun olduğunu düşünelim. Robotumuzun bu araziyi bataklığa düşmeden, ağaçlara çarpmadan geçebilmesi istenmektedir. Buna göre:

a) Bu robot nasıl bir yönetim sistemine sahip olmalıdır? Niçin?
b) Robotta hangi kontrol araçlarının bulunması istersiniz? Bu araçları niçin tercih ettiniz? Tartışınız.

1.5. Değerlendirme Soruları

1. **Bir robotun, kendi kendine (otonom) veya önceden programlanmış görevleri yerine getirebilmesi için aşağıda belirtilen özelliklerden hangisine sahip olması yeterlidir?**
   a) Çevresini algılayabilme yeteneğinin bulunması yeterlidir.
   b) Bulundukları ortamdan bilgi alabilmeleri ve bu bilgileri işleyerek tepkide bulunabilmeleri yeterlidir.
   c) İşlem yapma, işlemin sonucunu belirleme ve karar verme yeteneği bulunmalıdır.
   d) Aldıkları bilgileri genellikle anlamlı bir amaç için kullanabilmeleri yeterlidir.
   e) Bir operatörden bağımsız olarak işlem yapma yeteneklerinin bulunması yeterlidir.

2. **Robotları kontrol etmek için kullanılan farklı sistem ve yöntemler aşağıdakilerden hangisini oluşturmaktadır?**
   a) Robot kontrol teknolojilerini
   b) Robot kontrol yöntemlerini
   c) Robot kontrol sistemlerini
   d) Robot mimarisini
   e) Robot paradigmalarını

3. Uyaran-cevap ikililerinden oluşan kurallar içeren robot kontrol yöntemi aşağıdakilerden hangisidir?

a) Davranışsal (Behavioral) Kontrol
b) Tepkisel (Reactive) Kontrol
c) Karma (Hibrit) Kontrol
d) Bilinçli (Deliberative) Kontrol
e) Olasılıksal Kontrol

4. Aşağıdakilerden hangisi karma kontrole alternatif olarak sunulan robot kontrol yöntemidir?

a) Davranışsal (Behavioral) Kontrol
b) Tepkisel (Reactive) Kontrol
c) Karma (Hibrit) Kontrol
d) Bilinçli (Deliberative) Kontrol
e) Olasılıksal Kontrol

5. Aşağıdakilerden hangisi önce ayrıntılı olarak düşünen, sonra bu düşünce sonucuna göre hareket eden kontrol yöntemidir?

a) Davranışsal (Behavioral) Kontrol
b) Tepkisel (Reactive) Kontrol
c) Karma (Hibrit) Kontrol
d) Bilinçli (Deliberative) Kontrol
e) Olasılıksal Kontrol

6. Düşünme ve hareket işleminin paralel olarak yürütüldüğü kontrol yöntemi aşağıdakilerden hangisidir?

a) Davranışsal (Behavioral) Kontrol
b) Tepkisel (Reactive) Kontrol
c) Karma (Hibrit) Kontrol
d) Bilinçli (Deliberative) Kontrol
e) Olasılıksal Kontrol

7. Robotun çalışmakta olduğu çevrenin değişmediği sabit durumlar (örneğin endüstriyel robotlar) için oldukça uygun olan robot mimarisi aşağıdakilerden hangisidir?

a) Hiyerarşik Mimari
b) Tepkisel Mimari
c) Karma Mimari
d) Davranışsal Mimari
e) Olasılıksal Robotik

8. Daha önce karşılaşmadığı ortamlarda etkin bir şekilde çalışabilen robotların geliştirilmesini amaçlayan robotik alanı aşağıdakilerden hangisidir?

a) Hiyerarşik Mimari
b) Tepkisel Mimari
c) Karma Mimari
d) Davranışsal Mimari
e) Olasılıksal Robotik

9. Aşağıdakilerden hangisi robotun çevresiyle ilgili durumlar için programlanmasına gerek olmadığını savunan mimaridir?

a) Hiyerarşik Mimari
b) Tepkisel Mimari
c) Karma Mimari
d) Davranışsal Mimari
e) Olasılıksal Robotik

10. Çeşitli robot mimarilerin ortaya çıkmasının nedenini aşağıda verilen görüşlerden hangisi daha güçlü olarak açıklamaktadır?

a) Robot mimarileri robot kontrol yöntemlerindeki farklılıklardan ortaya çıkmıştır.
b) Belirli dönemlerde belirli mimarilere sahip robotların üretilmesi sonucu ortaya çıkmıştır.
c) Robotik anlama (Sense-algılama), planlama (Plan) ve hareket etme (Act-eylem) arasındaki ilişkilerin yorumlanma şeklinden ortaya çıkmıştır.
d) Robotik algılayıcılar tarafından üretilen bilişsel verilerin robotik sistem tarafından işlenmesindeki ve değerlendirilmesindeki farklardan ortaya çıkmıştır.
e) Robotların işlem yapma yeteneği, işlemin sonucunu belirleme yeteneği ve karar verme yeteneği arasındaki farklılıklardan ortaya çıkmıştır.

2. ROBOT TÜRLERİ VE EĞİTSEL AMAÇLI ROBOTLAR

Bu bölümün sonunda,

- ✓ Kullanılan uygulama alanlarına göre robot türlerine örnek gösterebilecek,
- ✓ Hareket mekaniğine göre robot türlerini açıklayabilecek,
- ✓ Eğitsel amaçlı robot türlerinin özelliklerini özetleyebilecek,
- ✓ Robot türlerini sınıflandırabilecek,
- ✓ Robot türlerini karşılaştırabileceksiniz.

2.1. Robot Türleri ve Eğitsel Amaçlı Robotlar

Günümüzde, robotlar pek çok alanda çok farklı görevler üstlenmekte ve robotlara devredilen işlerin sayısı sürekli olarak artmaktadır. Bu nedenle pek çok farklı ölçüte göre robotlar sınıflandırılabilmektedir (Benson, 2012; Robotpark, 2017; Robot Wiki, 2017). Örneğin hareketli (mobil) veya sabit olmasına göre, kullanılan alanlara göre, hareketin cinsine göre. Sınıflandırmada temel ölçüt robot için "Ne yapar?" ve "Bunu nasıl yapar?" sorularına verilen yanıt olmalıdır. Genellikle elde edilen yanıtlar robotları; uygulamaya göre robotlar ve hareket mekaniğine göre robotlar olmak üzere iki temel sınıfa ayırmaktadır. Özellikleri ve yapısı nedeniyle herhangi bir tür içerisine girmeyen veya bir tür içerisine henüz dâhil edilmeyen robotlar dikkate alınmamıştır. Eğitsel amaçlı robotlar ise özellikleri nedeniyle ayrı kategoride incelenmiştir.

2.2. Kullanılan Uygulama Alanlarına Göre Robotlar

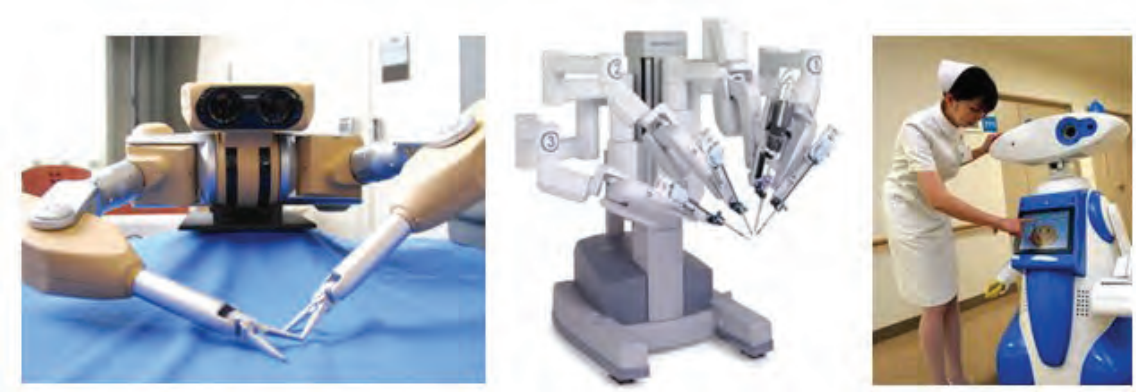
Endüstriyel Robotlar: Herhangi bir endüstriyel üretim ortamında kullanılan robotlardır. Endüstriyel robotların en önemli özelliği kollara sahip olmasıdır. Genellikle kaynak, birleştirme, boyama, eşya ve araç üretimi, montaj ve kontrol uygulamalarında kullanılmaktadır. Bu uygulamalar için gerekli olan malzeme taşıma, malzeme yükleme, kesme, tutma, yerleştirme, şekil verme, değiştirme, yüzey kaplama, silindirik ve düzlem yüzey taşlama gibi imalat işlemleri bu kollar aracılığıyla gerçekleştirilmektedir.

Resim 2.1: Endüstriyel robotlar

Ev Robotları: Evde kullanılmak için geliştirilmiş robotlardır. Elektrikli süpürge, havuz temizleyici, bahçe süpürgeleri, oluk temizliği ve diğer ev ve bahçe işlerini yapabilen robotları içerir. Bu ortamda kullanılan ayrıca, bazı gözetim ve Telepresence robotlar ev robotları olarak kabul edilebilir. Telepresence robotlar insanların fiilî olarak bulunmaması gereken nükleer, kimyasal felaketler gibi senaryolarda, sağlık alanında, askerî casusluk gibi birçok görevde kullanılması öngörülmüş insan kontrolünde çalışan robotlardır. Ürün satışı ve reklam, tur rehberi, gece bekçisi, fabrika müfettişi ve sağlık danışmanlığı için de kullanılmaktadır. Bir uzaktan eğitim sınıfında, bir telepresence robotu, sınıf içi eğitmen olabilir, sınıfın etrafında dolaşabilir ve öğrencilerle yüz yüze etkileşimde bulunabilirler. Kablosuz internet bağlantısı olan uzaktan kumandalı ve tekerlekli yapıdadırlar. Genellikle, bu robotlar video ve ses yetenekleri sağlamak için bir tablet kullanırlar.

Resim 2.2: Ev robotları

Tıbbi Robotlar: İlaç üretiminde ve dağıtımında, tıbbi kurumlarda, hastanelerde malzeme taşımak, doktorlara yardımcı olmak için kullanılan robotlardır. Bu robotların ilk ve en önemlisini cerrahi robotlar oluşturur. Cerrahi operasyonlarda doktorların en önemli yardımcısı konumundadır.



Resim 2.3: Tıbbî robotlar

Servis Robotları: Kullanım şekli açısından diğer türlere girmeyen robotlardır. Bu robotlar özerk üretim faaliyetlerinde kullanılmaz. İnsan tarafından yapılan tehlikeli ve zor işlerde insana yardımcı olması için geliştirilmiştir. İnsan refahını sağlamaya dönük her tür yararlı hizmeti gerçekleştirmek için tam veya yarı hizmet desteği veren robotlardır.

Resim 2.4: Servis robotları

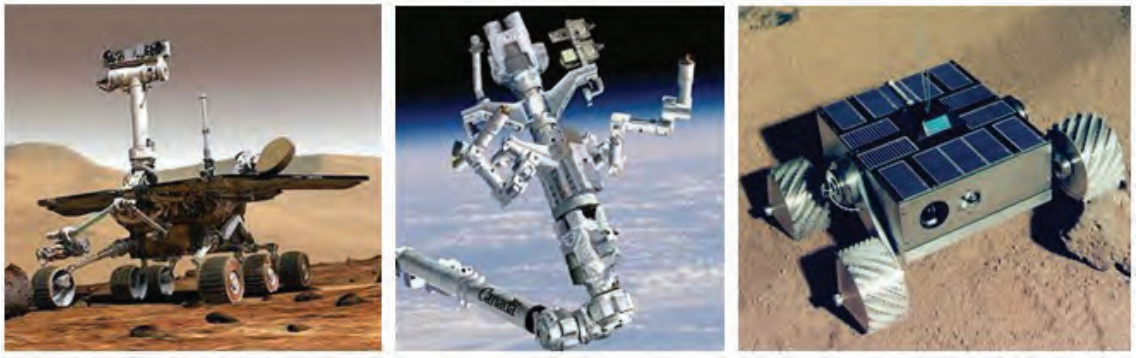
Askerî Robotlar: Askerî kullanım için geliştirilmiş robotlardır. Bomba imha robotları, farklı ulaşım robotları, robotik keşif uçağı bu tipte robotlardır. Genellikle başlangıçta askerî amaçlar için oluşturulan bu robotlar kolluk, arama kurtarma ve diğer ilgili alanlarda da kullanılabilmektedir.

Resim 2.5: Askerî robotlar

Eğlence Robotları: Bunlar herhangi bir hizmette kullanılmayıp çoğunlukla eğlence ve oyun arkadaşlığı için tercih edilen robotlardır. Bu robotlar çok geniş bir yelpazede yer almaktadır. AIBO, Poo-Chi gibi robotik köpekler ve hayvanlar, ses tanıma ve yürüme gibi bazı gelişmiş özellikleri sahip QRIO, Robosapien gibi insansı oyuncak robotlar, hareket simülatörleri olarak kullanılan belden robot kolları gibi fonksiyonel robotlar da bu kategoride değerlendirilmektedir.

Resim 2.6: Eğlence robotları

Uzay Robotları: Uzayda kullanılmak için üretilen robotlardır. Bu tür robotlar Uluslararası Uzay İstasyonu'nda, Mars'ın keşfinde ve diğer uzay görevlerinde kullanılmaktadır. Bu anlamda uzay sondaları da birer robottur.



Resim 2.7: Uzay robotları

Hobi ve Yarışma Robotları: Kişisel olarak yapılan robotlardır. Çizgi takipçileri, sumo-botlar, uçan robotlar gibi sadece eğlence ve herhangi bir görevi yerine getirme konusunda yarışmak için yapılan robotlar bu kategoride değerlendirilmektedir. Birçok ulusal ve uluslararası yarışma bu amaçla gerçekleştirilmektedir.

Resim 2.8: Hobi ve yarışma robotları

Sanal Robotlar: Sanal robotlar gerçek hayatta fiziksel olarak bulunmayan robotlardır. Sanal robotların yapı taşları bilgisayar programlarıdır. Sanal robotlar, gerçek bir robot simülasyonunu ya da sadece tekrarlanan bir görevi gerçekleştirebilirler. İnternet üzerinde kullanabileceğiniz sohbet robotları, çağrı merkezleri için müşteri temsilcisi robotları gibi pek çok örneği kullanılmaktadır. Robot simülatörleri kullanılarak maliyet ve zaman tasarrufu sağlanmaktadır.

Resim 2.9: Sanal robotlar

2.3. Hareket Mekaniğine Göre Robotlar

Sabit Robotlar: Sabit robotlar sürekli tekrarlayan görevlerini pozisyonlarını değiştirmeden yapan robotlardır. Robotun sabit olması ile anlatılmak istenen robotun temelinin sabit olmasıdır. Yoksa robotun kolları hareket hâlindedir. Çoğu sabit robotlar sanayi ortamlarında imalat ve montaj sektöründe kullanılmaktadır. Bu türün içine Kartezyen / Portal robotlar, Silindirik robotlar, Küresel robotlar, SCARA robotlar, Belden robotlar (robotik kollar) ve Paralel robotlar girmektedir.

Resim 2.10: Sabit robotlar

Tekerlekli Robotlar: Tekerlekli robotlar pozisyonlarını tekerlekleri ile değiştirebilen mobil robotlardır. Tekerlekli hareketi mekanik olarak sağlamak üretim açısından kolay ve düşük maliyetlidir. Aynı zamanda tekerlekli hareketin kontrolü diğer mobil robotlara oranla daha kolaydır. Bu nedenle tekerlekli robotlar en sık karşılaşılan mobil robot tiplerindendir. Bu robot sınıfı kendi içerisinde çoğunlukla tekerlek sayısına göre sınıflandırılır. Bu türün içerisinde tek tekerlekli robotlar, mobil top robotlar, iki tekerlekli robotlar, üç ve daha fazla tekerlekli robotlar, çok tekerlekli robotlar bulunmaktadır. Bu robotlar düz alanlarda çok etkili olup arazi koşullarında pek yararlı olamaz.

Resim 2.11: Tekerlekli robotlar

Paletli Robotlar: Paletli robotlar tekerlekli olmasalar da çalışma prensibi açısından tekerlekli robotlara çok benzer olarak çalışır. Bu robotlar hareket etmek için tekerlekleri yerine tanklar gibi paletlerini kullanır. Bu hareket yöntemi düzensiz, yumuşak, kaygan, karlı ya da çamurlu zeminlerde tekerlekli robotlara göre daha fazla avantaj sağlamaktadır. Paletler yerle temas alanını genişlettiği için robotun ağırlığı daha geniş bir yüzeye dağılmakta bu tür zeminlere saplanmasını engellemektedir. Bu nedenle paletli robotlar tekerlekli robotlara göre daha fazla ağırlık taşıyabilir.

Resim 2.12: Paletli robotlar

Ayaklı Robotlar: Ayaklı robotlar da tekerlekli robotlar gibi mobil robotlardandır ancak hareket yöntemleri ve teknolojisi çoğunlukla tekerlekli robotlara daha üstün ve karmaşıktır. Ayaklı robotlar gelişmiş robotlardır. Hareketlerini sağlamak için ayaklarından faydalanırlar ve tekerlekli robotlara göre sorunlu olan pek çok zeminde hareket edebilirler. Bu tip robotlarda denge en önemli unsurdur. Bu robotların üretim ve kontrolü daha karmaşık ve maliyeti tekerlekli robotlara göre daha yüksektir. Bu tür içerisinde tek ayaklı robotlar, iki ayaklı robotlar (insansı –humanoid-robotlar), üç ayaklı robotlar, dört ayaklı robotlar, altı ayaklı robotlar ve çok ayaklı robotlar sayılabilir. Günümüzde birçok kurum ve üniversite tarafından araştırılan ve geliştirilen robot tipidir.

Resim 2.13: Ayaklı robotlar

Yüzen Robotlar: Yüzen robotlar, suda hareket edebilen robotlardır. Bu robotlar balıklar gibi yüzgeçlerini kullanarak su içerisinde manevra yapabilmektedir. Genellikle uzaktan kumandayla kontrol edilmekle birlikte otonom olarak da hareket edebilmektedir. Bunlar deniz kaynakları ve balık türleriyle ilgili araştırma ve incelemelerde, su altı arkeolojik keşiflerinde, su altı fotoğrafçılığı, su altı haritacılığı, petrol platformlarını denetleme, inceleme ve olası hasarların tespitinde kullanılmak için tasarlanmış deneysel robotlardır.

Resim 2.14: Yüzen robotlar

Uçan Robotlar: Uçan robotlar; kanat, pervane ya da balonları ile havada asılı kalarak ve manevra yaparak hareketlerini sağlayan hareket eden robotlardır. Bu robotlara örnek olarak uçak benzeri kanatlı robotlar, kuş/böcek benzeri kanatlı robotlar, pervaneli multikopterler, insansız hava araçları ve balonlu robotlar verilebilir. Bu robotlar doğal afetlerde arama-kurtarma, araştırma, bilgi edinme görevlerinde, insanlar tarafından yapılması gereken tehlikeli görevlerin yerine getirilmesinde, mal ve ürünlerin dağıtımında ve gözetiminde, tarımsal alanların kontrolünde, eğlence ve hobi amacıyla pek çok alanda kullanılmaktadır.

Resim 2.15: Uçan robotlar

Yılan Robotlar: Bu robotlar sahip oldukları hareket yetenekleri ile her tür ortamda çok yönlü olarak kullanılabilmektedir. Duvarlar ve boşluklar arasında dolaşabilmeleri, arama ve kurtarma faaliyetlerinde bilgi almak için çok uygun yapıda olmaları bu robotların geliştirilme nedenlerini oluşturmaktadır.

Resim 2.16: Yılan robotlar

Yumuşak Elastik Robotlar: Hareket organları ve yapıları esnek robotlardır. Genellikle gövdeleri silikondan, diğer organları (el, kol vs.) ise elektrik akımıyla uyarıldığında boyut veya şekilde değişiklik yapan bir tür plastikten - elektroaktif polimer - üretilmiş robotlardır. Bu tür, robotik alanında kendilerine yeni yeni yer bulan robotlardır. Bu robotların esin kaynakları genellikle kalamar ve toprak solucanı gibi hayvanlardır. Kendilerine özgü davranışlara sahiptirler.

Resim 2.17: Yumuşak elastik robotlar

Mobil Küresel Robotlar (Robotik Toplar): Bu robotlar görünüş olarak topa benzeyen robotlardır. Kar veya kum gibi zeminlerde tekerlekli robotlara göre daha fazla performans gösterdikleri, ayrıca düşme riski daha az olduğu için tercih edilmektedir. Daha çok bilimsel araştırmalarda, tehlikeli ve zor arazi koşullarında (gezegen keşiflerinde) kullanım için tasarlanmakla birlikte oyun amacıyla geliştirilen çok fazla çeşidi de bulunmaktadır.

Resim 2.18: Mobil küresel robotlar

Hibrit Robotlar: Bu tanım hem birden fazla hareket mekaniğine sahip robotlar için hem de sibernetik robotlar için kullanılmaktadır. Sibernetik robotlar hem elektronik hem de biyolojik (canlı) elemanları içermektedir. Biyolojik elemanlar olarak deney hayvanlarının nöronları (genellikle fare) kullanılmaktadır. Bu nöronlara bağlı çipler robotik sistemin temelini oluşturmaktadır. Bu robotların birer sibernetik organizma olduğu rahatlıkla söylenebilir. Üniversite ve araştırma kuruluşlarında geliştirilen araştırma amaçlı deneysel robotlardır.

Resim 2.19: Hibrit robotlar

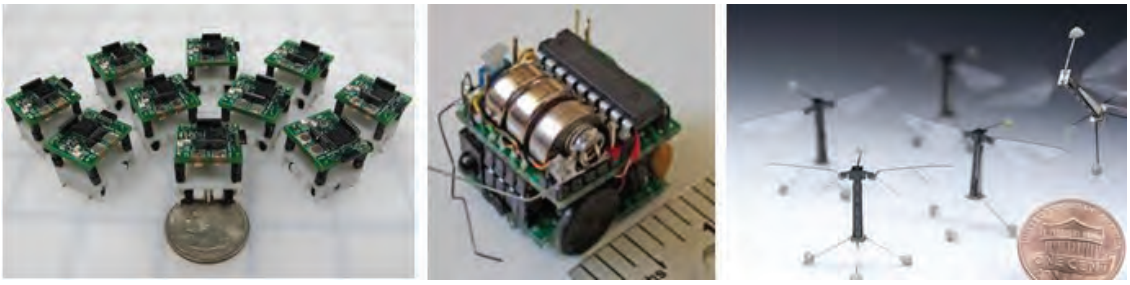
Sürü Robotları: Sürü robotları, yapı olarak birleşik ve tek olmak yerine çok sayıda benzer ve basit fonksiyonellikte robotun ortak çalışmaları ile işleyen robotlardır. Modüler robotlarla benzerlikler gösterseler de sürü robotlarının elemanları çok daha fazla sayıda ve fonksiyonel açıdan çok daha basittir.

Resim 2.20: Sürü robotları

Modüler Robotlar: Modüler robotlar da sürü robotlar gibi robotik sistemi değişik robotik parçalara dağıtılmış robot sistemleridir. Bu tür robotlar yeni koşullara uyum veya yeni görevleri gerçekleştirmek amacıyla kendilerini yeniden yapılandırabilmektedir. Bu amaçla kendi parçalarının bağlantılarını yeniden düzenleyerek kendi şeklini değiştirebilmektedir. Bu robotların sürü robotlarından farkı ise, parçaların daha gelişmiş ve nispeten daha az sayıda olmasıdır. Modüler robotların bir diğer özelliği ise parçalar arası birleşimlerle oluşturdukları konfigürasyonların değişik fonksiyonları bulunan farklı robotlar oluşturabilmesidir. Modüler yapı blokları genellikle tutucular, ayaklar, tekerlekler, kameralar, yük ve enerji depolama gibi birimlerden oluşmaktadır.

Resim 2.21: Modüler robotlar

Mikro Robotlar: Mikro robotlar, hem mikro hassasiyette işlem yapabilen farklı boyutlardaki robotları hem de mikrometre boyutlarında olup mikro hassasiyette işlem yapabilen robotları ifade etmektedir. Bu tür robotlar uzay çalışmalarında, tıpta, askerî uygulamalarda ve daha pek çok yerde kullanılmaktadır. Mikro robotların kullanıldığı en önemli alan, tıbbi mikro robot uygulamalarıdır. Bu alan insan vücudundaki çeşitli hastalıkları insana rahatsızlık vermeden tanıyıp, doğrudan hasta olan noktaya ilaç verebilecek, biyopsi ve cerrahi müdahale yapabilecek küçük kablosuz robotların geliştirilmesini amaçlamaktadır.

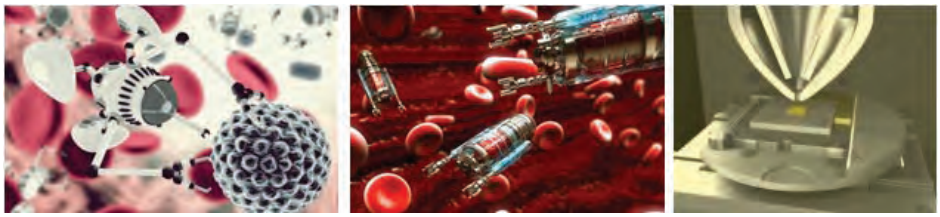


Resim 2.22: Mikro robotlar

Nano Robotlar: Nano robotlar nanometre düzeyinde hassasiyetle işlem yapabilen çok hassas robotlardır. Bu tür robotlar boyut olarak nanometre düzeyinde ifade edilen çok küçük ölçülerde (atom ve molekül boyutlarında) yapılmış olabildiği gibi nano ölçekte doğrulukla hareket edebilen makro veya mikro ölçekli robotlar da olabilmektedir. Bu robotlar nanoteknoloji, biyoteknoloji ve biyomedikal alanlarının gelişimine katkıda bulunmak için yine bu alanlardaki gelişmelerden yararlanarak üretilmektedir. Mikron ve nanometre boyutlarında cisimleri, parçaları ve biyolojik maddeleri çok hassas olarak manipüle edebilecek nano robotların geliştirilmesi çalışmaları sürdürülmektedir.

Resim 2.23: Nano robotlar

Beam Robotlar: Beam (Biology, Electronics, Aestetics, Mechanics) robotlar yapılarında temel elektronik bileşenlerin kullanıldığı robotlardır. Bu nedenle beam robotların yapımında genellikle programlanabilir mikroişlemci veya mikrodenetleyici kullanılmaz. Bu tür robotlar temel elektronik elemanlarıyla (foto-diyotlar, kapasitörler, tersleyiciler ve transistörler gibi) yapılan basit lojik devrelerle tıpkı bir sinir ağı gibi oluşturulur. Genellikle oluşturulan mantık devreleri ile algılayıcılardan algıladıkları sinyalleri yorumlayarak kendinden içgüdümle hareket ederler. Genellikle güneş enerjisinden güçlerini alırlar. Bu tür robotlar doğadan esinlenerek yapılmaktadır.



Resim 2.24: Beam robotlar

2.4. Eğitsel Amaçlı Robotlar

Yüzyılımızda robotların eğitsel amaçlarla kullanımı giderek artmaktadır. Eğitsel amaçlarla geliştirilen ve kullanılan çok fazla sayı ve türde eğitsel robot, robot kiti ve seti ortaya çıkmıştır. Robotlar eğitimde daha çok FTMM (Fen, Teknoloji, Mühendislik ve Matematik) eğitimini desteklemek amaçlı kullanılmaktadır. Fakat günümüzde 21. yüzyıl becerileri olarak adlandırdığımız problem çözme, birlikte çalışma, karar verme, bilgi-işlemsel düşünme gibi çeşitli becerilerin kazanılmasında da etkili olduklarının belirlenmesiyle diğer eğitim alanlarında da (sosyal bilimlerde) kullanımı yaygınlaşmaya başlamıştır. Özellikle öğrencilerin keşfetme, eleştirel düşünebilme ve sosyal becerilerini geliştirmedeki etkileri dikkati çekmektedir. Bu konuda yapılan çalışmaların büyük bir bölümü robotların eğitime olan pozitif etkisini ortaya koymasıyla sonuçlanmıştır. Örneğin programlama dilleri öğretiminde robot kullanımıyla birlikte öğrencilerin problem çözme becerilerinin geliştiğini, iş birliği içerisinde takım çalışmalarıyla bilgiyi paylaşarak öğrendikleri belirlenmiştir. Bunların sağlanmasında kullanılabilecek eğitsel robotlar farklı şekillerde sınıflandırılmaktadır. Bazı sınıflamalar robotun kullanılacağı eğitim türüne göre, bazı sınıflamalar robotun yapısına (Elektronik Robot Kitleri, Mekanik Robot Kitleri, İnsansı –Hümanoid– Robotlar gibi), bazı sınıflamalar maliyetine göre, bazı sınıflamalar da kullanılabilecek yaş guruplarına göre yapılmaktadır.

Blok (LEGO Benzeri) Tabanlı Robot Montaj Setleri: Öğrencilerin kendi robotlarını tasarlamaları, inşa etmeleri ve onları programlayarak harekete geçirmeleri için birbirine kolayca bağlanabilen parçalardan oluşan robot setleridir. Bu tür robotik setler oldukça fazla sayıda yapı ve hareket bileşenlerinden oluşmaktadır. Örneğin VEX IQ Süper Kit içerisinde 850 adet yapısal ve hareket bileşeni, 4 adet akıllı motor, 7 çeşit algılayıcı, robot kontrol kumandası, robot kontrol kartı ve piller bir saklama kutusu içerisinde yer almaktadır. LEGO® MINDSTORMS® EV3 Education Ana Set toplam 541 parçadan oluşmakta, içerisinde yapısal bileşenler, EV3 programlanabilir kontrolör, renk algılayıcı, ultrasonik algılayıcı, buton algılayıcı ve jiroskop algılayıcı bulunmaktadır. Yine aynı şekilde Fischertechnik ROBOTICS TXT Discovery Set 310 parçadan oluşmaktadır.

Resim 2.25: Blok (LEGO benzeri) tabanlı tobot montaj setleri

Düşük Maliyetli Programlanabilir Robotik Kol Setleri: Robotik kollar insan kollarından esinlenerek tasarlanmış ve benzer fonksiyonlara sahip robotik sistemlerdir. Robotik kol, programlanabilir yapıda, mekanik parçaların bütünü ya da karmaşık bir robotun bir parçası olarak nitelendirilebilir. Bir kol sistemi farklı eklemlerin birbirlerine bağlanması ile oluşmaktadır. Eklemlerin bağlantı noktalarında bulunan motorların hareketleri robot kolun yapabileceği hareketlerin göstergesini oluşturmaktadır. Robot kolların uçlarında gerçekleştirilmesi istenen işlemlere uygun bir araç bulunur. Bu araç kavrama, kaldırma, boyama, resim çizme veya yazma gibi değişik işlemler için kullanılabilir. Bu sayede temel robotik ilkelerin ve programlamanın öğretilmesinde kullanmak mümkündür. Bu amaçla gerçekleştirilmiş montajlı veya montajsız olarak bulunabilen pek çok kit satılmaktadır.

Resim 2.26: Düşük maliyetli programlanabilir robotik kol setleri

Düşük Maliyetli Minimum Özelliklerde Mobil Robot Tasarım Kitleri: Pek çok firmanın ürettiği bu tür eğitsel robotlar kullanıma hazır ama tamamen montajlanmamış şekilde satışa sunulmaktadır. Temel düzeyde özelliklere ve algılayıcılara sahip, ancak genişleme özellikleri ile sonradan herhangi bir bileşenin eklenmesine olanak veren kitlerdir. Parallax Robotics Kitleri (Robotics Arduino Shield Kit, Boe-Bot Robot Kit, ActivityBot), Pololu Robot Kitleri (Zumo Robots, 3pi Robot) ve Makeblock (mBot - STEM Educational Robot Kit, mBot Ranger, Starter Robot Kit) bunlara örnek olarak verilebilir.

Resim 2.27: Düşük maliyetli minimum özelliklerde mobil robot tasarım kitleri

Açık Kaynaklı Düşük Maliyetli Mobil Robot Platformları: Eğitim amaçlı bu robotlar tamamen açık kaynak kodlu (mekanik ve elektronik yapı) ve açık kaynak yazılım araçları (OpenScad, FreeCAD ve Kicad) ile özel olarak tasarlanmış ve paylaşıma sunulmuş robotlardır. Bu robot platformları öğrenci-

lerin robot programlamayı öğrenmelerine, aynı zamanda kolayca kasayı, yapıyı değiştirebilmelerine ve yeni özel parçaları oluşturmalarına izin vermektedir. Açık kaynak, donanım ve yazılım robotun serbestçe değiştirilebilmesine, kopyalanabilmesine ve İnternet üzerinden paylaşılabilmesine olanak vermektedir. Genel olarak son derece ekonomik bileşenlerden oluşturulmaktadır. Teknoloji ve robot marketlerde satılan onlarca model dışında, Mini Skybot Robot V1, Miniskybot 2, MIT SEG: An Origami-Inspired Segway Robot gibi tanınmış modeller bu tür robotlara örnek olarak verilebilir.

Resim 2.28: Açık kaynaklı düşük maliyetli mobil robot platformları

Düşük Maliyetli, Tam Monte Edilmiş Mobil Robotlar: Bu robotlar tamamen montajı yapılmış, kullanıma hazır olarak satışa sunulan eğitsel robotlardır. Bazılarında kendine özgü görsel veya metin tabanlı programlama araçları kullanılırken bazılarında açık kaynak programlama araçları kullanılabilmektedir. Genişleme özellikleri daha sınırlı olabilmektedir.

Resim 2.29: Düşük maliyetli, tam monte edilmiş mobil robotlar

Modüler Eğitsel Robot Kitleri: Modüler eğitsel robotların robotik sistemi değişik robotik parçalara ayrılmıştır. Bu tür robotlar uygun modüllerin eklenmesi veya çıkarılmasıyla farklı iş ve işlemleri için yeniden yapılandırılabilmektedir. Öğrenciler farklı parçaları bir araya getirerek farklı yapıda robotlar ortaya çıkarabilmektedir. Kinematics Modular Robotic Construction Kit, MOSS Modular Robot Construction Kit, Modular Robotics tarafından geliştirilen Cubelets bu tür robotlara örnek olarak verilebilir.

Resim 2.30: Modüler eğitsel robot kitleri

Açık Kaynaklı Minyatür Sürü Robotlar: Sürü robotları, yapı olarak birleşik ve tek olmak yerine çok sayıda benzer ve basit fonksiyonellikte robotun ortak çalışmaları ile işleyen robotlardır. Daha çok üniversite düzeyinde robotik araştırmacılar için çok sayıda robotun, merkezi bir kontrole ihtiyaç duymadan birlikte çalışarak, sürü seviyesinde hareket etme davranışlarının incelenmesi, algoritmaların denenmesi ve testlerinin yapılması için tasarlanmış robotlardır. Kilobot, Robomote ve Alice bu tür robotlara örnek olarak verilebilir.

Resim 2.24: Açık kaynaklı minyatür sürü robotlar

2.5. Düşünelim / Araştıralım

Robot programlama dersinde kullanmak üzere eğitsel amaçlı olarak sunulmuş montaj setlerinden, kol setlerinden, tasarım kitlerinden, robot platformlarından, robot kitlerinden birini seçin veya doğrudan bir eğitsel robot seçimi yapınız. Bu seçim için İnternet'te araştırma yapınız. Niçin bu seçimi yaptığınızı, eğitsel robotun hangi özelliklerinin bu seçiminizde etkili olduğunu açıklayınız.

2.6. Değerlendirme Soruları

1. **Hangi robot türünün en önemli özelliği kollara sahip olmasıdır?**
   a) Eğitsel robotlar
   b) Servis robotlar
   c) Endüstriyel robotlar
   d) Savaş robotlar
   e) Hibrit robotlar

2. **Aşağıda verilen robot türlerinden hangisi kullanılan uygulama alanlarına göre yapılan sınıflamaya girmez?**
   a) Endüstriyel robotlar
   b) Nano robotlar
   c) Ev robotları
   d) Tıbbi robotlar
   e) Servis robotları

3. **Aşağıda verilen robot türlerinden hangisi hareket mekaniğine göre yapılan sınıflamaya girmez?**
   a) Sabit robotlar
   b) Tekerlekli robotlar
   c) Mobil küresel robotlar
   d) Uzay robotları
   e) Hibrit robotlar

4. **İnsanların fiilî olarak bulunmaması gereken nükleer, kimyasal felaketler gibi senaryolarda, sağlık alanında, askerî casusluk gibi birçok görevde kullanılması öngörülmüş insan kontrolünde çalışan robot türü aşağıdakilerden hangisidir?**
   a) Telepresence robotlar
   b) Endüstriyel robotlar
   c) Tıbbi robotlar
   d) Hibrit robotlar
   e) Modüler robotlar

5. **Düzensiz, yumuşak, kaygan, karlı ya da çamurlu olabilen zor zeminlerde hangi robot türü diğerlerine göre daha fazla avantaj sağlamaktadır?**
   a) Tekerlekli robotlar
   b) Paletli robotlar
   c) Uçan robotlar
   d) Mobil küresel robotlar
   e) Çok ayaklı robotlar

6. Sabit robotlar, sürekli tekrarlayan görevlerini pozisyonlarını değiştirmeden yapan robotlar için aşağıdaki ifadelerden hangisi yanlıştır?

a) Sabit robotların temeli bulundukları yüzeye sabitlenmiştir.
b) Sabit robotların kolları hareket halindedir.
c) Sabit robotların robotik sistemi değişik robotik parçalara dağıtılmış robot sistemleridir.
d) Silindirik robotlar, küresel robotlar, SCARA robotlar, belden robotlar (robotik kollar) ve paralel sabit robotlar gurubuna girmektedir.
e) Çoğu sabit robotlar sanayi ortamlarında imalat ve montaj sektöründe çalışmaktadır.

7. Hangi tür eğitsel robotlar uygun modüllerin eklenmesi veya çıkarılmasıyla farklı iş ve işlemler için yeniden yapılandırabilmektedir?

a) Açık kaynaklı minyatür sürü robotlar
b) Modüler eğitsel robot kitleri
c) Düşük maliyetli programlanabilir robotik kol setleri
d) Düşük maliyetli minimum özelliklerde mobil robot tasarım kitleri
e) Açık kaynaklı düşük maliyetli mobil robot platformları

8. Aşağıda verilen eğitsel robot türlerinden hangisi robotun serbestçe değiştirilebilmesine, kopyalanabilmesine ve İnternet üzerinden paylaşılabilmesine olanak vermektedir?

a) Düşük maliyetli, tam monte edilmiş mobil robotlar
b) Açık kaynaklı minyatür sürü robotlar
c) Modüler eğitsel robot kitleri
d) Açık kaynaklı düşük maliyetli mobil robot platformları
e) Blok (LEGO Benzeri) tabanlı robot montaj setleri

9. Kavrama, kaldırma, boyama, resim çizme veya yazma gibi değişik işlemler için kullanılabilir eğitsel robot türü aşağıdakilerden hangisidir?

a) Düşük maliyetli programlanabilir robotik kol setleri
b) Düşük maliyetli minimum özelliklerde mobil robot tasarım kitleri
c) Açık kaynaklı düşük maliyetli mobil robot platformları
d) Blok (LEGO Benzeri) tabanlı robot montaj setleri
e) Modüler eğitsel robot kitleri

10. Öğrencilerin farklı parçaları bir araya getirerek farklı yapıda robotlar ortaya çıkarabilmeleri için hangi tür eğitsel robota ihtiyacı bulunmaktadır?

a) Açık kaynaklı minyatür sürü robotlara
b) Düşük maliyetli minimum özelliklerde mobil robot tasarım kitlerine
c) Açık kaynaklı düşük maliyetli mobil robot platformlarına
d) Blok (LEGO Benzeri) tabanlı robot montaj setlerine
e) Modüler eğitsel robot kitlerine

3. EĞİTSEL ROBOTTA MEKANİK BİLEŞENLER

Bu bölümün sonunda,

- ✓ Eğitsel robotta kullanılan yapısal bileşenleri listeleyebilecek,
- ✓ Yapısal bileşenlerin görevlerini açıklayabilecek,
- ✓ Eğitsel robotta kullanılan montaj bileşenlerini tanımlayabilecek,
- ✓ Montaj bileşenlerinin görevlerini örneklendirebilecek,
- ✓ Eğitsel robotta kullanılan hareket/eylem bileşenlerini sıralayabilecek,
- ✓ Hareket/eylem bileşenlerinin görevlerini açıklayabileceksiniz.

3.1. Eğitsel Robotta Mekanik Bileşenler

Eğitsel robotta kullanılan mekanik bileşenler; gövde veya iskeleti oluşturan şasi, mekanik kollar, aktüatörler ve robot mekanik parçaları gibi yapısal bileşenler, vida, somun, rondela gibi bağlantı parçalarından oluşan bağlantı bileşenleri ile tekerlek, palet ve ayak gibi parçalardan oluşan mekanik hareket/eylem bileşenleridir.

3.2. Yapısal Bileşenler (Gövde, İskelet)

Yapısal bileşenler; robotun gövdesini, ana yapıyı oluşturan, diğer bileşenleri üstünde taşıyan gövde, iskelet gibi yapılardır. Plastikten, metalden veya her ikisinden de yapılabilir. Üreticiler tarafından satılan hazır gövdeler, gövde elemanları ve kitleri bulunmaktadır. Standart olarak üretilen pek çok yapısal bileşen bulunmaktadır:

1. Şaseler: Robot gövdesini oluşturmak üzere kullanılan çeşitli türde plastik veya metal delikli plakalar veya biçimlendirilerek gerekli bağlantı delikleri açılmış montaja hazır gövdelerdir. Kare, dikdörtgen veya yuvarlak çeşitleri vardır. Ayrıca kullanıma hazır fakat üzerinde herhangi bir elektronik bileşenin bulunmadığı veya gövdeyle birlikte sadece motorların yer aldığı kit şeklinde olanları da bulunmaktadır.

Resim 3.1: Şaseler

2. Mekanik Kollar, Aktüatörler: Robotun bir nesneyi tutması, kaldırması, sürüklemesi sağ sol, yukarı aşağı (pan/tilt) hareketi yapması için kullanılan mekanik bileşenlerdir. Genellikle iki kıskaçtan oluşan kol şeklindedir. Fakat daha fazla uzvu bulunan el şeklinde kollar da bulunmaktadır. Elektronik bileşenleri olmayan veya sadece adım veya servo motorlara sahip çeşitleri birçok üretici tarafından hazır olarak sunulmaktadır.

Resim 3.2: Mekanik kollar, aktüatörler

3. Robot Mekanik Parçaları: Robota ve robot gövdesine (şase) ekleme yaparak robotik platformu istenilen şekilde oluşturmayı ve geliştirmeyi amaçlayan yapısal bileşenlerdir. Bağlantı elemanları kul-

lanılarak robota mekanik eklemeler yapılabilmektedir. Aşağıdaki fotoğrafta yer alan tekerlekli robota mekanik parçalar eklenerek, ayaklı robot haline getirilmiştir.

Resim 3.3: Mekanik robot parçaları

3.2.1.Yapısal Bileşenlerin Görevleri

Yapısal bileşenlerin temel görevi robot için ana taşıyıcı yapıyı oluşturmaktır. Gerektiği zaman eklemeler yapılmasına olanak sağlayarak robotun geliştirilmesini, yeni eklemeler yapılabilmesini sağlamaktır. Şasi ve mekanik parçalarının üzerinde bulunan çeşitli deliklerin yardımıyla kullanılacak bileşenlerin montajını oldukça kolaylaştırırlar. Robot bileşenlerinin kolay ve hızlıca montajına izin veren bir yapıya sahiptirler.

3.3. Montaj Bileşenleri (Bağlantı Parçaları)

Robotu meydana getiren bileşenleri gövdeye veya birbirine bağlamak için kullanılan vida, somun, rondela, yükselteç, küçük delikli levha gibi elemanlardır. Metal veya plastik çeşitleri bulunmaktadır.

Resim 3.4: Montaj Bileşenleri (Bağlantı Parçaları)

3.3.1. Montaj Bileşenlerinin (Bağlantı Parçaları) Görevleri

Montaj bileşenlerinin görevleri robotu meydana getiren bileşenleri gövdeye veya birbirine bağlayarak bir bütün oluşturmalarını sağlamaktır. Bu sayede hem bileşenler bir arada olurken hem de hareket esnasında robotun zarar görmesi önlenmektedir. Ayrıca bileşenlerin istenilen şekilde bağlanmasını sağladıkları için daha esnek kullanım sunarlar. Örneğin yükselteçler kullanarak bileşenleri gövde üzerinde daha yüksek veya daha yakın bağlamak mümkün olur.

3.4. Mekanik Hareket/Eylem Bileşenleri (Tekerler, Paletler, Ayaklar)

Robotun tercih edilen hareketine uygun olarak kullanılan mekanik bileşenlerdir. Tekerleğe dayalı hareket için çok çeşitli ölçü ve türlerde tekerlekler veya paletler kullanılırken, yürümeye dayalı hareket için servo veya step (adım) motor içeren çeşitli türlerde ayaklar kullanılmaktadır. Tekerlekli, paletli, iki veya daha çok bacaklı mekanik kitler bulunmaktadır.

Resim 3.5: Mekanik Hareket/Eylem Bileşenleri (Tekerler, Paletler, Ayaklar)

3.4.1. Mekanik Hareket/Eylem Bileşenlerinin (Tekerler, Paletler, Ayaklar) Görevleri

Hareket/eylem bileşenlerinin temel görevi robotun hareketini (yürümesini) sağlamak için gerekli mekanik yapıyı sağlamaktır. Robotun sınıf ortamında kullanımında tekerlekli çözümler tercih edilirken, dış mekân kullanımında paletli çözümlerin tercih edilmesi yüzey şartları nedeniyle daha uygun olabilir. İnsansı robotlarda ise hareket için ayaklı çözümler tercih edilebilir. Bu tür robotların hareketlerinin programlanması diğerlerine göre daha zor olabilir.

3.5. Düşünelim / Araştıralım

Robot programlama dersinde kullanmak üzere bir eğitsel robot yapacağınızı düşünerek gerekli olabilecek mekanik bileşenlerin seçimi için İnternet'te araştırma yapınız. Niçin bu bileşenleri seçtiğinizi, bileşenlerin hangi özelliklerinin seçiminizde etkili olduğunu açıklayınız.

3.6. Değerlendirme Soruları

1. **Robotun gövdesini, ana yapıyı oluşturan diğer bileşenleri üstünde taşıyan gövde, iskelet gibi yapıların genel adı aşağıdakilerden hangisidir?**
   a) Elektromekanik bileşenler
   b) Yapısal bileşenler
   c) Montaj bileşenleri
   d) Mekanik hareket/eylem bileşenleri
   e) Elektronik bileşenler

2. **Robotun gövdesini oluşturmak üzere kullanılan çeşitli türde plastik veya metal delikli plakalar veya biçimlendirilerek gerekli bağlantı delikleri açılmış montaja hazır bileşenlere ne ad verilir?**
   a) İskelet
   b) Gövde
   c) Şase
   d) Montaj bileşeni
   e) Aktüatör

3. **Robotun bir nesneyi tutması, kaldırması, sürüklemesi sağ-sol, yukarı-aşağı (pan/tilt) hareketi yapması için kullanılan mekanik bileşenlere ne ad verilir?**
   a) İskelet
   b) Gövde
   c) Şase
   d) Montaj bileşeni
   e) Aktüatör

4. **Robota gövdesine çeşitli mekanik eklemeler yaparak, robotik platformu istenilen şekilde oluşturmayı veya geliştirmeyi amaçlayan bileşenlere ne ad verilir?**
   a) Robot mekanik parçaları
   b) İskelet
   c) Gövde
   d) Şase
   e) Montaj bileşeni

5. **Aşağıdakilerden hangisi yapısal bileşenlerin görevlerinden biri değildir?**
   a) Robot için ana taşıyıcı yapıyı oluşturmaktır.
   b) Gerektiği zaman eklemeler yapılmasına olanak sağlamaktır.
   c) Robotun yeteneklerinin geliştirilmesini, yeni özellikler kazanmasını sağlamaktır.
   d) Kullanılacak bileşenlerin montajını kolaylaştırmaktır.
   e) Robot bileşenlerinin kolay ve hızlıca adaptasyonunu sağlamaktır.

6. Robotu meydana getiren bileşenleri gövdeye veya birbirine bağlamak için kullanılan vida, somun, rondela, yükselteç, küçük delikli levha gibi elemanlara ne ad verilir?

a) Elektromekanik bileşenler
b) Mekanik hareket/eylem bileşenleri
c) Yapısal bileşenler
d) Montaj bileşenleri
e) Elektronik bileşenler

7. Aşağıdakilerden hangisi montaj bileşenlerin görevlerinden biri değildir?

a) Robotu meydana getiren bileşenleri gövdeye veya birbirine bağlamaktır.
b) Robotun mekanik tasarımını kolaylaştırmaktır.
c) Robotu meydana getiren bileşenlerin bir bütün oluşturmalarını sağlamaktır.
d) Hareket esnasında robotun zarar görmesini önlemektir.
e) Bileşenlerin istenilen şekilde bağlanmasını sağlayarak daha esnek kullanım olanağı sunmaktır.

8. Düzgün olmayan yüzeylerde hızlıca hareket etmesi için geliştirilen bir robot için uygun hareket/eylem bileşeni aşağıdakilerden hangisidir?

a) Tekerlek
b) İki ayak
c) İkiden fazla ayak
d) Palet
e) Kanat

9. Eğitsel robotta kullanılan mekanik bileşenler için aşağıdaki ifadelerden hangisi yanlıştır?

a) Mekanik bileşenleri olmayan robot yapmak imkânsızdır.
b) Mekanik bileşenler robotun bir bütün olmasını sağlar.
c) Mekanik bileşenler sağlam robotlar yapmak için gereklidir.
d) Mekanik bileşenler sayesinde modüler robotlar geliştirilebilmektedir.
e) Mekanik bileşenler metal, plastik veya ağaç gibi materyallerden meydana gelebilir.

10. Aşağıdakilerden hangisi hareket/eylem bileşenlerinden biri değildir?

a) Tekerlekler
b) Ayaklar
c) Paletler
d) Kanatlar
e) Kollar

4. EĞİTSEL ROBOTTA ELEKTROMEKANİK BİLEŞENLER

Bu bölümün sonunda,

- ✓ Buton, anahtarlar ve konektör bileşenlerinin görevlerini açıklayabilecek,
- ✓ Güç bileşenlerini listeleyebilecek,
- ✓ Güç bileşenlerinin görevlerini örneklendirebilecek,
- ✓ DC motorların görevlerini tanımlayabilecek,
- ✓ Servo motorların görevlerini sıralayabilecek,
- ✓ Adım (Step) motorların görevlerini açıklayabileceksiniz.

4.1. Eğitsel Robotta Elektromekanik Bileşenler

Eğitsel robotlarda kullanılan mekanik bileşenler; butonlar, anahtarlar ve konektörler gibi bağlantı bileşenleri; pil, akü, batarya gibi güç bileşenleri; hareket sağlamak için kullanılan doğru akım, servo ve adım motor gibi bileşenlerdir.

4.2. Bağlantı Bileşenleri (Butonlar, Anahtarlar, Konektörler ve Klemensler)

1. Butonlar: Üzerine basıldığında, robottaki veya yazılımdaki önceden belirlenmiş mekanik veya elektronik bir sürecin başlamasını, sonlanmasını veya kontrolünü sağlayan basit kontak mekanizmalarıdır. Butonların pek çok çeşitleri bulunmaktadır. Fakat hepsi itme veya üzerine uygulanan kuvvet karşısında tepki veren yay sisteminden oluşurlar. Genellikle butonların tek konumu ve tek kontağı vardır. Stop (durdurma) butonları buna örnek olarak verilebilir.

Resim 4.1: Butonlar

2. Anahtarlar: Elektrikle çalışan bütün sistem ve devrelerde, devreyi açıp kapatmaya yarayan elemanlardır. Basmalı veya çevirmeli tiplerde pek çok çeşidi bulunmaktadır. Örneğin robotun veya kontrol devrelerinin açılması ve kapatılması için kullanılan iki yollu anahtarların iki konumu ve normalde kapalı ve açık olmak üzere iki kontağı vardır. Kalıcı tip anahtarlar ise bir yollu ve iki yollu yapılabilmektedir. Bir yollu çeşitlerinde; anahtara basılınca veya çevrilince kontak açık ise kapanır, kapalı ise açılır. İki yollu çeşitlerinde; anahtara basılınca veya çevrilince, kontaklardan biri açılır, diğeri kapanır.

Resim 4.2: Anahtarlar

3. Konektörler ve Klemensler: Robotun yapısında kullanılan dc, servo veya adım motor gibi elektromekanik ve robotik kontrol kartları, algılayıcılar, güç kaynakları ve motor sürücüleri gibi elektronik bileşenlerin birbirine bağlantısı için kullanılan kablo bağlantı elemanlarıdır. Her türlü bileşenin kablolarla birbirine bağlanması için geliştirilmiş çok fazla tür ve sayıda konektör çeşidi bulunmaktadır. Klemensler ise kabloların birbirine bağlanması için kullanılmaktadır.

Resim 4.3: Konektörler

4.2.1. Bağlantı Bileşenlerinin (Butonlar, Anahtarlar ve Konektörler) Görevleri

Butonların görevi, üzerine basıldığında robottaki veya yazılımdaki önceden belirlenmiş mekanik veya elektronik bir sürecin başlamasını, sonlanmasını veya kontrol edilmesini sağlamaktır. Anahtarların görevi ise elektrikle çalışan bütün sistem ve devrelerde, devreyi açıp kapatmaktır. Konektörlerin görevi ise her türlü donanımın kablolarla birbirine bağlanmasını sağlamaktır.

4.3. Güç Bileşenleri (Pil, Akümülatör, Batarya)

1. Piller: Kimyasal enerjinin depolanabilmesi ve elektriksel forma dönüştürülebilmesi için kullanılan küçük hacimli temel güç kaynaklarıdır. Piller, bir veya daha fazla elektrokimyasal hücre, yakıt hücreleri veya akış hücreleri gibi, farklı elektrokimyasal yapılardan meydana gelir. Genel olarak kullanıldıktan sonra atılan (Non-rechargeable) ve tekrar şarj edilebilen (Rechargeable) piller olarak ikiye ayrılır. Eğitsel robotların enerji kaynağı olarak bu pillerin her iki türü de oldukça yaygın olarak kullanılmaktadır.

Resim 4.4: Piller

2. Akümülatörler: Elektrik enerjisini kimyasal enerji olarak depolayıp, istenildiğinde bunu tekrar elektrik enerjisi olarak geri veren pillerden daha güçlü enerji kaynaklarıdır. Yüksek güç tüketimi olan robotların enerji ihtiyaçlarını karşılamak için kullanılmaktadır. Piller gibi elektrokimyasal yapılardan meydana gelirler.

Resim 4.5: Akümülatörler

3. Bataryalar: Paralel ya da seri bağlanan birden çok pil veya akümülatör gibi kimyasal enerjiyi elektrik enerjisine dönüştüren üreteçlerden oluşturulan güç kaynaklarıdır. Robotlarda, genel olarak tablet ve taşınabilir bilgisayarda yaygın olarak bataryalar kullanılmaktadır.

Resim 4.6: Bataryalar

4.3.1.Güç Bileşenlerinin (Pil, Akümülatör, Batarya) Görevleri

Güç bileşenlerinin görevi robotun çalışması için ihtiyaç duyduğu elektrik enerjisini karşılamaktır. Bu amaçla gerekli voltaj ve akım değerlerinin karşılanması güç bileşenlerinin görevidir. Kesintisiz ve/veya yedek enerji ihtiyaçları için elektrik enerjisinin depolanması ve gerektiğinde geri alınması (kullanılması) yine güç bileşenlerinin görevidir. Hareketsiz ve sabit robotların elektrik ihtiyacı için yukarıda açıklanan güç bileşenleri yerine şehir şebekesinden adaptörle elektrik alınması daha uygun seçenek olacaktır.

4.4. Hareket Bileşenleri (Doğru Akım -DC-, Servo ve Adım Motorlar)

1. Doğru Akım (DC) Motorlar: Doğru akım elektrik enerjisini dairesel mekanik enerjiye dönüştüren makinelerdir. Robotun hareketi için kullanılan temel bileşenlerden biridir. Düşük maliyetli robotlar

üretmek için uygundur. ırçalı, fırçasız, reduktörlü, enkoderli, enkoderli ve reduktörlü çeşitleri bulunmaktadır. Fırçalı motor, motorun hareketli olan bölümüne elektrik akımını aktarılabilmek için fırça ve kolektör kullanılan motor türüdür. Fırçasız motor ise motorun hareketli olan bölümüne elektrik akımı aktarılabilmek için fırça ve kolektör yerine elektronik aksam kullanılan motor türüdür. Reduktörlü motor, şanzıman, dişli kutusu veya dişli sistemi kullanılan motor türüdür. Enkoderli motor ise dönme hareketini ardışık sayısal sinyallere çevirerek dönme hızı ve dönme sayısı hakkında bilgi veren motor türüdür. Standart robot uygulamaları için fırçalı motorlar kullanılırken, yüksek performans isteyen uygumalar için fırçasız motorlar kullanılmaktadır. Motorun devir hızını azaltarak daha yüksek tork (motordan tekerleğe iletilen itme -dönme momenti- kuvveti) elde etmeyi gerektiren uygulamalar için ise redüktörlü bulunan motorlar tercih edilmektedir. Dönme hızı ve dönme sayısını kontrol etmeyi gerektiren uygulamalar için enkoderli motorlar kullanılmaktadır.

Resim 4.7: Doğru akım (DC) motorlar

2. Servo Motorlar: Hareket kontrolü yapılabilen (dönüş yönü, mekaniksel konum, hız veya ivme gibi parametrelerin kontrol edilebildiği) motor çeşitleridir. Bu amaçla gerekli olan sürücü ve kontrol devresi motor içerisinde bulunmaktadır. Bu motorlar, DC motorlardan farklı olmak üzere istenilen pozisyonda sabit kalacak şekilde tasarlanmıştır. Çoğunlukla 0 ile 180 derece arası açılarda çalışırlar. Robotun bileşenlerinin hareketi (kol, ayak, dönen gövde, baş gibi) ve bunların hassas pozisyon kontrolü için kullanılan temel bileşenlerden biri olduğu için robot teknolojisinde en çok kullanılan motor çeşididir. Yürüyen robotlar için yine bu tip motorlar kullanılmaktadır.

Resim 4.8: Servo motorlar

3. Adım (Step) Motorlar: Çok hassas konum kontrol olanağı ve düşük devirde yüksek tork sağlayan motorlardır. Bu motorlarda dönme hareketi istenildiği kadar açıya bölünerek, açısal konumu adımlar halinde değiştirilebilmekte, hassas konum ve pozisyon düzenlemeleri yapılabilmektedir. Adım açısı motorun yapısına bağlı olarak 90°, 45°, 18°, 7.5°, 1.8° veya daha değişik açılarda olabilmektedir. Örneğin robotun kolunun 17° dönmesini istiyorsak adım motor kullanılmalıdır. Adım motor kullanarak tekerlekli robotların daha hassas ve ölçülebilir manevralar yapabilmesi de sağlanmaktadır.

Resim 4.9: Adım (Step) motorlar

4.4.1. Hareket Bileşenlerinin (Doğru Akım -DC-, Servo ve Adım Motorlar) Görevleri

Hareket bileşenlerinin görevi robotun hareketi için gerekli motor gücünü sağlamaktır. Bu amaçla mekanik hareket/eylem bileşenlerinin ihtiyaç duyduğu türde dairesel mekanik enerji, hareket bileşenleri tarafından karşılanır. Bu dairesel enerji robotun hareket biçimine göre değiştirilebilmektedir. İstenildiğinde doğrusal şekle de dönüştürülebilmektedir. Örneğin robotun hareketi için tekerlek kullanılıyorsa tekerleği döndürmek, ayakla yürüyorsa ayakları yürütmek bu bileşenlerin görevidir.

4.5. Düşünelim / Araştıralım

Robot programlama dersinde kullanmak üzere bir eğitsel robot yapacağınızı düşünerek gerekli olabilecek elektromekanik bileşenlerin seçimi için İnternet'te araştırma yapınız. Niçin bu bileşenleri seçtiğinizi, bileşenlerin hangi özelliklerinin seçiminizde etkili olduğunu açıklayınız.

4.6. Değerlendirme Soruları

1. **Her türlü elektrik ve elektronik bileşenin kablolarla birbirine bağlanması için geliştirilmiş kablo bağlantı yapılarına ne ad verilir?**
   a) Buton
   b) Anahtar
   c) Konektörler
   d) Klemens
   e) Duy

2. **Aşağıdakilerden hangisi bağlantı bileşenlerinin görevi değildir?**
   a) Önceden belirlenmiş bir sürecin başlamasını, sonlanmasını veya kontrol edilmesini sağlamak
   b) Bütün elektrik ve elektronik sistem ve devrelerde, devreyi açıp kapatmak
   c) Her türlü donanımın kablolarla birbirine bağlanmasını sağlamak
   d) Her türlü kablonun birbirine bağlanmasını sağlamak
   e) Robotun bileşenlerini birbirine bağlamak

3. **Kimyasal enerjinin depolanabilmesi ve elektriksel forma dönüştürülebilmesi için kullanılan küçük hacimli temel güç kaynakları aşağıdakilerden hangisidir?**
   a) Fotovoltaik panel
   b) Akümülatör
   c) Batarya
   d) Pil
   e) Yakıt hücresi

4. **Elektrik enerjisini kimyasal enerji olarak depolayıp, istenildiğinde bunu tekrar elektrik enerjisi olarak geri veren güçlü enerji kaynaklarına ne ad verilir?**
   a) Fotovoltaik panel
   b) Akümülatör
   c) Batarya
   d) Pil
   e) Yakıt hücresi

5. **Pillerin bir araya gelerek oluşturdukları pil gruplarına ne ad verilmektedir?**
   a) Batarya
   b) Fotovoltaik panel
   c) Akümülatör
   d) Pil
   e) Yakıt hücresi

6. **Yüksek güç tüketimi olan robotların enerji ihtiyaçlarını karşılamak için aşağıdaki seçeneklerden hangisinin kullanılması daha uygundur?**
   a) Batarya
   b) Akümülatör
   c) Fotovoltaik panel
   d) Pil
   e) Yakıt hücresi

7. **Motorun devir hızını azaltarak daha yüksek tork elde etmeyi gerektiren uygulamalar için hangi motor türü tercih edilmelidir?**
   a) Fırçalı motor
   b) Fırçasız motor
   c) Servo motor
   d) Enkoderli motor
   e) Redüktörlü motor

8. Aşağıdakilerden hangisi hareket kontrolü yapılabilen (dönüş yönü, mekaniksel konum, hız veya ivme gibi parametrelerin kontrol edilebildiği) motor çeşididir?

a) Fırçasız motor
b) Step motor
c) Enkoderli motor
d) Servo motor
e) Redüktörlü motor

9. Dönme hareketini istenildiği kadar açıya bölerek, açısal konumu adımlar hâlinde değiştirebilen, hassas konum ve pozisyon düzenlemeleri yapabilen motor çeşidi aşağıdakilerden hangisidir?

a) Fırçasız motor
b) Step motor
c) Enkoderli motor
d) Servo motor
e) Redüktörlü motor

10. Dönme hızı ve dönme sayısını kontrol etmeyi gerektiren uygulamalar için hangi tür motorlar kullanılmalıdır?

a) Redüktörlü motor
b) Enkoderli motor
c) Fırçasız motor
d) Fırçalı motor
e) Adım motor

5. EĞİTSEL ROBOTTA ELEKTRONİK BİLEŞENLER

Bu bölümün sonunda,

- ✓ Motor sürücü katlarının görevlerini listeleyebilecek,
- ✓ USB-UART çeviricilerin görevlerini tanımlayabilecek,
- ✓ Kablosuz iletişim bileşenlerinin görevlerini özetleyebilecek,
- ✓ Algılayıcı çeşitlerini listeleyebilecek,
- ✓ Algılayıcı çeşitlerinin görevlerini açıklayabilecek,
- ✓ Robotik programlamada kullanılan işlemcileri tanımlayabilecek,
- ✓ Robotik programlamada kullanılan işlemcilerinin görevlerini yorumlayabilecek,
- ✓ Robot kontrol kartlarını listeleyebilecek,
- ✓ Robot kontrol kartlarının görevlerini açıklayabileceksiniz.

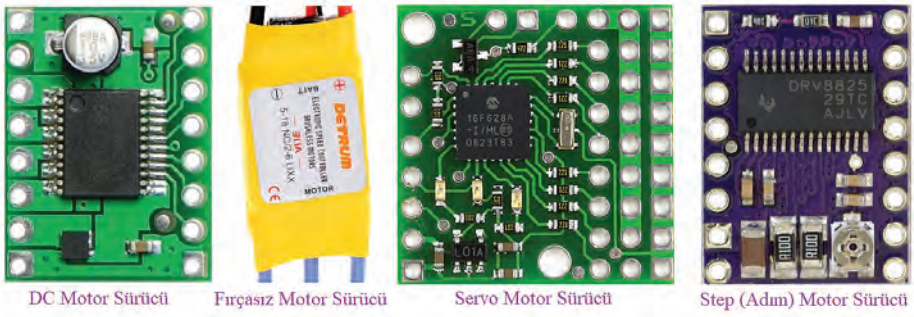
5.1 Eğitsel Robotta Elektronik Bileşenler

Bu bölümde eğitsel robotta kullanılan elektronik bileşenler ve bu bileşenlerin görevleri açıklanmıştır. Bu kapsamda motor sürücü kartları, usb-uart çeviriciler, kablosuz iletişim bileşenleri, robotik uygulamalarda kullanılan algılayıcılar (sensörler), algılayıcıların mikrodenetleyici kartlarla haberleşmesi/bağlanması, robotik programlamada kullanılan işlemciler, mikrodenetleyici kartlar (geliştirme kartları), mikrodenetleyici kartlar için kalkanlar (shields) konuları ele alınmıştır.

5.2. Motor Sürücü Kartları ve Görevleri

Robotlarda kullanılan motorların kontrol edilebilmesi (çalışma, durma, ileri geri hareket etme, hızlanma, yavaşlama vb.) için kullanılan bileşenlerdir. Ayrı bir kart olarak alınabileceği gibi, robot kontrol kartlarının ya da mikro kontrolör kartlarının dâhilî bir bileşeni olarak da bulunabilmektedir. Tek bir motorun kontrolünden, çok sayıda ve türde motorun kontrolüne kadar çok çeşitli yapıda motor kontrol kartları bulunmaktadır. Birden fazla sayıda ve türde motorun hız ve yönlerini birbirinden bağımsız olarak kontrol edebilmektedir.

Tercih edilecek motorun türüne göre farklı motor sürücü kartlarının kullanılması gerekmektedir. Fırçalı doğru akım motorları için DC Motor Sürücüler, fırçasız doğru akım motorları için Fırçasız Motor Sürücüler (Bunlara Electronic Speed Controller, ESC adı verilmektedir.) kullanılmaktadır. Aynı şekilde Servo motorlar için Servo Motor Sürücüler ve Adım (Step) motorlar için Adım Motor Sürücülerin kullanılması gerekmektedir. Fırçasız doğru akım motorları hariç diğer türlerin kontrolü için ortak kullanımlı (her üç tür motoru bir arada kontrol edebilen) kartlar bulunmaktadır.



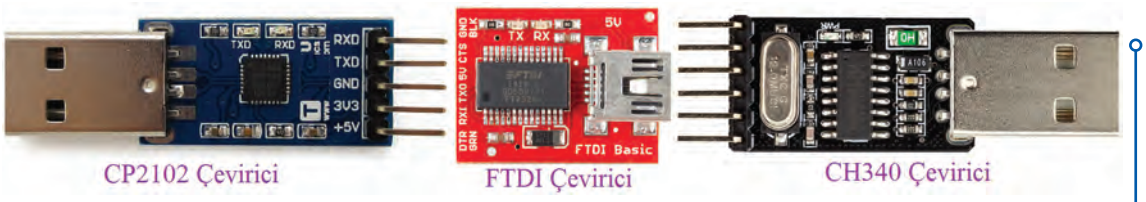
Resim 5.1: Motor sürücü kartları

5.3. USB-UART Çeviriciler ve Görevleri

Bilgisayar ve ona bağlanabilen her türlü çevresel aygıt seri haberleşme tekniğini (seri iletişim) kullanmaktadır. Bu amaçla bilgisayar ve çevresel aygıtların üzerinde seri iletişim bağlantı noktaları bulunmaktadır. Günümüzde kullanılan seri iletişim bağlantı noktası temelde USB'dir (Universal Serial Bus-Evrensel Seri Veriyolu).

Robotik programlamada kullanılan işlemcilerin, bunların üzerinde bulunduğu mikrodenetleyici kartların ve robotik kontrol kartların bilgisayara bağlanıp programlanabilmesi için de USB bağlantı noktası kullanılmaktadır. USB'nin görevi, bilgisayar ile kontrol kartı (örneğin Arduino) üzerinde yer alan mikrodenetleyici arasında iletişimi sağlamaktır. Bu sayede kartların programlanması ve kontrolü gerçekleşebilmektedir. Fakat bazı mikrodenetleyici kartlarda ve robotik kontrol kartlarında (aynı şekil-

de mikroişlemcilerde) USB bağlantı seçeneği bulunmamaktadır. Yalnızca UART (Universal Asynchronous Receiver/Transmitter - Evrensel Asenkron Alıcı / Verici) bulunmaktadır. Bu durumda bu tür birimlerle iletişim kurulabilmesi için USB-UART çeviricilere ihtiyaç duyulmaktadır. Ancak iletişimin sağlanabilmesi için bilgisayarın kullanılan çevirici ile nasıl haberleşeceğini biliyor olması, başka bir deyişle çeviricinin aygıt sürücülerinin bilgisayarda yüklü olması gerekmektedir. Farklı türlerde USB-UART çeviriciler bulunmaktadır.



Resim 5.2: USB-UART çeviriciler

5.4. Kablosuz İletişim Bileşenleri ve Görevleri

Robotun kontrol edileceği, programlanacağı aygıtlara (Bilgisayar, tablet veya akıllı telefon olabilir.) kablosuz olarak bağlanabilmesi için kullanılan haberleşme bileşenlerdir. Genellikle Wi-Fi, Bluetooth, XBee ve ZigBee parçaları bu amaçla tercih edilmektedir. Bu parçalar kullandıkları protokole, haberleşme frekansına, anten tiplerine ve güçlerine göre sınıflandırılmaktadır. Mikrodenetleyici kartların ve robotik kontrol kartların bilgisayara bağlanıp kontrol edilebilmesi için bu teknolojilerin hepsi de kullanılabilmektedir.

Wi-Fi (Wireless Fidelity-Kablosuz Bağlantı Alanı) kişisel bilgisayar, tablet, video oyunu konsolları, dijital ses ve video oynatıcıları ve akıllı telefonlar gibi cihazların kablosuz olarak İnternet'e ve birbirlerine bağlanması için kullanılmaktadır. Wifi teknolojisi ile 4900 Mbps'ye kadar ses ve veri iletimi yapabilmektedir. Wi-Fi destekli cihazların ve parçaların etkin olduğu mesafe, kapalı alanlarda en fazla 300 metre civarındadır.

Bluetooth kişisel bilgisayar, çevre birimleri ve diğer cihazların birbirleri ile kablo bağlantısı olmadan haberleşmelerine olanak sağlayan kısa mesafe radyo frekans (RF) teknolojisidir. Bluetooth teknolojisi ile 24 Mbps'ye kadar ses ve veri iletimi yapabilmektedir. Bluetooth destekli cihazların etkin olduğu mesafe, yaklaşık 10 ile 100 metre arasındadır. Robotik uygulamalarda yaygın olarak kullanılmaktadır.

Resim 5.3: Kablosuz iletişim bileşenleri

XBee ve ZigBee düşük maliyetli, düşük güçlü kablosuz kısa mesafe radyo frekans (RF) teknolojisidir. Düşük maliyetli teknoloji olduğu için, kablosuz aygıtların kontrol ve izleme uygulamalarında kulla-

nılmaktadır. Düşük güç kullanımı daha küçük pil ile daha uzun ömür sunmaktadır. XBee ve ZigBee destekli 2. Nesil cihazların etkin olduğu mesafe düşük veri iletişim hızlarında (10-20 kbit/sn) ve yüksek kazançlı antenler kullanılarak 45 km'ye kadar ulaşabilmektedir. Genellikle veri iletim hızı çeşitlerine göre 20 ile 1000 kilobit/saniye arasında değişmektedir. Oldukça küçük yapıda üretilebilmektedir.

5.5. Robotik Uygulamalarda Kullanılan Algılayıcılar (Sensörler)

Robot teknolojisinin veya genel anlamda otomasyon sistemlerinin en önemli kısımlarından birisi algılamadır. Algılamayı sağlayan aygıtlara sensör ya da algılayıcı adı verilmektedir. Algılayıcıları bu sistemlerin duyu organları olarak değerlendirebiliriz. Çünkü insanların çevrelerinde olup bitenleri duyu organlarıyla algılamasına benzer biçimde, robotlar ve otomasyon sistemleri de çevresindeki sıcaklık, basınç, hız, yön, eğim ve benzeri değişkenleri algılayıcıları vasıtasıyla algılarlar. Algılama algılananları ölçme ve ölçümleri kontrol aygıtına (mikroişlemci) iletme şeklinde gerçekleşir. Mikroişlemci algılananları yorumlamak ve ona göre karar döngülerini yürütmek zorundadır. Algılanması gereken farklı değişkenler farklı tiplerde algılayıcılar gerektirir. Neyin ya da nelerin algılanacağı kullanılan algılayıcının seçimine bağlıdır. Algılayıcı seçimi robotun görevine uygun olarak yapılır. Örneğin robotun herhangi bir engele çarpmadan dolaşabilmesini istiyorsak bir mesafe ölçüm algılayıcısının kullanılması gerekmektedir. Algılayıcılar ile algılanan çok farklı türde değişken bulunmaktadır. Bu değişkenler şu şekilde özetlenebilir:

- Mekanik Değişkenler: Uzunluk, alan, miktar, kütlesel akış, kuvvet, tork (moment), basınç, hız, ivme, pozisyon, ses dalga boyu ve yoğunluğu gibi değişkenlerin ölçülmesidir.
- Termal Değişkenler: Sıcaklık, ısı akışı gibi değişkenlerin ölçülmesidir.
- Elektriksel Değişkenler: Voltaj, akım, direnç, endüktans, kapasitans, dielektrik katsayısı, polarizasyon, elektrik alanı ve frekans gibi değişkenlerin ölçülmesidir.
- Manyetik Değişkenler: Alan yoğunluğu, akı yoğunluğu, manyetik moment, geçirgenlik gibi değişkenlerin ölçülmesidir.
- Işıma Değişkenleri: Yoğunluk, dalga boyu, polarizasyon, faz, yansıtma, gönderme gibi değişkenlerin ölçülmesidir.
- Kimyasal Değişkenler: Yoğunlaşma, içerik, oksidasyon/redaksiyon, reaksiyon hızı, pH miktarı gibi değişkenlerin ölçülmesidir.

5.5.1. Robotik Algılayıcı Türleri

Günümüzde çok çeşitli algılayıcı bulunmaktadır. Bunları birbirinden farklı birçok sınıfa ayırmak mümkündür. Genelde ölçülen büyüklüğe göre, çıkış büyüklüğüne göre, besleme ihtiyacına göre yapılan sınıflandırmalar kullanılmaktadır. Örneğin besleme ihtiyaçlarına göre algılayıcılar, pasif ve aktif algılayıcılar; çalıştıkları sinyallere göre ise dijital ve analog algılayıcılar olarak iki grupta sınıflandırılır.

Özellikle mobil robotlar için kullanılan algılayıcılar işlevlerine göre sınıflandırılmaktadır. Bazı algılayıcılar, bir robotun elektroniğinin iç sıcaklığı veya motorların dönme hızı gibi basit değerleri ölçmek için kullanılır. Bazıları ise robotun çevresi hakkında bilgi edinmek veya robotun küresel konumunu doğrudan ölçmek gibi daha karmaşık değerleri ölçmek için kullanılabilir. Robotik algılayıcılar bu iki önemli fonksiyonel eksende propriyoseptif ve exteroseptif algılayıcılar olarak ayrılmaktadır.

Propriyoseptif Algılayıcılar: Robotik sistemin içindeki motor hızı, tekerlek yükü, robot kolu eklem açısı ve akü gerilimi gibi değerleri ölçmek için kullanılan algılayıcılardır.

Eksteroseptif Algılayıcılar: Robotun bulunduğu ortamdan bilgi alan algılayıcılardır. Örneğin mesafe ölçümleri, ışık yoğunluğu ve ses dalga genliği ölçümü gibi işlemleri yaparlar. Bu nedenle, eksteroseptif algılayıcı ölçümleri, anlamlı çevre özelliklerini çıkarmak için robot tarafından yorumlanırlar.

Pasif Algılayıcılar: Dışarıdan harici hiçbir güç kaynağına ihtiyaç duymadan çevrelerinden aldıkları fiziksel ya da kimyasal sinyalleri ölçen algılayıcılardır. Başka bir deyişle, algılayıcıya giren çevre ortam enerjisini ölçerler. Pasif algılayıcı çeşitlerine en basit örnek ise buton ve anahtardır. Bunlardan farklı olarak potansiyometre, limit anahtarları, ısı, ışık, basınç algılayıcıları, dokunma algılayıcılar, mikrofonlar, CCD veya CMOS kameralar örnek olarak verilebilir. Bu algılayıcıların çalışması için harici hiçbir enerjiye ihtiyaç yoktur. Bu algılayıcılar sadece giriş değişkenlerini ölçerek tepki verirler.

Aktif Algılayıcılar: Sinyallerini kendileri üretip çevrelerine yayar ve bu sinyallerin çevreleriyle olan etkileşimlerini ölçen algılayıcılardır. Aktif algılayıcılar sinyallerini kendileri yaydıklarından daha fazla enerjiye ihtiyaç duyarlar. Bu nedenle fiziksel ya da kimyasal değerleri ölçmek için dışarıdan haricî bir güç kaynağı kullanılmaktadır. Bu algılayıcıların en önemli özelliklerinden biri, zayıf sinyalleri oldukça hassas biçimde ölçmek için kullanılabilmeleridir. Kızılötesi algılayıcılar, mesafe algılayıcılar, enkoderler, lazer mesafe bulucular ve ultrasonik uzaklık algılayıcıları aktif algılayıcılara örnek olarak verebiliriz. Aktif algılayıcılar ürettiği sinyal türüne göre analog veya dijital sinyal çıkışı vermektedir.

Dijital Sinyal Veren Algılayıcılar: Dijital algılayıcılar ayrık sinyaller üretir. Bu, değerlerin sınırlı sayıda ve kesikli olduğu anlamına gelir. Dijital algılayıcılardan alınan ham bilgiler belli adımlarla yükselen değerlere sahiptirler. Örneğin bir dijital pusula 360 farklı değer üretirken, dijital algılayıcı olan anahtarlar açık ya da kapalı olarak iki değer üretirler.

Analog Sinyal Veren Algılayıcılar: Analog algılayıcılar, devre 0 V - 5 V arasında ya da 4 mA - 20 mA arasındaki değerleri algılayacak şekilde çalışırlar ve bu durumda bu iki değer arasındaki tüm değerleri okuyabilirler. Analog sinyal belli iki değer arasında herhangi bir değerdir. Sürekli sinyal ürettikleri için sinyaller arası aralık yoktur. Analog algılayıcılar kullanıldığında bunları mikroişlemcilere yönlendirmeden önce analog/dijital (A/D) çeviriciler kullanılarak analog sinyallerin dijital sinyallere çevrilmeleri gerekir. Çünkü mikroişlemciler dijital sinyallerle çalışırlar.

5.5.2. Yaygın Kullanılan Robotik Algılayıcılar ve Görevleri

Burada genel olarak robotik uygulamalarda en fazla kullanılan algılayıcı çeşitleri ve görevleri kısaca açıklanmıştır. Sınıflandırma, algılayıcıların aktif veya pasif olmasına göre yapılmış analog veya dijital sinyal üretmesi gibi özelliklerine değinilmemiştir. Çünkü aynı amaç için kullanılan fakat farklı özellikler taşıyan çok fazla sayıda ve türde algılayıcı bulunduğu gibi hem analog hem de dijital sinyali birden verenleri de bulunmaktadır.

5.5.3. Aktif Algılayıcılar

Çizgi Takip Algılayıcıları (Line Sensors): Robot uygulamalarında, robotun kalınca çizgilerle çizilen belirli bir alan içerisinde kalması veya çizilen çizgileri izlemesi için kullanılan algılayıcılardır.

Resim 5.4: Çizgi takip algılayıcı

Engel Kaçınma Algılayıcıları (Obstacle Avoidance Sensors): Robotun bir engele çarpmadan önce onu algılayıp kaçınması için kullanılan algılayıcılardır.

Resim 5.5: Engel kaçınma algılayıcı

Enkoder Algılayıcılar (Encoder Sensors): Robotik uygulamalarda motorların dönüş yönünü, hızlarını ve tur sayılarını belirlemek için kullanılan, motor kontrol sistemleri için geri bildirim sağlayan algılayıcılardır. Optik ve manyetik yöntemle çalışan çeşitleri bulunmaktadır. Doğrusal ve döner olmak üzere ikiye ayrılırlar.

Resim 5.6: Enkoder algılayıcı

Hareket Algılayıcılar (PIR Motion Sensors): İnsan ve hayvanların robot tarafından algılanması için kullanılan algılayıcılardır. PIR (Passive Infrared Sensor) algılayıcılar insanlar veya sıcakkanlı hayvanlar tarafından üretilen kızılötesi ışığı algılarlar. Algılayıcının ön yüzünde ısı ışınlarını IR algılayıcı üzerinde çeşitli noktalara odaklayan çok sayıda fresnel mercekler bulunmaktadır.

Resim 5.7: Hareket algılayıcı

Hareket Kontrol Algılayıcılar (Gesture Sensors): Robotun elle yapılan hareketlerle kontrol edilebilmesi için kullanılan algılayıcılardır. Bu algılayıcılar, kullanıcıdan yansıyan kızılötesi ışınları tespit ederek basit el hareketlerini robotun tanımasını sağlar.

Resim 5.8: Hareket kontrol algılayıcı

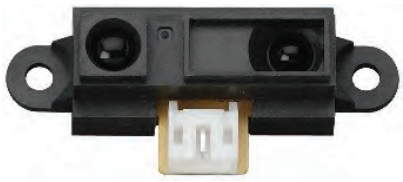
Işık Kesici Algılayıcılar (Photo Interrupter Sensors): Algılayıcının kolları arasında bulunan kızılötesi ışık demeti arasından bir nesne geçtiğinde ışının kırılması sonucu robotun o nesneyi algılamasını sağlayan algılayıcılardır.

Resim 5.9: Işık kesici algılayıcı

Kızılötesi Termometre Algılayıcılar (Infrared Thermometer Sensors): Robotun temassız olarak (uzaktan) ortam sıcaklığını algılaması, vücut ısısı ölçümü veya hareket algılaması gibi uygulamaları için kullanılan algılayıcılardır.

Resim 5.10: Kızılötesi termometre algılayıcı

Kızılötesi Yakınlık Algılayıcılar (Infrared Proximity Sensors): Robotun belirli bir nesneye veya duvara olan mesafesini ölçmek için kullanılan algılayıcılardır. Genellikle 3 ile 150 cm aralığındaki uzunluğu ölçebilmektedir.



Resim 5.11: Kızılötesi yakınlık algılayıcı

Lazer Tarama Algılayıcılar (Laser Scanner Senors): Robotun engellerden kaçınması, bulunduğu ortamı haritalaması, lokalizasyon, rota planlaması gibi işlemleri yapabilmesi için kullanılan algılayıcılardır. Robot 360° tarama yaparak bulunduğu ortamın 2 veya 3 boyutlu gerçek görüntülerini oluşturmaktadır.

Resim 5.12: Lazer tarama algılayıcı

Mikrodalga Hareket Dedektörü Algılayıcılar (Microwave Motion Detector Sensors): Robotun mikrodalgalar kullanılarak cansız hareketli nesneleri algılaması, hız ölçmesi için kullanılan algılayıcılardır. Sistemin çalışma mantığı Doppler Efektine dayanır.

Resim 5.13: Mikrodalga hareket detektörü algılayıcı

Optik Algılayıcılar (Optical Detectors): Bu algılayıcılar robotun yansıyan kızılötesi sinyalleri algılaması için kullanılır. Siyah beyaz renk geçişlerini algılama veya yakındaki cisimleri (0,5-1 cm) tespit etmek için de kullanılmaktadır.

Resim 5.14: Optik algılayıcı

Sonar Mesafe Bulucular (Sonar Range Finders): Robotun belirli bir nesneye veya duvara olan mesafesini ölçmek için kullanıldıkları gibi algılama bölgesindeki nesneleri tespit etme ve bir nesne (bir kişi gibi) algılama bölgesine girdiğinde rapor vermek için de kullanılan algılayıcılardır. 0 ile 765 cm aralığındaki uzunluğa kadar 2,5 mm hassasiyete ölçme yapabilen, bu mesafeler içerisindeki engelleri algılayabilen çeşitli modelleri bulunmaktadır.

Resim 5.15: Sonar mesafe algılayıcı

Ultrasonik Uzaklık Algılayıcılar (Ultrasonic Distance Sensors): Robotun belirli bir nesneye veya duvara olan mesafesini ölçmek için kullanılan algılayıcılardır. Genellikle 2 ile 400 cm aralığındaki uzunluğu 3 mm hassasiyete ölçebilmekte, bu mesafeler içerisindeki engelleri algılayabilmektedir.

Resim 5.16: Ultrasonik uzaklık algılayıcı

Yansıtıcı Optik Algılayıcılar (Reflective Optical Sensors): Robotun siyah beyaz renk değişimini algılaması için kullanılan algılayıcılardır. Genelde çizgi izleyen robotlar için kullanılmaktadır.

Resim 5.17: Yansıtıcı optik algılayıcı

Tampon Algılayıcılar (Bumper Sensors): Robotun herhangi bir nesneye veya yapıya çarpmadan önce onu algılaması için kullanılan algılayıcılardır. Algılama çarpmadan önce gerçekleşmektedir.

Resim 5.18: Tampon algılayıcı

5.5.4. Pasif Algılayıcılar

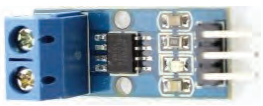
Açısal Algılayıcılar (Angular Sensors): Robotun bir bağlantı mekanizmasının açısal değerini veya robota ait bir eklemin açı değerini tespit için tasarlanmış algılayıcılardır.

Resim 5.19: Açısal algılayıcı

Ağırlık Algılayıcılar (Load Sensors): Robotun ağırlıkları algılayabilmesi, ölçebilmesi için kullanılan algılayıcılardır. Çok çeşitli tür ve ağırlık kapasitelerinde üretilmektedir.

Resim 5.20: Ağırlık algılayıcı

Akım Algılayıcılar (Current Sensors): Robotun kendi genel güç tüketimlerini ölçmek ve değerlendirmek için kullandığı algılayıcılardır.



Resim 5.21: Akım algılayıcı

Alev Algılayıcılar (Flame Sensors): Robotun alevi, ateşi uzaktan algılaması için kullanılan algılayıcılardır.

Resim 5.22: Alev algılayıcı

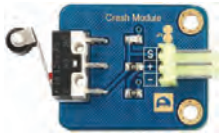
Basınç / Yükseklik Algılayıcılar (Barometric Pressure /Altitude Sensors): Robotun barometrik basınç ölçmesi için kullanılan algılayıcılardır. Basınç yükseklik ile değiştiği için aynı zamanda bir altimetre (yükseklikölçer) olarak da kullanılabilmektedir.

Resim 5.23: Basınç algılayıcı

Buhar Algılayıcılar (Steam Sensors): Robotun ortamdaki nem ve buhar varlığını algılaması için kullanılan algılayıcılardır. Nem ve buhar miktarının ölçümü için kullanılabilmektedir.

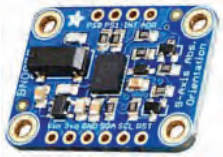
Resim 5.24: Buhar algılayıcı

Çarpma Algılayıcılar (Crash Sensors): Robotun herhangi bir nesneye veya yapıya çarptığını algılaması için kullanılan algılayıcılardır. Algılama çarptıktan sonra gerçekleşmektedir.



Resim 5.25: Çarpma algılayıcı

Çoklu Algılayıcılar (IMU-Inertial Measurement Unit- Atalet Ölçüm Birimi): Robotun gerçek dünyadaki konumu, hızı, yüzeyle olan açısı ve yüksekliği gibi bilgileri algılamasını sağlayan entegre algılayıcılardır. 3 eksen jiroskop, 3 eksen ivmeölçer, 3 eksen pusula ve dijital barometre algılayıcılarının birleştirildiği bir mini kart şeklindedir.



Resim 5.26: Çoklu algılayıcı

Dokunma Algılayıcılar (Touch Sensors): Robotun kendisine dokunulduğunu anlamasını sağlayan algılayıcılar-

Resim 5.27: Dokunma algılayıcı

dır. İnsan derisine duyarlıdırlar. Açma/kapama düğmesi kullanmadan bir açma/kapama işlemi yapmak veya robotun insan eliyle dokunmaya duyarlı bir eylem veya hareketi yapması için kullanılmaktadır.

Eğim Algılayıcılar (Tilt Sensors): Robotun bulunduğu yerdeki eğimi, eğimin yönünü veya sarsıntıyı tespit edebilmesi için kullanılan algılayıcılardır.

Resim 5.28: Eğim algılayıcı

Esnek Kuvvet, Güç, Basınç Algılayıcılar (Flexiforce Pressure Sensors): Robotun kuvvet, güç ya da üzerine uygulanan basıncı algılayabilmesi için kullanılan algılayıcılardır. Robot üzerindeki belirli bir alana (kare veya dairesel olabilir) uygulanan, güç ya da basıncın algılanması söz konusudur.

Resim 5.29: Esnek algılayıcı

Gaz Algılayıcılar (Gas Sensors): Havadaki Karbon Monoksit (CO), Azot dioksit (NO2), Doğalgaz (CNG), Hidrojen (H2), sıvılaştırılmış petrol gazı (LPG), Bütan, Propan, Metan (CH4), Alkol, Amonyak (NH3) ve duman gibi gazlarla, toksik gazları algılamak için kullanılan algılayıcılardır. Hava kalitesini ölçmek için kullanılan çeşitleri de bulunmaktadır.

Resim 5.30: Gaz algılayıcı

Görüntü Algılayıcılar (Image Sensors): Robotun nesneleri tanıması, öğrenmesi ve istenildiğinde bulması için kullanılan görme sistemleridir. Öğretilen nesneleri gördüğünde algılamaktadır. Gerçek zamanlı görüntü işleme görevleri için kullanılmaktadır.

Resim 5.31: Görüntü algılayıcı

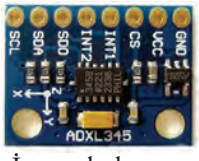
GPS Algılayıcılar (GPS Sensors): Robotun bulunduğu noktayı enlem ve boylam olarak tespit edebilmesi, kendine verilen rota doğrultusunda hareket edebilmesi, gerçek hızı ve yüksekliğini belirleyebilmesi için kullanılan küresel konumlandırma (Global Positioning System -GPS) algılayıcılarıdır.

Resim 5.32: GPS algılayıcı

Işık Algılayıcılar (Light Sensors): Robotun ortamdaki ışık miktarını, yoğunluğunu ölçmesi, buna göre herhangi bir eylem veya hareket yapması için kullanılan algılayıcılardır. Kızılötesi ve normal ışık için kullanılan çeşitleri bulunmaktadır.

Resim 5.33: Işık algılayıcı

İvme Algılayıcılar (Accelerometer Sensors): İvme ölçmek için kullanılan algılayıcılardır. Robotun eklem hareketlerini, eğilme derecesini ve titreşimleri algılayabilmesini



Resim 5.34: İvme algılayıcı

sağlayan algılayıcılardır. X, Y ve Z eksenlerinde yapılan temel hareketleri algılamak için bu algılayıcılar kullanılmaktadır. Tek, iki veya üç eksende oluşan ivmeyi ölçebilen çeşitleri bulunmaktadır.

Jiroskop Algılayıcılar (Gyroscope Sensors): Robotun yön ölçümü ve ayarlanmasında konumsal ve hareketsel yönünü hesaplamayı sağlayan algılayıcılardır. Bu amaçla X, Y ve Z eksenleri arasındaki açısal oranların ölçümü yapılmaktadır. Jiroskop dış etkenlerden, yer çekiminden ve merkezkaç kuvvetinden etkilenmeyen bir referans etkeni sağlamaktadır.

Resim 5.35: Jiroskop algılayıcı

Konuşma, Ses Tanıma Algılayıcıları (Speech, Voice Recognition Sensors): Robotun sesle verilen emirleri anlayıp uygulayabilmesi için sesi ve konuşmayı tanımasını sağlayan algılayıcılardır. Bu sayede robotla konuşarak iletişim kurmak ve istenileni yaptırmak mümkün hâle gelmektedir.

Resim 5.36: Konuşma, ses tanıma algılayıcı

Manyetik Alan Algılayıcılar (Hall Effect Sensors): Robotun manyetik malzeme ve ortamları algılamasını sağlayan algılayıcılardır. Robotun manyetik alana duyarlı bir eylem veya hareketi yapması için kullanılmaktadır.

Resim 5.37: Manyetik alan algılayıcı

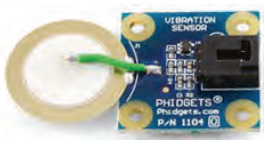
Nem Algılayıcılar (Humidity Sensors): Robotun ortamdaki nem miktarını ölçmesi için kullanılan algılayıcılardır.

Resim 5.38: Nem algılayıcı

Parlaklık Algılayıcılar (Luminosity Sensors): Robotun ışığın parlaklık düzeyini algılaması ve ölçmesi için kullanılan algılayıcılardır.

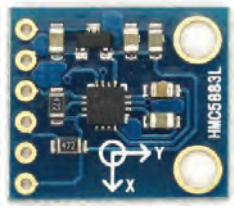
Resim 5.39: Parlaklık algılayıcı

Piezo Titreşim Algılayıcılar (Piezo Vibration Sensors): Robotun esneme, dokunma, titreşim ve şok ölçümleri yapabilmesi, çarpışmaları algılayabilmesi veya esnek anahtar uygulamaları için kullanılan algılayıcılardır.



Resim 5.40: Piezo titreşim algılayıcı

Pusula, Manyetometre Algılayıcılar (Compass, Magnetometer Sensors): Dijital yön algılayıcılardır. Dünyanın manyetik alanına ilişkin ölçmeye dayalı yönlendirme ile robotun her zaman otomatik veya programlı olarak istenilen gerçek fiziksel yönde hareket etmesi için kullanılır. Tek, iki veya üç eksen ölçen çeşitleri bulunmaktadır.



Resim 5.41: Pusula algılayıcı

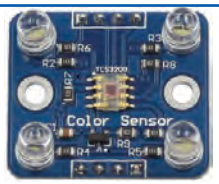
Ses Algılayıcılar (Sound Sensors): Robotun sesi algılaması, sese duyarlı bir eylem veya hareketi yapması için kullanılan algılayıcılardır. Bu algılayıcılar sesi tanımlayamaz, anlayamaz, sadece sesi fark eder.

Resim 5.42: Ses algılayıcı

Sıcaklık Algılayıcılar (Temperature Sensors): Robotun ortam ve çalışma sıcaklığını ölçmesi için kullanılan algılayıcılardır.

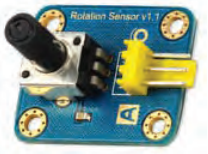
Resim 5.43: Sıcaklık algılayıcı

Renk Algılayıcılar (Color Sensors): Robotun renkleri algılaması, tanımlaması ve renk ölçümlerini doğru yapabilmesi için kullanılan algılayıcılardır.



Resim 5.44: Renk algılayıcı

Rotasyon Algılayıcılar (Rotation Sensors): Robotun herhangi bir bileşeninin (kol, ayak, baş, gövde vb.) kaç derece hareket ettiğini mekanik bağlantıyla algılaması için kullanılan algılayıcılardır.



Resim 5.45: Rotasyon algılayıcı

Titreşim Algılayıcılar (Vibration Sensors): Robotun meydana gelen titreşimleri ve hızlanmayı algılaması için kullanılan algılayıcılardır. Titreşim miktarının veya hızlanmanın ölçümü için kullanılmaz.

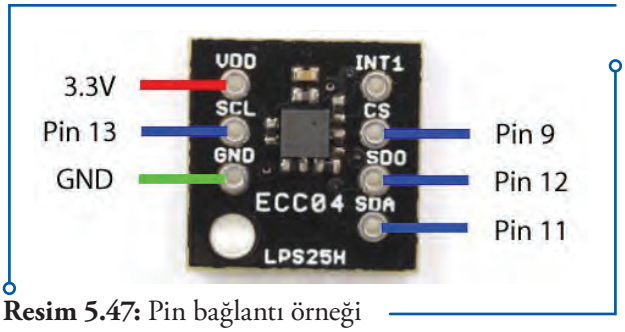
Resim 5.46: Titreşim algılayıcı

5.5.5. Algılayıcıların Mikrodenetleyici Kartlarla Haberleşmesi/Kartlara Bağlanması

Algılayıcıların mikrodenetleyici kartlarla haberleşmesi ile ilgili birçok standart protokol bulunmaktadırdır. SPI (Seri Çevresel Arayüz - Serial Peripheral Interface) ve I²C (Inter-Integrated Circuit) protokolleri en fazla kullanılan haberleşme protokolleridir. Arduino kartları, diğer Arduino kartlarıyla veya algılayıcılarla (sensörlerle) haberleşmek için bu haberleşme protokollerini kullanmaktadır. Bu protokollerin kullandıkları uç sayıları, ulaşabilecekleri maksimum hızları ve kullanım şekilleri birbirinden farklılık göstermektedir. Pasif algılayıcılar genellikle (+ volt), [- volt (toprak hattı)] ve (S -Sinyal) olmak üzere 3 pin üzerinden Arduino kartlarla haberleşirler. Algılayıcının analog veya dijital çıkış vermesine bağlı olarak Arduino kartının analog veya dijital çıkışlarına bağlanmaları gerekmektedir. Bağlantı algılayıcının voltaj girişlerinin Arduinonun Vcc (+Volt) ve Gnd (-Volt) pinlerine, sinyal çıkışlarının analogsa Arduinonun analog pinlerine (A0, A1, A2, A3 gibi), dijitalse Ardıninonun dijital pinlerine (D2, D3, D4 gibi) yapılması gerekmektedir. Bazı algılayıcılarda hem analog hem de dijital sinyal çıkışı bulunabilmektedir. Bu durumdaki algılayıcıların hangi çıkış tipi kullanılacaksa ona uygun pine bağlantılarının yapılması gerekmektedir.

Aktif algılayıcıların mikrodenetleyici kartlara haberleşmesi ise daha fazla pin kullanımını gerektirebilir. Eğer algılayıcıda I²C protokolu kullanılıyorsa Arduino kartlarla haberleşme için + Volt (Vcc) ve – Volt (toprak hattı) dışında SDA (SerialDAta) ve SCL (SerialCLock) olmak üzere iki bağlantı hattının daha kullanılması gerekmektedir. Bu algılayıcıların haberleşmesi için kullanılan Arduino kart türüne

göre hangi pinlerin SDA ve SCL pini olduğunun bilinmesi ve bağlantıların buna göre yapılması gerekmektedir. Eğer algılayıcılar Arduino kartla haberleşmek için SPI protokolü kullanıyorlarsa, kullanılan Arduino kartın türüne göre hangi pinlerin MISO (Master In Slave Out), MOSI (Master Out Slave In) ve SCLK (Serial Clock) ve SS (Slave Select) pinleri olduğunun bilinmesi ve bağlantılarının buna göre yapılması gerekmektedir. Genellikle algılayıcı ile ilgili dokümanlarda bağlantılar ve kullanılacak pinler aşağıdali örnekte olduğu gibi gösterilmekte veya açıklanmaktadır. Konu ile ilgili ayrıntılar Arduino IDE konusunda verilmektedir.



Resim 5.47: Pin bağlantı örneği

5.6. Robotik Programlamada Kullanılan İşlemciler

Robotik programlamada kullanılan işlemcilere mikrodenetleyici (Microcontroller) adı verilmektedir. Mikrodenetleyiciler endüstriye ve elektronik sanayisine yönelik olarak kontrol ve otomasyon işlemlerini gerçekleştirmek için tasarlanmış özel mikroişlemciler olarak tanımlanabilmektedir. Bir mikroişlemci sadece işlem ve hafıza birimlerinden oluşurken bu özel mikroişlemciler birçok bileşenden oluşmaktadır.

Mikrodenetleyicilerin içerisinde; aritmetik ve mantıksal işlemler yapan bir mikroişlemci CPU (Central Process Unit), sistemin komutlarının kalıcı olarak tutulduğu ROM (Read Only Memory) bellek, geçici verilerin ve sonuçların tutulduğu RAM (Random Access Memory) bellek bulunmaktadır. Ayrıca seri giriş ve çıkış birimleri I/O (input – output), SPI (Serial Peripheral Interface) ile bilgi alışverişi için kullanılan programlanabilen seri iletişim (UART-USART/Ethernet) ara birimleri, Analogdan Dijitale (A/D) ya da Dijitalden Analoğa (D/A) çeviriciler (konvektör), sayıcılar (counter), PWM sinyal üretici gibi çevre birimlerini de türüne göre yer almaktadır. Bu özellikleri ile değerlendirildiğinde mikrodenetleyici programlanabilme, bir programı içerisinde depolayıp daha sonra çalıştırabilme özelliklerine sahip tek bir işlemciden (tümleşik devre) oluşan bir mikro bilgisayardır. Bu özellikleriyle mikrodenetleyiciler mikroişlemcilerden ayrılmaktadır.

Günümüzde birçok üretici (Intel, Atmel, Michrochip, National Semiconductror, Texas Instruments, vb.) çeşitli tür ve modellerde 8, 16 veya 32 bit mikrodenetleyiciler üretmektedir. Bunlardan en yaygın olanları, Microchip firmasının PIC (Peripheral Interface Controller) ailesini oluşturan PIC10, 12, 16, 17, 18, 24 ve PIC32M model mikrodenetleyiciler, Atmel firmasının AVR ailesini oluşturan tinyAVR, Mega AVR, XMEGA, AVR32 serisi mikrodenetleyiciler, Texas Instruments firmasının MSP430 ailesini oluşturan mikrodenetleyiciler ile ARM tabanlı TI, ST ve ATMEL mikrodenetleyicileridir.

Resim 5.48: Robotik programlamada kullanılan işlemciler

5.6.1. Robotik Programlamada Kullanılan İşlemcilerin Görevleri

Öncelikli olarak tek başlarına çalışmaları, bütün iş ve işlemleri tek başına yapmaları gerekmektedir. Ayrıca donanımı oluşturan diğer elektronik devrelerle (düşük hızlı çevre birimleri, örneğin algılayıcı) iletişim kurmak mikrodenetleyicinin görevidir. Bunun için gerekli iletişim yapılarını sağlaması, algılayıcılardan gelen analog verileri dijital verilere dönüştürmesi gerekmektedir. Ayrıca algılayıcılardan gelen her türlü verinin toplanması ve işlenmesini yapmak, uygulamanın gerektirdiği bütün fonksiyonları gerçekleştirmek, üzerinde çalışan programda bir sorun olduğu takdirde programın sıfırlanmasını sağlamak yine temel görevlerini oluşturmaktadır.

5.7. Mikrodenetleyici Kartlar (Geliştirme Kartları) ve Görevleri

Mekanik, elektromekanik ve elektronik sistemlerin veya bunların bileşeni olan robotların kontrolü için kullanılabilen, üzerinde 8, 16 veya 32 bit mikrodenetleyicilerin bulunduğu, çeşitli fiziksel boyutları olan genelde mini bir kart şeklindeki elektronik platformdur. Her amaca uygun farklı büyüklük ve özelliklerde farklı tür ve modeli olan single-board (bütün üyeleri tek bir kart üzerinde olan sistem) mini bilgisayardır. Kartlara göre farklılık göstermekle beraber kart ile bilgisayar arasındaki bağlantı için genellikle USB iletişim birimi kullanılmaktadır. Dâhilî Wi-Fi veya bluetooth parçası olan çeşitleri de bulunmaktadır.

Geliştirme kartları için farklı programlama ortamları (bilgisayar üzerinde, web üzerinde) ve programlama dilleri bulunmaktadır. Kendine özgü kolay blok veya metin tabanlı programlama dilleri yanında C/C++, Pyhton gibi yüksek seviyeli dillerle de programlanabilmektedir. Hemen hemen bütün işletim sistemleri ile kullanılabilmektedir. Bu geliştirme ortamları kodları derleyip kolayca mikrodenetleyiciye yüklemeyi sağlamaktadır. Geliştirme kartları için oluşturulan kütüphaneler, birçok işlemi donanım seviyesine inmeden mikrodenetleyicinin kaydedicileri üzerinde işlemler yapmaya gerek duymadan yapmayı sağlayacak şekilde oluşturulmuştur. Birçok işlem bu kütüphane fonksiyonları ile yapıldığından kullanıcı daha az kodla ve kolayca programlama yapabilmektedir. Bu tür kartların en büyük özelliğinin kullanım kolaylığı olduğunu belirtebiliriz. Ardunio UNO, Raspberry PI, Beagle Bone robotik uygulamalar için yaygın olarak kullanılan kartlardan bazılarıdır.

Resim 5.49: Mikrodenetleyici kartlar (Geliştirme kartları)

5.8. Mikrodenetleyici Kartlar (Geliştirme Kartları) İçin Kalkanlar (Shields) ve Görevleri

Mikrodenetleyici kartların özelliklerini geliştirmek, yeni fonksiyon ve özellikler kazandırmak veya kolayca diğer kart yapısındaki bileşenleri eklemek için kullanılan, doğrudan mikrodenetleyici kart üzerine takılabilen (eklenebilen katmanlar) farklı tür ve çeşitlerde kartlardır. Örneğin bluetooth shield

adından da anlaşılabileceği gibi mikrodenetleyici kartla bluetooth kullanarak haberleşmeyi, veri alışverişi yapmayı sağlarken aynı şekilde ethernet shield de mikrodenetleyici kartla ethernet üzerinden haberleşmeyi sağlamaktadır. Bazı kalkanlar katmanlar şeklinde üst üste takılabilmektedir. Kalkanların mikrodenetleyici kart üzerine takılmasında herhangi bir kablolama ihtiyacı bulunmamaktadır. Soket yapıdaki ürünler birbirine geçirilerek kullanılmaktadır.

Resim 5.50: Mikrodenetleyici kartlar (Geliştirme kartları) için kalkanlar (Shields)

5.9. Robot Kontrol Kartları

Özellikle robotik uygulamalar için geliştirilmiş olup üzerinde bir mikrodenetleyici, motor sürücü, Wi-Fi veya Bluetooth gibi kablosuz iletişim parçası bulunan kartlardır. Bazılarında her üç bileşen bulunabildiği gibi, daha az veya daha çok bileşen bir arada bulunabilir. Bazı çeşitlerde bir robotu programlayarak kontrol etmek için gerekli tüm elektronik donanımlar kart üzerinde yer alabilmektedir. Genellikle giriş çıkış bağlantıları (I/O portları) soketli olarak yapılmıştır. Bu sayede soketli bileşenler soketli birimlere kolayca bağlanabilir. Bunun yanında pin içeren bağlantılar da kullanılabilir. Robot kontrol kartları üzerindeki motor sürücüler ile kullanılacak motorlar doğrudan bağlanabilmektedir. Bu tür kartlar robot yapımı ve kontrolünü oldukça kolaylaştırır.

Resim 5.51: Çeşitli robot kontrol kartları

5.9.1. Robot Kontrol Kartlarının Görevleri

Robot kontrol kartlarının görevi, robot için gerekli elektronik bileşenlerin tamamını veya önemli bir kısmını karşılayarak robot yapımını kolaylaştırmaktır. Ayrıca kart üzerinde bulunmayan bileşenler için

bağlantı ortamı oluşturmak, çok bileşenli yapıdan daha az bileşenli yapıya geçişi sağlayarak karmaşıklığı azaltmak, kullanımı kolaylaştırmak, fiziksel büyüklükten tasarruf sağlamak gibi görev ve yaraları bulunmaktadır.

5.10. Düşünelim / Araştıralım

Robot programlama dersinde kullanmak üzere bir eğitsel robot yapacağınızı düşünerek gerekli olabilecek elektronik bileşenlerin seçimi için İnternet'te araştırma yapınız. Niçin bu bileşenleri seçtiğinizi, bileşenlerin hangi özelliklerinin seçiminizde etkili olduğunu açıklayınız.

5.11. Değerlendirme Soruları

1. **Robotlarda kullanılan motorların kontrol edilebilmesi (çalışma, durma, ileri geri hareket etme, hızlanma, yavaşlama vb.) için kullanılan bileşenlere ne ad verilir?**
   a) Robot kontrol kartı
   b) Mikrokontrolör
   c) Motor sürücü
   d) Motor denetleyici
   e) Mikroişlemci

2. **Bilgisayara bağlamak istediğimiz bir çevresel aygıtın (örneğin mikrodenetleyici kartın) üzerinde seri iletişim bağlantı noktası bulunmuyorsa aşağıdaki seçeneklerden hangisinin kullanılması uygundur?**
   a) Paralel bağlantı noktası
   b) Wi-Fi
   c) USB-Seri çevirici
   d) USB-UART çevirici
   e) Bluetooth

3. **Robotun kontrol edileceği, programlanacağı aygıtlara (bilgisayar, tablet veya akıllı telefon) kablosuz olarak bağlanabilmesi için yaygın olarak kullanılan haberleşme bileşeni aşağıdakilerden hangisidir?**
   a) Bluetooth
   b) Wi-Fi
   c) XBee
   d) ZigBee
   e) WiMAX

4. Robotun bulunduğu ortamdan bilgi alan algılayıcılara ne ad verilir?

a) Propriyoseptif algılayıcılar
b) Eksteroseptif algılayıcılar
c) Pasif algılayıcılar
d) Aktif algılayıcılar
e) Dijital sinyal veren algılayıcılar

5. Aşağıdakilerden hangisi dışarıdan haricî hiçbir güç kaynağına ihtiyaç duymadan çevrelerinden aldıkları fiziksel ya da kimyasal sinyalleri ölçen algılayıcılardır?

a) Propriyoseptif algılayıcılar
b) Eksteroseptif algılayıcılar
c) Pasif algılayıcılar
d) Aktif algılayıcılar
e) Analog sinyal veren algılayıcılar

6. Aşağıdaki algılayıcıların hangisi mikrodenetleyici kartların haberleşmesi için kullanılan standart protokollerden biridir?

a) USB
b) UART
c) Seri
d) Paralel
e) I^2C

7. Mikrodenetleyicilerde aşağıdaki bileşenlerden hangisi yer almaz?

a) I/O (İnput – Output-Seri Giriş ve Çıkış Birimleri)
b) GUI (Grafiksel Kulanıcı Arayüzü)
c) ROM (Read Only Memory-Sadece Okunabilir Bellek)
d) RAM (Random Access Memory-Rastgele Erişimli Bellek)
e) SPI (Serial Peripheral Interface-Seri Çevresel Arayüz)

8. Mekanik, elektromekanik ve elektronik sistemlerin veya bunların bileşeni olan robotların kontrolü için kullanılabilen, üzerinde 8, 16 veya 32 bit mikrodenetleyicilerin bulunduğu çeşitli fiziksel boyutlardaki temelde mini bir kart şeklinde elektronik platformlara ne ad verilir?

a) Mikrobilgisayar kartı
b) Mikroişlemci kartı
c) Mikroprogramlayıcı kart
d) Mikrodenetleyici kart
e) Geliştirme kiti

9. Robotikte kullanılan kartların özelliklerini geliştirmek, yeni fonksiyon ve özellikler kazandırmak veya kolayca diğer kart yapıdaki bileşenleri eklemek için kullanılan ve doğrudan mevcut kartın üzerine takılabilen kartlara ne ad verilir?

a) Mikrobilgisayar kartı
b) Mikroişlemci kartı
c) Mikroprogramlayıcı kart
d) Mikrodenetleyici kart
e) Kalkan (Shields) kart

10. Robot için gerekli elektronik bileşenlerin tamamını veya önemli bir kısmını karşılayarak robot yapımını kolaylaştıran kart türü aşağıdakilerden hangisidir?

a) Robot kontrol kartı
b) Mikrobilgisayar kartı
c) Mikroprogramlayıcı kart
d) Mikrodenetleyici kart
e) Kalkan (Shields) kart

6. BLOK TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI

Bu bölümün sonunda,

- ✓ Blok tabanlı ortamların ve yazılımların temel yapısını tanımlayabilecek,
- ✓ mBlock robot programlama ortam ve yazılımının kullanım özelliklerini açıklayabilecek,
- ✓ mBlock'ta programlama yapılan bilgisayarla robot ve geliştirme kartları arasında bağlantı oluşturulabilecek,
- ✓ mBlock'ta programlama yapılan bilgisayarla robot ve geliştirme kartları arasında bağlantı ayarlarını yapabilecek,
- ✓ Genel programlama yapılarının çalışma mantığını özetleyebilecek,
- ✓ Harekete yönelik yapıların çalışma mantığını tanımlayabilecek,
- ✓ Harekete yönelik yapıları programlayabilecek,
- ✓ Görünüme yönelik yapıların çalışma mantığını açıklayabilecek,
- ✓ Görünüme yönelik yapıları programlayabilecek,
- ✓ Sese yönelik yapıların çalışma mantığını tanımlayabilecek,
- ✓ Sese yönelik yapıları programlayabilecek,
- ✓ Veriye yönelik yapıları, geliştirdiği program için uygun şekilde kullanabilecek,
- ✓ Olaylara yönelik yapıları, geliştirdiği program için uygun şekilde kullanabilecek,
- ✓ Kontrol yapılarını, geliştirdiği programa uygun şekilde kullanabilecek,
- ✓ Algılama yapılarının çalışma mantığını açıklayabilecek,
- ✓ Algılama yapılarını programlayabilecek,
- ✓ İşlem yapılarını, geliştirdiği program için uygun şekilde kullanabilecek,
- ✓ Robota özgü yapıları listeleyebilecek,
- ✓ Robota özgü yapıların çalışma mantığını açıklayabilecek,
- ✓ mBlock robot programlama ortam ve yazılımında geliştirilen programları yorumlayabilecek,
- ✓ mBlock robot programlama ortam ve yazılımında geliştirilen programları yeniden düzenleyebilecek,
- ✓ mBlock robot programlama ortam ve yazılımında program geliştirilebileceksiniz.

6.1. Blok Tabanlı Robot Programlama Yazılımları ve Ortamları

Robot programlama için oluşturulmuş metin tabanlı programlama yazılımlarına göre daha kullanıcı dostu, öğrenici ve hobi kullanıcılarına hitap eden diller ve ortamlar da bulunmaktadır. Birçoğu ücretsiz olan bu araçlarla hiçbir kod kullanmadan, sürükle bırak veya yapboz oynar gibi programlar oluşturmak olanaklı hale gelmiştir. Bu tür ortamlara blok programlama ortamları adı verilmektedir. Lego NXT-G, EV3, Enchanting, Robo Pro, Modkit, miniBlog, Ardublock, Snap4Arduino, Srach for Arduino (S4A) ve mBlock bu ortamlara örnek olarak verilebilir. Burada blok tabanlı programlama yazılımı ve ortamı olarak mBlock tercih edilmiştir.

6.2. mBlock Grafik Programlama Yazılımı ve Ortamı

mBlock Makeblock tarafından geliştirilen Arduino projelerinde programlamayı ve etkileşimli uygulamalar oluşturmayı kolaylaştıran grafik ara yüzlü görsel programlama yazılımı ve ortamıdır. Fiziksel dünya ile etkileşim içinde interaktif uygulamalar (oyun, hikâye, animasyon) ve kablosuz olarak programlanabilen robotlar oluşturmak için modüler ve geliştirilebilir şekilde tasarlanmıştır. Gerçek zamanlı kod üretme desteğine sahiptir. Scratch tarzındaki (sürükle-bırak) açık kaynak kodlu bu kod yazma programı Makeblock tarafından geliştirilen mBot eğitim robotlarının programlanmasında kullanıldığı gibi Arduino temelli robotların programlanmasında da kullanılabilmektedir. Bu amaçla Arduino Board Standartlarını desteklemektedir. Arduino UNO, Leonardo, Nano, Mega128, Mega 2560, PicoBoard, Makeblock mCore ve Arduino uyumlu diğer kartlarla kullanılabilmektedir.

Bu görsel programlama ortamı, açık haberleşme protokolleri ve kaynak kodları kullanmaktadır. Windows ve MAC uyumlu güncel sürümü 20'den fazla dili desteklemektedir. Ücretsizdir ve kaynak kodları açıktır. Herhangi bir yardımcı ek uygulama olmaksızın kullanılabilmektedir. Kablosuz haberleşme protokolleri de desteklendiği için daha esnek kullanım olanağı sunmaktadır.

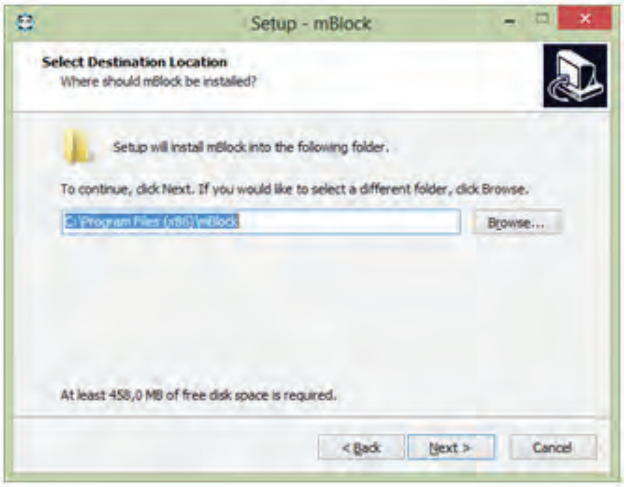
mBlock ortamında hazırlanan örnek uygulamalar dersin sitesinde resim numarası ile yer almaktadır. Bilgisayarınıza kopyalayarak farklı parametrelerle test ediniz.

6.3. mBlock Grafik Programlama Yazılımının Yüklenmesi

mBlock yazılımının son sürümü http://www.mblock.cc/download sitesinden, kullanılacak işletim sistemi (Windows veya Mac) seçilerek ücretsiz olarak indirilebilmektedir. Hem Windows hem de Mac işletim sistemlerinin eski ve güncel sürümlerinde çalışabilmektedir. Seçimden sonra doğrudan exe dosyası olarak indirilmektedir. İndirildikten sonra mBlock_win_V3.4.5.exe'ye tıklayarak programın kurulması yeterlidir. Kurulum işlemi tamamlandıktan sonra program kullanılmaya hazırdır. Önemli kurulum aşamaları aşağıda verilmiştir.

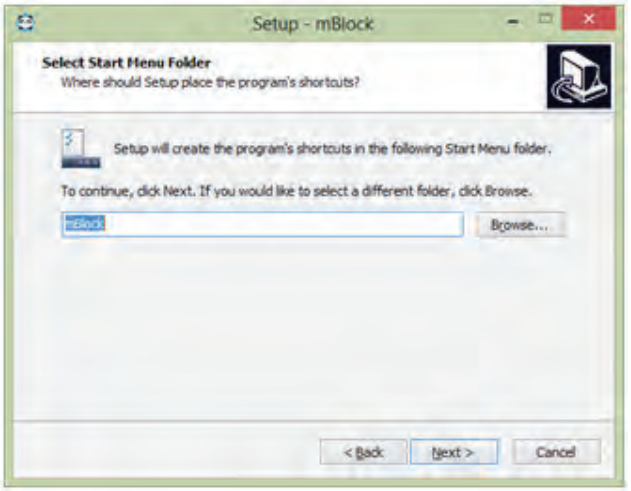
Resim 6.1: mBlock indirilme ekranı

Dil seçimi yapıldıktan sonra ilk aşamada lisans sözleşmesi onaylanmalıdır. Daha sonra üçüncü aşamada kurulum klasörü seçilmelidir. İstenirse olduğu gibi bırakılabilir.



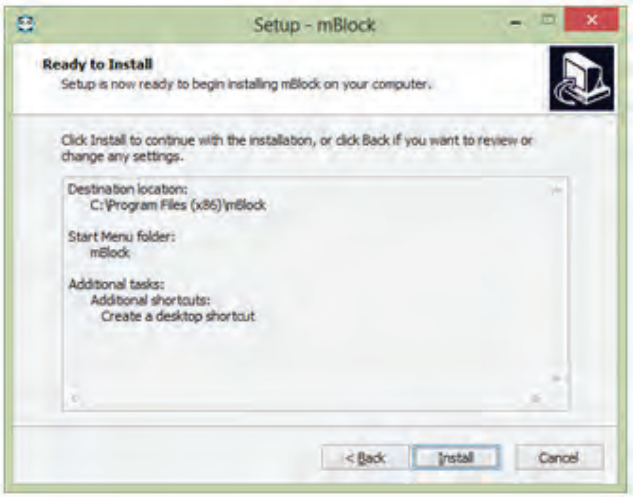
Resim 6.2: mBlock kurulum klasörü seçimi

Kurulumun bu aşamasında programın başlangıç menüsü kısayolu için klasör seçimi yapılabilir. İstenirse olduğu gibi bırakılabilir.



Resim 6.3: mBlock başlangıç klasörü seçimi

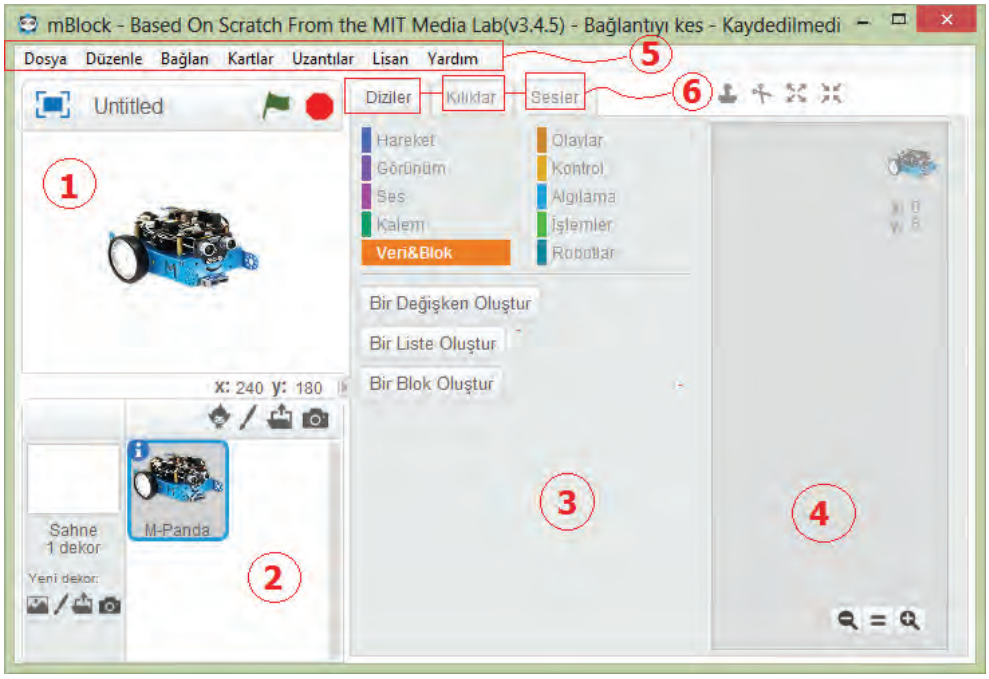
Kurulum yapılacak başlangıç menüsü klasörü seçildikten sonra programın kurulumu install seçeneğinin tıklanmasıyla başlar. Bittiği zaman programın masaüstündeki kısa yoluna tıklanarak program çalıştırılır.



Resim 6.4: mBlock kurulum ekranı

6.4. mBlock Programının Arayüzü, Yapısı ve Temel Özellikleri

Program çalıştırıldığı zaman karşımıza aşağıdaki arayüz çıkmaktadır. Arayüz üzerindeki yapılar ve görevleri aşağıda açıklanmıştır.



Resim 6.5: mBlock programının arayüzü ve yapısı

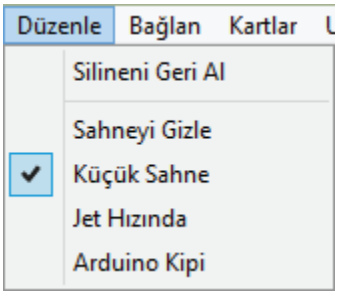
Programlama ortamı menü yapısında (5 numaralı alan) sunulmakta, 4 temel bölümden oluşmaktadır (Görsel 6.65). 1 numaralı bölümde seçilen dekor içerisinde seçilen figür (kukla) yer almaktadır. 2

numaralı bölümde sahne (dekor) ve figür (kılık, kukla) seçimi yapılmaktadır. 3 numaralı bölüm programlama ortamını oluşturan blok yapılara ayrılmıştır. Programın yazıldığı alan ise 4 numara ile belirtilmiştir. mBlock programlama ortamı (6 numaralı alan) diziler (blok kategorileri), sahneler (dekor), figürler (kılık, kukla) ve seslerden oluşan bir yapı içerisinde sunulmaktadır. Bu ortamı kullanarak etkileşimli hikâyelerle, oyunlarla, animasyonlarla ve robotlarla eğlenceli ve kolay şekilde programlama yapmak veya yapmayı öğretmek mümkündür. mBlock'ta robot programlamayı öğrenmek veya öğretmek için kullanabilecek yapılar "Diziler" başlığı altında; "Hareket", "Görünüm", "Ses", "Kalem", "Veri&Blok", "Olaylar", "Kontrol", "Algılama", "İşlemler" ve "Robotlar" olmak üzere toplam 10 kategoride toplanmıştır. Bunların kullanımıyla programlar oluşturulmaktadır. Seksenden fazla sahne, yüzden fazla kılık ve ses; çeşitli kategori, tema veya türlere göre ortamın kütüphanesinden seçilebilmekte, istenildiğinde bilgisayarda bulunan veya oluşturulan dekor, kukla ve sesler kullanılabilmektedir. İstenirse programın sunduğu dekor, kukla ve ses aracıyla da oluşturulabilmektedir.

Dosya menüsü yeni proje açmak, proje yüklemek, proje kaydetmek, projeyi farklı kaydetmek gibi temel dosya işlemlerine ayrılmıştır. Bu menü kullanılarak oluşturulan resimler bilgisayara veya bilgisayardan programa aktarılabilmektedir.

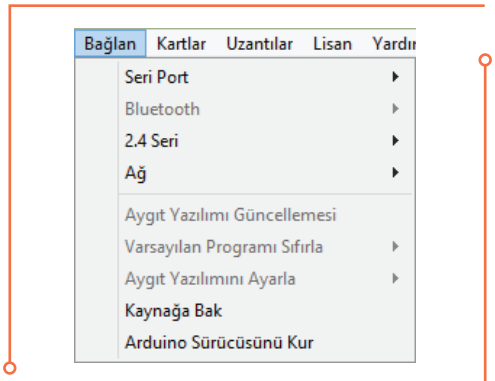
Resim 6.6: mBlock dosya menüsü

Düzenle menüsü ile silinen sahnelerin geri alınması, sahnenin gizlenmesi, küçük sahne seçimi, jet hızında ve Arduino kipinde çalışma gibi seçenekler yer almaktadır.



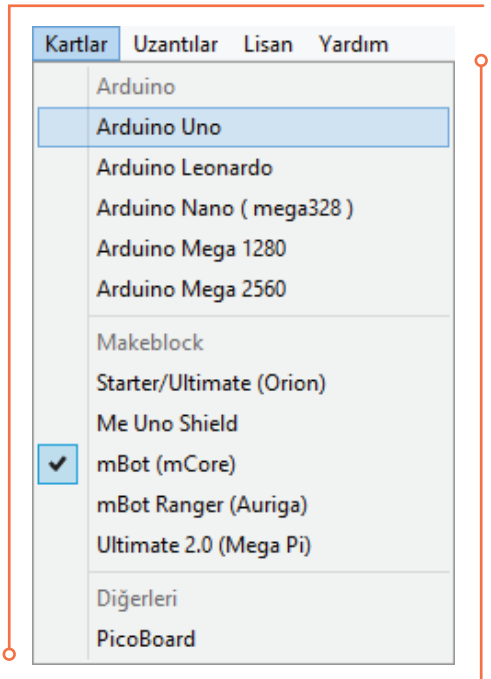
Resim 6.7: mBlock düzenle menüsü

Bağlan menüsü robotun mBlock programlama ortamına bağlanması için kullanılabilecek seçenekleri barındırmaktadır. Seri port (USB portu üzerinden), bluetooth, 2.4 seri (USB Dongle üzerinden) ve ağ üzerinden bağlantı seçenekleri sunulmaktadır. Makeblock tarafından sunulan robotlar için aygıt yazılım güncellemeleri, varsayılan programın sıfırlanması, aygıt yazılımının ayarlanması, aygıt yazılımının kaynağının görüntülenmesi ile Arduino sürücülerin kurulması yine bu menüden yapılmaktadır. Kurulum sırasında mevcut Arduino modellerinin sürücüleri standart olarak kurulmaktadır.



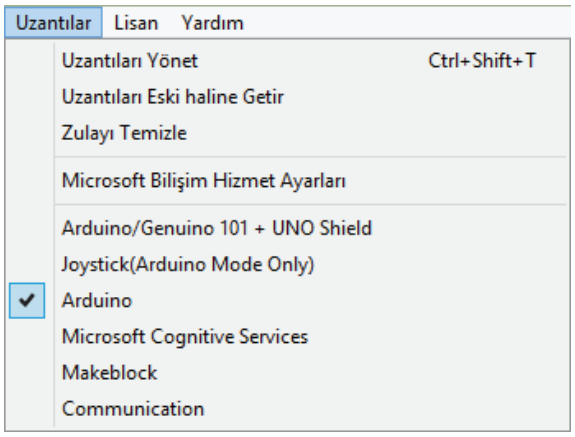
Resim 6.8: mBlock bağlan menüsü

Kartlar menüsü ile kullanılan Arduino kart modelleri, Arduino uyumlu kartlar ve Makeblock tarafından üretilen robot kontrol kartları içerisinden seçim yapılmaktadır. Robotik kontrol kartı hangi model Arduino içeriyorsa o kart modeli seçilmelidir. Arduino uyumlu kartlar için Arduino, diğer karlar için diğerleri seçilebilir. Burada Scratch ile programlanabilen sensör kartı PicoBoard desteği de yer almaktadır.



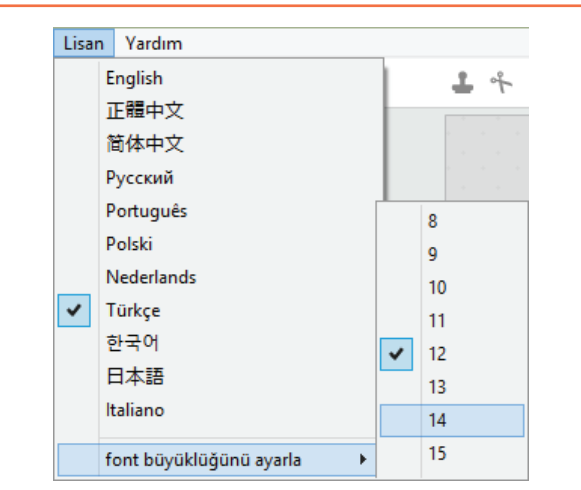
Resim 6.9: mBlock kartlar menüsü

Uzantılar menüsü ile kullanılacak kartın türü seçilmektedir. Seçilen türe özgü kod blokları dizeler sekmesinde kullanılabilir hâle gelmektedir. Örneğin Arduino seçildiğinde bunun için oluşturulmuş kod blokları dizeler sekmesine eklenecektir. Birden fazla kart türü seçilebilmektedir.



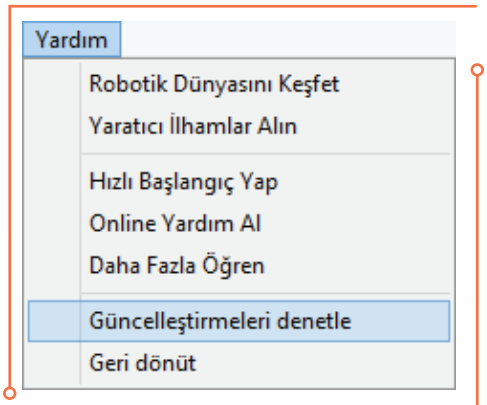
Resim 6.10: mBlock uzantılar menüsü

Lisan menüsü ile istenilen dil ve font büyüklüğü seçilebilmektedir. mBlock şu anda yirmi dilde kullanılabilmektedir.



Resim 6.11: mBlock lisan menüsü

Yardım menüsünde yardım konularının yanında güncelleştirmelerin kullanımı ve üretici firmanın robotik ürünlerinin tanıtıldığı site ile örnek uygulamaların bulunduğu siteye yönlendirme yapılmaktadır.



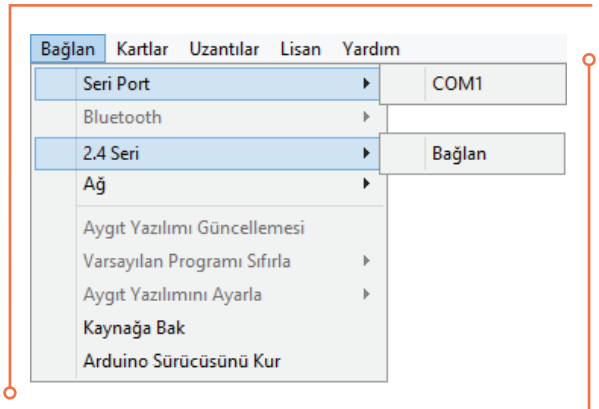
Resim 6.12: mBlock yardım menüsü

6.5. mBlock Yüklü Bilgisayarla Robot Arasında Bağlantının Oluşturulması

mBlock ile robotun programlanması ve kontrol edilmesi için kablolu veya kablosuz olarak gerçekleştirilebilecek dört farklı seçenek sunulmaktadır. Bunlar; USB port ve Ağ üzerinden kablolu bağlantı, Bluetooth modülü ile ve 2.4 Ghz Dongle modülü üzerinden kablosuz bağlantı seçenekleridir. Kullanılacak robot hangi bağlantı türünü destekliyorsa o seçilmelidir. Kablolu bağlantılar için USB, kablosuz bağlantılar için Bluetooth en çok tercih edilen bağlantı şekilleridir.

USB bağlantı Arduino temelli robotlara program yüklemek için gereklidir. Ayrıca Makeblock tarafından üretilen robotların programlanması, güncellemesi için yine USB bağlantı türü kullanılmaktadır. Bağlantı gerçekleştirildiğinde blokların üzerinde yer alan kırmızı noktanın rengi yeşile dönmektedir.

Bağlan menüsü (Görsel 6.73) robotun mBlock programlama ortamına bağlanması için kullanılabilecek seçenekleri barındırmaktadır. Seri port seçeneği robotun USB kablosu ile bilgisayara bağlanması için kullanılmaktadır. Uygun olan COM (iletişim kapısı) listeden seçilmelidir. Bluetooth seçeneği ile Bluetooth üzerinden, 2.4 seçeneği ile USB Dongle (Mbot robot ile sunulmaktadır) üzerinden kablosuz bağlantı, ağ seçeneği ile de ağ üzerinden bağlantı seçenekleri sunulmaktadır. Makeblock tarafından sunulan robotlar için aygıt yazılım güncellemeleri, varsayılan programın sıfırlanması, aygıt yazılımının ayarlanması, aygıt yazılım kaynağının görüntülenmesi ile Arduino sürücülerin kurulumu yapılmaktadır. Kurulum sırasında mevcut Arduino modellerinin sürücüleri standart olarak kurulmaktadır.



Resim 6.13: mBlock ile robot arasındaki bağlantının oluşturulması

6.6. mBlock Programlama Dilinin Temel Özellikleri ve Programlama Yapısı

Blok tabanlı programlama dilleri ve ortamlarında komutu oluşturan bloklar, bulunduğu alandan programlama alanına sürükle bırak tekniği ile aktarılır. Bütün bloklar kullanıma açık olmalarına karşın robotik uygulamalarda kullanılmayan bloklar vardır. Bunlar kullanılmak istendiğinde herhangi bir işlev göstermeyecektir.

mBlock'ta robot programlama yapmak ve robotları kontrol etmek için kullanılabilecek komut blokları "Diziler" başlığı altında; "Hareket", "Görünüm", "Ses", "Kalem", "Veri&Blok", "Olaylar", "Kontrol", "Algılama", "İşlemler" ve "Robotlar" olmak üzere toplam on kategoride toplanmıştır. Burada yer alan blok yapıları fiziksel bir robotun programlanması ve kullanımının sağlanması dışında sanal bir robotun (kukla, figür kullanılarak) programlanması ve kullanımını da içermektedir.

6.6.1. Hareket Alt Başlığı Altında Verilen Komut Blokları

"Hareket" alt başlığı altında verilen komut blokları sanal bir robotun (kütüphaneden kukla, figür kullanılarak veya oluşturularak) hareket kullanımı ve diğer uygulamalar için kullanılmaktadır. Buradaki blokların kullanılmasıyla sanal robotun istenilen yöne döndürülmesi, belirli bir koordinata gitmesi, belirli bir zaman aralığında belirli bir koordinata yavaşça ilerlemesi (süzülebilmesi) sağlanabilir. Ayrıca sanal robotun kenara geldiği zaman sekmesi de buradan gerçekleştirilebilir. Sanal robotun hangi yönlere dönmesine izin verilip verilmeyeceği de belirlenebilir. "Hareket" alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

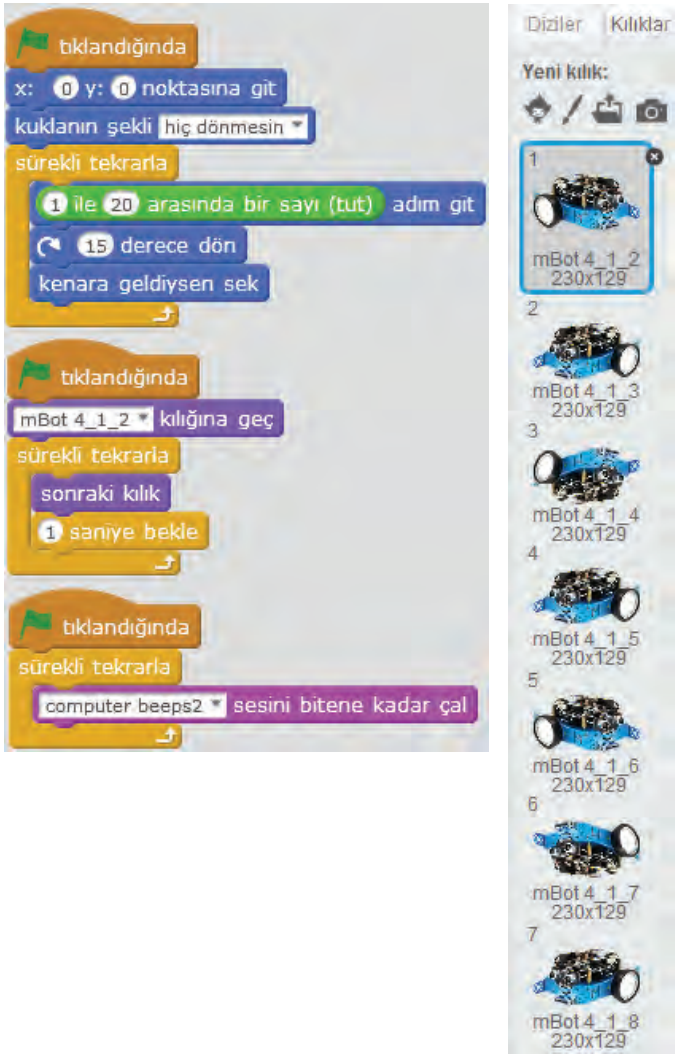
10 adım git	Belirtilen adım kadar hareket etmesi için kullanılır.
15 derece dön	Belirtilen açıda soldan sağa dönmesi için kullanılır.
15 derece dön	Belirtilen açıda sağdan sola dönmesi için kullanılır.
90 yönüne dön	89 Derce sağa, -90 derece sola, 0 derce yukarı ve 180 derce aşağı dönmesi için kullanılır.
'ye doğru dön	Kuklanın fare okuna, Kuklaya, rasgele yatay, dikey ve sahne noktasına dönmesi için kullanılır.
x: 0 y: 5 noktasına git	Belirtilen x ve y noktasına gitmesi için kullanılır.
fare oku 'na git	Kuklanın fare okuna, Kuklaya, rasgele yatay, dikey ve sahne noktasına gitmesi için kullanılır.
1 sn.de x: 0 y: 5 a süzül	Belirtilen süre içerisinde, belirtilen x ve y noktalarına süzülmesi için kullanılır.
x'i 10 arttır	x'i belirtilen değer kadar artırmak için kullanılır.
x, 0 olsun	x'i belirtilen nokta yapmak için kullanılır.
y 'yi 10 arttır	y'i belirtilen değer kadar artırmak için kullanılır.
y, 0 olsun	y'i belirtilen nokta yapmak için kullanılır.
kenara geldiysen sek	Kenara geldiyse sekmesi için kullanılır.
kuklanın şekli sağa-sola dönebilsin	Sağa sola dönebilmesi, hiç dönememesi veya her yöne dönebilmesi için kullanılır.
x konumu	x konumunun ekranda gösterimi için kullanılır.
y konumu	y konumunun ekranda gösterimi için kullanılır.
yönü	Yönünün ekranda gösterimi için kullanılır.

Tablo 6.1: Hareket alt başlığı altında verilen komut blokları

Hareket Örneği: Bu örnekte sanal robot, ortamdaki ses seviyesi 10 birimden yüksek olunca 250 adım gitmekte; 1 sn bekledikten sonra ekrana "Merhaba!" yazısı 2 sn boyunca çıkmaktadır. 1 sn daha bekledikten sonra geldiği noktaya geri dönmektedir.

Resim 6.14: Hareket örneği

Yandaki diğer örnekte ise basit bir animasyon çalışması görülmektedir. Kullanılan mBot kuklası 1 ile 20 arasında adımla hareket etmekte, 15 derece dönmekte ve eğer kenara geldiyse sekmektedir. Her işlemden sonra bir sonraki kılığa geçerek animasyon oluşturulmaktadır. Her hareketin ardışık olarak gerçekleşmesi için birbirini izleyen hareket görüntülerinden oluşan kukla resimleri kullanılmalıdır. Animasyon sırasında "computer beeps2" sesini çıkarmaktadır.



Resim 6.15: mBlock hareket örneği

6.6.2. Görünüm Alt Başlığı Altında Verilen Komut Blokları

"Görünüm" alt başlığı altında verilen bloklar sanal bir robotun (kütüphaneden kukla, figür kullanılarak veya oluşturarak) görünüm kullanımı ve diğer uygulamalar için kullanılmaktadır. Buradaki blokların kullanılmasıyla sanal robotların veya sahnelerin görünümünde değişiklik yapılabilir. Belirli bir süre ya

da sürekli olarak bir mesaj verilebilir, bir düşünme cümlesi belirtilebilir. İhtiyaca göre istenilen sanal robotun sahnede görünüp gizlenmesi sağlanabilir. Sahnede birden fazla dekor (arka plan, hareketsiz nesneler) ve birden fazla sanal robot bulunabilir. Bunlar arasında geçişler yapılabilir. Sanal robotun veya dekorun renk etkisi (0-200 arası) değiştirilebilir. Sanal robotun büyüklüğünde değişiklik yapılabilir ya da yapılacak uygulamalara göre belirli bir büyüklük seçilebilir. Sahnede bulunacak nesnelerin istenilen oranla önde ya da arkada görünmesi sağlanabilir. Görünüm alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

Merhaba ! de 2 saniye	Belirtilen süre kadar, belirtilen ifadeyi konuşma balonu içerisinde yazdırmak için kullanılır.
Merhaba ! de	Belirtilen ifadeyi konuşma balonu içerisinde yazdırmak için kullanılır.
Eee... diye düşün 2 saniye	Belirtilen süre kadar, belirtilen ifadeyi düşünme balonu içerisinde yazdırmak için kullanılır.
Eee... diye düşün	Belirtilen ifadeyi düşünme balonu içerisinde yazdırmak için kullanılır.
görün	Gizlenmiş (ekranda görünmeyen) kuklanın görünmesi için kullanılır.
gizlen	Kuklanın gizlenmesi (ekranda görünmmemesi) için kullanılır.
Panda-b kılığına geç	Kuklanın seçilen kılığa geçmesi için kullanılır.
sonraki kılık	Kuklanın sonraki kılığa geçmesi için kullanılır.
dekor1 dekoruna geç	Dekorun program içerisinde değiştirilmesi için kullanılır.
renk etkisini 25 arttır	Belirtilen efekt etkisini belirtilen oranda artırmak için kullanılır.
renk etkisi 0 olsun	Belirtilen efekt etkisini belirtilen değere getirmek için kullanılır.
görsel etkileri temizle	Görsel etkileri temizlemek için kullanılır.
10 birim büyüt	Kuklayı belirtilen birim kadar büyütmek için kullanılır.
büyüklüğü % 100 yap	Kuklanın büvüklüğünü belirtilen yüzdelik değer kadar ayarlamak için kullanılır.
üste çık	Kuklanın üst katmana çıkması için kullanılır.
1 katman alta in	Kuklanın belirtilen katman alta inmesi için kullanılır.

Tablo 6.2: Görünüm alt başlığı altında verilen komut blokları

Görünüm Örneği: Bu küçük örnekte sanal robot ortamdaki ses şiddeti 10 birimden büyük olunca, 250 adım gidip 1 sn sonra merhaba demekte ve görüntüsü 25 birim büyümektedir. 5 sn bekledikten sonra büyüklüğü %50 oranında değiştirilmektedir.

Resim 6.16: Görünüm örneği

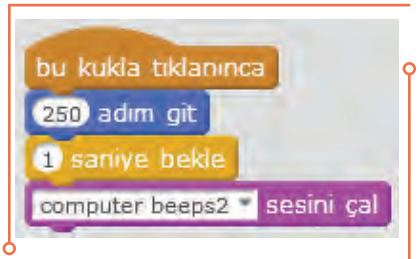
6.6.3. Ses Alt Başlığı Altında Verilen Komut Blokları

"Ses" alt başlığı altında verilen bloklar sanal bir robotun (kütüphaneden kukla, figür kullanılarak veya oluşturarak) sesle kullanımı, ses şiddetine dayalı ve sesle ilgili diğer robotik uygulamalar için kullanılmaktadır. Bilgisayarda bulunan mikrofondan elde edilen sesler de kullanılabilir. Buradaki blokların kullanılmasıyla sahnelere veya sanal robotlara ses eklenebilir. Sanal robot bir hedefe ulaştığında ses çıkarması ya da uygulama devam ettiği sürece arka fon müziği bulunması sağlanabilir. Uygulamaya istenirse ses kaydı ya da var olan bir ses dosyası eklenebilir. Ayrıca mBlock içinde bulunan çalgı aletleri de uygulamalara eklenebilir. Sesin temposuyla ilgili düzenlemeler yapılabilir. "Ses" alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

patlat sesini çal	Belirtilen sesi çalmak için kullanılır.
patlat sesini bitene kadar çal	Belirtilen ses bitene kadar çalmak için kullanılır.
tüm sesleri durdur	Tüm sesleri durdurmak için kullanılır.
8 davulunu 0.25 vuruş çal	Belirtilen ses kaynağını (18 adet) belirtilen vuruş oranı kadar çalmak için kullanılır.
0.25 vuruş sus	Belirtilen vuruş oranı kadar susması için kullanılır.
60 notasını 0.5 süresince çal	Belirtilen notayı belirtilen süre boyunca çalmak için kullanılır.
çalgıyı 1 yap	Müzik enstrumanını (21 adet) değiştirmek için kullanılır.
sesi -10 birim yükselt	Sesi düzeyini belirtilen birim kadar değiştirmek için kullanılır.
ses şiddeti % 100 olsun	Ses şiddetini belirtilen oran kadar değiştirmek için kullanılır.
☐ ses şiddeti	Ses şiddetin ekranda göstermek için kullanılır.
tempoyu 20 arttır	Tempoyu belirtilen oranda değiştirmek için kullanılır.
tempo 60 vuruş/dk olsun	Tempoyu belirtilen vuruş/dakika oranına ayarlamak için kullanılır.
☐ tempo	Tempoyu ekranda göstermek için kullanılır.

Tablo 6.3: Ses alt başlığı altında verilen komut blokları

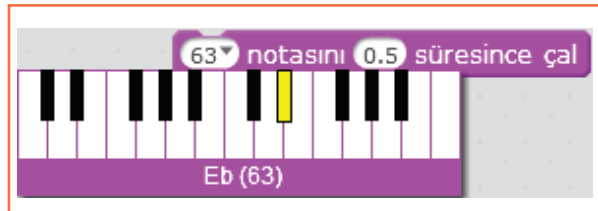
Ses Örneği: Bu basit örnekte sanal robot görüntüsüne tıklanınca 250 adım ilerlemekte ve 1 sn bekledikten sonra "computer beeps2" sesini çalmaktadır. Ses kaynağı ses kütüphanesinden örnek olarak seçilmiştir.



Resim 6.17: Ses örneği

Aşağıdaki diğer örnekte ise belirlenen üç ayrı tuşa basılmasıyla piyanodan seçilen üç ayrı nota çalınmaktadır. Diğer notalar da aynı şekilde eklenerek bir piyano yapılabilir. Hangi tuş için hangi sesin seçileceği aşağıdaki blokun nota numarasına tıklanarak belirlenebilir.

Resim 6.18: 2. Ses örneği

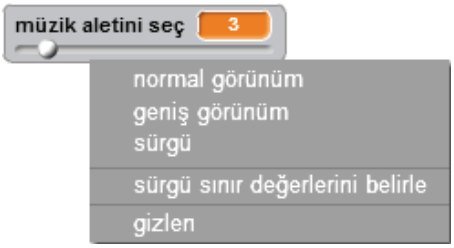


Resim 6.19: 2. Ses örneği için nota

Örnekte piyanonun farklı müzik aletleri ile kullanılabilmesi için "müzik aletini seç" adında bir değişken oluşturulmuştur. Değişkenin nasıl oluşturulacağı "Veri&Blok" alt başlığı altında verilen komut blokları konusunda açıklanmıştır.

Resim 6.20: 2. Ses örneği için değişken oluşturulması

"müzik aletini seç" değişkeni ile farklı müzik aletlerinin seçimi sürgünün kaydırılması ile yapılmaktadır. Seçim için değişken sağ tıklanarak "sürgü" görünümü tercih edilmiştir.



Resim 6.21: 2. Ses örneği için müzik aleti seçimi

6.6.4. Kalem Alt Başlığı Altında Verilen Komut Blokları

"Kalem" alt başlığı altında verilen bloklar, sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robotun kalem kullanımı, çizim ve grafik uygulamaları ile diğer uygulamaları için kullanılmaktadır. Buradaki blokların kullanılmasıyla geometrik çizimler yapılabilir. Kalemin rengi, tonu ve kalınlığı değiştirilebilir. Ayrıca sanal robotun olduğu yerde izini bırakması sağlanabilir. "Kalem" alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

temizle	Kalemin bıraktığı izleri temizlemek için kullanılır.
iz bırak	Kalemin çizmeye başlaması için kullanılır.
kalemi bastır	Kalemın kullanılacak yüzeyde çizmeye başlaması ıçın kullanılır.
kalemi kaldır	Kalemi kaldırarak çizimi durdurmak için kullanılır.
kalem rengini ☐ yap	Kalemin rengini değiştirmek için kullanılır.
kalem rengini 10 değiştir	Kalemın rengını belırtılen değer kadar değıştırmek ıçın kullanılır.
kalem rengini 0 yap	Kalemin rengini belirtilen renk yapmak için kullanılır.
kalem tonunu 10 arttır	Kalemin tonunu belitirlen oranda artırmak için kullanılır.
kalem tonunu 50 yap	kalemin tonunu belitirlen değer yapmak için kullanılır.
kalem kalınlığını 1 arttır	Kalemin kalınlığını belirtilen değerde artırmak için kullanılır.
kalem kalınlığını 1 yap	Kalemin kalınlığını belirtilen değere getirmek için kullanılır.

Tablo 6.4: "Kalem" alt başlığı altında verilen komut blokları

Kalem Örneği: Bu örnekte ses şiddeti ile ekranda ses grafiği çizilmektedir. X koordinatında -240 noktasından başlayıp, +239 olunca ekranı temizlemektedir. Y koordinatı sıfır olarak alınmıştır (Koordinat düzlemi en solda -240, en sağda +240, en yukarıda +180, en aşağıda -180 arasında değişmektedir). Kalem kalınlığı 1, kalem rengi olarak da siyah seçilmiştir. Tanımlana "Ses" değişkeni ile her ekran

temizlemesinden sonra eğer ses varsa program çalışması sağlanmaktadır. Elde edilen grafik boyutunun büyütülmesi için ses şiddeti 4 kat oranında artırılmıştır. Ses kaynağı olarak bilgisayara bağlı bulunan mikrofon veya web kamera mikrofonu kullanılabilmektedir. Çizilen ses grafik örneği aşağıda verilmiştir. Çizilen noktanın x ve y konumlarını göstermek için hareket blok gurubunda bulunan x ve y konumu blokları işaretlenmiştir.

Resim 6.22: Kalem uygulaması

Resim 6.23: Kalem örneğinin ekran görüntüsü

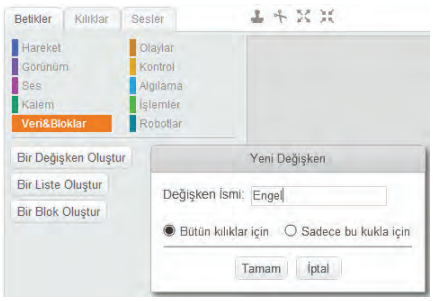
6.6.5. Veri&Blok Alt Başlığı Altında Verilen Komut Blokları

"Veri&Blok" alt başlığı altında verilen bloklar sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robotun programlanması ile diğer uygulama programlarında gerekli olabilecek "değişken", değişken listesi "liste" ve "blok" oluşturulması ve bunların düzenlenmesi için kullanılmaktadır.

Bir Değişken Oluştur	Bir değişken oluşturmak için kullanılmaktadır.
Bir Liste Oluştur	Bir liste oluşturmak için kullanılmaktadır.
Bir Blok Oluştur	Bir blok oluşturmak için kullanılmaktadır.

Tablo 6.5: Veri&Blok alt başlığı altında verilen komut blokları

Değişkenler ve Oluşturulması: Girilen değerleri alan veya programın çalışmasıyla bazı değerlerin atandığı veri tutucularından oluşan temel yapılarından biridir. Değişkenlerin taşıdığı değerler programın akışı içinde farklılaşabilir. Değişkenler, değişken adı ve değeri olmak üzere iki kısımdan oluşurlar. Basit değişken tipleri; sayısal, metin ve boolean tipindedir. Aşağıda "Engel" adında bir değişkenin oluşturulması işlemi gösterilmiştir.

mBlock'ta değişkenler "Veri&Bloklar" kategorisinde "Bir Değişken Oluştur" seçeneğinde bulunmaktadır. "Bir Değişken Oluştur" butonuna tıklandıktan sonra, değişken için bir ad girilmelidir.	
Değişkenler programın sol tarafında listelenmektedir. Eklenen her bir değişken için değer atama blokları otomatik olarak eklenmektedir.	
Değer atama bloklarından hangi değişkene değer atanacağı aşağı okuyla seçilmektedir.	

Tablo 6.6: Değişken oluşturulması aşamaları

Değişken Oluşturma Örneği: Bu örnekte robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 30 cm'den büyük ise robot her tıklamada 100 rpm hıza göre 0.2 saniye ileri doğru ve 0.2 saniye geriye doğru 2 defa harekat ettikten sonra durmaktadır. Engele olan uzaklık 30 cm'den küçük ise 0.65 saniye sağa dönüp durmaktadır. Örnekte engele olan uzaklık için "mesafe" adında bir değişken oluşturulmuştur. "Bir Değişken Oluştur" butonuna tıklandıktan sonra, değişken için "mesafe" ad olarak girilmiştir. "Mesafe" değişkeni için değer atama blokları bu aşamada otomatik olarak eklenmiştir. Bu bloklar program içinde kullanılarak,

Resim 6.24: Değişken oluşturma örneği

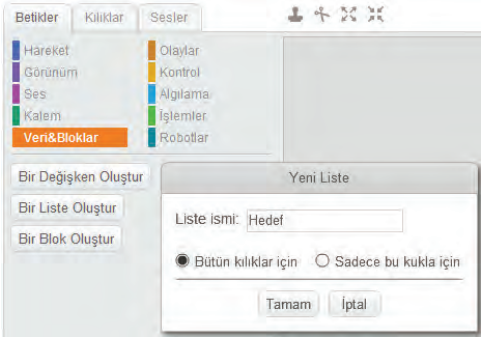
ultrasonik algılayıcı mesafesi "mesafe" değişkeni ile ifade edilmiştir. Uzaklık şartı olan 30 cm "işlemler" kategorisinde bulanan  işlem bloğu kullanılarak yazılmış ve "mesafe" değişkenine eşlenmiştir.

Listeler, Diziler ve Oluşturulması: Çok sayıda değişkenle çalışmak için oluşturulmuş temel yapılarından biridir. Listeler değişkenlerden farklı olarak birden fazla değer taşırlar. Dizilerse köşeli parantez içinde virgülle ayrılmış değerler taşır. Aşağıda "Hedef" adında bir değişkenin oluşturulması işlemi gösterilmiştir.

mBlock'ta listeler "Veri&Bloklar" kategorisinde "Bir Liste Oluştur" seçeneğinde bulunmaktadır. "Bir Liste Oluştur" butonuna tıklandıktan sonra, liste için bir ad girilmelidir.

Liste değişkenleri ekranın sol tarafında listelenmektedir. Listeyle yapılabilecek işlemlere ait bloklar otomatik olarak eklenmektedir.

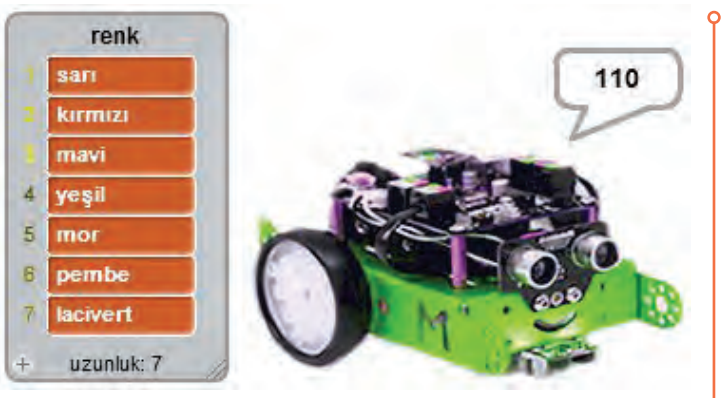
Liste sol üst köşede yer almaktadır. Diziye değerler bu tablonun sol alt köşesindeki + işaretine tıklayarak oluşan kutucuğa yazıyla girilebilir. İstenirse program akışı içerisinde de listeye değer eklenip çıkarılabilmektedir.

mBlock'ta listeler "Veri&Bloklar" kategorisinde "Bir Liste Oluştur" seçeneğinde bulunmaktadır. "Bir Liste Oluştur" butonuna tıklandıktan sonra, liste için bir ad girilmelidir.t	
Liste değişkenleri ekranın sol tarafında listelenmektedir. Listeyle yapılabilecek işlemlere ait bloklar otomatik olarak eklenmektedir.	

Liste sol üst köşede yer almaktadır. Diziye değerler bu tablonun sol alt köşesindeki + işaretine tıklayarak oluşan kutucuğa yazıyla girilebilir. İstenirse program akışı içerisinde de listeye değer eklenip çıkarılabilmektedir.	

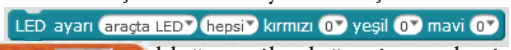
Tablo 6.7: Liste ve dizi oluşturulması aşamaları

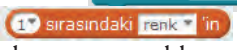
Liste Oluşturma Örneği: Bu örnekte robotun üzerinde bulunan ışık algılayıcısı kullanılarak, ortamdaki ışık miktarına göre robotun renk değiştirmesi, aynı zamanda oluşturulan renk listesindeki renklere göre mBot üzerinde bulunan RGB LED'lerin sıra ile yanması ve mBot üzerinde ışık seviyesinin konuşma balonu şeklinde görülmesi sağlanmıştır. Bu amaçla; "Görünüm" kategorisinde bulunan Merhaba ! de bloğu seçilerek buraya "Merhaba" yerine "Robotlar" kategorisinde bulunan ışık algılayıcı araçta ışık sensörü bloğu yerleştirilmiştir. Böylece ışık algılayıcının kullanılması sağlanmıştır. Yine "Görünüm" kategorisinde bulunan renk etkisini 25 arttır bloğu seçilerek üzerine "İşlemler" kategorisinde bulanan ○/○ işlem bloğu 10/100 oranıyla yerleştirilmiş, bunun üzerine de ışık algılayıcı araçta ışık sensörü bloğu yerleştirilerek renk etkisi artırılarak araç gövdesinin ışık miktarına göre renk değiştirmesi sağlanmıştır. "Veri&Bloklar" kategorisinde bulunan "Bir Liste Oluştur" seçeneği



Resim 6.25: Liste oluşturma örneği ekran görüntüsü

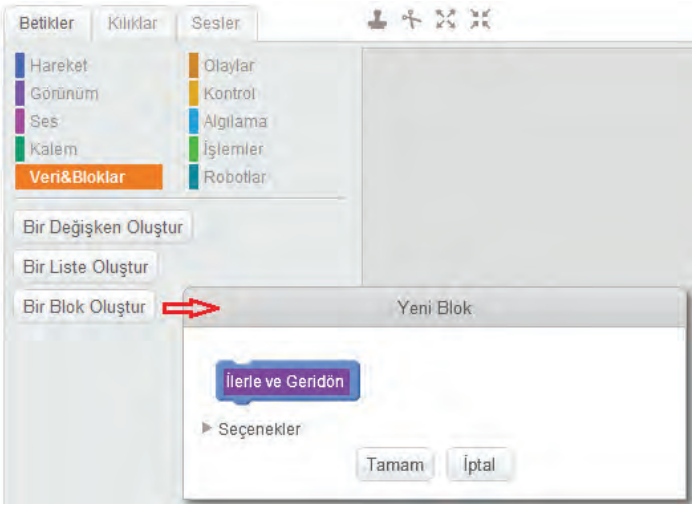
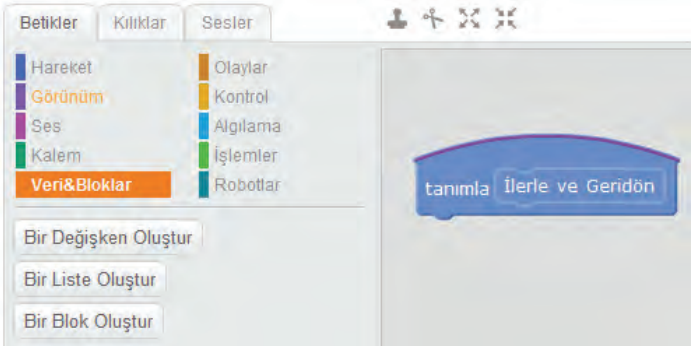
Resim 6.26: Liste oluşturma örneği

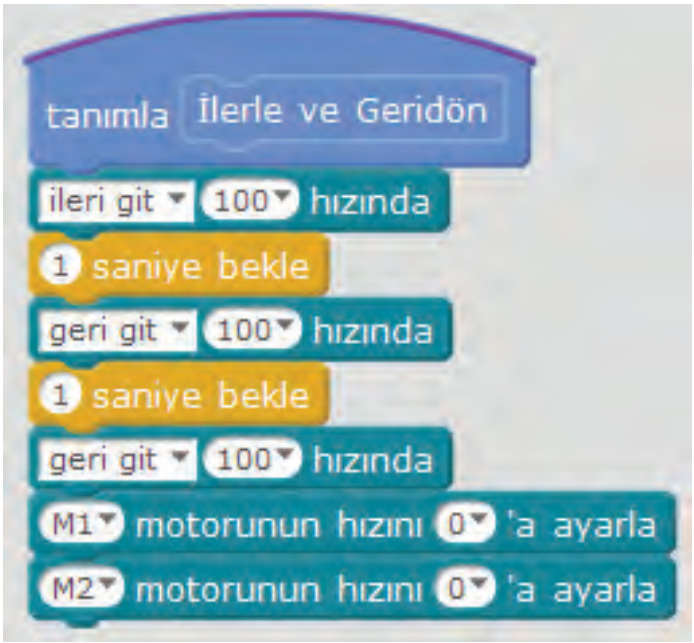
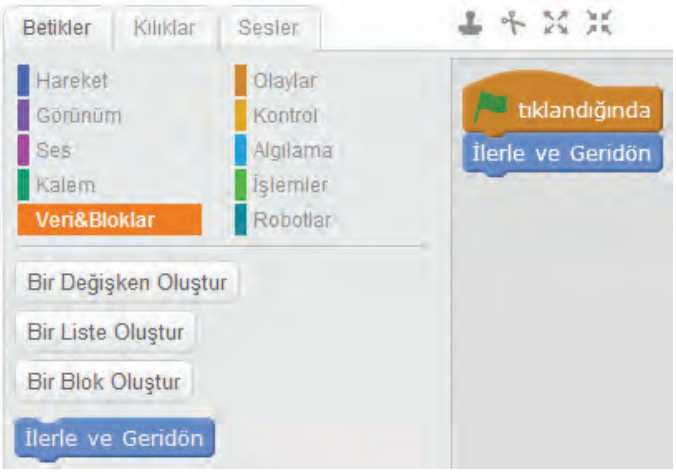
kullanılarak "renk" adını taşıyan bir liste yapılmıştır. "renk" listesini oluşturan değişkenler yazılım tarafından programın sol tarafında otomatik olarak listelenmiştir. "renk" adlı listeyle yapılabilecek işlemlere ait bloklar yine yazılım tarafından otomatik olarak eklenmiştir. Liste sol üst köşede tablo şeklinde yer almıştır. Buraya değerler (renk adları) tablonun sol alt köşesindeki + işaretine tıklayarak oluşan kutucuğa yazılarak girilmiştir. "Robotlar" kategorisinde bulunan 

bloğu seçilerek "hepsi" yerine, oluşturulan listeden 

bloğu seçilerek üzerine yerleştirilmiştir. Bu işlem listedeki 7 renk için tekrarlanarak programa eklenmiştir. Renkler için RGB kodları girilerek LED'lerin listede belirtilen renkte yanması sağlanmıştır.

Blok (Prosedür) Oluşturma: Kodu bir kez yazıp defalarca kullanmak için ortaya konmuş temel yapılardır. Program akışı içinde tekrarlayan ifadelerin her seferinde tekrar tekrar yazılması yerine, bir kere ayrı bir yerde yazılıp tekrarlanan her yerde kullanmak için uygundurlar. Aşağıda "İlerle ve Geridön" adında bir blok oluşturulması işlemi gösterilmiştir. Burada bu blokun oluşturulmasıyla yapılmak istenilen, bir işlem gurubunu tek bir blok olarak belirleyip her seferinde aynı blokların ayrı ayrı kullanılmasını ortadan kaldırmaktır. Örneğimizde robotun önce ileri gidip sonra geri gelme işlemi 7 adımdan oluşmaktadır ancak İlerle ve Geri Dön bloğu oluşturulduğunda bunu tek bir blokla gerçekleştirmek mümkün olmaktadır.

mBlock'ta blok oluşturulması "Veri&Bloklar" kategorisinin altında "Bir Blok Oluştur" seçeneğinde bulunmaktadır. "Bir Blok Oluştur" butonuna tıklandıktan sonra, oluşan "Yeni Blok" penceresindeki isimsiz alana blok için bir ad girilmeli ve onaylanmalıdır.	
Programda kullanılacak ve çağrılabilecek blok otomatik olarak eklenmektedir.	

Bu şekilde blok oluşturulması yapıldıktan sonra blok tarafından yapılacak işlemlerin blok altında tanımlanması gerekmektedir. Örnekte robotu 1 sn ileri ve 1 sn geri getirdikten sonra durmasını sağlayan bir blok oluşturulmuştur.	
Blok kullanılmak istendiğinde ana programdan çağrılmaktadır.	

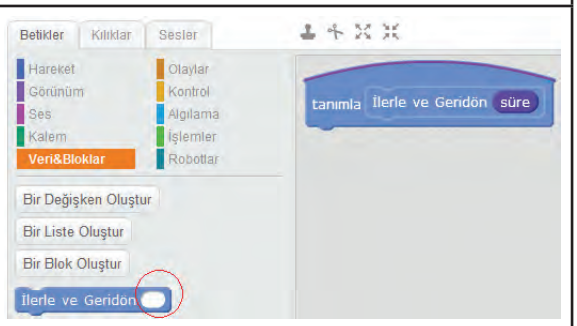
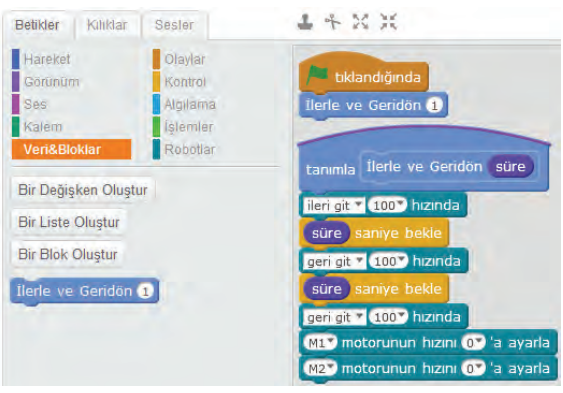
Tablo 6.8: Blok oluşturma aşamaları

Blok (Prosedür) Oluşturma Örneği: Bu örnekte robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 10 cm'den büyük ise program çalışmaktadır. Engele olan uzaklık 10 cm'den büyük ise her tıklamada 100 rpm hıza göre 1 saniye ileri doğru 10 cm kalıncaya kadar gitmektedir. Koşul sağlanınca "Dur" prosedürü ile robot durmaktadır. Bu prosedür "Veri&Bloklar" kategorisinde bulunan "Bir Blok Oluştur" seçeneği ile oluşturulmuş ve prosedür tanımlanması "Dur" şeklinde yapıldıktan sonra kullanacağımız işlemler prosedürün al-

Resim 6.27: Blok oluşturma örneği

tında M1 ve M2 motorlarının hızlarını 0'a ayarla şeklinde tanımlanmıştır. Bu işlemlerin tanımlanması için "Robotlar" kategorisinde bulunan yandaki blok kullanılmıştır. Bloğun bir kopyası oluşturularak M1 ve M2 seçenekleri seçilmiştir. Uzaklık şartı olan 10 cm "İşlemler" kategorisinde bulanan  işlem bloğu kullanılarak yazılmıştır.

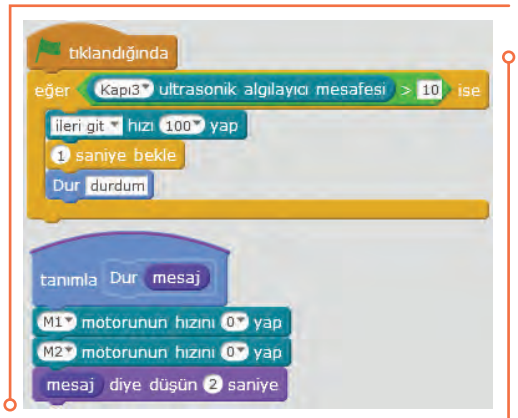
Oluşturulan Blok (Prosedür) İçin Parametre Tanımlanması: Prosedürlere değer taşıyan değişkenlere parametre adı verilir. Bir prosedür çağrıldığı zaman aynı zamanda parametre değerlerini de vermek gerekmektedir. Oluşturulan "İlerle ve Geridön" bloğu bu anlamda bir prosedürdür. Bu prosedüre parametre ekleme işlemi aşağıdaki örnekte açıklanmıştır.

mBlock'ta oluşturulan bloka parametre tanımlanması için "Bir Blok Oluştur" seçeneğiyle açılan "Yeni Blok" penceresinde blok oluşturulup ad verilmesinden sonra yine bu pencerede bulunan "Seçenekler" düğmesine tıklanmalıdır. Uygun olan sayısal veya sözel parametre tipi seçildikten sonra parametre adı girilmelidir.	
Yeni tanımlanan bu prosedürde yapılacak işler parametre değeri ile birlikte tanımlanır.	
Hazırlanan prosedüre parametre değeri vererek kullanılır. Bunun için prosedür bloğu içinde tanımlanan kutucuğun içerisine doğrudan bir değer girerek ya da bir değişken atayarak kullanılır. Girilen bu değer prosedüre aktarılacak ve bu değerle işlem yapılacaktır.	

Tablo 6.9: Prosedüre parametre ekleme aşamaları

Parametre Örneği: Bu örnekte robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 10 cm'den büyük ise program çalışmaktadır. Engele olan uzaklık 10 cm'den büyük ise her tıklamada 100 rpm hıza göre 1 saniye ileri doğru, engele 10 cm kalıncaya kadar gitmektedir. Koşul sağlanınca oluşturulan "Dur" değişkeni ile robot durmaktadır. Bu aşamaya kadar

olan işlemler için yandaki uygulama örneği kullanılmıştır. Uzaklık şartı olan 10 cm "İşlemler" kategorisinde bulanan  işlem bloğu kullanılarak yazılmıştır.



Resim 6.28: Parametre örneği

Resim 6.29: Parametre örneği ekran görüntüsü

Bu prosedüre parametre tanımlamak için "Dur" değişkenine sağ tıklanmış ve açılan "Düzenle" düğmesinde bulunan "Seçenekler" içerisinden "Sözel girdi ekle" kullanılarak parametre tipi seçilmiştir. Prosedür bloğu içinde tanımlanan kutucuğun içerisine "durdum" yazılarak parametre tamamlanmıştır. Program çalıştırılıp koşul sağlanınca programdaki kukla "durdum" ifadesini 2 saniye boyunca ekrana yazmaktadır.

6.6.6. Olaylar Alt Başlığı Altında Verilen Komut Blokları

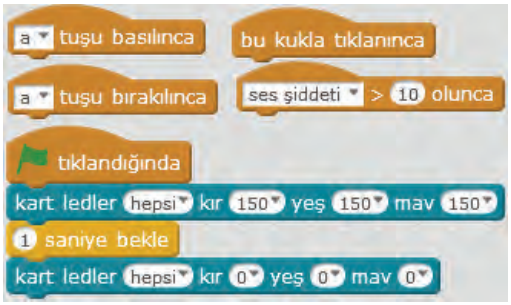
Olaylar alt başlığı altında verilen bloklar sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robot için oluşturulan programların ve diğer uygulamaların çalıştırılmasında kullanılmaktadır. Buradaki blokların kullanılmasıyla bir tuşun basılı olup olmaması veya seçilen nesnenin aktif olup olmaması durumuna göre uygulamanın çalışması sağlanabilir. Belirlenen bir haberin gelmesi durumunda yapılacak işlemler kontrol edilebilir. Olaylar alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

tıklandığında	Herhangi bir uygulamanın tıklandığında çalışması için kullanılır.
boşluk tuşu basılınca	Herhangi bir uygulamanın belirlenen tuşa basılmasıyla çalışması için kullanılır.
boşluk tuşu bırakılınca	Herhangi bir uygulamanın belirlenen tuşun bırakılmasıyla çalışması için kullanılır.
bu kukla tıklanınca	Herhangi bir uygulamanın kuklaya tıklanmasıyla çalışması için kullanılır.
dekor dekor1 olunca	Herhangi bir uygulamanın belirlenen dekorun kullanılmasıyla çalışması için kullanılır.

ses şiddeti > 10 olunca	Herhangi bir uygulamanın ses şiddeti, süre ölçer veya video hareketi belirlenen değerden büyük olunca çalışması için kullanılır.
message1 haberi gelince	Herhangi bir uygulamanın belirtilen ileti veya belirtilen yeni ileti gelince çalışması için kullanılır.
message1 haberini sal	Herhangi bir uygulamada belirtilen ileti veya belirtilen yeni ileti gelince açıklanması için kullanılır.
message1 haberini sal ve bekle	Herhangi bir uygulamada belirtilen ileti veya belirtilen yeni ileti gelince açıklanması ve beklenmesi için kullanılır.

Tablo 6.10: Olaylar alt başlığı altında verilen komut blokları

Olay Örneği: Robot kontrol kartı üzerindeki RGB LED'lerin rengini kontrol edebilen bu uygulama tıklandığında, a tuşu basılınca, a tuşundan el çekince, bunun için oluşturulmuş kuklaya tıklanınca veya ses şiddeti 10'dan büyük olunca çalıştırılabilir. Bunun için yapılması gereken tek şey istenilen bloku tıklandığında blokuyla yer değiştirmektir.



Resim 6.30: Olay örneği

6.6.7. Kontrol Alt Başlığı Altında Verilen Komut Blokları

Kontrol alt başlığı altında verilen bloklar sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robot için hazırlanan programların ve diğer uygulamaların oluşturulmasında kullanılan temel programlama komutlarından oluşmakta ve tüm programlama uygulamalarında kullanılmaktadır. Buradaki blokların kullanılmasıyla birden fazla yapılması gereken döngüsel işlemler gerçekleştirilebilir. Koşullu durumunda gerçekleştirilmesi gereken işlemler yapılabilir. Ayrıca çalışan tüm kodların durdurulması ve nesnenin ikizi ile ilgili işlemlerin yapılması sağlanabilir. Robotik uygulamalarda, robot hareketinin durdurulması için "durdur" bloğu kullanılamamaktadır. Bu durumlara motor hızlarının sıfırlanarak durdurulması pratik bir çözüm olabilir. Aşağıda verilen örnekleri inceleyip uygulayınız. Kontrol alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

1 saniye bekle	Programın belirtilen saniye kadar beklemesi için kulanılır.
10 defa tekrarla	Döngüler tekrarlanan işleri yapmak için kullanılan temel yapılardır. Bu döngü bloğu belirtilen sayı kadar işlemi tekrarlamak için kullanılır.
sürekli tekrarla	Verilen işlemi sürekli tekrarlamak için kullanılan döngü bloğudur.
eğer ise	Olumlu koşul ifadesi "eğer" "ise" koşulu gerçekleşene kadar işlemi tekrarlamak için kullanılır. Döngü bloğunda bir koşul tanımlanır ve o koşul gerçekleşene kadar döngü devam eder.

eğer ise değilse	Olumlu koşul ifadesi "eğer" "ise" ve olumsuz koşul ifadesi "değilse" koşulu gerçekleşene kadar işlemi tekrarlamak için kullanılır.
olana kadar bekle	Belirtilen koşul gerçekleşene kadar işlemi bekletmek için kullanılır.
olana kadar tekrarla	Belirtilen koşul gerçekleşene kadar işlemi tekrarlamak için kullanılır. Koşul komutları program akışını farklı durumlara göre değiştirmek, yönlendirmek için kullanılan temel karar yapılardır.
hepsi durdur	Çalışan betik, kuklanın diğer betikleri veya hepsinde işlemi durdurmak için kullanılır.
ikiz olarak başladığımda	Program ikiz olarak başlatıldığında kullanılır.
kendim in ikizini oluştur	Programcının kendisi veya kullanılan kukla tarafından ikizinin oluşturulması için kullanılır.
bu ikizi sil	Oluşturulan ikizin silinmesi için kullanılır.

Tablo 6.11: Kontrol alt başlığı altında verilen komut blokları

6.6.7.1. Kontrol Örnekleri-Döngüler

Verdiğimiz Sayı Kadar İşlemi Tekrarlayan Döngü Örneği: Bu örnekte robot 100 rpm hıza göre 1 saniye ileri, 1 saniye de geri hareket etmekte ve toplamda bunu 2 defa tekrarlamaktadır. Tekrarın sonunda geri gelme hızını sıfırlayarak durmaktadır.

Resim 6.31: Verdiğimiz sayı kadar işlemi tekrarlayan döngü örneği

İşlemi Sürekli Tekrarlayan Döngü Örneği: Bu örnekte robot 100 rpm hıza göre 1 saniye ileri, 1 saniye geri hareket etmekte ve bu işlemi sürekli olarak tekrarlamaktadır.

Resim 6.32: İşlemi sürekli tekrarlayan döngü örneği

Belirli Bir Şart Gerçekleşene Kadar Bekleme Örneği: Bu örnekte robot ses şiddeti 10'dan büyük olunca çalışmaya başlamaktadır. Robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 20 cm'den büyük ise robot beklemeye geçmektedir. Eğer engele olan uzaklık 20 cm'den küçük olursa (örneğin elinizi ultasonik algılayıcıya yaklaştırırsanız) koşul gerçekleşince 100 rpm hıza göre 1 saniye ilerleyip M1 ve M2 motorlarının hızlarını sıfıra düşürmektedir. Uzaklık şartı olan 20 cm "İşlemler" kategorisinde bulanan  işlem bloğu kullanılarak yazılmıştır.

Resim 6.33: Belirli bir şart gerçekleşene kadar bekleme örneği

Belirli Bir Şart Gerçekleşene Kadar Döngü Örneği: Bu örnekte robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 20 cm'den büyük ise program çalışmaktadır. Engele olan uzaklık 20 cm'den küçük olana kadar 100 rpm hıza göre 1 saniye 1 saniye ilerleyip koşul gerçekleşince M1 ve M2 motorlarının hızlarını sıfıra düşürmektedir. Uzaklık şartı olan 20 cm "İşlemler" kategorisinde bulanan  işlem bloğu kullanılarak yazılmıştır.

Resim 6.34: Belirli bir şart gerçekleşene kadar döngü örneği

6.6.7.2. Kontrol Örnekleri -Koşullar

Olumlu Koşul İfadesi "eğer" "ise" Örneği: Bu örnekte robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 20 cm'den küçük ise program çalışmaktadır. Engele olan uzaklık 20 cm'lik alan içerisinde kalıncaya kadar her tıklamada 100 rpm hıza göre 1 saniye

gerileyip, M1 ve M2 motorlarının hızlarını sıfıra düşürmektedir. Uzaklık şartı olan 20 cm "İşlemler" kategorisinde bulanan  işlem bloğu kullanılarak yazılmıştır.

Resim 6.35: Olumlu koşul ifadesi "eğer" "ise" örneği

Olumlu Koşul İfadesi "eğer" "ise" ve Olumsuz Koşul İfadesi "değilse" Örneği: Bu örnekte de robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 10 cm'den büyük ise her tıklamada 100 rpm hıza göre 1 saniye ileri doğru 10 cm kalıncaya kadar gitmektedir. Eğer engele olan uzaklık 10 cm'den küçük ise robot geriye doğru 10 cm oluncaya kadar 100 rpm hıza 1 saniye boyunca çalışmaktadır. Koşul sağlanınca M1 ve M2 motorlarının hızlarını sıfıra düşürmektedir. Uzaklık şartı olan 10 cm "İşlemler" kategorisinde bulanan  işlem bloğu kullanılarak yazılmıştır.

Resim 6.36: Olumlu koşul ifadesi "eğer" "ise" ve olumsuz koşul ifadesi "değilse" örneği

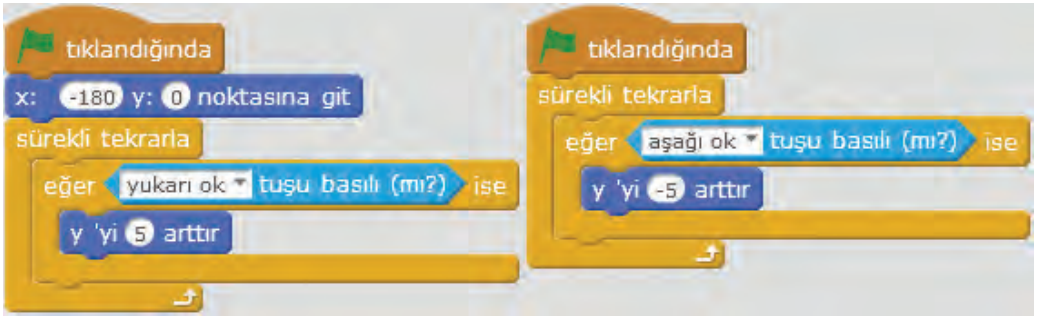
6.6.8. Algılama Alt Başlığı Altında Verilen Komut Blokları

Algılama alt başlığı altında verilen bloklar sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robot için hazırlanan programların ve diğer uygulamaların oluşturulmasında ve tüm uygulama programlarında kullanılmaktadır. Buradaki blokların kullanılmasıyla bir nesneye, bir renge değme durumu ya da fareye olan mesafe algılanabilir. Mesaj vermek için kullanılabilir. Klavye ve fare kullanımına dayalı işlemler yapılabilir. Bilgisayara bağlı kameranın aktif olması ve sanal robotun video hareketlerine göre değişiklik göstermesi sağlanabilir. Ayrıca nesnelerin konumunu algılama işlemleri de yapılabilmektedir. Algılama alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

a değdi (mi?)	Kuklanın fare okuna, sağ, sol, üst veya alt kenara değdiğini belirlemek için kullanılır.
rengine değdi (mi?)	Kuklanın belirlenen herhangi bir renge değdiğini belirlemek için kullanılır.
rengi rengine değdi (mi?)	Kuklanın belirlenen herhangi bir renginin belirlenen alanda herhangi bir renge değdiğini belirlemek için kullanılır.
'a mesafe	Fare okuna, kuklaya, rasgele yatay, dikey ve sahne noktasına olan mesafeyi belirlemek için kullanılır.
İsmin ne? diye sor ve bekle	Belirtilen ifadeyi (ad, isim, yer vs.) sormak ve beklemek için kullanılır.
yanıt	Yanıtını ekranda almak için kullanılır.
boşluk tuşu basılı (mı?)	Belirtilen tuşun basılı olup olmadığını belirlemek için kullanılır.
fareye basılı mı?	Farenin tuşunun (aktif tuş) basılı olup olmadığını belirlemek için kullanılır.
farenin x'i	Farenin x koordinatını belirlemek için kullanılır.
farenin y'si	Farenin y koordinatını belirlemek için kullanılır.
ses şiddeti	Ses şiddetini ekranda görmek için kullanılır.
video hareket on bu kukla	Kullanılan kukla veya sahne üzerinde video hareketi ve yönü belirtmek için kullanılır.
videoyu aç	Videoyu açmak, kapatmak ve açıp solu sağ yapmak için kullanılır.
video saydamlığı % 50 olsun	Video saydamlığını belirtilen orana ayarlamak için kullanılır.
süre ölçer	Süre ölçeri ekran üzerinde açmak için kullanılır.
süre ölçeri sıfırla	Süre ölçeri sıfırlamak için kullanılır.
x konumu değeri mBot 4_1_1 'in	x, y konumu, yönü, kılık no, kılığın ismi, büyüklük ve ses şiddeti değerleri mevcut kukla veya sahne için kullanılır.
şimdiki dakika	Ekranda saniye, dakika, saat, haftanın günü, tarih, ay ve yıl bilgilerini göstermek için kullanılır.
2000'den beri geçen gün	2000 yılından beri geçen gün sayısını belirlemek için kullanılır.

Tablo 6.12: Algılama alt başlığı altında verilen komut blokları

Algılama Örneği: Bu örnekte yukarı ve aşağı ok tuşları kullanılarak sanal robotun dikey yönde aşağı veya yukarı hareket etmesi sağlanmıştır. Program çalıştırıldığında robot x=-180, y=0 konumuna gitmektedir. Yukarı ve aşağı hareket y değerinin artırılıp azaltılmasıyla sağlanmaktadır.



Resim 6.35: Algılama örneği

Bu örnekten yararlanarak her yöne hareket edebilen bir sanal robot tasarlayınız.

6.6.9. İşlemler Alt Başlığı Altında Verilen Komut Blokları

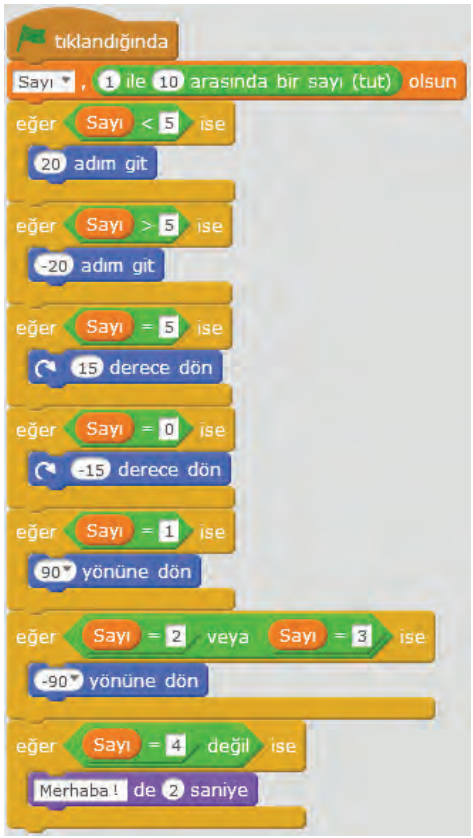
İşlemler alt başlığı altında verilen bloklar sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robot için hazırlanan programların ve diğer uygulamaların oluşturulmasında ve tüm uygulama programlarında kullanılmaktadır. İşlemler bloğu matematiksel işlemlerin bulunduğu bloktur. Dört işlem gerçekleştirme, iki değer arasında rastgele değer üretme, karşılaştırma yapma, birden fazla durumu veya bir durumu seçmek veya seçmemek için kullanılmaktadır. Ayrıca iki farklı ifadeyi birleştirme, karakter uzunluğu belirtme işlemleri ve basit matematiksel hesaplamalar (mod alma, yuvarlama, karekök alma gibi) yapılabilmektedir. İşlemler alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

+	Toplama işlemleri için kullanılır.
-	Çıkarma işlemleri için kullanılır.
*	Çarpma işlemleri için kullanılır.
/	Bölme işlemleri için kullanılır.
1 ile 10 arasında bir sayı (but)	Belirtilen değerler arasında rasgele sayı üretmek için kullanılır.
<	"<" karşılaştırma işlemleri için kullanılır.
=	"=" karşılaştırma işlemleri için kullanılır.
>	">" karşılaştırma işlemleri için kullanılır.
ve	Mantıksal işlemler için kullanılır. Her iki koşul da doğru olduğunda "ve" yapı taşı doğru olacaktır. Aksi takdirde yanlış olur.

veya	Mantıksal işlemler için kullanılır. Her iki durumdan biri doğru olduğunda, "veya" yapı taşı doğrudur. Aksi takdirde yanlış olur.
değil	Mantıksal işlemler için kullanılır. Matıksal işlem "değil" ise yapı taşı doğrudur.
hello ile world i birleştir	Belirtlen iki kelimeyi birleştirmek için kullanılır.
1 . harfi world in	Belirtilen kelimenin belirtilen harfiyla işlem yapmak için kullanılır.
world in uzunluğu	Belirtilen kelimenin uzunluğuyla işlem yapmak için kullanılır.
index of world in hello world	Belirtilen kelime içinden alanan kelimenin indeksi ile işlem yapmak için kullanılır.
oyuncu 1 i dizgeye	Belirtilen oyuncuyu dizgeye taşımak için kullanılır.
modüler	Mod işlemleri yapmak için kullanılır.
i yuvarla	Belirtilen değeri yuvarlamak için kullanılır.
karekök 9 nin	Belirtilen sayının karakökünü, mutlak değerini almak, aşağı veya yukarı yuvarlamak ve açısal değerleri için kullanılır.

Tablo 6.13: İşlemler alt başlığı altında verilen komut blokları

İşlem Örneği: Bu örnekte 1 ile 10 arasında rastgele oluşturulan sayıya göre kuklanın hareketi sağlanmıştır. Bu amaçla "Sayı" adında bir değişken oluşturulmuştur. Tutulan sayı 5'ten küçükse kukla 20 adım ilerlemektedir. Eğer tutulan sayı 5'ten büyükse kukla 20 adım geri gitmektedir. Eğer sayı 5'e eşitse 15 derece dönmekte, eğer sayı 0'a eşitse -15 derece dönmekte, eğer sayı 1'e eşitse 90 (sağa) yönüne dönmektedir. Eğer sayı 2 veya 3 ise -90 (sola) yönüne dönmektedir. Eğer sayı 4 değilse ekrana 2 sn boyunca Merhaba! şeklinde yazmaktadır.



Resim 6.38: İşlem örneği

6.6.10. Robotlar Alt Başlığı Altında Verilen Komut Blokları

Robotlar alt başlığı altında verilen bloklar Arduino uyumlu kartlar için uygulama programları hazırlanmasında, fiziksel bir robot için programların hazırlanmasında, Makeblock tarafından üretilen robot ve robot kontrol kartları ve kalkanların (shield) programlanmasında ve diğer tüm donanım tabanlı programlama uygulamalarında kullanılmaktadır. Kullanılan kart türüne göre desteklenen bloklar değişmekte olup aşağıda

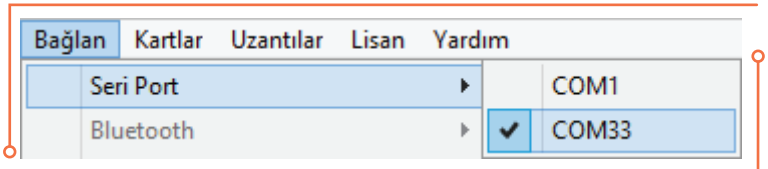
Arduino uyumlu kartlar için kullanılabilecek bloklar verilmiştir. Robotlar alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

Arduino Programı	Arduino programları için kullanılır.
9 sayısal pini oku	Belirtilen sayısal pini okumak için kullanılır.
(A) 0 analog pini oku	Belirtilen analog pini okumak için kullanılır.
13 darbe pini oku, zaman aşımı 20000 olsun	Belirtilen PWM pinini, verilen zaman aşımı içinde okuması için kullanılır.
9 sayısal pini YÜKSEK yap	Belirtilen sayısal pini LOW veya HIGH yapmak için kullanılır.
5 pwm pini 0 yap	Belirtilen PWM pinini, 0,50,100,150 veya 255 değerlerine ayarlamak için kullanılır.
9 ses tonu pini C4 notasında Yarım vuruş çal	Belirtilen ses tonu pini, istenilen notada, istenilen vuruş kadar çalması için kullanılır.
9 servo pini açısını 90 yap	Belirtilen servo pini açısını 0, 45, 90, 135 veya 180 derece yapmak için kullanılır.
seri porta merhaba yaz	Verilen ifadeyi seri porta yazdırmak için kullanılır.
seri portta byte var	Seri porttan gelen byte'ı tespit için kullanılır.
seri porttan byte oku	Seri porttan gelen byte'ı okumak için kullanılır.
ultrasonik 13 tetik pini 12 okuma pini	Ultrasonik sensörün tetikleme ve okuma pinini belirtmek için kullanılır.
süre ölçer	Süre ölçümü için kullanılır.
süre ölçeri sıfırla	Süre ölçeri sıfırlamak için kullanılır.

Tablo 6.14: Robotlar alt başlığı altında verilen komut blokları

6.7. mBlock ile Arduino Kullanımı

Arduino kullanım örnekleri UNO R3 üzerinden verilmiştir. mBlok üzerinden Arduino kullanımı için "Bağlan" sekmesinden Arduino'nun bağlandığı Seri Port, ayrıca "Kartlar" sekmesinden de "Arduino Uno" seçilmelidir.

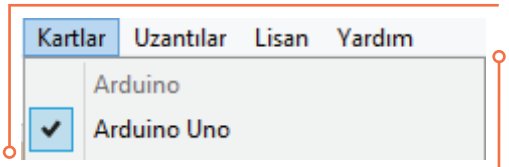


Resim 6.39: Arduino kullanımı için port seçimi

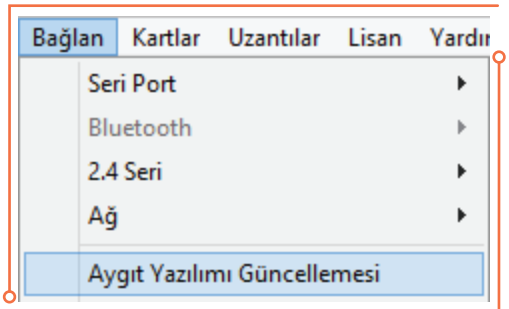
Arduino ile kullanılacak elektronik bileşenlerin önceden hazırlanması, bağlantılarının yapılması sonra da uygulamanın hazırlanarak Arduino'ya yüklenmesi gerekmektedir. Yüklemek için "Arduino Prog-

ramı” 

blokunun üzerine tıklanarak açılan ekrandan “Arduinoya Yükle” seçeneğinin tıklanması gerekmektedir. Bu durumda uygulama ekranın sağında Arduino kodları şeklinde görülecek ve yükleme bitince “Yükleme Bitti” şeklinde uyarı verecektir. Bu şekildeki kullanımda sadece Arduino için oluşturulmuş bloklar kullanılabilir. Yüklenen uygulama mBlock olmadan Arduino üzerinde kullanılabilir. Fakat istenirse “Bağlan” sekmesinden “Aygıt Yazılımı Güncellemesi” seçeneği ile Arduino kullanım kodları yüklenerek Arduino blokları ve diğer bloklar birlikte kullanılabilir. Bu durumda hazırlanan uygulama Arduino hafızasında yer almayacak yalnızca mBlock üzerinden kullanılabilecektir.

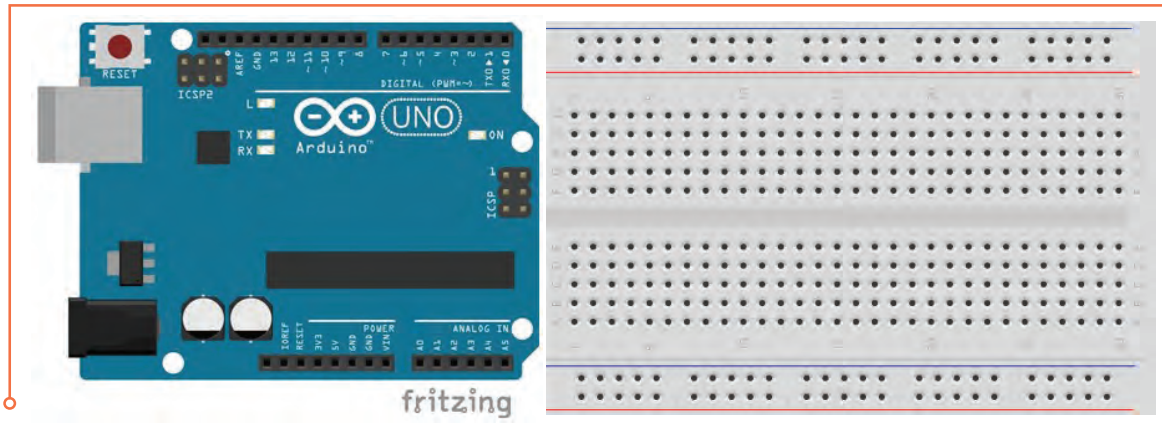


Resim 6.40: Arduino Uno seçimi



Resim 6.41: Aygıt yazılımının güncellenmesi

Arduino uygulamalarında kullanılacak elektronik bileşenler için Breadboard kullanılması uygulamaların hızlı, kolay ve en önemlisi lehim yapmadan yapılmasına olanak tanıyacaktır. Breadboardların kenarında bulunan alanlar voltaj bağlantıları için enine bağlı, diğer alanlar ise dikine bağlıdır. Bu bağlantı noktalarını kullanarak LED, direnç ve benzeri elektronik bileşenlerin birbirine bağlanması oldukça kolaydır. Aşağıda Arduino UNO R3 ve bir breadboard örneği yer almaktadır.

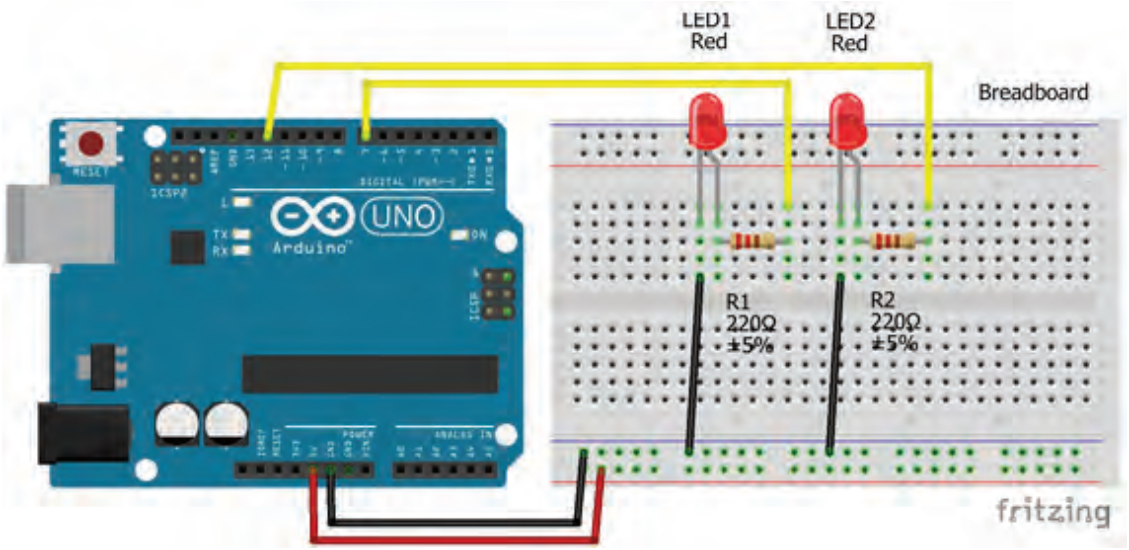


Resim 6.42: Arduino UNO R3 ve Breadboard örneği

Dijital Kontrol Pimlerin Ayarlanması ve Okunması: Arduino UNO kartında toplamda 14 dijital kontrol pimi (6 adet PWM dâhil) bulunmaktadır. Dijital pinlerin çıkış değerini 0 (DÜŞÜK) ya da 1 (YÜKSEK) olarak ayarlamak ve aynı zamanda, dijital pinlerin giriş değerlerini ekranda okumak için yandaki örneği inceleyiniz.

Resim 6.43: Dijital kontrol pimlerin ayarlanması ve okunması

Aşağıda verilen iki aynı örnekte Arduino’nun 7 ve 12 numaralı dijital pinlerine bağlı 2 LED’in 1 sn aralıklarla yanıp sönmesi sağlanmıştır. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir. LED’ler düşük güçle çalıştıkları ve zarar görmemeleri için birer 220 ohm dirençle bağlanmıştır.

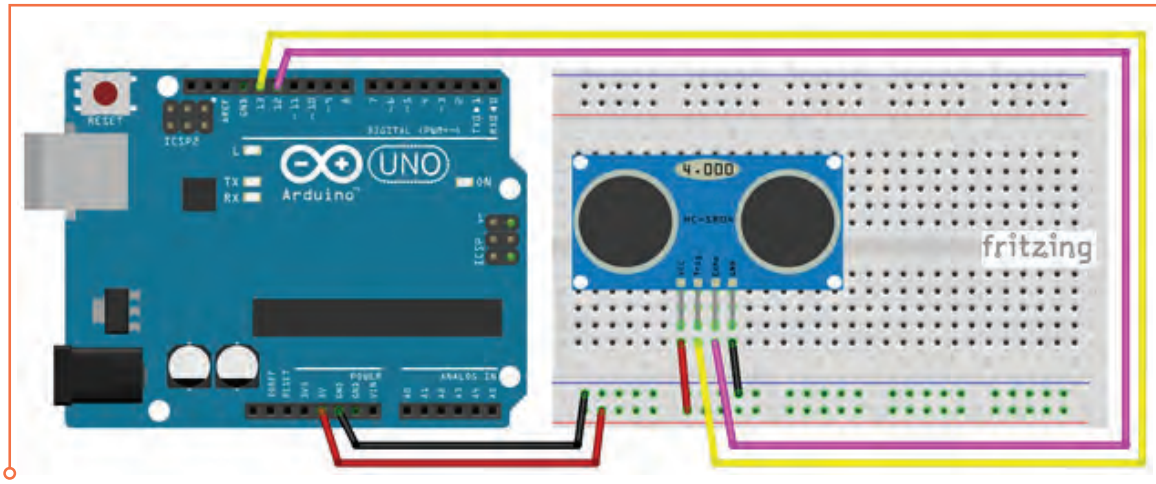


Resim 6.44: Arduino UNO ile LED uygulaması Breadboard çizimi

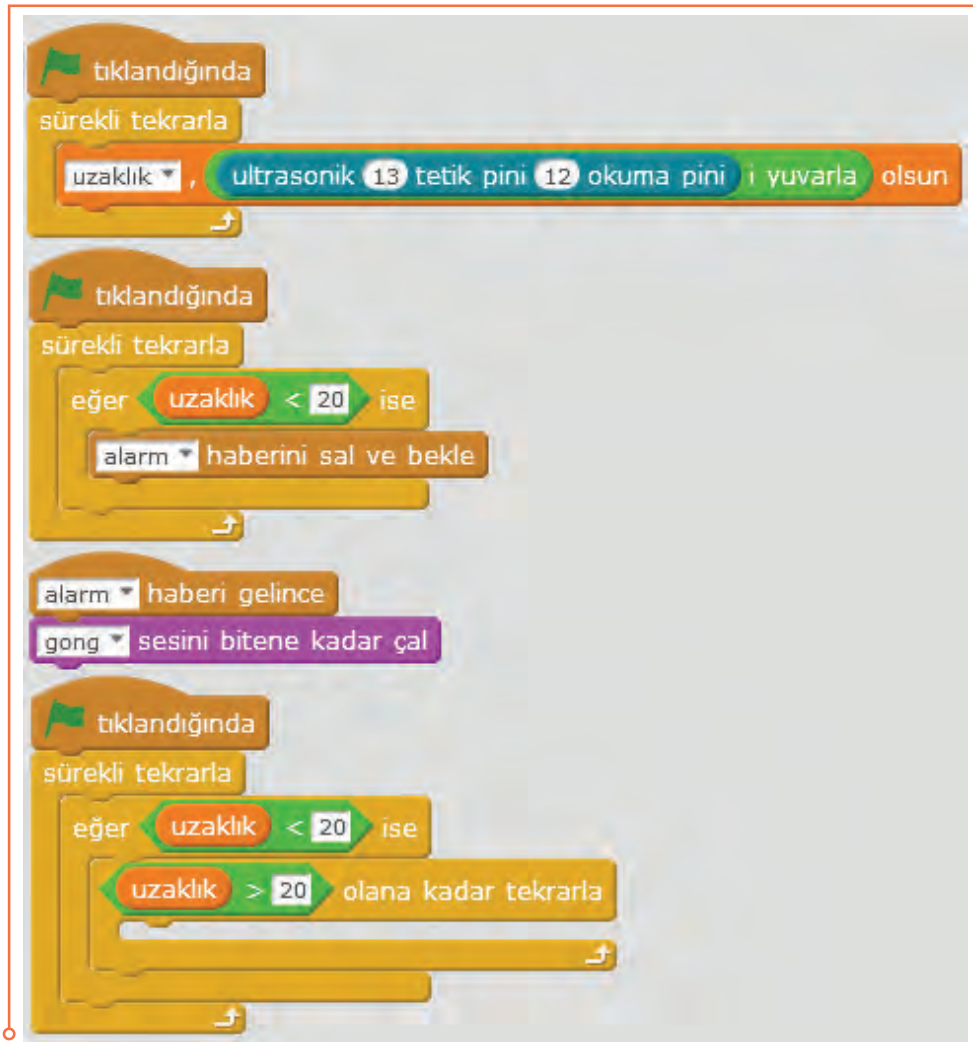
Aşağıdaki LED uygulamasının çalışması için "Arduino Programı" blokunun üzerine tıklanarak açılan ekranda "Arduinoya Yükle" seçeneğinin tıklanması gerekmektedir. Örnekteki diğer programın tıklandığında çalışabilmesi için önceden "Bağlan" sekmesinden "Aygıt Yazılımı Güncellemesi" seçeneği ile güncelleme yapılması gerekmektedir.

Resim 6.45: Arduino UNO ile LED uygulaması örneği

Aşağıda verilen örnekte Arduino'nun dijital pinlerine bağlı HY-SRF 05 ultrasonik sensör kullanılarak bir alarm uygulaması hazırlanmıştır. Ultrasonik sensörün tetik (Trig) pini 13 numaralı, okuma (Echo) pini 12 numaralı Arduino UNO pinine bağlanmıştır. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir. Eğer ultrasonik sensöre olan uzaklık 20 cm'den küçük ise alarm devreye girmekte ve uzaklık 20 cm'den büyük oluncaya kadar devam etmektedir.



Resim 6.46: Arduino UNO ile Ultrasonik Sensör uygulaması Breadboard çizimi

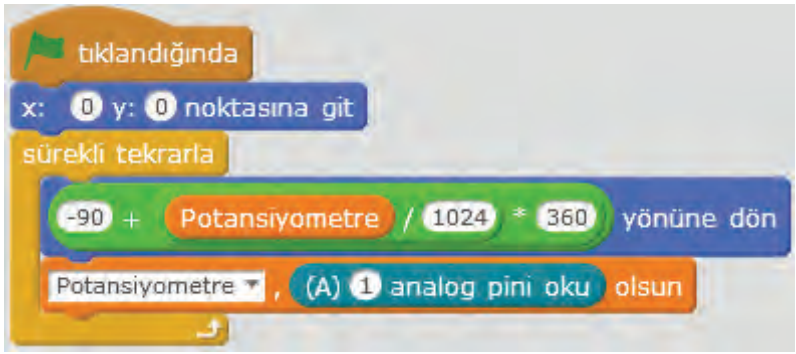


Resim 6.47: Arduino UNO ile Ultrasonik Sensör uygulaması örneği

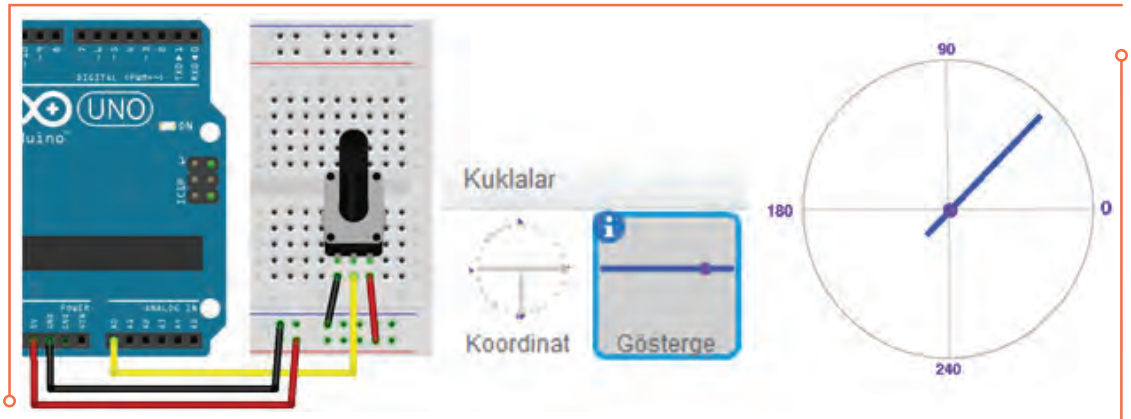
Analog Kontrol Pimlerin Ayarlanması ve Okunması: Arduino UNO kartında toplamda 6 analog kontrol pimi bulunmaktadır (A0, A1, A2, A3, A4, A5, A6 numaralı pinler). Analog pinlerin çıkış değerini 0 ile 1023 arasında ayarlamak ve aynı zamanda, analog pinlerin giriş değerlerini ekranda okumak için yandaki örneği inceleyiniz.

Resim 6.48: Analog pimlerin ayarlanması ve okunması

Aşağıda verilen örnekte A1 pinine bağlı bir potansiyometre kullanılarak 3600'lik bir açıölçer uygulaması yer almaktadır. Potansiyometrenin hareketi ile çizilen koordinat sistemi üzerinde gösterge istenen açıda ayarlanabilmektedir. Arduino ile okunan analog değerler 0 ile 1023 arasında olduğu, koordinat sisteminde ise 3600 istendiği için dönüştürme işlemi uygulanmıştır. Bunun için potansiyometre değeri 1024'e bölünerek 360 ile çarpılmıştır. Böylece 0 ile 1023 arasındaki bir değer 0 ile 360 arasına taşınmıştır. Potansiyometrenin Arduino ile bağlantısı ve koordinat sistemi aşağıda gösterilmektedir. Kullanılan göstergenin kaç derecelik açıyla başlayacağını belirlemek için "yönüne don" blokunda -90° alınmıştır. Kullanılacak koordinat sistemi ve gösterge şekline göre istenilen açıdan başlanabilir.

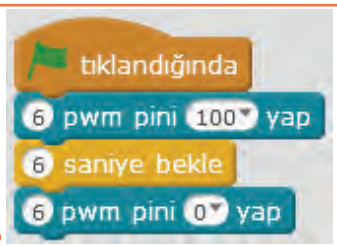


Resim 6.49: Arduino UNO ile Potansiyometre uygulaması örneği



Resim 6.50: Arduino UNO ile Potansiyometre uygulaması Breadboard çizimi ve ekran görüntüleri

PWM Dijital Kontrol Pimlerin Ayarlanması ve Okunması: Arduino UNO kartında toplamda 6 adet PWM dijital kontrol pimi bulunmaktadır (D3, D5, D6, D9, D10, D11 numaralı pinler). PWM pinlerin çıkış değerini 0 ile 1023 arasında ayarlamak ve aynı zamanda, PWM pinlerin giriş değerlerini ekranda okumak için hazırlanan yandaki örneği inceleyiniz.

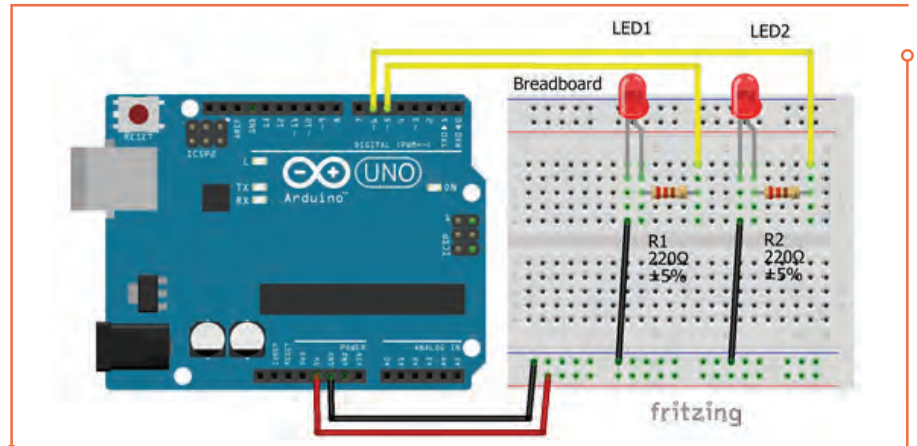


Resim 6.51: PWM Dijital kontrol pimlerin ayarlanması ve okunması işlemi

Aşağıda verilen örnekte Arduino'nun 5 ve 6 numaralı pwm pinlerine bağlı 2 LED'in boşluk tuşuna basıldığı sürece ışık seviyesinin 0.5 birim artarak yanması sağlanmıştır. Bunun için ışık seviyesi adını taşıyan bir değişken oluşturulmuştur. Örnek çalıştırldığı zaman boşluk tuşunun basılı olup olmadığı kontrol etmekte, eğer basılı değilse ışık seviyesini sıfırlamakta, basılı ise ışık düzeyini artırmaktadır. Aynı zamanda boşluk tuşuna basıldığı süre, süre ölçer kullanılarak ekrana yazılmaktadır. Süre ölçer değeri 100'e bölünerek ve tam sayıya yuvarlanarak sürenin 1'den başlaması sağlanmıştır. Eğer ışık seviyesi 30 birim olursa ışık düzeyi sıfırlanmaktadır.

Resim 6.52: Arduino UNO ile PWM uygulaması

Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için birer 220 ohm dirençle 5 ve 6 numaralı PWM pinine bağlanmıştır.

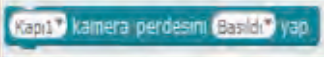
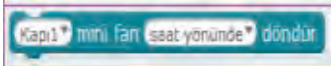
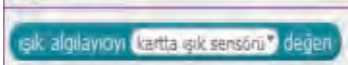
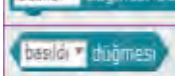


Resim 6.53: Arduino UNO ile PWM uygulaması Breadboard çizimi

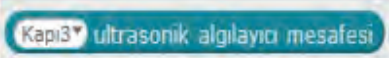
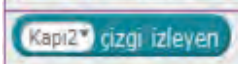
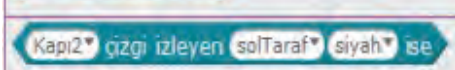
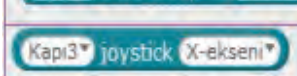
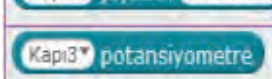
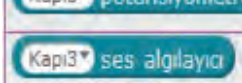
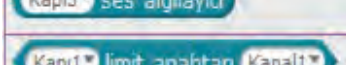
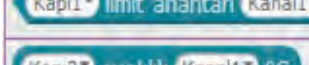
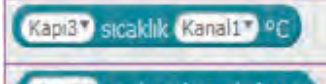
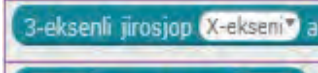
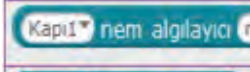
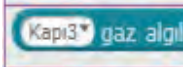
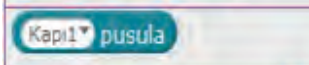
6.8. Robotlar Alt Başlığı Altında Verilen mBot Komut Blokları

Makeblock tarafından üretilen mBot robot, robot kontrol kartları ve kalkanların (shield) programlanmasında kullanılabilecek bloklar aşağıda iki bölüm halinde açıklanmıştır.

mBot Programı	mBot programını çalıştırmak için kullanılır.
ileri git hızı 0 yap	Robotun belirtilen hızda (0 ile 255 arası), ileri veya geri gitmesi, sağ veya sola dönmesi için kullanılır.
M1 motorunun hızını 0 yap	Belirtilen moturun (M1 veya M2) hızını (0 ile 255 arası) ayarlamak için kullanılır.
Kapı1 Kanal1 servo açısını 90 yap	Belirtilen kapıya bağlı, seçilen kanaldaki servo motorun açısını (0, 45, 90, 135 veya 180) derece yapmak için kullanılır.
kart ledler hepsi kır 0 yeş 0 mav 0	Robot kontrol kartı üzerinde bulunan RGB ledlerin renklerini ayarlamak için kullanılır.
Kapı1 led hepsi kır 0 yeş 0 mav 0	Genişleme kapılarına bağlı bulunan RGB ledlerin (4 adet) renklerini ayarlamak için kullanılır.
Kapı1 şerit led Kanal2 — hepsi kır 0 yeş 0 mav 0	Genişleme kapılarına bağlı bulunan serit RGB ledlerin (4 adet), seçilen kanaldaki renklerini ayarlamak için kullanılır.
ses tonunu C4 notasında Yarım vuruş çal	Ses tonunu belirtilen notada, istenen vuruş kadar çalmak için kullanılır.
Kapı1 de 0 numaralı yüzü göster	Belirtilen kapıda, belirtilen numaralı yüzü göstermek için kullanılır.
Kapı1 de x: 0 y: 0 konumunda Hi harf(ler)ini göster	Belirtilen kapıda, belirtilen x ve y konumunda belirtilen kelimeyi göstermek için kullanılır.
saati göster Kapı1 saat: 10 : dakika: 20	Belirtilen kapıda, belirtilen saati göstermek için kullanılır.
Kapı1 de x: 0 y: 0 konumunda çizimi göster	Belirtilen kapıda, belirtilen x ve y konumunda belirtilen çizimi göstermek için kullanılır.
Kapı1 7 parçalı displeyde 100 yaz	Belirtilen kapıya bağlı 7 parçalı displeye, belirtilen sayıyı yazdırmak için kullanılır.

	Kapı 3 veya 4'te bulunan ışık algılayıcıyı açmak veya kapatmak için kullanılır.
	Belirtilen kapıya bağlı kamera perdesini basıldı, bırak, odakları veya kaydır yapmak için kullanılır.
	Belirtilen kapıya bağlı mini fanı saat yönünde, saat yönünün tersinde döndürmek vaya durdurmak için kullanılır.
	Robot kontrol kartı üzerinde bulunan veya kapılara bağlı olan ışık sensorunun değeri ile ilgili işlem yapmak için kullanılır.
	Basıldı düğmesine basıldığında veya serbest bırakıldığında programı çalıştırmak için kullanılır.
	Basıldı düğmesine basıldığında veya serbest bırakıldığında işlem yapmak için kullanılır.

Tablo 6.15: Robotlar alt başlığı altında verilen mBot komut blokları 1

	Belirtilen kapıya bağlı ultrasonik algılayıcı mesafesi ile ilgili işlemler yapmak için kullanılır.
	Belirtilen kapıya bağlı çizgi izleme algılayıcılarıya ilgili işlemler yapmak için kullanılır.
	Belirtilen kapıya bağlı sağ veya sol çizgi izleme algılayıcılarıya ilgili işlemler yapmak için kullanılır.
	Belirtilen kapıya bağlı joystick'in x ve y ekseni ile ilgili işlemler yapmak için kullanılır.
	Belirtilen kapıya bağlı potansiyometre ile ilgili işlemler yapmak için kullanılır.
	Belirtilen kapıya bağlı ses algılayıcı ile ilgili işlemler yapmak için kullanılır.
	Belirtilen kapıya bağlı limit anahtarı ile ilgili işlemler yapmak için kullanılır.
	Belirtilen kapıya bağlı sıcaklık algılayıcı ile ilgili işlemler yapmak için kullanılır.
	Belirtilen kapıya bağlı pır hareket algılayıcı ile ilgili işlemler yapmak için kullanılır.
	Belirtilen kapıya bağlı 3 eksenli jiroskopun x, y ve z ekseniyle ilgili işlemler için kullanılır.
	Belirtilen kapıya bağlı nem ve sıcaklık algılayıcıyla ilgili işlemler için kullanılır.
	Belirtilen kapıya bağlı alev algılayıcıyla ilgili işlemler için kullanılır.
	Belirtilen kapıya bağlı gaz algılayıcıyla ilgili işlemler için kullanılır.
	Belirtilen kapıya bağlı pusula algılayıcıyla ilgili işlemler için kullanılır.
	Belirtilen kapıya bağlı dokunma algılayıcıyla ilgili işlemler için kullanılır.
	Belirtilen kapıya bağlı butonlarla ilgili işlemlerler için kullanılır.

kızıl ötesi kumandanın A düğmesi basıldı	Kızıl ötesi kumandayla ilgili işlemler yapmak için kullanılır.
merhaba mesajını mBot'a gönder	Belirtilen mesajı mBot robota göndermek için kullanılır.
mBot iletisi alındı	Alınan mBot iletisi ile ilgili işlemler için kullanılır.
süre ölçer	Süre ölçümü için kullanılır.
süre ölçeri sıfırla	Süre ölçümünü sıfırlamak için kullanılır.

Tablo 6.16: Robotlar alt başlığı altında verilen mBot blokları 2

6.8.1. mBlock ile Arduino ve mBot Uyumlu Robot Kullanımı

Makeblock tarafından üretilen mBot robot, mBot Ranger robot, robot kontrol kartları ve kalkanların (Me Uno Shield) programlanmasında kullanılabilecek bloklar gruplar halinde verilmektedir. Hangi robot ya da kalkan kullanılıyorsa onun mBlock Kartlar sekmesinden seçilmesi gereklidir. Aşağıda Arduino UNO uyumlu robot ve mBot robot için programlama örnekleri verilmiştir.

Robot Hareketi için DC Motor Kullanımı: mBot robot Arduino uyumlu olduğu için buradaki bloklar Arduino uyumlu robotların kullanımı için de kullanılabilmektedir. DC motorlar doğrudan kontrol kartına bağlanmazlar. Bu nedenle, uygun DC motor sürücüsü ile istenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) ve uygun kanala (Kanal 1 veya 2) bağlayarak kullanılması gerekmektedir. Yandaki örnekte uyumlu robot 1 sn boyunca ileri, 1 sn boyunca da geri hareketini iki defa tekrar ettikten sonra hızı sıfırlanarak durmaktadır.

Resim 6.54: Robot hareketi için DC motor kullanımı örneği

mBot Robotun Hareketi için DC Motor Kullanımı: mBot robot üzerinde bulunan M1 ve M2 motorları kullanılarak robotun hareket kullanımı sağlanmaktadır. Yandaki örnekte mBot robot 1 sn boyunca ileri, 1 sn boyunca da geri hareketini iki defa tekrar ettikten sonra motor hızları sıfırlanarak durmaktadır.

Resim 6.55: mbot Robotun hareketi için DC motor kullanım örneği

mBot Robot Üzerindeki RGB LED Kullanımı: mBot robot üzerinde bulunan RGB LED1 ve RGB LED2 istenilen renkleri üretmek için programlanabilmektedir. Ayarlanması gereken üç parametre vardır: Led seçimi (sağ, sol veya hepsi), kırmızı değer (0 ile 255 arası), mavi değer (0 ile 255 arası), yeşil değer (0 ile 255 arası). Aşağıdaki örnekte RGB LED'lere kırmızı, yeşil ve mavi renk değerlerinin verilmesi gösterilmiştir.

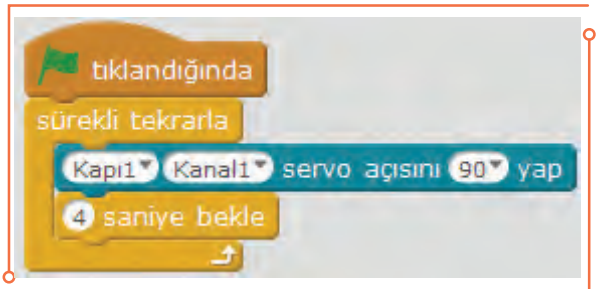
Resim 6.56: mbot Robot üzerindeki RGB LED kullanım örneği

8*16 LED Matris Kullanımı: Bu parça sekiz âdeti dikeyde, 16 âdeti yatayda olmak üzere toplam 128 adet mavi LED'den oluşmaktadır. Bu parça mBot robot üzerinde kayan yazılar yazmak, algılayıcı değerlerini veya işlem sonuçlarını göstermek için kullanılabilir. Bu amaçla parçanın istenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) bağlayarak kullanılması gerekmektedir. Yandaki örnekte istenilen metnin bu parça üzerinde yazdırılması gösterilmiştir. Önce metin girişi yapılmakta, girilen metinin (yanıt)

uzunluğu tespit edilmekte ve bu değer 6 ile çarpılarak buna "yer" bilgisi eklenmektedir. Her bir karakter yatayda 6 sıra LED kullandığı için 6 ile çarpılmaktadır. Yer bilgisi için "yer" adında bir değişken oluşturulmuş olup, değişkeninin sağdan başlanarak ("yer" 16 olsun bloğu) her harf için -1 azaltılarak yeni konumuna yerleştirilmesi sağlanmıştır. Yanıtın kayarak gösterilmesi için kapı 4'de x:yer y:0 konumunda yanıt harflerini göster bloğu kullanılmıştır.

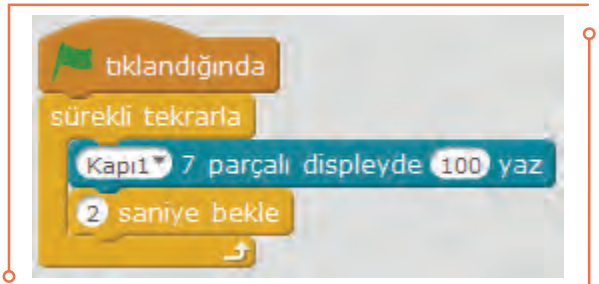
Resim 6.57: 8*16 LED Matris kullanım örneği

Servo Motor Kullanımı: Servo motorlar doğrudan kontrol kartına bağlanmazlar. Bu nedenle, uygun servo motor sürücüsü ile istenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) ve uygun kanala (Kanal 1 veya 2) bağlanarak kullanılması gerekmektedir. Yandaki örnekte servo motor açısının 90 derece yapılması gösterilmiştir.



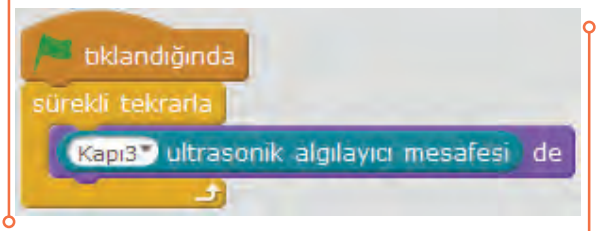
Resim 6.58: Servo motor kullanım örneği

7 Segment Ekran Kullanımı: 7 segmentli ekranlar, genellikle hızı, zamanı, algılayıcıların değerini veya puanı göstermek için robot projelerinde kullanılmaktadır. Doğrudan istenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) bağlanabilirler. Yandaki örnekte 7 segmentli ekrana 100 yazılması gösterilmiştir.



Resim 6.59: Servo motor kullanım örneği

Ultrasonik Algılayıcı Kullanımı: Ultrasonik algılayıcılar genellikle algılayıcı ile engeller arasındaki mesafeyi ölçmek için kullanılırlar. Direkt olarak (Kapı 1, 2, 3 veya 4) bağlanabilirler. Yandaki örnekte engele olan uzaklığın ölçülmesi gösterilmiştir.



Resim 6.60: Ultrasonik algılayıcı ile mesafe ölçme örneği

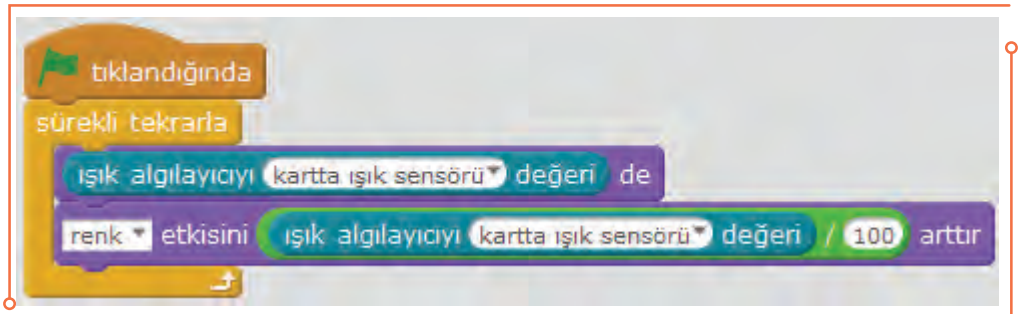
Aşağıdaki diğer örnekte ultrasonik algılayıcıdan alınan mesafe bilgisi kullanılarak robotun herhangi bir nesneye çarpmadan dolaşması sağlanmıştır. Bu amaçla oluşturulan "hız" değişkeni ile istenilen hız belirlenmekte "mesafe" değişkeni ile ölçülen uzunluğa göre robotun yönlendirme işlemi sağlanmaktadır. Buna göre nesneye 20 cm'den az kalmışsa robot geri dönmekte, 20 ile 40 cm arasındaki mesafeler için sağa dönmekte, 40 ile 60 cm mesafeler için sola dönmekte ve 60 cm'den büyük mesafe varsa ileri gitmektedir. Her hareket eylemini de sesle belirtmektedir. Ses dosyaları önceden kaydedilerek mBlock ortamına aktarılmıştır.

Resim 6.61: Ultrasonik algılayıcı kullanarak engele çarpmadan dolaşma uygulaması

Işık Algılayıcı Kullanımı: Işık algılayıcı genellikle ortamdaki ışığın yoğunluğunu ölçmek için kullanılır. Direkt olarak karttaki ışık algılayıcı kullanılabildiği gibi (Kapı 3 veya 4) de bağlanabilirler. Yandaki örnekte ortamdaki ışık seviyesinin ölçülmesi gösterilmiştir.

Resim 6.62: Işık algılayıcı kullanım örneği

Aşağıdaki diğer örnekte ışık algılayıcı kullanılarak robot kuklanın ışık değerine göre renk değiştirmesi sağlanmıştır. Bu amaçla "Görünüm" blok gurubunda bulunan "… etkisini artır" bloğu alınmış, burada renk seçeneği kullanılmıştır. Işık değeri / 100 oranında artırılmıştır.



Resim 6.63: Işık algılayıcı kullanarak renk değiştirme örneği

Ses Algılayıcı Kullanımı: Ses algılayıcı, ortamdaki ses şiddetini ölçmek için tasarlanmıştır. Direkt olarak (Kapı 3 veya 4) de bağlanabilirler. Yandaki örnekte ortamdaki ışık seviyesinin harici bir ses algılayıcı bağlanarak ölçülmesi gösterilmiştir. İstenirse ses seviyesine bağlı uygulama yapmak için bilgisayarda bulunan mikrofon veya web kamera mikrofonu da kullanılabilir.

Resim 6.64: Ses algılayıcı kullanım örneği

Yandaki diğer örnekte bilgisayara bağlı mikrofon kullanılarak ses seviyesine göre ekrandaki kuklayı zıplatan bir uygulama gösterilmiştir.

Kuklanın x ekseninde sıfır noktasında, y ekseninde "Ses_Seviyesi" düzeyinde olması için "Hareket" blok gurubunda bulunan "x:0 y: noktasına git" bloğu kullanılmıştır. "Ses_Seviyesi" değişkeninin işaretlenmesi ile ölçüm sonucunun bilgisayar ekranında gösterilmesi sağlanmıştır.

Resim 6.65: Ses seviyesine göre ekrandaki kuklayı zıplatan uygulama

Resim 6.66: Ses seviyesine göre ekrandaki kuklayı zıplatan uygulama

Bu amaçla "Ses_Seviyesi" adını taşıyan bir değişken oluşturulmuştur. Bu değişkene "Algılama" blok gurubunda bulunan "ses şiddeti" bloğu eklenerek ses şiddetinin tespiti sağlanmıştır.

Resim 6.67: Uygulama için değişken oluşturulması

Çizgi Algılayıcı Kullanımı: Çizgi algılayıcılar, robota kalınca çizilmiş çizgileri veya yakında bulunan nesneleri algılama yeteneği kazandırmak, belirli bir yolu takip etmesini sağlamak için kullanılırlar. Direkt olarak (Kapı 1, 2, 3 veya 4) bağlanabilirler. Yandaki örnekle algılayıcıların çizgi üzerinde olup olmadıkları tespit edilmektedir.

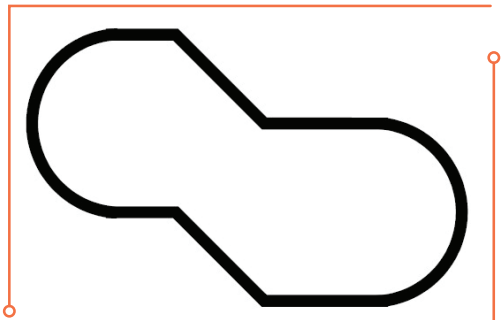
Resim 6.68: Çizgi algılayıcı kullanım örneği

Aşağıda verilen örnekte ise robotun çizgi algılayıcıları çizgi üzerinde veya çizgi dışında olup olmadıkları her bir algılayıcı için belirlenip ekrana yazılmaktadır. Kullanılan blokları göstermek için ilk eğer işlemini oluşturan bloklar yerlerine konmadan gösterilmiştir.

Resim 6.69: Robotun çizgi algılayıcıları ile çizginin neresinde olduğunu belirleyen uygulama örneği

Aşağıda verilen örnekte ise bir çizgi takip uygulaması bulunmaktadır. Robotun çizgileri takip edebilmesi için önce algılayıcıların her birinin konumu belirlenmekte, buna göre de robotun yönlendirilmesi sağlanmaktadır. Bu amaçla "hiz" ve "alan" adında iki değişken oluşturulmuştur. "hiz" değişkeni ile hız belirlenmekte, "alan" değişkeni ise algılayıcıların konumuna göre hareketin yönünün belirlenmesinde kullanılmaktadır. Algılayıcılar tarafından; eğer her iki algılayıcı çizgi üzerinde ise "0", sol algılayıcı çizgi

üzerinde ise “1”, sağ algılayıcı çizgi üzerinde ise “2” ve her iki algılayıcı çizgi dışında ise “3” değeri üretildiği için yönlendirmeler buna göre yapılmaktadır. Aşağıda örnek bir çizgi takip haritası yer almaktadır.



Resim 6.70: Çizgi takip haritası

Resim 6.71: Çizgi takip uygulaması

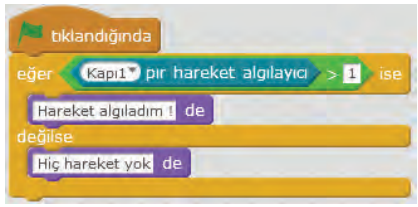
Potansiyometre Kullanımı: Potansiyometre potun dönüşüne bağlı olarak sürekli değişen bir değer üreten elektronik parçadır. Döndürülmek suretiyle 0 ile 1023 arasında çıkış değeri üretir. Direkt olarak (Kapı 3 veya 4) bağlanabilirler. Yandaki örnekle portun döndürülmesi ile oluşan potansiyometre değeri ekranda okunabilmektedir.

Resim 6.72: Potansiyometre kullanım örneği

Sıcaklık Algılayıcı Kullanımı: Sıcaklık algılayıcı uç noktalardaki ve ortamdaki sıcaklığı ya da uzaktaki sıcak bir şeyi algılamak için kullanılmaktadır. İstenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) ve uygun kanala (Kanal 1 veya 2) bağlayarak kullanılması gerekmektedir. Yandaki örnekle ısı değeri ekranda okunabilmektedir.

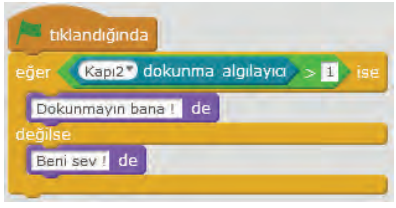
Resim 6.73: Sıcaklık algılayıcı örneği

PIR Hareket Algılayıcı Kullanımı: PIR hareket algılayıcı, hayvanların ve insanların hareketini yaklaşık 6-7 metre uzaklıktan algılamak için kullanılmaktadır. Yandaki örnekle hareket algılandığında ekrana "Hareket algıladım !" ifadesi yazılmaktadır.



Resim 6.74: PIR Hareket algılayıcı kullanım örneği

Dokunma Algılayıcı Kullanımı: Dokunma algılayıcı, insanların dokunma hareketini algılamak için kullanılmaktadır. Yandaki örnekle dokunma algılandığında ekrana "Dokunmayın bana !" ifadesi yazılmaktadır.



Resim 6.75: Dokunma algılayıcı kullanım örneği

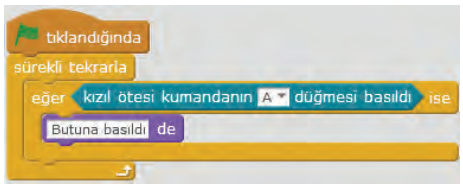
Limit Anahtarı Kullanımı: Limit anahtarları, belirlenen seviyeye ulaşılıp ulaşılmadığının kullanımı için kullanılmaktadır. Yandaki örnekle limite ulaşıldığında ekrana "Limite ulaşıldı !" ifadesi yazılmaktadır.

Resim 6.76: Limit anahtarı kullanım örneği

Buton Kullanımı: Buton parçası üzerinde bulunan butonların kullanımı için kullanılmaktadır. Butonlar robot kontrol etmek için, işlev düğmesi veya yön düğmesi olarak çalışabilirler. Yandaki örnekle butona basıldığında ekrana "Butona basıldı!" ifadesi yazılmaktadır.

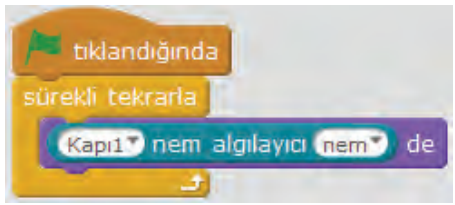
Resim 6.77: Buton kullanım örneği

Kızılötesi Kumanda Kullanımı: Kızılötesi kumandanın kullanımı için kullanılmaktadır. Kızılötesi kumanda robot kontrol etmek için, işlev düğmesi veya yön düğmesi olarak da çalışabilirler. Yandaki örnekle kızılötesi kumandanın A düğmesine basıldığında ekrana "Butona basıldı!" ifadesi yazılmaktadır.



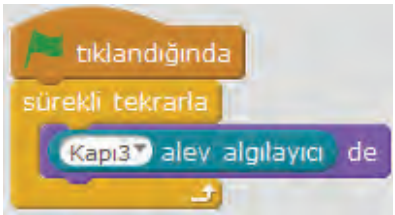
Resim 6.78: Kızılötesi kumanda kullanım örneği

Nem ve Sıcaklık Algılayıcı Kullanımı: Nem ve sıcaklık algılayıcı ortamın sıcaklığı ya da nem durumunu algılamak için kullanılmaktadır. İstenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) bağlayarak kullanılması gerekmektedir. Yandaki örnekle nem veya sıcaklık değeri ekranda okunabilmektedir.



Resim 6.79: Nem ve sıcaklık algılayıcı kullanım örneği

Alev Algılayıcı Kullanımı: Alev algılayıcı yangın ve ateş durumunu algılamak için kullanılmaktadır. İstenilen bağlantı noktalarına (Kapı 3 veya 4) bağlayarak kullanılması gerekmektedir. Yandaki örnekle alev, ateş algılandığında ekranda okunabilmektedir.



Resim 6.80: Alev algılayıcı kullanım örneği

Gaz Algılayıcı Kullanımı: Gaz algılayıcı ortamdaki çeşitli gazların (kullanılan türe göre) varlığını algılamak için kullanılmaktadır. İstenilen bağlantı noktalarına (Kapı 3 veya 4) bağlayarak kullanılması gerekmektedir. Yandaki örnekle gaz algılandığında ekranda okunabilmektedir.

Resim 6.81: Gaz algılayıcı kullanım örneği

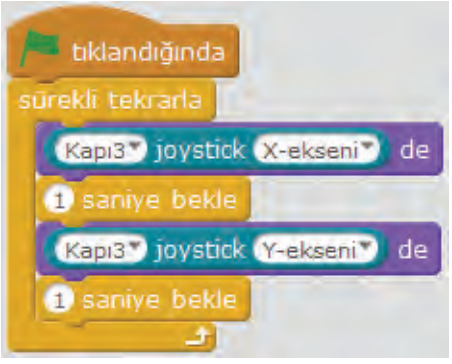
Pusula Algılayıcı Kullanımı: Pusula algılayıcı yön belirlemek için robot uygulamalarında kullanılmaktadır. İstenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) bağlayarak kullanılması gerekmektedir. Yandaki örnekle pusula algılandığında ekranda okunabilmektedir.

Resim 6.82: Pusula algılayıcı kullanım örneği

Jiroskop Algılayıcı Kullanımı: Jiroskop algılayıcı robotun hareket ve duruş algılaması, dengelenmesi için kullanılmaktadır. Genellikle 3 eksenli ivmeölçer ile 3 eksenli açısal hız algılayıcı ve hareket işlemcisi birlikte bulunmaktadır. İstenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) bağlayarak kullanılması gerekmektedir. Yandaki örnekle x, y ve z eksen değerleri algılandığında ekranda okunabilmektedir.

Resim 6.83: Jiroskop algılayıcı kullanım örneği

Joystick Kullanımı: Joystick, yönsel hareketleri analog değerlere (0 ile 1023 arasında) dönüştürebilen elektronik bir parçadır. Robotu X ve Y ekseni yönünde kontrol etmek için kullanılmaktadır. İstenilen bağlantı noktalarına (Kapı 3 veya 4) bağlayarak kullanılması gerekmektedir. Yandaki örnekle x ve y eksenindeki joystick hareketleri ekranda okunabilmektedir.



Resim 6.84: Joystick kullanım örneği

6.9. Düşünelim / Araştıralım / Uygulayalım

mBlock blok tabanlı robot programlama yazılımını öğrenmek ve bu ortamda programlama yapabilmek için her bölümde verilen örnekleri ve uygulamaları aynı şekilde hazırlayınız. Eğer elinizde eğitsel robot bulunmuyorsa birçok uygulama için sanal robot kullanabileceğinizi unutmayınız. Tercih edilen fiziksel bir robotun olmasıdır. Örnekleri ve uygulamaları parametreleri değiştirerek tekrar tekrar deneyiniz. mBlock sitesinde verilen örnekleri ve İnternette yer alan uygulamaları araştırınız. Bu ortamda hazırlayabileceğiniz özgün uygulamalar düşünerek gerçekleştirmeye çalışınız.

6.10. Değerlendirme Soruları

1. **mBlock yüklü bilgisayarla Arduino destekli robot arasında bağlantının oluşturulması için aşağıdaki seçeneklerden hangisi kullanılabilir?**
   a) USB port
   b) Ağ üzerinden bağlantı
   c) Bluetooth modülü ile
   d) 2.4 Ghz Dongle modülü
   e) Hepsi

2. **mBlock yüklü bilgisayarla Arduino destekli robotun güncellenmesi ve programlanması için aşağıdaki seçeneklerden hangisi kullanılabilir?**
   a) USB port
   b) Ağ üzerinden bağlantı
   c) Bluetooth modülü
   d) 2.4 Ghz Dongle modülü
   e) Hepsi

3. **Girilen değerleri alan veya programın çalışmasıyla bazı değerlerin atandığı veri tutucularından oluşan temel programlama yapılarına ne ad verilir?**
   a) Fonksiyon
   b) Liste
   c) Değişken
   d) Dizi
   e) Prosedür

4. **Çok sayıda değişkenle çalışmak için oluşturulmuş temel programlama yapılarına ne ad verilir?**
   a) Fonksiyon
   b) Liste
   c) Değişken
   d) Dizi
   e) Prosedür

5. **Program akışı içinde tekrarlayan ifadelerin her seferinde tekrar tekrar yazılması yerine, bir kere ayrı bir yerde yazılıp tekrarlanan her yerde kullanmak için aşağıdakilerden hangisinin yapılması uygundur?**
   a) Prosedür oluşturmak
   b) Dizi oluşturmak
   c) Liste oluşturmak
   d) Blok oluşturmak
   e) Fonksiyon oluşturmak

6. Prosedürlere değer taşıyan değişkenlere ne adı verilir?

a) Blok
b) Fonksiyon
c) Dizi
d) Liste
e) Parametre

7. Sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robot için oluşturulan programların ve diğer uygulamaların çalıştırılması için hangi alt başlık altında verilen blok gurupları kullanılmaktadır?

a) Hareket alt başlığı altında verilen blok gurupları
b) Görünüm alt başlığı altında verilen blok gurupları
c) Olaylar alt başlığı altında verilen blok gurupları
d) Kontrol alt başlığı altında verilen blok gurupları
e) İşlemler alt başlığı altında verilen blok gurupları

8. Arduino blokları ile diğer blokların Arduino uygulamalarında birlikte kullanılabilmesi için mBlock ortamında aşağıdakilerden hangisinin yapılması gereklidir?

a) Aygıt Yazılımı Güncellemesi yapılmalıdır
b) Arduino Programı blokunun üzerine tıklanmalıdır
c) Arduino'ya Yükle seçeneğini üzerine tıklanmalıdır
d) Varsayılan Program sıfırlanmalıdır
e) Arduino sürücüsü yüklenmelidir

9. Arduino uygulamalarında elektronik bileşenlerin lehim yapmadan kullanılabilmesi için aşağıdakilerden hangisi kullanılmalıdır?

a) Soket
b) Konnektör
c) Delikli kart
d) Breadboard
e) Pertinaks

10. Aşağıdakilerden hangisi blok tabanlı robot programlama yazılımlarından ve ortamlarından biri değildir?

a) miniBlog
b) ModKit
c) Ardublock
d) Snap4Arduino
e) Srach for Arduino (S4A)

7. METİN TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI

Bu bölümün sonunda,

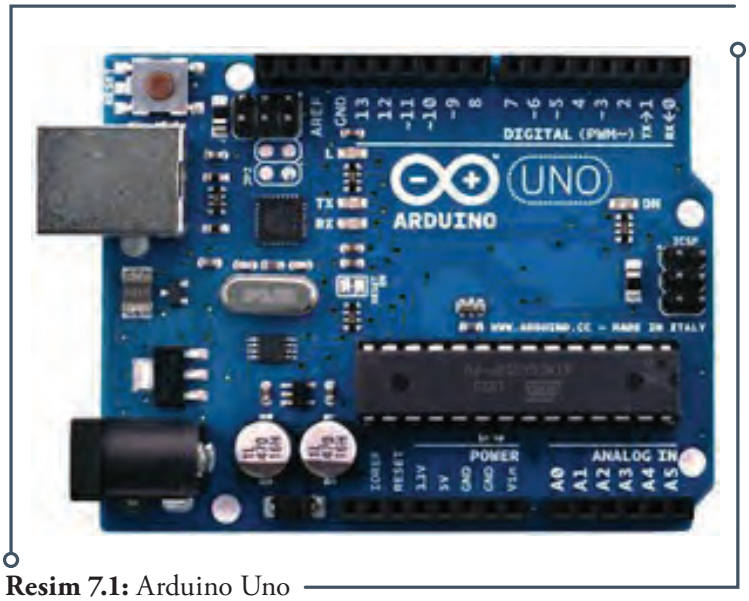
- ✓ Arduino tümleşik geliştirme ortamının temel yapısını tanımlayabilecek,
- ✓ Arduino tümleşik geliştirme ortamının kullanım özelliklerini açıklayabilecek,
- ✓ Arduino UNO geliştirme kartının kullanım özelliklerini ve yapısını açıklayabilecek,
- ✓ Geliştirme yapılan bilgisayarla Arduino kartları arasında bağlantı oluşturulabilecek,
- ✓ Geliştirme yapılan bilgisayarla Arduino kartları arasında bağlantı ayarlarını yapabilecek,
- ✓ Arduino tümleşik geliştirme ortamının program yapısını açıklayabilecek,
- ✓ Arduino tümleşik geliştirme ortamının program yapısını oluşturan bileşenleri geliştirilen programa uygun şekilde kullanabilecek,
- ✓ Arduino tümleşik geliştirme ortamının değişken yapısının açıklayabilecek,
- ✓ Arduino tümleşik geliştirme ortamının değişken yapısını oluşturan bileşenleri geliştirilen programa uygun şekilde kullanabilecek,
- ✓ Arduino tümleşik geliştirme ortamının fonksiyon yapısının açıklayabilecek,
- ✓ Arduino tümleşik geliştirme ortamının fonksiyon yapısını oluşturan bileşenleri geliştirilen programa uygun şekilde kullanabilecek,
- ✓ Arduino tümleşik geliştirme ortamında kullanılan seri haberleşme protokollerini karşılaştırabilecek,
- ✓ Arduino kartları ile elektronik bileşenler arasında seri iletişim sağlayabilecek,
- ✓ Arduino kartları ile elektronik bileşenler arasında I2c veri yolu ile iletişim sağlayabilecek,
- ✓ Arduino kartları ile elektronik bileşenler arasında SPI veri yolu ile iletişim sağlayabilecek,
- ✓ Arduino kütüphanelerini program geliştirme projelerinde kullanabilecek,
- ✓ Arduino tümleşik geliştirme ortamında geliştirilen programları yeniden düzenlebilecek,
- ✓ Arduino tümleşik geliştirme ortamında geliştirilen programları yorumlayabilecek,
- ✓ Arduino tümleşik geliştirme ortamında program geliştirilebileceksiniz.

7.1. Metin Tabanlı Robot Programlama Yazılımları ve Ortamları

Robot programlamak amacıyla kullanılan pek çok programlama yazılımı ve ortamı bulunmaktadır. Robot programlama için kullanılan diller incelendiğinde geleneksel dillere robotik kontrolleri kolaylaştıran yapıların eklenmesiyle oluşturduğu görülür. Robot C ve Parallax Propeller C bu alanda dikkat çeken C dilinin robotik programlamaya uyarlanmış metin temelli sürümleridir. Burada metin tabanlı programlama yazılımı olarak Arduino IDE tercih edilmiştir. Arduino IDE robotik programlamada oldukça yaygın kullanılan tümleşik geliştirme ortamıdır.

7.2. Arduino Geliştirme Kartları

Arduino, ileri derecede elektronik ve mikrodenetleyici bilgisi gerektirmeden çok çeşitli projelerin uygulanabileceği açık kaynaklı, donanımında Atmel firması tarafından üretilen AVR mikrodenetleyici içeren bir elektronik geliştirme platformudur. Kolaylıkla analog ve dijital sinyaller alınarak işlenebilmektedir. Bununla birlikte Arduino ile birlikte kullanılacak devre elemanlarının çalışma mantığı ve bağlantı yapısının bilinmesi gerekmektedir. Algılayıcılardan gelen sinyalleri kullanarak, çevresiyle etkileşim içerisinde olan robotlar ve sistemler tasarlanabilmektedir. Tasarlanan projeye özgü olarak dış dünyaya hareket, ses, ışık gibi tepkiler oluşturulabilmektedir. Arduino'nun farklı ihtiyaçlara çözüm üretebilmek için tasarlanmış çeşitli kartları ve modülleri bulunmaktadır. Örneğin daha az donanımın yeterli olduğu projelerde Arduino Nano gibi modeller, çok sayıda giriş çıkış bağlantısına (pin) ihtiyaç duyulduğunda Arduino Mega gibi modeller kullanılmalıdır.

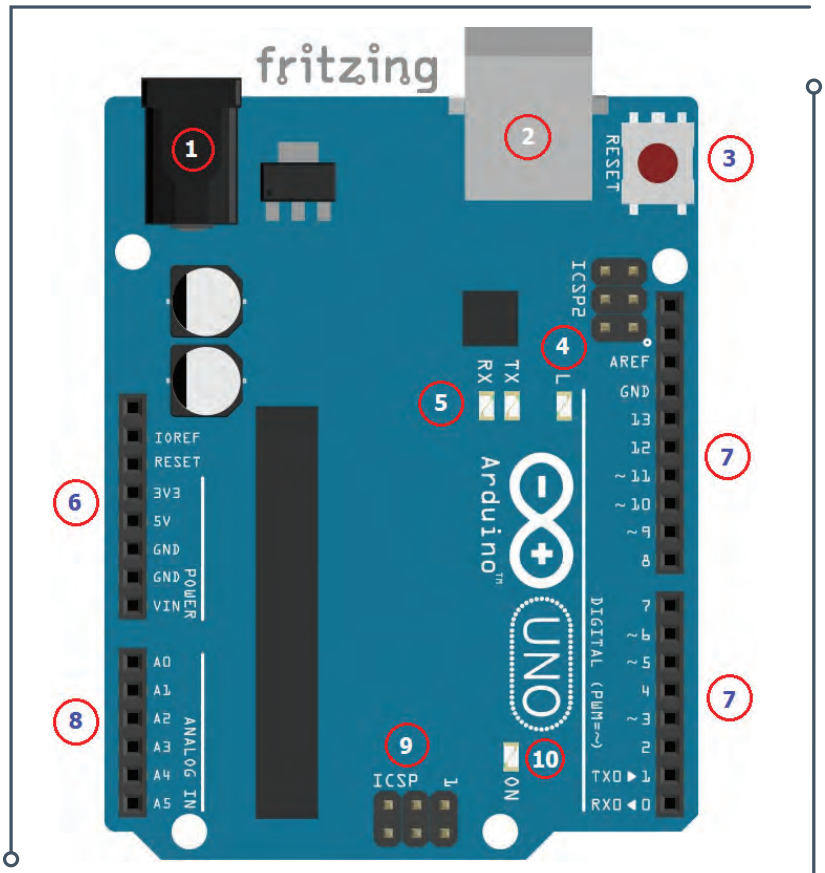


Resim 7.1: Arduino Uno

Birçok robotik uygulamada Arduino ya da Arduino uyumlu Atmel AVR mikrodenetleyici içeren kartlar kullanıldığı için Arduino robot programlamada, programlama öğretiminde de iyi bir seçenektir. Arduino kartlarla çok çeşitli kütüphaneleri yardımı ile kolaylıkla programlama yapılabilmektedir. Bu kitapta gerek konu anlatımında gerekse uygulamalarda çok yaygın olarak kullanılan Arduino UNO modeli tercih edilmiştir.

7.3. Arduino UNO Geliştirme Kartı

Arduino Uno Atmel Atmega 328P mikrodenetleyicisine sahip mikrodenetleyici karttır. Kart üzerinde temel olarak; 14 adet dijital giriş / çıkış pini (6 adeti PWM (Pulse Width Modulation-Darbe / Sinyal Genişlik Modülasyonu), 6 adet analog giriş pini, 16 MHz saat hızı için osilatör, bir adet USB bağlantısı, bir adet DC güç girişi, bir adet ICSP bağlantı başlığı ve bir adet reset düğmesi bulunmaktadır. 32 KB kapasiteli bir flash belleğe sahiptir. Uno'nun mikrodenetleyicisinde diğer modellerde olduğu gibi önceden hazırlanmış bir bootloader programı yüklüdür. Bu nedenle programlama için harici bir programlayıcıya ihtiyaç duyulmaz. Kartın kolaylıkla kullanılabilmesi, bileşenlerin kablo bağlantılarının rahatlıkla yapılabilmesi için pin soket yapısı kullanılmaktadır. Arduino Uno'nun temel bileşenleri aşağıda gösterilmektedir.



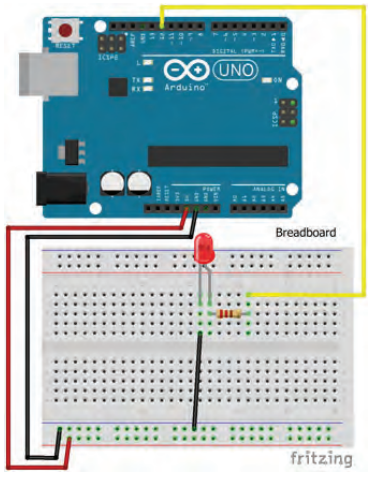
Resim 7.2: Arduino UNO bileşenleri

1. **Güç Girişi:** 7-12 Volt DC adaptör girişi.
2. **USB Bağlantı Konnektörü (USB Port):** UNO'ya program yüklemek ve bilgisayar ile haberleşmek için kullanılmaktadır.
3. **Reset Butonu:** Arduinoyu ve programı setup() fonksiyonundan itibaren yeniden başlatmak için kullanılmaktadır.
4. **LED (Light-Emitting Diodes):** 13 numaralı dijital pine bağlıdır. Programları test etmek için kullanılabilir.

5. **RX (Receiving) ve TX (Transmitting) LED'leri:** Seri haberleşme için kullanılan RX ve TX pininin durumunu gösteren LED'lerdir. Veri alışverişi olduğunda bu LED'ler yanar.
6. **Güç Bağlantıları:** Bu kısımda güç çıkışları yer almaktadır.
7. **Dijital Giriş /Çıkış Pinleri:** Yanında ~ işareti bulunan pinler aynı zamanda analog çıkış (PWM) almak için de kullanılabilir. Ayrıca analog-dijital çeviricinin referans giriş pini ve seri iletişim pinleri de (RX ve TX) buradadır.
8. **Anolog Giriş Pinleri:** 6 adet analog giriş pini bu bölümde bulunmaktadır.
9. **Kart Üzerinde Programlama (ICSP) Pinleri:** Atmega microdenetleyiciyi harici bir programlayıcı ile programlamak için kullanılan pinlerdir.
10. **Güç LED'i:** Kartın güç gösterge LED'idir.

Arduino UNO ile bilgisayar veya diğer cihazlar arasında seri iletişim yoluyla bağlantı kurulmaktadır. Tüm Arduino kartlarında en az bir seri bağlantı noktası (UART veya USART) bulunmaktadır. Seri bağlantı için dijital pin 0 (RX) ve pin 1 (TX) üzerinden bilgisayara USB portundan bağlanarak iletişim kurulur. Bu nedenle, USB bağlantısı kullanılırken dijital giriş veya çıkış için 0 ve 1 numaralı pinler kullanılamazlar. Çalışılan programa bağlı olarak seri iletişim işlemleri için Arduino ortamının dâhili seri monitörü kullanılır.

Arduino uygulamalarında kullanılacak elektronik bileşenler için breadboard (devre tahtası) kullanılması uygulamaların hızlı, kolay ve en önemlisi lehim yapmadan yapılmasına olanak tanıyacaktır. Breadboardların kenarında bulunan alanlar voltaj bağlantıları için enine bağlı, diğer alanlar ise dikine bağlıdır. Bu bağlantı noktalarını kullanarak LED, direnç ve benzeri elektronik bileşenlerin birbirine bağlanması oldukça kolaydır. Aşağıda örnekte Arduino UNO R3'ün 12 numaralı dijital pinine bağlı bir LED'in breadboard bağlantısı gösterilmektedir. Bağlantılar için breadboard kabloları kullanılmaktadır.



Resim 7.3: Arduino IDE'de breadboard kullanımı

Resim 7.4: Breadboard jumper kabloları

7.4. Arduino IDE (Tümleşik Geliştirme Ortamı-Integrated Development Environment)

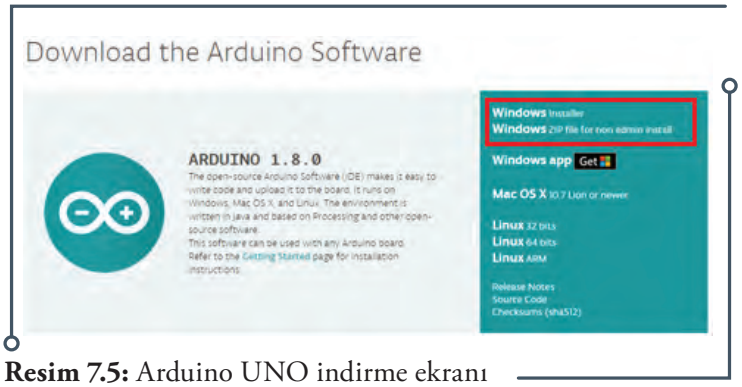
Arduino IDE; kod yazım editörü, tümleşik bir derleyici, yorumlayıcı ve hata ayıklayıcı olarak görev yapan, aynı zamanda derlenen programı karta yükleme işlemini de yapabilen, her platformda çalışabilen Java programlama dilinde yazılmış bir uygulamadır. Arduino tümleşik geliştirme ortamı (IDE); Arduino bootloader (Optiboot), Arduino kütüphaneleri, AVRDude (Arduino üzerindeki mikrodenetleyiciyi programlayan, derlenen kodları programlamak için kullanılan yazılım) ve derleyiciden (AVR-GCC) oluşmaktadır. Bu araçlar yazılımın derlenmesi, bağlanması, çalışmaya tümüyle hazır hale gelmesi ve daha birçok ek işi otomatik olarak yapabilmek amacıyla kullanılmaktadır.

Arduino yazılımı bir tümleşik geliştirme ortamı (IDE) ve mikrodenetleyici ve elektronik konusunda detaylı bilgi sahibi olmayı gerektirmeden herkesin programlama yapabilmesini sağlayan kütüphanelerden oluşmaktadır. Arduino kütüphaneleri, geliştirme ortamı ile birlikte gelmekte ve "libraries" klasörünün altında bulunmaktadır. IDE, Java dilinde yazılmıştır ve Processing adlı dilin ortamına dayanmaktadır. Kütüphaneler ise C ve C++ dillerinde yazılmıştır ve AVR-GCC ve AVR Libc ile derlenmiştir. Optiboot bileşeni Arduino 'nun bootloader bileşenidir. Bu bileşen, Arduino kartlarının üzerindeki mikrodenetleyicinin programlanmasını sağlayan bileşendir. Arduino C++ dili ile kolayca programlanabilmektedir. Kütüphaneler yardımıyla uygulama geliştirilmesi de oldukça kolay ve hızlıdır.

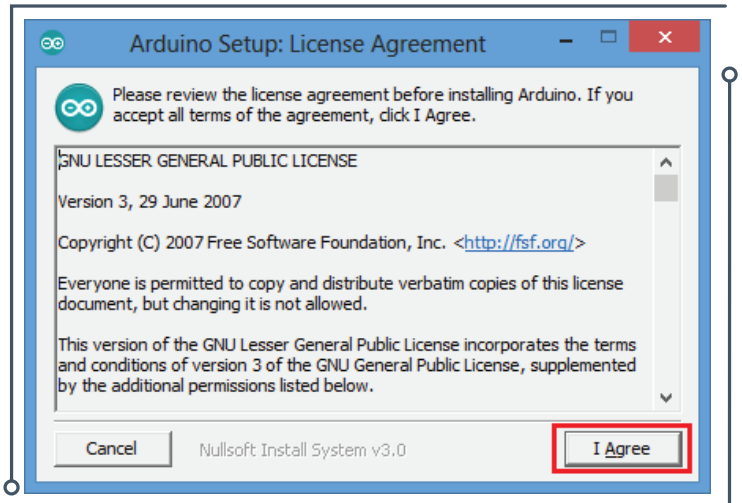
7.5. Arduino IDE Yazılımının Yüklenmesi

Arduino IDE yazılımının son versiyonu http://arduino.cc/en/main/software sitesinden kullanılacak işletim sistemi seçilerek ücretsiz olarak indirilebilmektedir. Örneğin Windows işletim sistemi için zip dosyası veya doğrudan exe dosyasından biri seçilerek indirilmelidir. Eğer zıp seçilmişse dosya ayıklandıktan sonra arduino.exe'ye tıklayarak programın kurulması yeterlidir. Kurulum işlemi tamamlandıktan sonra program kullanılmaya hazırdır. Önemli kurulum aşamaları aşağıda gösterilmiştir.

İlk aşamada lisans sözleşmesi onaylanmalıdır.

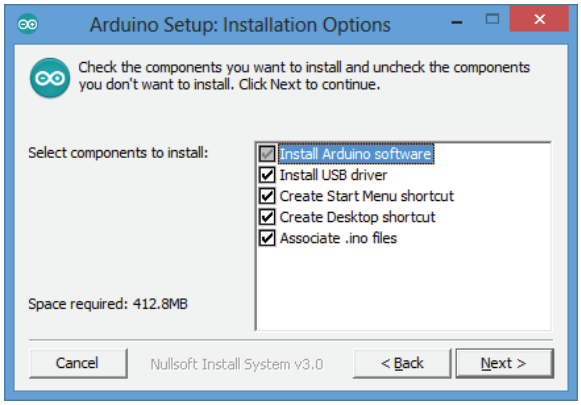


Resim 7.5: Arduino UNO indirme ekranı



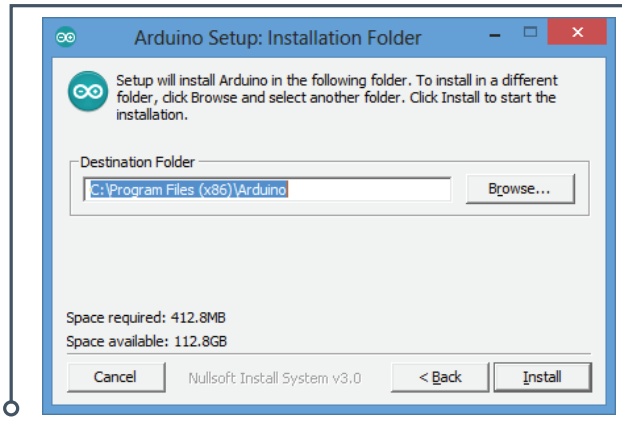
Resim 7.6: Arduino IDE kurulum aşaması

İkinci aşamada kurulum seçenekleri seçili değilse hepsi onaylanmalıdır.



Resim 7.7: Arduino IDE kurulum seçenekleri

Üçüncü aşamada kurulum klasörü seçilmelidir. İstenirse olduğu gibi bırakılabilir. Kurulum sırasında gerekli aygıt sürücüleri de kurulduğu için Arduino destekli bütün kartlar otomatik olarak tanınacaktır. Ayrıca tanıtılmasına gerek yoktur.

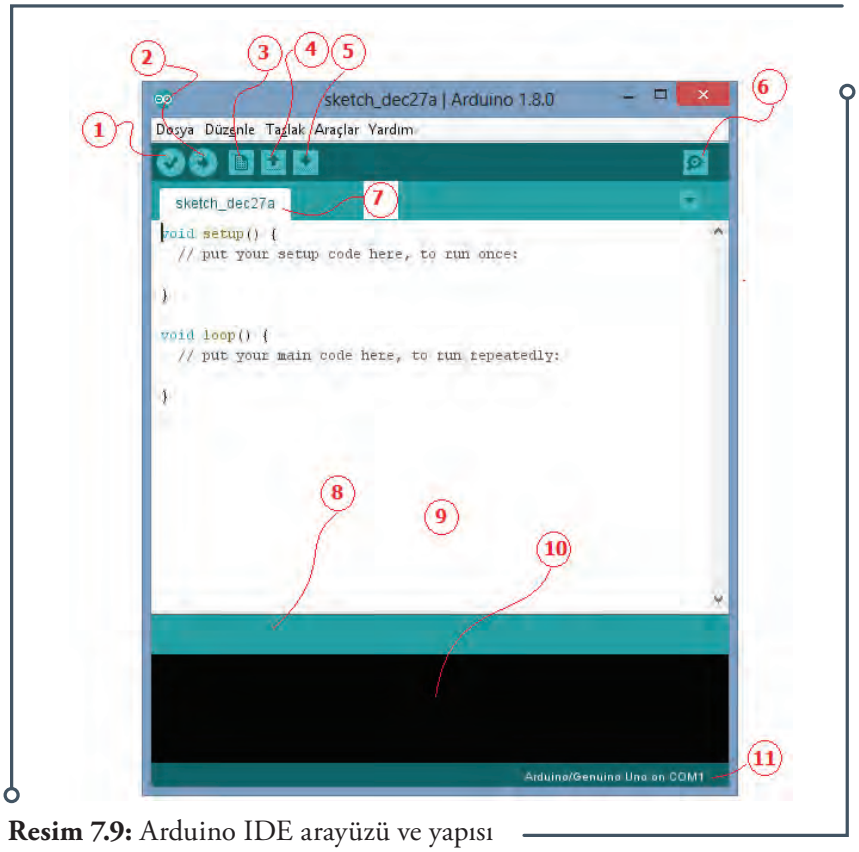


Resim 7.8: Arduino IDE kurulum klasörü

Program çalıştırıldığı zaman karşımıza aşağıdaki arayüz çıkmaktadır. Arayüz üzerinde bulunan sık kullanılan butonlar ve görevleri aşağıda açıklanmıştır.

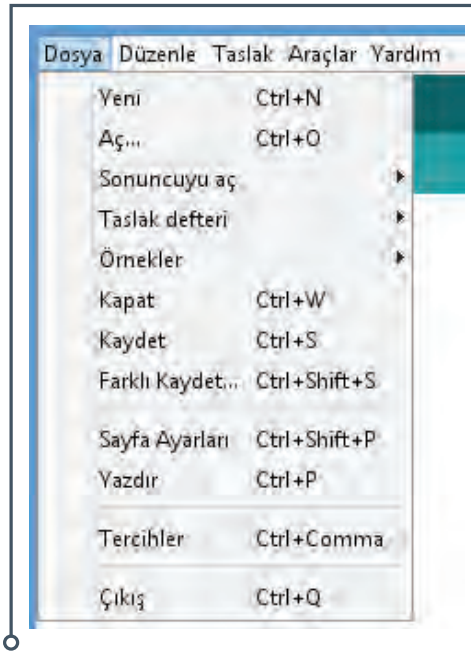
1. **Kontrol Et:** Yazılan kodları derler ve hataları bulur.
2. **Yükle:** Yazılan programı Arduino kartına yükler.
3. **Yeni:** Yeni çalışma sayfası açar.
4. **Aç:** Kayıtlı bir programı açar.
5. **Kaydet:** Yazılan programı kaydeder.
6. **Seri Port Ekranı:** Arduino ile seri iletişim yaparak ekran açar.
7. **Sketch:** Yazılan programın dosya ismini gösterir.
8. **Gösterge:** Yaptığı işlemin ilerleme durumunu gösterir.
9. **Boş alan:** Yazılacak program alanıdır.
10. **Rapor:** Derleme sonucu varsa yapılan hataları, yoksa programın yükleme sonrası mikrodenetleyicide kapladığı alanı gösterir.
11. **Gösterge:** Bilgisayara usb ile bağlanan Arduino'nun bağlandığı portu ve hangi Arduino modeli ile çalışılıyorsa onu gösterir.

Arayüzün üst kısmında temel bir menü çubuğu bulunmaktadır. Bu bölümde yer alan komutlarla da sık kullanılan butonların yaptığı işlevler yerine getirilebilmekte, buna ek olarak parça ayarlamaları, iletişim seçenekleri, ileri program ayarlamaları gibi işlevler de düzenlenebilmektedir.



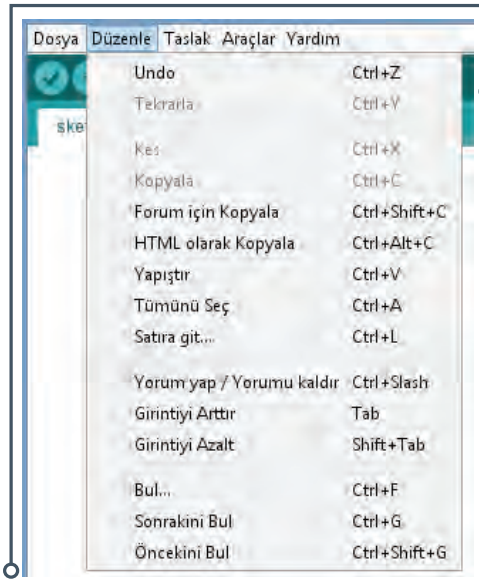
Resim 7.9: Arduino IDE arayüzü ve yapısı

Dosya menüsünde temel komutlar yer almaktadır. Ayrıca "Örnekler" komutu ile program kütüphanesinde yer alan kodlar incelenebilmektedir. Bu bölümde çok sayıda örnek bulunmakta, gerekli bağlantılar yapılarak görsel olarak incelenebilmekte, küçük değişiklikler yaparak etkisi görülebilmektedir. Bu bölüm yeni başlayan kullanıcılar için büyük kolaylık sağlamaktadır.



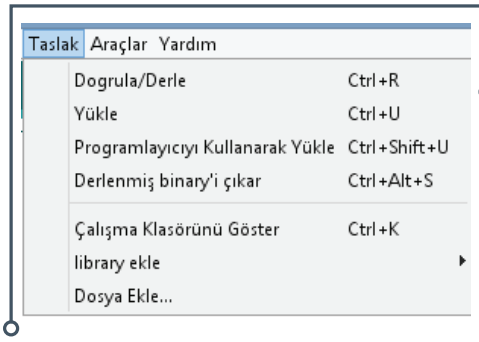
Resim 7.10: Arduino IDE dosya menüsü

Düzenle menüsünde standart işlemlerin dışında "Forum için Kopyala" ve "HTML olarak Kopyala" seçenekleri kullanılarak internet ortamında, Arduino arayüzünde yazılan biçimde (renk- yazı tipi) paylaşabilmektedir.



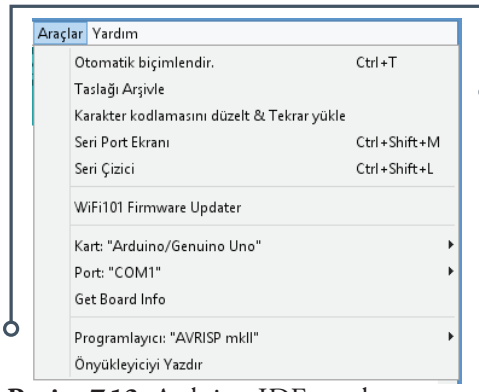
Resim 7.11: Arduino IDE düzenle menüsü

Taslak menüsünde standart işlemlerin dışında "library ekle" sekmesini kullanarak istenilen kütüphane otomatik olarak çalışılan koda eklenebilmektedir. Ayrıca başka bir kütüphane eklemek için "Dosya Ekle" kullanılabilmektedir.



Resim 7.12: Arduino IDE taslak menüsü

Araçlar menüsünde standart işlemlerin dışında "Kart" sekmesini kullanarak programlanacak Arduino modelinin seçimi, "Port" kısmından kullanılan Arduino'nun bilgisayarın hangi portuna bağladığı seçilmelidir. Arduino'nun bağlı olduğu port ile arayüzde seçili olan portun aynı olması gereklidir. Arduino destekli robotla bağlantı bu şekilde yapılmalıdır.



Resim 7.13: Arduino IDE araçlar menüsü

7.6. Arduino Tümleşik Geliştirme Ortamının Temel Özellikleri

- ✓ Arduino IDE tümleşik geliştirme ortamında basitleştirilmiş C++ kullanılır.
- ✓ Arduino programları genellikle tanımlamalar, kurulum ve ana program bloğu olmak üzere üç bölümden oluşur.
- ✓ Program yazımı belirli kalıpta, bloklar halinde gerçekleştirilir.
- ✓ Program kodları renkli olarak gösterilir. Kodların bulunduğu yerlerde gri renkte olan yazılar kodun ne işe yaradığı hakkında bilgi vermek için kullanılır.
- ✓ Arduino'ya yüklenen programlar kaldırılana kadar Arduino içinde kalır. Yüklemeden sonra bağımsız olarak çalıştırılabilir.
- ✓ Bloklar, { } parantezleri ile oluşturulur.
- ✓ Komutlar aynı veya alt alta satırlara yazılabilir. Fakat programın anlaşılabilirliği açısından alt alta yazmak daha uygundur.
- ✓ Tüm komutlar noktalı virgül (;) ile biter. Fakat blok başlatan ifadelerden sonra noktalı virgül kullanılmaz.
- ✓ Programda kullanılan tüm değişkenler ve bilgi tipleri bildirilir.
- ✓ Programın başında kullanılacak kütüphaneler aktifleştirilir /çağrılır.
- ✓ Açıklamalar "//" ve "/* */ " (birden fazla satır için) ile yazılır.
- ✓ Türkçe karakter kullanılmamalıdır. Fakat açıklama satırları içerisinde (derleme işlemine dâhil edilmediğinden) kullanılabilir.
- ✓ Eşdeğer ifadeler #define ile atanır.
- ✓ Kütüphaneler #include ile çağrılır.

7.7. Arduino Tümleşik Geliştirme Ortamının Bölümleri

Arduino programları yapı, değişkenler (değişkenler ve sabitler) ve fonksiyonlar olmak üzere üç ana bölümden oluşmaktadır. Her bölümde kullanılan yapılar, operatörler, işlev ve fonksiyonlar aşağıda verilmektedir.

A. PROGRAM YAPISI		
void setup() void loop()	**3. Aritmetik Operatörler**	**6. İşaretçi Operatörler**
1. Kontrol Yapıları	= Atama İşleci + Toplama - Çıkarma * Çarpma / Bölme % Modulo	* Referan dışı operatör & Referans operatörü
if if/else for switch/case while do/while break continue return goto	**4. Karşılaştırma Operatörleri**	**7. Bitsel Operatörler**
2. Söz Dizimi	== (eşit eşit) != (eşit değil) < (küçük) > (büyük) <= (küçük eşit) >= (büyük eşit)	& (Bitsel Ve) | | (Bitsel Veya) ^ (Bitsel Xor) ~ (Bitsel Değil) << (Bitshift Sol) >> (Bitshift Sağ)
; Noktalı Virgül {} Süslü Parantez // Çift Slash /**/ Yıldızlı Slash #define #include	**5. Boolean Operatörleri**	**8. Birleşik Operatörler**
	&&& (ve) || (veya) ! (değil)	++ (arttırma) -- (azaltma) +- (Birleşik artırma) -= (Birleşik çıkarma) *= (Birleşik çarpma) /= (Birleşik bölme) %= (Birleşik mod) &= (Bitsel Lojik Ve) |= (Bitsel Lojik Veya)

Tablo 7.1: Arduino IDE'nin program yapısı

B. DEĞİŞKENLER
1. Sabitler
HIGH | LOW
INPUT | OUTPUT | INPUT_PULLUP
LED BUILTIN
True | false
integer constants
floating point constants
2. Veri Tipleri
void
boolean
char
unsigned char
byte
int
unsigned int
word
long
unsigned long
short
dounle
string – char array
substring – object
array
3. Dönüşümler
char()
byte()
int()
word()
long()
float()
4. Değişken Kapsamları
static
volatile
const
5. Yardımcılar
PROGMEM
Sizeof()

C. FONKSİYONLAR	
1. Dijital Giriş Çıkışlar	**8. Rastgele Sayılar**
pinMode(pin,mod)	randomSeed()
digitalWrite(pin,değer)	random()
digitalRead(pin)	**9. Bit ve Bayt'lar**
2. Analog Giriş Çıkışlar	lowByte()
analogRead(pin,mod)	high(Byte()
analogWrite(pin,değer)	bitRead()
analogReference(tip)	bitWrite()
analogReadResolution ()	bitSet()
analogWriteResolution ()	bitClear()
3. Gelişmiş Giriş Çıkışlar	bit()
tone()	**10. Harici İnterruptlar (Kesmeler)**
noTone()	attachInterruption(interrupt, function, mode)
shiftOut()	detachInterrupt(interrupt)
shiftIn()	**11. İnterruptlar (Kesmeler)**
pulseIn()	interrupts()
4. Gecikmeler	noInterrupts()
delay(milisaniye)	**12. Seri Haberleşme**
unsigned long milis()	Serial.begin(hız)
delay(Microse conds (mikrosaniye)	int Serial.available()
5. Matematiksel İşlevler	int Serial.read()
min(x,y)	Serial.flush()
max(x,y)	Serial.print(data)
abs(x)	Serial.printIn(data)
constrain(x, a, b)	**13. Haberleşme Protokolleri**
map(value, fromLow fromHigh, toLow, toHigh)	I2C Veri Yolu
pow(base, esponent)	SPI Veri Yolu
sqrt(x)	
6. Trigonometri İşlevleri	
sin(raf)	
cos(rad)	
tan(rad)	
7. Karakterler	
isAlphaNumeric()	
isAlpha()	
isAscii()	
isWhiteSpace()	
isDigit()	
isGraph()	
isPrintable()	
isPunct()	
isSpace()	
isUpperCase()	
isHexaDecimalDigit()	

Tablo 7.2: Arduino IDE'de kullanılan değişken ve fonksiyon yapısı

7.8. Arduino Tümleşik Geliştirme Ortamının "Program" Yapısı

void setup()

void setup() fonksiyonu program yüklenip enerji verildikten veya tekrar başlatıldıktan sonra 1 defa çalışan fonksiyondur. "void se-

```
int buttonPin = 4; // butonu 4. pine tanımlandı
void setup()
{
  Serial.begin(9600); // Seri haberleşme başlatıldı
  pinMode(buttonPin, INPUT); // Pin modu tanımlandı
}
void loop()
{
  // ...
}
```

Resim 7.14: void setup() fonksiyonu örneği

tup()" ile başlayan satır, takip eden tırnak içindeki bölümde temel ayarların yapılacağını belirtmektedir. Bu fonksiyon içine pin modları, kütüphaneyi başlatma ve değişkenler yazılmaktadır. Burada yapılan ayarlamalarda hangi mikrodenetleyici pininin (veri bacağının) giriş (input-veri çekilen port-) ya da çıkış (output-veri gönderilen port-) olduğu belirtilmektedir. Örnekte (Resim 7.15) "void setup" içinde seri haberleşme başlatılmış ve pin modu tanımlanmıştır.

void loop()

Void loop() fonksiyonuna setup işleminden sonra eklenen ve mikrodenetleyici ya da Arduino'nun beslemesi devam ettiği sürece tekrarlanan komutlar yazılmaktadır. Buraya yazılan komutlar ile Arduino pinleri arasından karşılaştırma, ilişkilendirme, matematiksel işlemler vs. yapılmaktadır. Yazılan program burada sonsuz döngü içinde çalışmaktadır. Aşağıdaki örnekte tanımlanmış olan butuna basıldıysa seri ekrana "Basıldı", basılmadıysa "Basılmadı" yazan küçük bir program "void loop" içine yazılmıştır.

```
int buttonPin = 4; // butonu 4. pine tanımlandı

void setup()
{
  Serial.begin(9600); // Seri haberleşme başlatıldı
  pinMode(buttonPin, INPUT); // Pin modu tanımlandı
}

void loop()
{
  if (digitalRead(buttonPin) == HIGH) //okunan buton 1 ise
  Serial.write('Basıldı'); // Seri ekrana yaz
  else
  Serial.write('Basılmadı');
  delay(500);
}
```

Resim 7.15: void loop() fonksiyonu örneği

7.8.1. Kontrol Yapıları

#if

#if deyimi koşullu ifadeleri yürütmek için kullanılır. Örneğin belirlenen butona basıldıysa LED'i yak gibi durumlarda veya bir karşılaştırma operatörüyle birlikte karşılaştırmalarda kullanılır. Parantez içinde verilen sayı veya ifade ile belirlenen koşula ulaşılıp ulaşılamadığını test eder. Parantez içindeki ifadeler karşılanıyorsa, parantez içindeki ifadeler çalıştırılır. Değilse, program kod üzerinde atlanır. Aşağıda verilen örnekte Arduino UNO'nun 4 numaralı dijital pinine bağlı bir LED'in yanık kalma süresini kontrol eden #if deyimi uygulaması yer almaktadır. Arduino UNO'ya LED bağlantılarını yaptıktan sonra örneği dersin sitesinden (7.8.1.1 numaralı uygulama) kopyalayarak Arduino IDE'de test ediniz.

```
int buton = 4; // butonu 4. pine tanımladık
int led = 10; // ledi 10. pine tanımladık

void setup() {
pinMode(buton, INPUT); // 4. pin giriş oldu
pinMode(led, OUTPUT); // 10. pin çıkış oldu
}

void loop(){//sonsuz döngü
if (digitalRead(buton) == HIGH) { //okunan buton 1 ise
digitalWrite(led, HIGH); // ledi yak
}
}
```

Resim 7.16: #if örneği

```
int Led = 4; // Arduino 4 numaralı dijital pinine LED bağlanıyor
 // LED'in önüne 220/330 ohm değerinde bir direnç bağlanmalıdır

void setup()
 {
 pinMode(Led, OUTPUT);      // LED çıkış olarak tanımlanıyor
 }
 int gecikmeSuresi = 1000; // Başlangıç değeri 1000 olan bir değişken
tanımlanıyor

void loop()
 {
 gecikmeSuresi = gecikmeSuresi - 100; //Değişkenden her işlem için
100 değeri çıkartılıyor
 if(gecikmeSuresi <= 0)    // Eğer gecikme süresi sıfır veya daha
az ise
 {
 gecikmeSuresi = 1000;     //  Şart gerçekleşirse gecikme süresi
tekrar 1000 yapılıyor
 }
 digitalWrite(Led, HIGH);  // LED yanıyor
 delay(gecikmeSuresi);     // Gecikme süresi kadar bekleniyor
 digitalWrite(Led, LOW);   // LED söndürülüyor
 delay(gecikmeSuresi);     // Gecikme süresi kadar bekleniyor
}
```

#if/else

#if/else deyimi koşullu ifadeleri yürütmek için kullanılır. Temel kod akışı üzerinde daha fazla denetim sağlar. "if" eğer, "else" ise değil demektir. "if" ve "else" birlikte kullanılır. "else" tek başına kullanılamaz. Parantez içinde verilen sayı veya ifade ile belirlenen koşula ulaşılıyorsa bir eylem, ulaşılamıyorsa başka bir eylem yapılır. Aşağıda verilen örnekte Arduino UNO'nun 4 numaralı dijital pinine bağlı bir LED'in yanık kalma süresini kontrol eden if/else deyimi uygulaması yer almaktadır. #if için yapılan örneğe #else eklenerek oluşturulmuştur. Arduino UNO'ya LED bağlantılarını yaptıktan sonra örneği dersin sitesinden (7.8.1.2

```
int buton = 4; // butonu 4. pine tanımlandı
int led = 10; // ledi 10. pine tanımlandı

void setup() {
pinMode(buton, INPUT); // 4. pin giriş yapıldı
pinMode(led, OUTPUT); } // 10. pin çıkış yapıldı

void loop(){//sonsuz döngü
if (digitalRead(buton) == HIGH) { //okunan buton 1 ise
digitalWrite(led, HIGH);} // ledi yak
else { // değil ise
digitalWrite(led, LOW); }} // ledi söndür
```

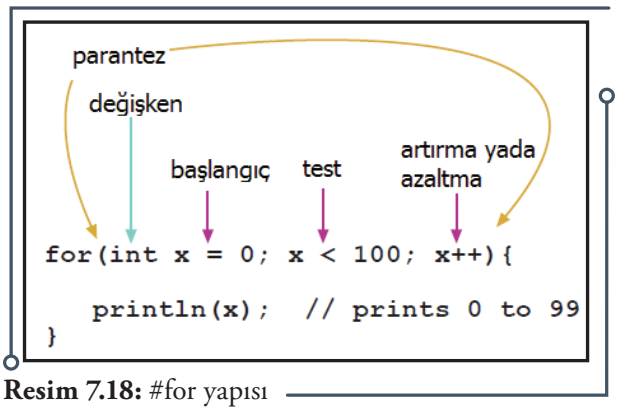
Resim 7.17: #if /else örneği

numaralı uygulama) kopyalayarak Arduino IDE'de test ediniz. İlk örnekle aralarındaki benzerlik ve farklılıkları inceleyiniz.

```
int Led = 4; // Arduino 4 numaralı dijital pinine LED bağlanıyor
 // LED'in önüne 220/330 ohm değerinde bir direnç bağlanmalıdır

void setup()
{
 pinMode(Led, OUTPUT);      // LED çıkış olarak tanımlanıyor
 }
 int gecikmeSuresi = 1000; //  Başlangıç  değeri  1000  olan  bir
değişken tanımlanıyor

void loop() {
 if(gecikmeSuresi <= 0)      // Eğer gecikme süresi sıfır veya daha az
ise
 {
 gecikmeSuresi = 1000;      //  Şart  gerçekleşirse  gecikme  süresi
tekrar 1000 yapılıyor
 }
 else                        // Değilse
 {
 gecikmeSuresi = gecikmeSuresi - 100; // Değişkenden her işlem için
100 değeri çıkartılıyor }
 digitalWrite(Led, HIGH);   // LED yanıyor
 delay(gecikmeSuresi);      // Gecikme süresi kadar bekleniyor
 digitalWrite(Led, LOW);    // LED söndürülüyor
 delay(gecikmeSuresi);      // Gecikme süresi kadar bekleniyor
}
```

#for

#for deyimi küme parantezi içine alınmış bir deyim bloğunu tekrarlamak için kullanılır. Döngüyü artırmak ve sonlandırmak için genellikle bir artış sayacı kullanılır. For ifadesi tekrar eden işlemler için yararlıdır ve genellikle veri / pin topluluğu üzerinde çalışmak için dizilerle birlikte kullanılır. For döngüsü üstbilgisinde başlatma, koşul ve artırma olmak üzere üç bölüm bulunmaktadır. İlk başlatma tam olarak bir kez olur. Döngüde her defasında koşul test edilir; eğer doğruysa, ifade bloğu ve artım yürütülür, koşul tekrar test edilir. Koşul yanlış olduğunda ise döngü sona erer. Yandaki örneği inceleyiniz. Aşağıda verilen diğer örnekte Arduino UNO'nun 4 numaralı dijital pinine bağlı bir



Resim 7.18: #for yapısı

LED'in birer saniye aralıklarla 10 defa yandıktan sonra döngüye tekrar başlayan #for deyimi uygulaması yer almaktadır. Arduino UNO'ya LED bağlantılarını yaptıktan sonra örneği dersin sitesinden (7.8.1.3 numaralı uygulama) kopyalayarak Arduino IDE'de test ediniz.

```
// PWM pini ile LED kontrolu
int PWMpin = 13; // LED 13. Pin'e tanımlandı
void setup()
{
  // setup gerekli değildir
}
void loop()
{
   for (int i=0; i <= 255; i++){
      analogWrite(PWMpin, i);
      delay(10);
   }
}
```

Resim 7.19: #for örneği

```
int Led = 4; // Arduino 4 numaralı dijital pinine LED bağlanıyor
 // LED'in önüne 220/330 ohm değerinde bir direnç bağlanmalıdır
void setup() {
 pinMode(Led, OUTPUT);       // LED çıkış olarak tanımlanıyor
 }
 int gecikmeSuresi = 1000;  //  Gecikme  süresi  1sn  olarak  bir
değişkenle tanımlanıyor

void loop() {
 for(int m = 0; m < 10; m++) // Döngü boyunca m değeri artırılıyor
 {
 digitalWrite(Led, HIGH);    // LED yanıyor
 delay (gecikmeSuresi);      // Gecikme süresi kadar (1 sn) bekleniyor
 digitalWrite(Led, LOW);     // LED söndürülüyor
 delay(gecikmeSuresi);       // Gecikme süresi kadar (1 sn) bekleniyor
 }
 delay(2000);                // 2 sn bekleniyor
}
```

#switch/case

#switch/case bir ifadenin sabit değerlerinden birisiyle eşleşip eşleşmediğini test eden çok yönlü bir karar verme yapısıdır. Bir programda çok sayıda koşul kontrolü ve bunların sonucuna göre gerçekleştirilmesi gereken işlemler varsa kullanılır. Bir "switch" deyimi bir değişkenin değerini "case" ifadelerinde belirtilen değerlerle karşılaştırır. Değişkenin değeriyle eşleşen bir "case" ifadesi bulunursa, bu "case" ifadesinin kodu çalıştırılır. Bir switch/case yapısından çıkışı sağlamak ya da sonlandırmak için "break" ya da "return" kullanılmalıdır. "switch" ifadesinden hemen sonra gelen ifade parantez içinde yer almalı ve bir tamsayı veya değişken olmalıdır. "case" ifadesini izleyen ifadeler tamsayı türünde ifadeler olmalıdır, yani değişken içermemelidir. Aşağıdaki örneği inceleyiniz.

```
/* Arduino'nun A1 girişine bir potansiyometrenin
   çıkış pini girdiğimizi düşünelim */
void setup() {
Serial.begin(9600); //Seri haberleşme
}
void loop() {
int okunandeger = analogRead(A1);
//map fonksiyonu okuduğumuz değeri (0-1023 aralığını) 2 ye böler
//0-511 ise aralığa 0 değerini verir.
//512-1023 ise aralığa 1 değerini verir.
int aralik= map(okunandeger, 0, 1023, 0, 1);
switch (aralik) {
case 0: //Pot döndürülmesi 0-49% arasında ise
Serial.println("DEĞER DÜŞÜK"); //Seri port ekranına yazdır
break; // döngüden çıkış
case 1: // Pot döndürülmesi 50-100% arasında ise
Serial.println("DEĞER YÜKSEK"); //Seri port ekranına yazdır
break; } // döngüden çıkış
delay(1); // stabil çalışması için 1 milisaniye beklenir
}
```

Resim 7.20: #switch/case örneği

#while

#while döngüsü başlamadan önce koşul kontrol edilir. Belirtilen koşul doğru olduğu sürece, döngü içindeki işlemler gerçekleştirilir. Örnekte while döngüsü; LED'i yakacak, 1 saniye bekleyip LED'i söndürecek ve işlemi 1 arttıracaktır. While döngüsünde 10 kere aynı işlem yapıldıktan sonra döngüden çıkılacaktır. Aşağıda verilen örnekte Arduino UNO'nun 4 numaralı dijital pinine bağlı bir LED'in yanık kalma süresini kontrol eden #while deyimi uygulaması yer almaktadır. Arduino UNO'ya LED bağlantılarını yaptıktan sonra örneği dersin sitesinden (7.8.1.4 numaralı uygulama) kopyalayarak Arduino IDE'de test ediniz. #if örneğindeki uygulama ile aralarındaki benzerlik ve farklılıkları inceleyiniz.

```
int deneme=0; int led=0;
void setup() { }
void loop() { while (deneme<10)
digitalWrite(led, HIGH); // ledi yak
delay(1000); //1 saniye bekle
digitalWrite(led, LOW); // ledi söndür
deneme=deneme+1;
}
```

Resim 7.21: #while örneği

```
int Led = 4; // Arduino 4 numaralı dijital pinine LED bağlanıyor
 // LED'in önüne 220/330 ohm değerinde bir direnç bağlanmalıdır
void setup() {
 pinMode(Led, OUTPUT); }        // LED çıkış olarak tanımlanıyor
int gecikmeSuresi = 1000;       // Başlangıç değeri 1000 olan bir
değişken tanımlanıyor

void loop() {
 while (gecikmeSuresi > 0) {   // Gecikme süresi sıfırdan büyükse
 digitalWrite(Led, HIGH);      // LED yanıyor
 delay(gecikmeSuresi);         // Gecikme süresi kadar bekleniyor
 digitalWrite(Led, LOW);       // LED söndürülüyor
 delay(gecikmeSuresi);         // Gecikme süresi kadar bekleniyor
gecikmeSuresi = gecikmeSuresi - 100; // Değişkenden her işlem için
100 değeri çıkartılıyor
 }
 while (gecikmeSuresi < 1000) {  // Gecikme süresi 1000'den küçükse
gecikmeSuresi = gecikmeSuresi + 100; // Değişkenden her işlem için
100 değeri ekleniyor
 digitalWrite(Led, HIGH);      // LED yanıyor
 delay(gecikmeSuresi);         // Gecikme süresi kadar bekleniyor
 digitalWrite(Led, LOW);       // LED söndürülüyor
 delay(gecikmeSuresi);         // Gecikme süresi kadar bekleniyor
 }
}
```

#do/while

#do/while operatörü karşılaştırmalarda koşul içeriyorsa parantez içinde belirtilen "do" ifadesine göndererek tekrar hesaplama yaptırarak yeni değeri karşılaştırır. Örnekteki program "do" döngüsüne girecek sayıyı 2 arttıracak, seri ekranına yazdıracak "while" ile durumu kontrol edecektir. Sayı 10'dan büyük olduğu zaman döngüden çıkacaktır.

```
void setup() {
int sayi = 0;
Serial.begin(9600); // seri haberleşme
do { // 10 e kadar 2 şer sayma
sayi = sayi+ 2;
Serial.print("sayi = ");
Serial.println(sayi);
delay(500);
// 500ms bekleme
} while (sayi< 10);
}
void loop() {
}
```

Resim 7.22: #do/while örneği

#break

#break do, for ve while döngülerinde döngü çalışması bittiğinde döngü dışına çıkmak için kullanılmaktadır. Switch/case yapısında da kullanılır. Örnekte sensör değeri 200'den büyük olursa döngü dışına çıkılacaktır.

```
for (x = 0; x < 255; x ++)
{
    analogWrite(PWMpin, x);
    sens = analogRead(sensorPin);
    if (sens > 200){ // sensor çıkışı tespit ediliyor
       x = 0;
       break;
    }
    delay(50);
}
```

Resim 7.23: #break örneği

#continue

#continue do, for ve while döngülerinde bir satırın, işlem yapılmadan geçilmesini istediğimiz durumlarda kullanılmaktadır. Döngünün geri kalan kısmını kontrol etmeye devam eder.

```
for(x=0;x<255;x++){
 if(x>40&&x<100){
 continue; // x değeri 100 den büyük ve
 //100 küçükse atla
 }
 analogWrite(PWMpin, x);
 delay(50);
}
```

Resim 7.24: #continue örneği

#return

#return bir fonksiyon sonlandırılmak istenirse "return" ile döndürülecek değer belirtilmelidir. Returnun ikinci bir kullanım şekli de belli bir yerden sonra kodların çalışmasının istendiği durumlarda kullanılmasıdır. Yandaki örnekte okunan algılayıcı değeri 100'den büyük ise 1 değerine, 100'den küçük ise sıfır değerine döndürülmektedir. Aşağıda ise çalışması istenmeyen kodların nasıl yazılacağı gösterilmiştir.

```
int sensorkontrol () {
  if(analogRead(0) > 100) {
return 1 ;
}
  else
return 0;
}
```

Resim 7.25: #return örneği

```
void loop ( ) {
// çalışması isteten kodlar
return; //çalışması istenmeyen kodlar buraya yazılır
```

Resim 7.26: #return diğer kullanım örneği

#goto

#goto program akışını istenilen yöne yönlendirmek için kullanılmaktadır. Derin iç içe geçmiş döngülerde veya "if" mantık bloklarından belirli bir şartla ayrılma durumunda kullanışlı olmakta ve kodlamayı basitleştirmek için tercih edilmektedir.

```
void loop() {
int x = analogRead(1);
if (x < 100) {
goto git;
}
git:
delay(1000);
// çalışması istenilen kod}
```

Resim 7.27: #goto örneği

7.8.2. Söz Dizimi

; Noktalı Virgül

C programlama dilinde her satır programından sonra noktalı virgül konulması, sonlandırılması gerekmektedir. Noktalı virgül konulmadığı durumlarda derleme hatası oluşmaktadır.

```
void loop() {
int x = analogRead(1);
```

Resim 7.28: ;noktalı virgül örneği

{} Süslü Parantez

Fonksiyonlarda, döngülerde ve koşullu ifadelerin bildirilmesinde süslü parantez kullanılmaktadır. İç içe olan fonksiyonlarda en dıştaki süslü parantezin en baştaki fonksiyona ait olduğuna dikkat edilmeli, aksi takdirde derleme hatası ortaya çıkmaktadır.

```
void fonksiyonlarim()
{
//yapılacaklar }
while () {
//yapılacaklar }
do {
//yapılacaklar
}
for (x=0;x<=100;x++)
{
//yapılacaklar }
```

Resim 7.29: {} süslü parantez örneği

// Çift Slash

Program satırından, program başında veya herhangi bir yerde programın çalışma şekli hakkında açıklama yapmak isteniyorsa çift slash kullanılmalıdır. Çift slashtan sonra yazılanlar program kodu olarak dikkate alınmaz. Derleyici tarafından yok sayılır ve mikroişlemciye aktarılmaz. Örneği (Resim 7.30) inceleyiniz.

```
X = 5; // Bu, tek satırlı bir açıklamadır. Çift slash sonrasındaki her şey bir açıklamadır
       // satırın sonuna çift slash eklenmelidir.
/ * bu satırlı bir açıklamadır - tüm kod bloklarını açıklamak için kullanabilirsiniz
(gwb == 0) {  // Tek satırlık açıklama tek satır olarak kullanılabilir
X = 3;        / * Ancak başka satırlar kullanılacaksa - yukarıdaki şekilde geçersizdir.
Bu şekilde kullanılmalıdır. * /
}
// "kapanış" slahsını unutmayın
* /
```

Resim 7.30: / / Çift slash örneği

/**/ Yıldızlı Slash

/* Buraya yazılan her türlü satır, sütun dahil program olarak alınmamakta, açıklama olarak dikkate alınmaktadır. */ Yukarıdaki örneği inceleyiniz.

#define

#define ön işlemci komutu olup kullanılan bir isim yerine başka bir ismin kullanımını ve değişimini sağlamaktadır. Programın derlenmesinden önce programcının sabit bir değere isim vermesine izin vermektedir. "#" işaretinin kullanılması gereklidir. #define deyiminden sonra hiçbir noktalı virgül veya eşittir bulunmamalıdır. Yanda verilen örneğe göre, programda ledpin görüldüğü yere 12 rakamı yerleştirilmelidir.

```
#define ledpin 12
```

Resim 7.31: #define örneği

#include

#include Arduino için yazılmış kütüphanelere erişimi sağlamak için kullanılmaktadır. Diğer bir ifadeyle üzerinde çalışılan program dışındaki kütüphanelere erişmek için kullanılan fonksiyondur. Örneğin yapılan programda SPI kütüphanesini kullanmak ve onun içerisindeki komutlara ulaşmak istendiğinde program başı tanımlanması ("...") ya da (<...>) şekilde olmak üzere iki şekilde yapılmaktadır. #include deyiminden sonra hiçbir noktalı virgül veya eşittir bulunmamalıdır.

```
#include "SPI.h" //Tanımlama 1. tanımlama şekli
#include <SPI.h> //Tanımlama 2. tanımlama şekli
```

Resim 7.32: #include kullanım şekli

7.8.3. Aritmetik Operatörler

= Atama İşleci

C programlama dilindeki tek eşit işareti atama operatörü olarak adlandırılır. Bir denklem veya eşitlik gösterdiği cebir sınıfından farklı bir anlam taşır. Atama işleci, mikrodenetleyiciye eşittir işaretinin sağ tarafında bulunan herhangi bir değer veya ifadeyi değerlendirmesini ve eşit işaretin solundaki değişkende saklamasını söyler.

```
int algiDeg;              // algiDeg adını taşıyan bir tamsayı değişkeni
algiDeg = analogRead(1); // analog pin 1 in giriş gerilimi algiVal de saklanmaktadır
```

Resim 7.33: = Atama işleci kullanım örneği

+ Toplama, - Çıkarma, * Çarpma ve / Bölme

C'de programlama yaparken matematiksel işlemlerde aritmetik operatörler kullanılır. İnt ya da float (virgüllü) değerinden sonuçlar bulunabilir.

```
Y = y + 4;
X = x - 8;
I = j * 7;
R = r / 6;
```

Resim 7.34:
4 işlem örneği

% Mod

Bir tam sayı belirlenen bir bölüme ayrıldığında kalanını hesaplar. Belirli bir aralıktaki bir değişkeni tutmak için de kullanılmaktadır.

```
x = 7% 5; // X şimdi 2 içerir
x = 9% 5; // X şimdi 4 içerir
x = 5% 5; // X şimdi 0 içerir
x = 4% 5; // x şimdi 4 içerir
```

Resim 7.35: % Mod örneği

7.8.4. Karşılaştırma Operatörleri

== (eşit eşit), != (eşit değil), < (küçük), > (büyük), <= (küçük eşit), >= (büyük eşit)

Karşılaştırma operatörlerinin kullanımı aşağıda verilmiştir. Genellikle if içerisinde karşılaştırma yaparken kullanılan operatörlerdir. "if" karşılaştırma operatörüyle birlikte kullanıldığında, belli bir değerin üzerinde olup olmadığı test edilir. Parantez içindeki ifadeler doğruysa parantez içindeki ifadeler çalıştırılır. Doğru değilse program kod üzerinde atlanır.

```
x == y (x, y ye eşit)
x != y (x, y ye eşit değil )
x < y (x, y den küçük)
x > y (x, y den büyük)
x <= y (x, y den küçük eşit)
x >= y (x, y den büyük eşit)
```

Resim 7.36: Karşılaştırma operatörlerinin kullanım örneği

```
if (degisken> 100)
{
//değişken 100 den büyükse buraya girilir
}
```

Resim 7.37: Karşılaştırma operatörlerinin "if" içinde kullanım şekli

7.8.5. Boolean Operatörleri

&& (mantıksal ve)

Boolean ifadeleri genellikle "if" yapısının içerisinde ve belirli şart gerektiren durumlarda kullanılmaktadır. Yalnızca her iki işlem de doğruysa "if" şartı doğrudur. Bu durumda "if" içerisindeki komut çalıştırılır. Herhangi bir durum ya da ikisi de false, yanlış sonuç ise if yapısı çalıştırılmaz. Örnekte oku1 ve oku2 HIGH ise LED yanmaktadır.

```
void loop(){ //sonsuz döngü
int oku1= digitalRead(1);
int oku2= digitalRead(2);
if (oku1 == HIGH && oku2 == HIGH) {
digitalWrite(led, HIGH); // ledi yak
}
}
```

Resim 7.38: && (mantıksal ve) kullanım örneği

|| (mantıksal veya)

Her iki işlemden herhangi birisi doğruluk şartını taşıyorsa if içerisindeki komut çalıştırılır. Örnekte oku1 veya oku2 HIGH ise led yanmaktadır.

```
void loop(){ //sonsuz döngü
int oku1= digitalRead(1);
int oku2= digitalRead(2);
if (oku1 == HIGH || oku2 == HIGH) {
digitalWrite(led, HIGH); // ledi yak
}
}
```

Resim 7.39: || (mantıksal veya) kullanım örneği

! (mantıksal değil)

Değer olarak verilen ifadenin sıfır olma durumudur. İfadenin sıfır olma şartı sağlanıyor ise if yapısı çalıştırılmaktadır. Örnekte buton1'e basılmıyor ise LED'i söndürülmektedir.

```
if (!buton1) {
digitalWrite(led, LOW); // ledi söndür
}
```

Resim 7.40: ! (mantıksal değil) kullanım örneği

7.8.6. İşaretçi Operatörler

& Referans operatörü, * Referans dışı operatör

C programlama dilinde bazı veri yapılarını değiştirmek, bir değişkene başvuru yoluyla geçmek için bu işaretçilerin kullanılması kodun basitleştirilmesini sağlamaktadır. Örneğin bazı durumlarda bir değişkene bir şey yapılması ve diğer durumlarda da aynı şeyin başka bir değişkene yapılması gereken durumlarda kullanılmaktadır. Esasen iki farklı değişken üzerinde aynı işlemi gerçekleştiren iki kod sırası yerine, hangi değişkenin üzerinde çalışacağını seçen bir bit kodu bulunmaktadır. C / C ++ 'da değişken geçirmenin en temel biçimi değeri geçmektir ve bir değişken başka bir değişkene atandığında, aslında o değişkenin bir kopyası oluşturulduğu anlamına gelmektedir. Aşağıdaki örneği inceleyiniz.

```
int a = 1;
int b = a;
B + = 1;
// şimdi a == 2 ve b == 2
```

```
int a = 1;
int * b = & a;
* B + = 1;
// şimdi a == 2 (ve b'nin değeri, bir bellekteki konuma
// işaret eden bir işaretçidir)
```

Resim 7.41: & Referans ve * Referans dışı operatör kullanım örneği

7.8.7. Bitsel Operatörler

& (bitsel ve)

Bitsel operatörler hesaplamalarını değişkenlerin bit düzeyinde gerçekleştirir. Bitsel ve, işleme giren bitlerin ve'sini verirler. Yani her iki giriş biti de 1 ise, sonuç 1; aksi halde sonuç 0 olmaktadır. Örneği inceleyiniz.

```
0 0 1 1 operand1
0 1 0 1 operand2
----------
0 0 0 1 (operand1 & operand2) - sonuç
```

Resim 7.42: & (bitsel ve) kullanım örneği

```
int a = 92; // ikili olarak: 0000000001011100
int b = 101; // ikili olarak: 0000000001100101
int c = a & b; // sonuç: 0000000001000100, veya ondalık 68
```

Resim 7.43: & (bitsel ve) 2. kullanım örneği

| (bitsel veya)

Giriş bitlerinden biri veya her ikisi birden 1 ise iki bitlik bitsel veya 1, aksi halde sonuç 0 olmaktadır. Örneği inceleyiniz.

```
0 0 1 1 operand1
0 1 0 1 operand2
----------
0 1 1 1 (operand1 | operand2) - sonuç
```

Resim 7.44: | (bitsel veya) kullanım örneği

```
int a = 92; // ikili olarak: 0000000001011100
int b = 101; // ikili olarak: 0000000001100101
int c = a | B; // sonucu: 0000000001111101 veya ondalık sayı125
```

Resim 7.45: | (bitsel veya) 2. kullanım örneği

^ (bitsel xor)

Bu operatör, "bitsel ve" operatörüne çok benzemektedir. Yalnızca o pozisyon için her iki giriş biti 1 olduğunda verilen bir bit pozisyonu için 0 olarak değerlendirilir. Örnekleri inceleyiniz.

```
0 0 1 1 operand1
0 1 0 1 operand2
----------
0 1 1 0 (operand1 ^ operand2) - döndürülen sonuç
```

Resim 7.46: ^ (bitsel xor) kullanım örneği

```
int x = 12; // binary: 1100
int y = 10; // binary: 1010
int z = x ^ y; // binary: 0110 veya ondalık 6
```

Resim 7.47: ^ (bitsel xor) 2. kullanım örneği

~ (bitsel değil)

Bitsel değil operatörü sağdaki tek bir işlenene uygulanır. Bitsel değil uygulandığında her bit için tersi olmaktadır: 0 1 olur ve 1 0 olur. En yüksek bit 1 ise sayı negatif olarak yorumlanır. Örneği inceleyiniz.

```
0  1    operand1
----------
1  0   ~ operand1
```

Resim 7.48: ~ (bitsel değil) kullanım örneği

```
int a = 103;  // binary:  0000000001100111
int b = ~a;   // binary:  1111111110011000 = -104
```

Resim 7.49: ~ (bitsel değil) 2. kullanım örneği

<< (bitshift sol) ve >> (bitshift sağ)

Bu operatörler, sol işlenendeki bitlerin sağ işlenen tarafından belirtilen konum sayısına göre sola veya sağa kaydırılmasına neden olur.

```
int a = 5; // binary: 0000000000000101
int b = a << 3; // binary: 0000000000101000 veya ondalıklı 40
int c = b >> 3; // binary: 0000000000000101 veya başlanılan yere 5'e dönüş
```

Resim 7.50: << (bitshift sol) ve >> (bitshift sağ) kullanım örneği

7.8.8. Birleşik Operatörler

a++ (arttırma) ve -- (azaltma)

Bir değişkende arttırma veya azaltma yapılmasını sağlamaktadır. Aşağıdaki örnekleri inceleyiniz.

```
X = 2;
Y = ++ x; // şimdi x 3 içerir  // y 3 içerir
y = x--; // x tekrar 2 içerir, y 4'ü içerir
```

Resim 7.51: ++ (arttırma) ve -- (azaltma) kullanım örneği

```
X ++; // x değerini birer arttırır ve eski x değerini döndürür
++ x; // x değerini birer artırır ve x'in yeni değerini döndürür

X--; // x değerini birer azaltır ve eski x değerini döndürür
--x; // x değerini birer azaltır ve yeni x değerini döndürür
```

Resim 7.52: << (bitshift sol) ve >> (bitshift sağ) 2. kullanım örneği

+= (birleşik artırma), -= (birleşik çıkarma), *= (birleşik çarpma), /= (birleşik bölme) ve %= (birleşik mod)

Bu operatörler değişken üzerinde başka bir sabit veya değişkenle matematiksel işlem yapmak için kullanılmaktadır. Örneği inceleyiniz.

```
x = 2;
x += 4;      // x şimdi 6 içerir
x -= 3;      // x şimdi 3 içerir
x *= 10;     // x şimdi 30 içerir
x /= 2;      // x şimdi 15 içerir
x %= 5;      // x şimdi 0 içerir**

**Mod alınıyor. Mod alma x in y ile
bölümünden kalan sayıdır.
x=15%5; olduğunda x=0 olacaktır.
```

```
X + = y; // x = x + y ifadesine eşdeğerdir
X - = y; // x = x - y ifadesine eşdeğerdir
X * = y; // x = x * y ifadesine eşdeğerdir
X / = y; // x = x / y ifadesine eşdeğerdir
X% = y; //  x = x% y; ifadesine eşdeğerdir

X: herhangi bir değişken türü
Y: herhangi bir değişken türü veya sabit
```

Resim 7.53: += (birleşik artırma), -= (birleşik çıkarma), *= (birleşik çarpma), /= (birleşik bölme) ve %= (birleşik mod) kullanım örnekleri

&= (bitsel lojik ve)

Bitsel Lojik Ve değişkendeki belirli bitleri LOW (düşük) durumuna zorlamak için genellikle bir değişken ve bir sabitle kullanılır. Buna programlama kılavuzlarında "temizleme" veya "sıfırlama" bitleri denmektedir. Bitsel Lojik Ve bir değişkenin geri kalanını değiştirmeden bırakmak, değişkenin 0 ve 1 bitlerini sıfırlamak (sıfıra ayarlamak) için kullanılmaktadır.

```
1 0 1 0 1 0 1 0 değişken
1 1 1 1 1 1 0 0 maske
-----------------------
1 0 1 0 1 0 0 0

Değişken değişmez
Bitler silindi. Aşağıdaki örneğe bakın

MyByte = B10101010;
MyByte & = B11111100 == B10101000;
```

Resim 7.54: &= (bitsel lojik ve) kullanım örneği

|= (bitsel lojik veya)

Bitsel Lojik Veya bir değişkende belirli bitlere "ayar" (1 olarak ayarlanır) yapmak için genellikle bir değişkenle ve bir sabitle kullanılmaktadır.

```
1 0 1 0 1 0 1 0 değişken
0 0 0 0 0 0 1 1 maske
-----------------------
1 0 1 0 1 0 1 1

Değişken değişmez
                  Bit seti

Aşağıdaki örneğe bakın

MyByte = B10101010;
MyByte | = B00000011 == B10101011;
```

Resim 7.55: |= (bitsel lojik veya) kullanım örneği

7.9. Arduino Tümleşik Geliştirme Ortamının "Değişken" Yapısı

7.9.1. Sabitler

Sabitler Arduino dilinde önceden tanımlanmış ifadelerdir. Programların okunmasını kolaylaştırmak için kullanılırlar. Sabitler gruplar hâlinde sınıflandırılmaktadır.

HIGH | LOW

Okuma veya yazma yaparken dijital pine verilen aktif veya/pasif olma durumunu ifade eder. HIGH ile pin çıkışı aktif edilirken, LOW ile pin çıkışı pasif yapılmaktadır. Pim aktif hâle getirildiğinde (HIGH yapıldığında) 5 Volt ile çalışan kartlarda 3 volttan daha yüksek bir voltaj mevcutken 3.3 Volt ile çalışan kartlarda 2 volttan daha yüksek bir voltaj bulunmaktadır. LOW durumunda ise (LOW yapıldığında) 5 Volt ile çalışan boardlarda 3 volttan daha düşük bir voltaj mevcutken 3.3 Volt ile çalışan boardlarda 2 volttan daha düşük bir voltaj bulunmaktadır.

```
int led= 13;
digitalWrite(led, HIGH); // Ledi yakılıyor
digitalWrite(led, LOW);  // Ledi söndürülüyor
```

Resim 7.56: HIGH | LOW kullanım örneği

INPUT | OUTPUT

Temelde dijital pine verilen modunun giriş ya da çıkış olacağı belirlenmektedir. Dijital pinler INPUT, INPUT_PULLUP, veya OUTPUT olarak kullanılabilmektedir. Dijital pinlerin elektriksel davranışı pinMode() ile değiştirilebilmektedir.

```
int led=13;
int buton=4;
void setup()
pinMode(led, OUTPUT); // Çıkış olarak tanımlandı
pinMode(buton, INPUT); // Giriş olarak tanımlandı
```

Resim 7.57: INPUT | OUTPUT kullanım örneği

LED_BUILTIN

Arduino kartlarının bir çoğu, bir dirençle seri olarak bağlı bir LED'in bulunduğu bir pime sahiptir. Sabit LED_BUILTIN, yerleşik LED'in bağlandığı pinin numarasıdır. Genellikle bu LED dijital pin 13'e bağlanmıştır.

true | false

Arduino da doğruyu ve yanlışı belirtmek için kullanılan iki sabit mantıksal tanımlamadır. Yanlış false 0 (sıfır) olarak tanımlanır. Doğru ise true 1 olarak tanımlanır. Ancak doğru olarak tanımlamanın daha geniş bir anlamı vardır. Boolean anlamında, sıfır olmayan herhangi bir tam sayı doğrudur. Dolayısıyla -1, 2 ve -200 tümüyle Boolean anlamında doğru olarak tanımlanır.

```
int m = true; // m doğru
int n = false; // n yanlıs
int led=13; // led 13. pine tanımlandı
int buton=12; // buton 12. pine tanımlandı
void setup() { } //ana kurulum yapıldı
void loop() { // sonsuz döngü sağlandı
if (buton == m) { // butona basıldı mı? Evet ise
digitalWrite(led, HIGH); // led e 5v ver
delay(1000); //1 saniye bekle
digitalWrite(led, LOW); } //led söndür
}
}
```

Resim 7.58: True | False kullanım örneği

integer constants

Sayı sistemleri için kullanılırlar. Tamsayı sabitleri, 123 gibi doğrudan bir eskizde kullanılan rakamlardır. Varsayılan olarak bu rakamlar int olarak kabul edilir, ancak bunlar U ve L değiştiricileriyle değiştirilebilmektedir. Normalde tamsayı sabitleri taban 10 (ondalık) tamsayılar olarak kabul edilirler, ancak diğer tabanlara sayı girmek için özel gösterim (formatlayıcılar) kullanılabilmektedir.

Decimal kullandığımız 10luk sayı sistemidir.

101 == ((1 * 10^2) + (0 * 10^1) + 1)

Binary ikili sayı sistemidir. 0 ve 1 kullanılmaktadır.

B101 == ((1 * 2^2) + (0 * 2^1) + 1) decimal

Octal sekizlik sayı sistemidir. 0 dan 7 ye kadar sayılardan oluşmaktadır.

0101 == ((1 * 8^2) + (0 * 8^1) + 1) decimal

Hexadecimal on altılık sayı sistemidir. Sembollerden 10 tanesi rakamlarla (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), geri kalan 6 tanesi harflerle (A, B, C, D, E, F) temsil edilmektedir.

0x101 == ((1 * 16^2) + (0 * 16^1) + 1) ==257

Sayı Sistemi	Örnek	Formatı	Karakter
10'luk (decimal)	123	-	
2'lik (binary)	B1111011	"B"	8 bit (0-255)
8'lik (Octal)	0173	"0"	0-7 karakter
16'lık (hexadecimal)	0x7B	"0x"	0-9, A-F

Tablo 7.3: Integer Constants kullanım formatı

U & L

Sayı tanımlamaları varsayılan int olarak kabul edilmektedir. Başka bir veri türüne sahip olan sayıları belirtmek için U, L imzasız veri türü kullanılmaktadır.

- ✓ Sabiti imzalamamış bir veri biçimine zorlamak için bir 'u' veya 'U'. Örnek: 33u
- ✓ Sabiti uzun bir veri formatına zorlamak için bir 'l' veya 'L'. Örnek: 100000L
- ✓ Sabiti imzasız bir uzun sabit hâline getirmek için bir 'ul' veya 'UL'. Örnek: 32767ul

floating point constants

Tamsayı sabitlerine benzer şekilde, kayan nokta sabitleri kodu daha okunabilir hâle getirmek için kullanılmaktadır. 'E' ve 'e', geçerli üs göstergeleri olarak kabul edilir. Örnek

2.34E5=2.34 * 10^5 234000; 67e-12= 67.0 * 10^-12 .000000000067

7.9.2. Veri Tipleri

void

Void anahtar sözcüğü yalnızca işlev bildirimlerinde sadece fonksiyon tanımlanırken kullanılır. Bu işlevin çağrıldığı işleve herhangi bir bilgi döndürmemesi beklenmez. Yani fonksiyonun değer döndürmeyeceği anlamına gelir.

```
// Eylemler ve fonksiyonlar "kurulum" ve
// "döngü" içerisinde gerçekleştirilir.
// Ancak hiçbir bilgi yada program bildirilmez
void setup() {
  // ...
}
void loop() {
  // ...
}
```

Resim 7.59: void kullanım örneği

boolean

Bir boolean doğru veya yanlış olmak üzere iki değerden birini tutar. 0 veya 1 değerlerini "true" ve "false" olarak alır.

```
int LEDpin = 5;        // LED 5 nolu pine bağlanıyor
int switchPin = 13;    // anahtar 13 numaralı pine bağlanıyor
boolean running = false;
void setup()
{
  pinMode(LEDpin, OUTPUT);
  pinMode(switchPin, INPUT);
  digitalWrite(switchPin, HIGH);
}
void loop()
{
  if (digitalRead(switchPin) == LOW)
  {  // anahtara basıldığında normal olarak çalışıyor
    delay(100);                          // anahtar için gecikme sağlanıyor
    running = !running;                  // değişkenler arasında geçiş yapılıyor
    digitalWrite(LEDpin, running);       // LED yakılıyor
  }
}
```

Resim 7.60: boolean kullanım örneği

char

Bir karakter değeri saklayan (1 bayt bellek alan) veri tipidir. Karakter verisini tanımlamak için kullanılmaktadır. Karakter harfleri, tek tırnak işaretleriyle yazılır: 'A' (birden fazla karakter için çift tırnak kullanılır: "ABC"). Ancak karakterler sayı olarak saklanır.

```
char myChar = 'A';
char myChar = 65;   // Her ikiside eşdeğerdir
```

Resim 7.61: char kullanım örneği

unsigned char

1 baytlık belleği kaplayan işaretsiz bir veri türüdür. Bayt veri türü ile aynıdır. 0-255 arası değer alır.

```
unsigned char myChar = 230;
```

Resim 7.62: unsigned char kullanım örneği

byte

Bir bayt, 0'dan 255'e kadar 8 bitlik bir işaretsiz sayı verisi taşır. Binary olarak da işlem yapılabilir.

```
byte b = B10010;  // "B" binary biçimi (B10010 = 18 decimal)
```

Resim 7.63: byte kullanım örneği

int

Tamsayıları saklamak için kullanılan birincil veri türüdür. 16 bit işlemcilerde -32,768 ile 32,767 arası 32 bit işlemcilerde ise -2,147,483,648 ile 2,147,483,647 arasında değişen veri saklanabilir.

```
int ledPin = 13;

int var = val;
//var -int değişken adı
//val -o değişkene atanan değer
```

Resim 7.64: int kullanım örneği

unsigned int

Negatif sayıları saklamak yerine 16 bitlik işlemcilerde sadece 0 ile 65,535 arasında değişen 2 baytlık (16 bit) bir pozitif değeri saklar. 32 bit işlemcilerde 0 ila 4,294,967,295 arasında değişen 4 baytlık (32 bit) bir pozitifdeğeri saklar.

```
unsigned int ledPin = 13;

unsigned int var = val;
//var -işaretsiz int değişken adı
//val -o değişkene atanan değer
```

Resim 7.65: unsigned int kullanım örneği

word

16 bitlik işlemcisi bulunan kartlarda 16 bitlik bir işaretsiz sayı saklanır. 32 bitlik işlemcisi bulunan kartlarda 32 bitlik bir işaretsiz sayı saklanır.

```
word w = 10000;
```

Resim 7.66: word kullanım örneği

long

Sayı saklamak için genişletilmiş boyut değişkenleridir ve 32 bit (4 bayt), -2,147,483,648 ile 2,147,483,647 arasında değişen değeri saklar. Tamsayılar kullanılıyorsa sayıların en az biri "long" olmalı ve bir "L" tarafından takip edilmelidir.

```
long speedOfLight = 186000L;

long var = val;
// var -long değişken adı
// val -o değişkene atanan değer
```

Resim 7.67: long kullanım örneği

unsigned long

Unsigned long değişkenler sayı saklamak için genişletilmiş boyut değişkenleridir ve 32 bit (4 bayt) depolamaktadır. Standart "long"un aksine, unsigned long 0 ile 4,294,967,295 arasında değişen pozitif sayıları saklar, negatif sayıları saklamaz. unsigned long değişken adı = o değişkene atayan değer olarak yazılır.

```
unsigned long time;
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  Serial.print("Time: ");
  time = millis();
  // program başlatıldıktan sonra zamanı yazdırır
  Serial.println(time);
  // bir saniye bekleniyor
  delay(1000);
}
```

Resim 7.68: unsigned long kullanım örneği

short

Short, 16 bitlik (2 baytlık) bir veri türüdür. -32,768 ile 32,767 arasında değişen değeri saklar.

```
short ledPin = 13;

short var = val;
// var -short değişken adı
// val -o değişkene atanan değer
```

Resim 7.69: short kullanım örneği

float

Ondalıklı sayılar için kullanılan veri türüdür. Ondalıklı sayılar, tamsayılara göre daha yüksek çözünürlüğe sahip oldukları için genellikle analog ve sürekli değerleri yaklaştırmak için kullanılırlar. Ondalık sayıları 3.4028235E + 38 ile -3.4028235E + 38 arasında olabilirler. Bunlar 32 bit (4 bayt) bilgi olarak saklanırlar.

```
float myfloat;
float sensorCalbrate = 1.117;

float var = val;
// var -float değişken adı
// val -o değişkene atanan değer
```

Resim 7.70: float kullanım örneği

double

Double, ondalıklı sayılar için kullanılan veri türüdür. Burada hassasiyet 2 kat yüksektir. "double" uygulaması tam olarak "float" ile aynıdır. Uno ve diğer ATMEGA tabanlı kartlarda 4 bayt yer kaplar. Arduino Due'da "double" 8 bayt (64 bit) hassaslığa sahiptir.

string - char array

Yazı verisi depolamak için kullanılır. Karakter dizisidir. Metin dizeleri string ile gösterilir. Yandaki gösterimlerin tümü geçerli kullanım şekillerine örnektir.

```
char Str1 [15];
char Str2 [8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
char Str3 [8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\ 0'};
char Str4 [] = "arduino";
char Str5 [8] = "arduino";
char Str6 [15] = "arduino";
```

Resim 7.71: string - char array kullanım örneği

substring()

String, içindeki kelimeden kaç karakter alacağını belirtmek için kullanılır.

```
String cumle = "Robot Programlama Dersi";
if (cumle.substring(3,9) == "Programlama") {
//
}
```

Resim 7.72: substring() kullanım örneği

String – object

String sınıfı, metin dizelerini karakter dizilerinden daha karmaşık yollarla kullanılmasına ve değiştirilmesine olanak tanır. Dizeler bir araya getirebilir, onlara eklenebilir, alt dizeler aranıp değiştirilebilir. Basit bir karakter diziliminden daha fazla bellek kullanır ancak daha kullanışlıdır.

array

Bir dizi, bir dizin numarasıyla erişilen değişkenlerin bir toplamıdır. Diğer bir ifadeyle her birine indeks numarası ile ulaşılan aynı türdeki veri topluluğudur. Dizi, bilgisayar belleğinde aynı isim altında genellikle aynı tipten çok sayıda veriyi bir arada saklayan veri yapısıdır. Yandaki yöntemlerin tümü, bir diziyi oluşturmak (bildirmek) için kullanılabilecek geçerli biçimlerdir.

```
int myInts [6];
int myPins [] = {2, 4, 8, 3, 6};
int mySensVals [5] = {2, 4, -8, 3, 2};
char message [8] = "merhaba";

mySensVals [0] = 10; // Bir diziye değer atamak
x = mySensVals [4] ; // Bir diziden bir değer almak
```

Resim 7.73: array kullanım örneği

- ✓ Bir dizi "myInts"'de olduğu gibi dizi başlatılmadan tanımlanabilir. Burada 6 farklı "int (integer)" veri tipinde değer tanımlaması yapılmıştır.
- ✓ "myPins"'te açıkça bir boyut seçmeden bir dizi tanımlanmıştır. Bu durumda derleyici elemanları sayar ve uygun büyüklükte bir diziyi otomatik olarak oluşturur.
- ✓ "mySensVals" örneğinde olduğu gibi dizi başlatabilir ve boyutlandırabilir.
- ✓ char türündeki bir diziyi bildirirken, gerekli boş karakteri tutmak için fazladan bir eleman gerekmektedir.
- ✓ Tanımlamış değişkenlere erişim için köşeli parantez ile değişkenin sıra numarası belirtilmektedir. Yani başta tanımlanan yada boş bırakılan indis'i çağırma işlemi gerçekleştirilmektedir. Dikkat edilmesi gereken nokta indisin '0' dan başladığıdır. Yani örnekte yazdığımız 5 farklı değeri olan "mySensVals" kümemizde 4 numaralı elemana erişmek için köşeli parantez içinde [4] değilde [3] yazılmalıdır. Örnekte "mySensVals" kümemize 1. elaman olarak 10 değeri atanmıştır. Yine ayni diziden 4 numaralı değer (2) alınmıştır.

```
int i;
for (i = 0; i < 5; i = i + 1) {
  Serial.println(myPins[i]);
}
```

Resim 7.74: array'ın For döngüsünde kullanım örneği

Dizilerin for döngülerinde kullanımı için yukarıdaki şekli inceleyip, aşağıdaki örneği uygulayınız. Aşağıda verilen örnekte Arduino UNO'nun 2, 3, 4 ve 5 numaralı dijital pinlerine bağlı dört LED'in sırayla bir saniye aralıklarla yanıp sırayla sönmesini kontrol eden array uygulaması yer almaktadır. Arduino UNO'ya LED bağlantılarını yaptıktan sonra örneği dersin sitesinden (7.8.1.5 numaralı uygulama) kopyalayarak Arduino IDE'de test ediniz.

```
const int LEDsayisi = 4; // Dizide kaç eleman olacağı tanımlanıyor
const int pinLEDS[LEDsayisi] = {2,3,4,5}; // LED'ler 2, 3, 4, ve 5 nolu pinlere bağlanıyor

void setup()
 {
 for(int m = 0; m < LEDsayisi; m++) // m değişkeni LED sayısı kadar artırılıyor
 {
 pinMode(pinLEDS[m], OUTPUT);       // LED'lerin bağlı olduğu pinler çıkış olarak ayarlanıyor
 }
}

void loop()
 {
 for(int m = 0; m < LEDsayisi; m++) // m değişkeni LED sayısı kadar artırılıyor
 {
 digitalWrite(pinLEDS[m], HIGH);    // LED'le sırayla yanıyor
 delay(1000);                       // 1 sn bekleniyor
 }

 for(int m = LEDsayisi - 1; m >= 0; m--) // m değişkeni LED sayısı kadar azaltılıyor
 {
 digitalWrite(pinLEDS[m], LOW);     // LED'le sırayla sondürülüyor
 delay(1000);                       // 1 sn bekleniyor
 }
}
```

7.9.3. Dönüşümler

char()

Herhangi bir değeri char veri türüne dönüştürür.

```
char myChar = 'A';
char myChar = 65; // Eşdeğeri
```

Resim 7.75: char() kullanım örneği

byte()

Herhangi bir değeri byte veri türüne dönüştürür.

```
byte b = B10010; // "B" Binary biçimi
(B10010 = 18 ondalık biçimi)
```

Resim 7.76: byte() kullanım örneği

int()

Herhangi bir değeri int (integer-tam sayı) veri türüne dönüştürür.

```
int led= 4;
```

Resim 7.77: int() kullanım örneği

word()

Herhangi bir değeri word veri türüne dönüştürür ya da iki bayt bir kelime oluşturur. Kelime oluşturuken word (h, l) şeklinde ifade edilir. Burada H: sözcüğün üst sırası (en soldaki), L: sözcüğün en düşük (en sağdaki) baytını ifade eder.

```
word w = 10000;
```

Resim 7.78: word() kullanım örneği

long()

Herhangi bir değeri long veri türüne dönüştürür.

```
long sensor = 253000L;
```

Resim 7.79: long() kullanım örneği

float

Herhangi bir değeri float veri türüne dönüştürür.

```
float myfloat;
float sensorCalbrate = 1.123;
```

Resim 7.80: float() kullanım örneği

7.9.4. Değişken Kapsamları

static

Statik olarak tanımlanan değişkenler yalnızca bir işleve çağrıldığında (o işleve özel) ilk kez oluşturulur ve başlatılırlar. Bu değişkenler sonra silinmeyip bellekte tutulmaktadır ve daha sonra kullanılmak istendiğinde yeniden oluşturulmadan bellekten çağırılmaktadır.

```
static int sayi=1;
```

Resim 7.81: static kullanım örneği

volatile

Volatile ile tanımlanan değişkenler Arduino mikro kontrolorunun ram bölgesine kaydedilir. Kullanılmak istendiğinde doğrudan ram bellekten okunur. Volatile kullanıldığında değişkenin değerini interrupt ile değiştirmek gerekmektedir.

```
int pin = 13;
volatile int state = LOW;
```

Resim 7.82: volatile kullanım örneği

const

Const değişken türü diğer tüm değişkenler gibi kullanılır. Oluşturulan değişkenler sabitlenirler ve değerleri daha sonradan değiştirilemezler.

```
const float pi = 3.14;
const float fi= 1.61;
```

Resim 7.83: const kullanım örneği

7.9.5. Yardımcılar

sizeof()

Sizeof işleci, herhangi bir tipteki değişken türünün bayt sayısını (değişkenin kaç bayt olduğunu) verir veya bir dizinin işgal ettiği toplam bayt sayısını döndürür. Yandaki program bir seferde bir karakterlik bir metin dizisi yazmaktadır. Metnin ifadesini değiştirerek deneyiniz.

```
char mesaj[] = "Robot Programlama";
int i;
void setup(){
Serial.begin(9600); }
void loop() {
for (i = 0; i < sizeof(mesaj) - 1; i++){
Serial.print(i, DEC);
Serial.print(" = ");
Serial.write(mesaj[i]);
Serial.println();
}
delay(5000); }
```

Resim 7.84: sizeof() kullanım örneği

PROGMEM

Bilgiyi SRAM bellek yerine Flash bellekte depolamak için kullanılmaktadır. Programda uzun char yazıldığında sorun çıkarabilmektedir. Bu nedenle çok uzun metinlerin Flash belleğe kaydedilmesi gerekmektedir. PROGMEM değişken değiştiricidir, yalnızca pgmspace.h'de tanımlanan veri türleriyle birlikte kullanılmalıdır. Yani kullanılabilmesi için programa #include <avr/pgmspace.h> kütüphanesi eklenmelidir.

```
#include <avr/pgmspace.h>
const dataType variableName[] PROGMEM = {};   // uygun kullanım şekli
const PROGMEM  dataType  variableName[] = {}; // uygun kullanım şekli
```

Resim 7.85: PROGMEM kullanım örneği

7.10. Arduino Tümleşik Geliştirme Ortamının "Fonksiyon" Yapısı

7.10.1. Dijital Giriş Çıkışlar

pinMode()

Belirtilen pinin giriş (INPUT) yada çıkış (OUTPUT) olacağının tanımlandığı komuttur. A0, A1 vb. olarak adlandırılan Analog giriş pinleri dijital pimler olarak kullanılabilir. Aşağıdaki yazım şekillerinin hepsi kullanılanilir.

```
int ledPin = 13;           // LED digital pin 13'e bağlandı
void setup()
{
  pinMode(ledPin, OUTPUT); // digital pin çıkış olarak tanımlandı
  pinMode(5, OUTPUT);      // Çıkış olarak tanımladı
  pinMode(6, INPUT);       // Giriş olarak tanımladı
}
```

Resim 7.86: pinMode() kullanım örneği

digitalWrite()

Belirtilen pinin aktif (HIGH) yada pasif (LOW) olacağının tanımlandığı komuttur. A0, A1 vb. olarak adlandırılan Analog giriş pinleri dijital pimler olarak kullanılabilir. Aşağıdaki örnek ve yazım şekillerinin hepsi kullanılanilir.

```
int ledPin = 13;          // LED digital pin 13'e bağlandı
void setup()
{
  pinMode(ledPin, OUTPUT); // Digital pin çıkış olarak tanımlandı
void loop()
{
  digitalWrite(ledPin, HIGH); // LED aktif
  delay(1000);                // 1 sn bekleniyor
  digitalWrite(ledPin, LOW);  // LED pasif
  delay(1000);                // 1 sn bekleniyor
}
digitalWrite(13, HIGH); //13. pin aktif (bu şekilde de gösterilebilir)
digitalWrite(13, LOW);  //13. pin pasif (bu şekilde de gösterilebilir)
```

Resim 7.87: digitalWrite() kullanım örneği

digitalRead()

Belirtilen bir dijital pinden, aktif (HIGH) yada pasif (LOW) değerini okur. A0, A1 vb. olarak adlandırılan Analog giriş pinleri dijital pimler olarak kullanılabilir. Aşağıdaki örneği hazırlayınız.

```
int buton = 7; // Buton dijital 7'ye tanımlandı
void setup() {
Serial.begin(9600); //Seri haberleşme hızı
pinMode(buton, INPUT); // Buton pini giriş yapıldı
}
void loop() {
int butondurumu = digitalRead(buton); //Giriş pini okundu
Serial.println(butondurumu); // Seri ekrana yazıldı
delay(250);
}
```

Resim 7.87: digitalRead() kullanım örneği

7.10.2. Analog Giriş Çıkışlar

analogReference()

Analog giriş için kullanılan referans voltajını (yani giriş aralığının üstü olarak kullanılan değeri) konfigüre etmek için kullanılır. Seçenekler şunlardır:

- ✓ **DEFAULT**: Arduino bordlarında çalışma voltajına göre varsayılan 5 volt ya da 3,3 volt olarak belirlenmiştir.

- ✓ **INTERNAL**: Atmega168 ya da Atmega 328'de 1,1 volt, ATmega8'de 2,56 volt geçerlidir. Mega katlar hariçtir.
- ✓ **INTERNAL1V1**: Sadece Arduino Mega 1,1 volt olarak belirlenmiştir.
- ✓ **INTERNAL2V56**: Sadece Arduino Mega 5,56 volt olarak belirlenmiştir.
- ✓ **EXTERNAL**: Kartın üzerinde bulunan AREF pin (sadece 0 - 5V) referans olarak kullanılabilir.

analogRead()

Belirtilen analog pimden değeri okumak için kullanılır. Arduino Uno'da 6 adet, Mini ve Nano'da 8 adet ve Mega'da 16 adet 10 bit analog sinyali dijitale dönüştüren çevirici bulunmaktadır. Okuma işlemi analog bir girişi dijitale çevirerek yapılmaktadır. Bu 0 ile 5 volt arasındaki giriş voltajlarını 0 ile 1023 arasında tam sayı değerlerine bölünmesi anlamına gelmektedir. Böylece 5 volt / 1024 ünite veya birim başına 0,009 volt (4,9 mV) okuma arasında bir çözünürlük sağlamaktadır. Giriş aralığı ve çözünürlük, analogReference() kullanılarak değiştirilebilmektedir. Yandaki örnekte orta ucu analog pine (A1) takılmış bir potansiyometrenin çevrilmesi sonucu elde edilen voltaj seri ekrandan okunmaktadır.

```
void setup() {
Serial.begin(9600);
}
void loop() {
int sensor = analogRead(A1);
float volt = sensor * (5.0 / 1023.0);
Serial.println(volt);
```

Resim 7.88: analogRead() kullanım örneği

analogWrite() -PWM

Bir pine bir analog değer (PWM) yazar. Bir LED'i farklı parlaklıklarda aydınlatmak veya çeşitli hızlarda bir motoru sürmek için kullanılabilir. Belirtilen görev döngüsünde kararlı bir kare dalga oluşturulur. Çoğu pim üzerindeki PWM sinyalinin frekansı yaklaşık 490 Hz'dir. Uno ve benzeri kartlarda 5 ve 6 numaralı pinlerin frekansı ise yaklaşık 980 Hz'dir. Aşağıdaki örnekte orta ucu analog pine (A1) takılmış bir potansiyometrenin çevrilmesi ile 5 numaralı pine takılmış bir LED'in parlaklığı kontrol edilmektedir.

- ✓ Arduino Uno'da bu komut 3, 5, 6, 9, 10. pinler için kullanılabilir.
- ✓ Arduino Mega'da 11, 2-13 ve 44-46. pinler için kullanılabilir.
- ✓ Arduino Due 2 den 13. pine kadar kullanılabilir.

```
int ledPin = 5;       // LED pin 5'e bağlandı
int analogPin = 3;    // Potansiyometre analog pin 3'e bağlandı
int okunan = 0;       // Okunan değeri saklamak için değişken tanımlandı
void setup()
{
  pinMode(ledPin, OUTPUT);   // Çıkış pini ayarlandı
}
void loop()
{
  okunan = analogRead(analogPin);  // Giriş pini okundu
  analogWrite(ledPin, okunan / 4); // AnalogRead değeri 0 ile 1023 arasında
                                   // AnalogWrite değeri 0 ile 255 arasında
}
```

Resim 7.88: analogWrite() -PWM kullanım örneği

analogReadResolution() ve analogWriteResolution()

AnalogReadResolution() ve analogWriteResolution() Arduino Due ve Zero için Analog API'nin bir uzantısıdır. AnalogRead() tarafından döndürülen değerin boyutunu (bit olarak) ayarlar. AVR tabanlı kartlarda geriye dönük uyumluluk için varsayılan olarak 10 bit (0-1023 arasındaki değerlerler döndürür). AnalogWriteResolution(), analogWrite() işlevinin çözünürlüğünü ayarlar. AVR tabanlı kartlarla geriye dönük uyumluluk için varsayılan 8 bit'dir (0-255 arasındaki değerler). Arduino Due ve Zero 12-bit çözünürlüğünde oldukları için 0 ile 4095 arasında çözünürlük sağlamaktadır.

7.10.3. Gelişmiş Giriş Çıkışlar

tone()

Kare dalga üretilmesine olanak sağlar. İstenilen pin istenilen frekansa ayarlanabilmektedir. Verilen sinyal %50 görev döngüsüne sahiptir. Görev döngüsü 1 periyot boyunca HIGH ve LOW kalma süresidir. Zil sesleri çalmak için pin bir piezo ziline veya başka bir hoparlöre bağlanabilir.Bir seferde yalnızca bir ton üretilebilir. Farklı bir pinde zaten bir ton çalıyorsa, tone() çağrısının etkisi olmayacaktır. Yandaki örneği uygulayınız.

```
int hoparlorPin = 12;
int notaSayisi = 2;
int A = 440;
int B = 494;
int notalar[] = {A, B_};
void setup()
{
for (int i = 0; i < notaSayisi; i++)
{
tone(hoparlorPin, notalar[i]);
delay(500);
noTone(hoparlorPin);
delay(20);
}
noTone(hoparlorPin);
}
void loop()
{
}
```

Resim 7.90: tone() ve no tone() kullanım örneği

noTone()

tone() ile üretilen kare dalgayı sonlandırmaya yarar. Hiçbir ton üretilmediğinde hiçbir etkisi olmaz.

shiftOut()

Her seferinde bir bayt veri kaydırır. En büyük (en soldaki) veya en küçük (en sağdaki) önemli bitten başlanır. Her bit bir veri pinine yazılır ve daha sonra bit bulunduğunu belirtmek için bir saat darbesi bir saat pinine uygulanır. Bu şekilde 8 bitlik veri tek bir seri giriş pininden girilir ve işlem sonucunda seri olarak girilen veri paralel çıkışlardan alınır. Bu işlem için özel işlemci (74HC595 Shift Register Entegresi) kullanılır. Görevi Arduino'da az sayıda dijital çıkış pini kullanarak çok sayıda veriyi kontrol etmeyi sağlamaktır.

```
// MSBFIRST için
int data = 500;
// HIGH baytların dışarı kaydırılması
shiftOut(dataPin, clock, MSBFIRST, (data >> 8));
// LOW baytların dışarı kaydırılması
shiftOut(dataPin, clock, MSBFIRST, data);

// LSBFIRST için
data = 500;
// LOW baytların dışarı kaydırılması
shiftOut(dataPin, clock, LSBFIRST, data);
// HIGH baytların dışarı kaydırılması
shiftOut(dataPin, clock, LSBFIRST, (data >> 8));
```

Resim 7.91: shiftOut() kullanım örneği

shiftOut(dataPin, clockPin, bitOrder, value) şeklinde yazılır. dataPin: Data pinini belirler. clockPin: Clock pinini belirler. bitOrder: Bitleri kaydırmak için hangi sıra ile başlanacağını belirler (MSBFIRST: En büyük bitten başlanacağını, LSBFIRST: En küçük bitten başlanacağını belirler). Value: Kaydırılacak olan verilerdir (bayt).

shiftIn()

Bir seferde bir bit veri kaydırır. En çok (en soldaki) veya en az (en sağdaki) önemli bitten başlanır. Her bit için saat pimi HIGH alınır, sonraki bit veri satırından okunur ve saat pimi LOW alınır. Bu bir yazılım uygulamasıdır. Arduino donanım uygulaması daha hızlıdır ama sadece belirli pinler üzerinde çalışan bir SPI kütüphanesiyle sağlanmaktadır. Görevi Arduino'da az sayıda dijital giriş pini kullanarak çok sayıda veriyi kontrol etmeyi sağlamaktır.

byte incoming = shiftIn(dataPin, clockPin, bitOrder) şeklinde yazılır. dataPin: Data pinini belirler. clockPin: Clock pinini belirler. bitOrder: Bitleri kaydırmak için hangi sıra ile başlanacağını belirler (MSBFIRST: En büyük bitten başlanacağını, LSBFIRST: En küçük bitten başlanacağını belirler).

```
int data = 0;  // PIN for data
int clock = 1; // PIN for clock
int latch = 2; // PIN for latch
int value = 0;
void setup() {
 pinMode(data INPUT);
 pinMode(clock, OUTPUT);
 pinMode(latch, OUTPUT);
 Serial.begin(9600);
}
void loop() {
 digitalWrite(latch, LOW);
 // Kayıttan 8 bit (bir bayt) veri okunuyor
 value = shiftIn(data, clock, LSBFIRST, 8);
 digitalWrite(latch, HIGH);
 Serial.println(value);
 delay(1000);
}
```

Resim 7.92: shiftIn() kullanım örneği

pulseIn()

Bir pin üzerinde bir sinyalin (HIGH veya LOW) olduğunu belirlemek için kullanılır. Örneğin değer HIGH ise, pulseIn() pinin HIGH konumuna gelmesini bekler, zamanlamaya başlar, sonra pinin LOW olmasını bekler ve zamanlamayı durdurur. Sinyalin uzunluğu mikrosaniye cinsinden döndürülmektedir.

```
int pin = 7;
unsigned long zaman;
void setup()
{
pinMode(pin, INPUT);
}
void loop()
{
zaman = pulseIn(pin, HIGH);
}
```

Resim 7.93: pulseIn() kullanım örneği

7.10.4. Gecikmeler

millis()

milis, program başladıktan sonra geçen milisaniye sayısını belirlemek için kullanılır. Yaklaşık 50 gün sonra bu süre sıfırlanmaktadır. Komutun kullanımında herhangi bir parametre bulunmamaktadır. Yandaki örneği inceleyiniz.

```
unsigned long sure;
void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.print("Süre: ");
  sure = millis();
  Serial.println(sure);
  delay(1000);
}
```

Resim 7.94: millis() kullanım örneği

micros()

micros, program başladıktan sonra geçen mikrosaniye sayısını belirlemek için kullanılır. Yaklaşık 70 dakika sonra bu süre sıfırlanmaktadır. Komutun kullanımında herhangi bir parametre bulunmamaktadır. Yandaki örneği inceleyiniz. Not: Bir milisaniyede 1.000 mikrosaniye ve 1 saniyede 1.000.000 mikrosaniye vardır.

```
unsigned long sure;
void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.print("Süre: ");
  sure = micros();
  Serial.println(sure);
  delay(1000);
}
```

Resim 7.95: micros() kullanım örneği

delay()

delay program akışını milisaniye cinsinden duraklatmak için kullanılır. Örneğin 1 saniyelik gecikme ihtiyacı için delay(1000) şeklinde ifade edilir. Gecikme fonksiyonu süresince hiçbir algılayıcı, matematiksel hesaplama ya da pin okuması devam edemez. Bu nedenle 10'un milisaniyesinden daha uzun süren olayların zamanlaması için delay() kullanılmasına uygun değildir. Yanda verilen yanıp sonen LED örneğini inceleyiniz.

```
int ledPin = 13;                  // LED pin 13'e bağlı
void setup()
{
  pinMode(ledPin, OUTPUT);        // Pin çıkış olarak ayarlandı
}
void loop()
{
  digitalWrite(ledPin, HIGH);     // LED açık
  delay(1000);                    // 1 sn bekliyor
  digitalWrite(ledPin, LOW);      // LED kapalı
  delay(1000);                    // 1 sn bekliyor
}
```

Resim 7.96: delay() kullanım örneği

delayMicroseconds()

Programı parametre olarak belirtilen süre boyunca (mikro saniye olarak) duraklatmak için kullanılır. Şu anda doğru bir gecikme için üretilen en büyük değer 16383'tür. Bu işlev 3 mikro saniye ve üzeri aralıklarla çok doğru çalışır. Birkaç bin mikrosaniyeden daha uzun olan gecikmeler için bunun yerine delay() kullanılır. Aşağıdaki örneği inceleyiniz.

```
int outPin = 8;                 // Pin 8 belirlendi
void setup()
{
  pinMode(outPin, OUTPUT);      // Pin çıkış olarak ayarlandı
}
void loop()
{
  digitalWrite(outPin, HIGH);   // Pin açık
  delayMicroseconds(50);        // 50 mikrosanye duraklatıldı
  digitalWrite(outPin, LOW);    // pin kapalı
  delayMicroseconds(50);        // 50 mikosaniye duraklatıldı
}
```

Resim 7.97: delayMicroseconds() kullanım örneği

7.10.5. Matematiksel İşlevler

min()

X ve y sayısal değerlerinden en küçük olanını seçmek için kullanılır. min() genellikle bir değişken aralığının alt ucunu sınırlamak içindir. Aşağıdaki örneği inceleyiniz.

max()

X ve y sayısal değerlerinden en büyük olanını seçmek için kullanılır. max()genellikle bir değişken aralığın üst ucunu sınırlamak içindir. Aşağıdaki örneği inceleyiniz.

```
enkucuk=min(30,40); // En küçük 30 olarak belirlenir
enbuyuk=max(30,40); // En büyük 40 olarak belirlenir
sure = min(sure, 20); // Hiçbir zaman 20'yi geçmez
sure = max(sure, 20); // En az 20 olur
```

Resim 7.98: min() ve max() kullanım örneği

abs()

Bir sayının mutlak değerini hesaplamak için kullanılır. Sayı sıfırdan küçükse pozitif değerini, sıfırdan büyükse aynı sayıyı verir.

```
int m= -10;
int n= 10;
veri=abs(m); // Veri 10 olarak belirlenir
veri=abs(n); // Veri 10 olarak belirlenir
```

Resim 7.99: abs() kullanım örneği

constrain()

Bir sayıyı belirli aralıklarla sınırlamak için kullanılır. constrain(x,a,b) şeklinde tüm veri türleri için kullanılır. x: herhangi türdeki bir sayı, a: alt aralık, b: üst aralık. Örnekte sensör değeri 10 ile 150 arasında sınırlandırılmıştır. Sensör değeri 10'dan küçükse çıktı 10, 150'den büyükse 150 olur. 10 ile 150 arasında bir değer ise o değer çıktı olarak kullanılır.

```
sensor = constrain(sensor, 10, 150);
// Sensor değeri 10 ile 150 arasında sınırlandırılmış
```

Resim 7.100: constrain() kullanım örneği

map()

Sensor değerini belli aralıkta tutarak istenilen aralığa dönüştürmek için kullanılır. Bu amaçla gerekirse bir dizi aralığı tersine çevirmek için de kullanılabilir. Map() işlevi tamsayı matematiği kullanır, böylece kesirler üretmez, kesirli kalanlar kesilir ve yuvarlamaz veya ortalamalanmaz. map(value, fromLow, fromHigh, toLow, toHigh) şeklinde ifade edilir.

- ✓ **value:** Sensorden gelen değer
- ✓ **fromLow:** Değerin geçerli aralığının alt sınırı
- ✓ **fromHigh:** Değerin geçerli aralığının üst sınırı
- ✓ **toLow:** Değerin hedef aralığının alt sınırı
- ✓ **toHigh:** Dğerin hedef aralığının üst sınırı

```
void setup() {}
void loop()
{
  int sensor = analogRead(0);
  sensor = map(sensor, 0, 1023, 0, 255);
  analogWrite(9, sensor);
}
```

Resim 7.101: map() kullanım örneği

Yanda verilen örnekte sensörden okunan değer 0 ile 1023 arasındadir fakat 0 ile 255 arasında bir değere dönüştürülmektedir.

pow()

Bir sayının üstsel kuvvetini almak için kullanılır. Bir sayıyı kesirli bir kuvvete yükseltmek için de kullanılabilir. Değerlerin veya eğrilerin üstel haritalamasını üretmek için yararlı bir işlevdir. pow(base, exponent) şeklinde ifade edilir. Base: Taban sayısı, exponent: Üstsel kuvvet. Yanda "for döngüsü" boyunca bir değişkenin katlanarak kuvvetini alan bir örnek verilmiştir.

```
for (int i = 0; i < 10; i++) {
// For döngüsü devam ettiği için veri katlanarak büyür
veri = pow(2, i);
}
```

Resim 7.102: pow() kullanım örneği

sqrt()

Bir sayının karekökünü hesaplamak için kullanılır. Sayı herhangi bir veri türünde olabilir.

```
islem=sqrt(65536)// Sonuç 265'dır
```

Resim 7.103: sqrt() kullanım örneği

7.10.6. Trigonometri İşlevleri

Trigonometrik işlemlerin yaptırılabilmesi için "math.h" kütüphanesinin kullanılması gereklidir. Kütüphane #include "math.h" şeklinde kullanıma alınır.

sin()

Bir açının sinüsünü radyan cinsinden hesaplar. Sonuç -1 ile 1 arasında olacaktır.

cos()

Bir açının kosinusünü radyan cinsinden hesaplar. Sonuç -1 ile 1 arasında olacaktır.

tan()

Bir açının tanjantını radyan cinsinden hesaplar. Sonuç negatif sonsuzluk ile pozitif sonsuzluk arasında olacaktır.

```
#include "math.h"
islem=sin(65); // Sonuç 0,826828679 olacaktır
islem=cos(65); // Sonuç -0,562453851 olacaktır
islem=tan(65); // Sonuç -1,470038258 olacaktır
```

Resim 7.104: Trigonometrik işlev örnekleri

7.10.7. Karakterler

Bir karakterin ne tür bir karakter olduğunu kontrol etmek için kullanılmaktadır. Aşağıda verilen uygulama örneği dersin sitesinde 7.10.7 numarasıyla yer almaktadır. Uygulamayı indirerek test ediniz.

isAlphaNumeric(): Bir karakterin alphanumeric olup olmadığını kontrol eder.

isAlpha(): Bir karakterin alpha olup olmadığını kontrol eder.

isAscii(): Bir karakterin Ascii tablosundaki değerini verir.

isWhiteSpace(): Bir karakterin bir boşluk olup olmadığını kontrol eder.

isControl(): Bir karakterin kontrol karakteri olup olmadığını kontrol eder.

isDigit(): Bir karakterin dijital karakter olup olmadığını kontrol eder.

isGraph(): Bir karakterin grafik karakter olup olmadığını kontrol eder.

isLowerCase(): Bir karakterin küçük harfli bir karakter olup olmadığını kontrol eder.

isPrintable(): Bir karakterin yazdırılabilir bir karakter olup olmadığını kontrol eder.

isPunct(): Bir karakterin noktalama işareti olup olmadığını kontrol eder.

isSpace(): Bir karakterin boşluk olup olmadığını kontrol eder.

isUpperCase(): Bir karakterin büyük harfli bir karakter olup olmadığını kontrol eder.

isHexaDecimalDigit(): Bir karakterin geçerli heksedesimal (onaltılık) rakam olup olmadığını kontrol eder. Yukarıdaki örnek program gönderilen herhangi bir karakterin karakter analizini yapmaktadır.

7.10.8. Rastgele Sayılar

randomSeed()

randomSeed() rastgele sayı üretimini rastgele sıralamayla rastgele bir noktadan başlatır. Bu dizi çok uzun ve rastgele iken daima aynıdır. Üretilen bir dizi sıranın farklı olması istendiğinde rastgele sayı üreteci, bağlantısız bir pin üzerinden (analogRead() gibi) oldukça rastgele bir girdi ile başlatılabilir. Rastgele sayı üretilirken bir Seed değeri alınır. Bu algoritmalarla uzun bir sayı listesi hesaplanır. Seed belirtilmezse şu anki zaman alınır ve sayılar hep aynı sırada rastgele olarak üretilir. Yandaki örnek 0 ile 249 arasında rastgele sayı üretmektedir.

```
long rasgelesayi;
void setup(){
Serial.begin(9600);
randomSeed (analogRead (1));
}
void loop(){
rasgelesayi = random(250);
Serial.println(rasgelesayi);
delay(100);
}
```

Resim 7.106: randomSeed() kullanım örneği

```
Serial.println();}
void loop() {
if (Serial.available() > 0) {
int thisChar = Serial.read();
Serial.print("Gonderilen: '");
Serial.write(thisChar);
Serial.print("' ASCII Degeri: ");
Serial.println(thisChar);
if (isAlphaNumeric(thisChar)) {
  Serial.println("Alphanumerik Degeri");
}
if (isAlpha(thisChar)) {
  Serial.println("Bu Alfabetik!");
}
if (isAscii(thisChar)) {
Serial.println("Bu ASCII karakter");
}
if (isWhitespace(thisChar)) {
  Serial.println("Bu Bos Karakter");
}
if (isControl(thisChar)) {
  Serial.println("Bu kontrol karakteri");
}
if (isDigit(thisChar)) {
  Serial.println("Bu Dijital Karakter");
}
if (isGraph(thisChar)) {
Serial.println("Bosluk Degil Yazdirilabilir Karakter");
}
if (isLowerCase(thisChar)) {
  Serial.println("Kucuk Harf");
}
if (isPrintable(thisChar)) {
  Serial.println("Yazdirilabilir");
}
if (isPunct(thisChar)) {
  Serial.println("Noktalama Isareti");
}
if (isSpace(thisChar)) {
  Serial.println("Bosluk Karakteri");
}
if (isUpperCase(thisChar)) {
  Serial.println("Buyuk Harf");
}
if (isHexadecimalDigit(thisChar)) {
Serial.println("Gecerli Heksadesimal Dijit Var");
}
Serial.println();
Serial.println("Baska Bir Karakter Girin:");
Serial.println();
}
```

Resim 7.105: Karakterlerin kullanım örneği

random()

Minimum ve maksimum olarak belirtilen sayılar arasında rastgele sayılar üretmek için kullanılır. Üretilen bir dizi sıranın farklı olması istendiğinde rastgele sayı üreteci, bağlantısız bir pin üzerinden (analogRead() gibi) oldukça rastgele bir girdi ile başlatılabilir. Yandaki örnek 10 ile 39 arasında rastgele sayı üretmektedir.

```
long rasgelesayi;
void setup() {
Serial.begin(9600); }
void loop(){
rasgelesayi = random(10,40);
Serial.println(rasgelesayi);
delay(100);
}
```

Resim 7.107: random() kullanım örneği

7.10.9. Bit ve Bayt'lar

lowByte(): Bir değişkenin (örneğin bir sözcük) düşük sıralı (en sağdaki) baytı için kullanılır.

highByte(): Bir kelimenin üst sıra (en soldaki) baytını (veya daha büyük bir veri türünün en düşük ikinci baytını) ayıklamak için kullanılır.

bitRead(): Herhangi bir bit'i okumak için kullanılır.

bitWrite(): Bir sayısal değişken bit yazmak için kullanılır.

bitSet(): Bir sayısal değişkenin bit'ini ayarlamak için kullanılır.

bitClear(): Bir sayısal değişkenden bit silmek için kullanılır.

bit(): Belirtilen bit değerini hesaplamak için kullanılır.

7.10.10. İnterruptlar (Kesmeler)

Arduino üzerinde bir programı yürütürken yürümekte olan programın duraklatıp arada başka bir programın çalıştırılması için kesmeler kullanılır. Arada çalıştırılan program komutları yerine getirildikten sonra ana program kaldığı yerden devam eder. Arduino'da farklı görevlerde kullanılmak üzere çeşitli Interrupt'lar (kesmeler) bulunur. Zaman kesmesi (timer interrupt) ve dış kesmeler (external İnterrupt) en yaygın kullanılan Arduino kesmeleridir.

interrupts()

Kesmeler, bazı önemli görevlerin arka planda yapılmasını sağlamak için kullanılır.

noInterrupts()

Kesmeleri devre dışı bırakmak için kullanılır, (interrupts() komu ile yeniden etkinleştirilebilir). Yanda kullanım şekli gösterilmektedir.

```
void setup() {}
void loop()
{
  noInterrupts(); // Kritik zamana duyarlı kod burada
  interrupts();   // Diğer kodlar burada
}
```

Resim 7.108: interrupts() kullanım şekli

7.10.11. Harici Interruptlar (Kesmeler)

attachInterrupt()

Arduinonun belli bir pinine gelen sinyalle belli bir fonksiyonun ya da önceden belirlenmiş bir alt programın çalıştırılmasını sağlamak için kullanılır. Kullanılan Arduinonun türüne göre dijital pinlerden bazıları attachInterrupt pini olarak da işlev görmektedir. Aşağıdaki tabloda farklı Arduino modellerine göre kullanılabilir attachInterrupt pinleri listelenmiştir.

Kart Türleri	Dijital Pin Interrupt						
	Int.0	Int.1	Int.2	Int.3	Int.4	Int.5	...
Uno, Nano, Mini ve diğer 328 temelli kartlar	2	3					
Mega, MEga2560, MegaADK	2	3	18	19	20	21	
Micro, Leonardo, diğer 32u4 temelli kartlar	0	1	2	3	7		
Zero	4. hariç tüm pinler						
Due	Tüm dijital pinler						
MKR1000 Rev.1	0, 1, 4, 5, 6, 7, 8, 9, A1, A2 interrupt numarası = pin numarası						

Tablo 7.4: Arduino modellerine göre kullanılabilir attachInterrupt pinleri

Kullanımı attachInterrupt(pin, fonksiyon, mod) şeklindedir. **Pin:** Kullanılan interrupt pinidir. **Fonksiyon:** Interrupt tetiklendiğinde yapılacak işlemlerin içinde bulunduğu fonksiyonunu (Void loop'un altına yazılmalıdır), **Mod:** Kesmeye ne zaman girileceğini ifade eder. Kullanılan modlar şu şekidedir:

LOW: Dijital pindeki gerilim 0 olduğunda kesmeye girmesini sağlar.

CHANGE: Belirli dijital pinde oluşacak her gerilim değişiminde kesmeye girmesini sağlar. Yani pindeki gerilim 0'dan 5'e yükseldiğinde veya 5'ten 0'a düştüğünde kesmeye girer.

RISING: Yükselen kenar olduğunda kesmeye girmesini sağlar. Yani dijital pindeki gerilim 0'dan 5 volta çıktığında kesmeye girer.

FALLING: Düşen kenar olduğunda kesmeye girmesini sağlar. Yani dijital pindeki gerilim 5'ten 0 volta çıktığında kesmeye girer.

HIGH: Arduino Due, Zero ve MKR1000'da Dijital pindeki gerilim 5 volt olduğunda kesmeye girmesini sağlar.

```
int ledPin = 12;
boolean ledDurumu = LOW;
int butonPin = 3;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(butonPin,INPUT);
  attachInterrupt(1, yanson, CHANGE);
}
void loop() {
}
void yanson() {
  ledDurumu = !ledDurumu;
  digitalWrite(ledPin, ledDurumu);
}
```

Resim 7.109: attachInterrupt() kullanım örneği

Yukarıdaki örnekte bir Led, buton ile kontrol edilmektedir. Bunun için 1 adet interrupt, CHANGE modunda kullanılmıştır.

detachInterrupt()

Verilen kesmeyi kapatmak için kullanılır. Devredışı bırakılacak kesmenin pin numarası parantez içine girilmelidir

Timer Interrupt

Zaman kesmesi (timer interrupt), belirli süre aralıklarında belirli görevlerin yapılabilmesi için kullanılır. Örneğin bir Led'in saniyede bir yakıp söndürülmesi işleminde. Bu işlem için loop fonksiyonunun kullanılması yerine, zaman kesmesinin kullanılması ile kesme her saniyede bir Arduino'ya haber vererek, LED'in yakılıp söndürülmesini sağlamaktadır. Timer interruptu kullanmak için hazır kütüphane olan <TimerOne.h> kütüphanesi kullanılmalıdır. Yandaki örnek bir LED'i 1'er saniye aralıklarla yakıp söndürmektedir.

```
#include <TimerOne.h>
void setup()
{
pinMode(13, OUTPUT);
Timer1.initialize(100000); // 1 saniyede tetikleniyor
Timer1.attachInterrupt( kontrolet ); }
void loop() {
void kontrolet()}
{
digitalWrite(13);
digitalRead(13) ^ 1 );
}
```

Resim 7.110: Timer Interrupt() kullanım örneği

7.11. Arduino Tümleşik Geliştirme Ortamında Seri Haberleşme

Arduino kartlarında seri iletişim TX / RX pinleri ve USB aracılığıyla gerçekleştirilir.

Seri bağlantı Arduino kartı ile bir bilgisayar veya diğer cihazlar arasındaki iletişim için kullanılır. Tüm Arduino kartları, en az bir seri porta (aynı zamanda UART veya USART olarak da bilinir) sahiptir. Dijital pin 0 (RX) ve 1 (TX) üzerinden ve ayrıca bilgisayar üzerinden (USB) aracılığıyla iletişim kurulabilir. Bu nedenle USB kullanıldığı sürece, dijital giriş veya çıkış için 0 ve 1 numaralı pimler kullanılamazlar. Bir Arduino kartıyla iletişim kurmak için Arduino ortamının dahili seri monitörü de kullanılabilir. Bu amaçla araç çubuğundaki seri monitör düğmesi tıklanarak ve "begin()" çağrısında kullanılan aynı baud hızı seçilmelidir. Birçok seri haberleşme fonksiyonu bulunmaktadır. Burada temel olanlara yer verilmiştir.

Serial.begin()

Bilgisayar ile Arduino arasında seri iletişimi başlatmak için kullanılır. Seri veri iletimi için veri hızı saniyedeki bit sayısı (baud) cinsinden ayarlanır. Veri kaybı yaşanmaması için Arduino üzerinde ayarlanan baud oranı ile bilgisayar üzerinde ayarlanan baud oranı aynı olmalıdır. 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, ve 115200 baud oranlarından biri alınan ve gönderilen verileri hızı olarak seçilebilir.

if (Serial): Belirtilen seri portun hazır olup olmadığını gösterir.

Serial.available(): Seri bağlantı noktasından okumak için kullanılabilir bayt (karakter) sayısını öğrenmeyi sağlar.

Serial.availableForWrite(): Yazma işlemini engellemeden seri arabelleğe yazılabilecek bayt (karakter) sayısını öğrenmek için kullanılır.

```
void setup() {
  Serial.begin(9600); // Seri iletim başlatılır ve portun açılmasını beklenir
  while (!Serial) {    // Seri portun (USB) bağlanması beklenir.
    ;
  }
}
void loop() {
 // Program noormal olarak devam eder
}
```

Resim 7.111: if(Serial)ve Serial.begin kullanım şekli

```
int gelenByte = 0;   // Gelen seri veri için değişken tanımlanıyor
void setup() {
  Serial.begin(9600); // Seri port 9600 bps veri hızına ayarlanıyor
}
void loop() {
  if (Serial.available() > 0) {   // Eğer veri alınmışsa
  gelenByte = Serial.read();      // Gelen bayt okunuyor
  Serial.print("Aldım: ");        // Seri ekrana "Aldım: " yazılıyor
  Serial.println(gelenByte, DEC); // Alınan bayt seri ekranda gösteriliyor
}
}
```

Resim 7.112: Serial.available(), Serial.read(), Serial.print() ve Serial.printIn() kullanım örneği

Serial.begin(): Seri veri iletimi için veri hızını saniyedeki bit sayısı (baud) cinsinden ayarlamak için kullanılır.

Serial.end(): Seri haberleşmeyi devre dışı bırakmak için kullanılır. Fakat genel giriş ve çıkış işlemleri için RX ve TX pinlerinin kullanılmasına izin verir.

Serial.findUntil(): Verilen uzunluktaki hedef dizgi bulunana kadar seri arabelleğinden veri okumak için kullanılır.

Serial.flush(): Giden seri iletim verilerinin tamamlanmasını beklemek için kullanılır.

Serial.parseFloat(): ilk geçerli kayan nokta sayısını seri arabelleğinde döndürmek için kullanılır.

Serial.parseInt(): Gelen seri akıştan bir sonraki geçerli tam sayıyı aramak için kullanılır.

Serial.peek(): Gelen seri verilerin bir sonraki baytını (karakterini) dahili seri arabelleğinden kaldırmadan döndürmek için kullanılır.

Serial.print(): Veriyi seri porta okunabilir ASCII metni olarak yazdırmak için kullanılır.

Serial.println(): Veriyi seri porta okunabilir ASCII metni olarak yazdırmak için kullanılır. Fakat sonuna satır sonu ekleyerek alt satıra geçmesi sağlanır.

Serial.read(): Gelen seri iletim verilerini okumak için kullanılır.

Serial.readBytes(): Karakterleri seri porttan bir arabelleğe okumak için kullanılır. Arabelleğe yerleştirilen karakter sayısını döndürür.

Serial.readBytesUntil(): Seri arabellekteki karakterleri bir diziye okumak için kullanılır. Arabelleğe okunan karakter sayısını döndürür.

Serial.readString(): Seri arabellekteki karakterleri bir dizeye okumak için kullanılır.

Serial.readStringUntil(): Seri arabellekteki karakterleri bir dizeye okumak için kullanılır.

Serial.setTimeout(): Serial.readBytesUntil (), Serial.readBytes (), Serial.parseInt () veya Serial.parseFloat () işlevlerini kullanırken seri veri beklenecek maksimum milisaniye değerini ayarlamak için kullanılır. Varsayılan süre 1000 milisaniyedir.

Serial.write(): İkili verileri seri bağlantı noktasına yazmak için kullanılır. Bu veriler bir bayt veya bayt serisi olarak gönderilir.

```
void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.write(45); // Değeri 45 olan bir bayt yazılıyor
// "Merhaba" dizesini gönderir ve dizgenin uzunluğunu döndürür
  int bytesSent = Serial.write("Merhaba");
```

Resim 7.113: Serial.write() kullanım şekli

serialEvent(): Veri mevcut olduğunda çağırılması için kullanılır. Bu verileri okumak için Serial.read() kullanılır.

7.12.Arduino Tümleşik Geliştirme Ortamında Seri Haberleşme Protokolleri

Dijital sistemlerde kablolu seri haberleşme ile ilgili birçok standart protokol bulunmaktadır. SPI ve I²C protokolleri bunlara örnek olarak verilebilir. Arduino kartı, diğer Arduino kartlarla veya algılayıcılarla (sensörlerle) haberleşmek için bu haberleşme protokollerini kullanmaktadır. Bu protokollerin kullandıkları uç sayıları, ulaşabilecekleri maksimum hızları ve kullanım şekilleri birbirinden farklıdır.

7.12.1. I²C Veri Yolu

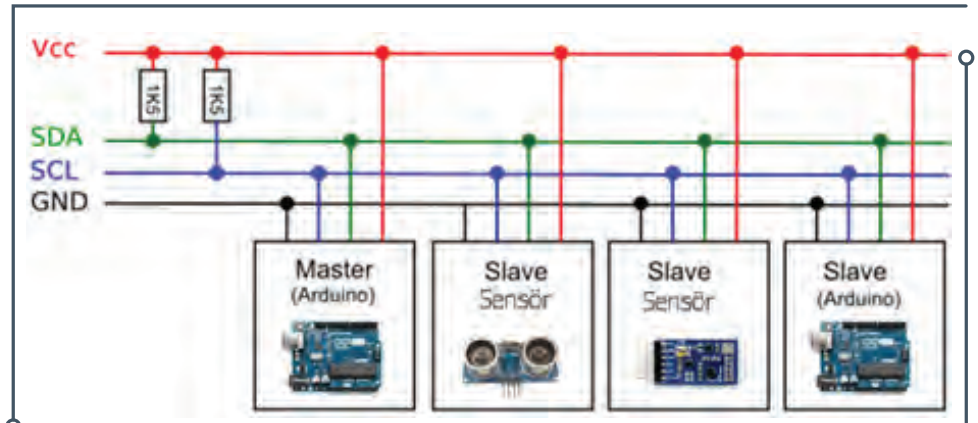
I²C (Inter-Integrated Circuit), oldukça hızlı veri aktarımına olanak tanıyan seri haberleşme türlerindendir. Bir arada çalışan, belirli aralıklarla birbiriyle haberleşen, çeşitli çevresel cihazların çok az harici donanım gereksinimiyle haberleşmelerini sağlar. Uzun mesafeli haberleşmelerde tercih edilmez. Genellikle kısa mesafeli ve düşük veri aktarım hızının yeterli olduğu yerlerde kullanılır. Haberleşme senkron (eş zamanlı) olarak gerçekleştirilir. Haberleşme için toprak hattı dışında SDA (SerialDAta) ve SCL (SerialCLock) olmak üzere iki hat bulunmaktadır. Haberleşme hızı 400kbps'ye kadar çıkabilmektedir.

I²C ile birden fazla cihaz adresleme planı içerisinde bulunduğu için kolayca haberleşebilmektedir. SDA ve SCL pinleri, kullanılan Arduino türüne göre değişiklik göstermektedir. Arduino türlerine göre SDA ve SCL pinleri aşağıdaki tabloda gösterilmiştir.

Arduino Kart Türü	SDA Pini	SCL Pini
Arduino Uno	A4	A5
Arduino Mega	20	21
Arduino Leonardo	2	3
Arduino Due	20	21
Arduino Nano	A4	A5

Tablo 7.5: Arduino türlerine göre SDA ve SCL pinleri

I²C haberleşmesinde, haberleşmeyi kontrol eden master cihazı bulunmalıdır. Haberleşmenin gerçekleşebilmesi için haberleşme hattına en az bir adet de slave (köle) cihaz bağlanmalıdır. Hatta bağlanan birden fazla slave cihazlardan hangisinin veri aktaracağına, master cihaz karar vermektedir. Böylece hat sayısında bir değişiklik olmadan birden fazla cihazla haberleşmenin yapılması sağlanır. Bağlantı şekilleri aşağıdaki tabloda gösterilmiştir. Haberleşme için #include ile <Wire.h> kütüphanesinin eklenmesi gereklidir. Haberleşmenin tüm hat boyunca hatasız bir şekilde sağlanabilmesi için SDA ve SCL hatları, pull-up dirençlerle VCC hattına bağlanmalıdır.



Resim 7.114: I²C veri yolu bağlantı şekilleri

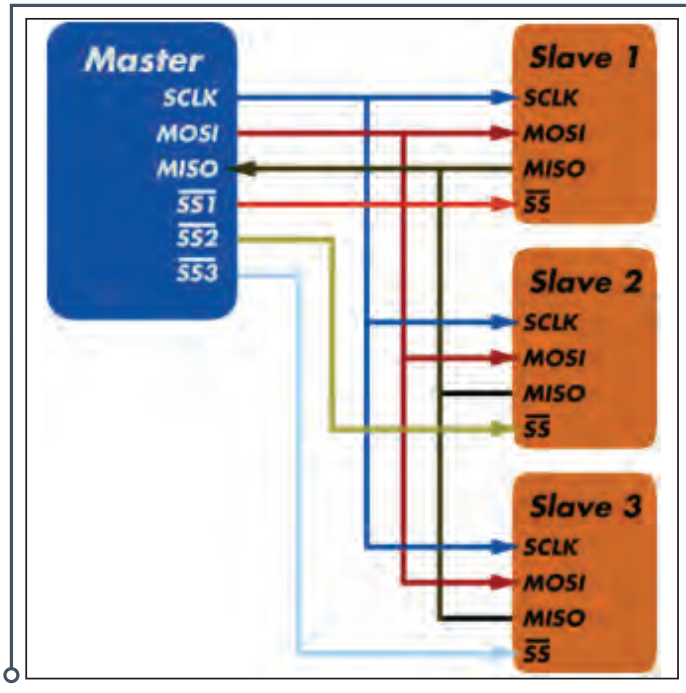
Aşağıdaki örnekte, robotik uygulamalarda yaygın olarak kullanılan MPU6050 3 eksenli gyro ve 3 eksen açısal ivme ölçer sensörünün I²C bağlantısı yapılarak test edilmesi sağlanmaktadır. Arduino Uno'da sda ve scl I²C pinleri sırayla A4 ve A5 pini olduğundan, sensör üzerindeki sda ve scl bağlantılarının o pinlere yapılması gerekmektedir. Bunların dışında güç pinlerinin bağlanması yeterlidir. Uygulama dersin sitesinde 7.12.1. numarasıyla yer almaktadır. Uygulamayı indirerek test ediniz. Gerekli kütüphaneyi https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050 adresinden indirebilirsiniz.

```
#include "I2Cdev.h"  // I2C kütüphanesi ekleniyor
#include "MPU6050.h" // Mpu6050 kütüphanesi ekleniyor
#include "Wire.h"    // Seri bağlantı kütüphanesi ekleniyor
MPU6050 accelgyro;   // Mpu6050 sensör tanımlanıyor
int16_t ax, ay, az;  // İvmeölçer tanımlanıyor
int16_t gx, gy, gz;  // Gyro sensör tanımlanıyor

void setup() {
Wire.begin();                          // Seri bağlantı kütüphanesi başlatılıyor
Serial.begin(9600);                    // Seri iletişim hızı tanımlanıyor
Serial.println("MPU6050 Başlatılıyor"); // Seri ekrana yazdırılıyor
accelgyro.initialize();                // Sensör başlatılıyor
Serial.println(accelgyro.testConnection() ? "MPU6050 Başarılı" : "MPU6050 Başarısız");
}

void loop() {
accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); // İvme ve gyro değerlerini okunuyor

// Açısal ivmeleri ve gyro değerlerini seri ekrana yazdırılıyor
  Serial.print("a/g:\t");
  Serial.print(ax); Serial.print("\t");
  Serial.print(ay); Serial.print("\t");
  Serial.print(az); Serial.print("\t");
  Serial.print(gx); Serial.print("\t");
  Serial.print(gy); Serial.print("\t");
  Serial.println(gz);
  delay (1000); // 1 sn bekletiliyor
}
```

Resim 7.115: I²C veri yolu uygulama örneği

7.12.2. SPI Veri Yolu

SPI (Seri Çevresel Arayüz - Serial Peripheral Interface), Arduino'nun desteklediği senkron (veri alma ve gönderme işleminin eş zamanlı olarak gerçekleştirildiği) seri haberleşme protokolüdür. Veri iletimi tek yönlü olarak sağlanmaktadır. Özellik ve kullanım olarak I²C ile oldukça benzerlik göstermektedir. Bir Arduino'nun diğer Arduino veya sensörlerle kısa mesafede haberleşmesini sağlamak için kullanılmaktadır. SPI protokolünde de I²C'de olduğu gibi bir adet Master cihaz bulunmaktadır. Bu cihaz hatta bağlı diğer çevresel cihazları kontrol etmektedir. SPI bağlantısı için 4 adet pin kullanılmaktadır.



Resim 7.116: SPI veri yolu bağlantı şekilleri

Master ve çevresel cihazlara bağlanan MOSI (Master Out Slave In),

MISO (Master In Slave Out) ve SCLK (Serial Clock) olmak üzere üç adet SPI hattı bulunmaktadır. MOSI: Master cihazdan yollanan verilerin çevresel cihazlara aktarıldığı hattır. MISO: Çevresel cihazlardan (slave) yollanan verilerin master cihaza aktarıldığı hattır. SCLK: SPI haberleşmesinde senkronu sağlayan saat sinyalinin bulunduğu hattır. Saat sinyali master cihaz tarafından üretilmektedir.

SPI protokolünde I²C'den farklı olarak veri hatları tek yönlüdür. Ayrıca çevresel cihazların (slave) adreslerinin olmasına gerek yoktur. Her çevresel cihazın seçim pini bulunur. Bu pine, SS (Slave Select) denir. Bu hattın sayısı kullanılan çevresel cihazların sayısı kadardır. Her cihaz için master cihazından ayrı SS hattı çıkar. SS hattı LOW (0 volt) düzeyinde olan çevresel cihaz, master cihaz ile iletişime başlar. Verilerin gönderilmesi ve alınması clock sinyali ile senkronize bir şekilde gerçekleşir. Veriler byte'lar halinde gönderilir / alınır. Yukarıdaki tabloda 3 adet slave çevresel aygıtın yer aldığı örnek bir SPI haberleşme hattı gösterilmiştir. Bu hatların bağlandığı SPI pinleri Arduino türüne göre değişiklik göstermektedir. Arduino türlerine göre değişen bu pinler aşağıdaki tabloda verilmiştir.

Arduino Kart Türü	MOSI	MISO	SCK	SS (Slave)	SS (Master)
Arduino Uno	11 veya ICSP4	12 veya ICSP1	13 veya ICSP3	10	-
Arduino Mega	51 veya ICSP4	50 veya ICSP1	52 veya ICSP3	53	-
Arduino Leonardo	ICSP-4	ICSP-1	ICSP-3	-	-
Arduino Due	ICSP-4	ICSP-1	ICSP-3	-	-

Tablo 7.6: Arduino türlerine göre kullanılan pinler

SPI veri yolu kullanarak haberleşmek için #include ile <SPI.h> kütüphanesinin uygulamaya eklenmesi gereklidir. Kütüphanenin uygulamaya eklenmesiyle kullanılabilecek fonksiyonlardan bazıları şunlardır:

SPI.begin(): SPI veri yolu haberleşmesini başlatmak için kullanılır.

SPI.end (): SPI veri yolunu devre dışı bırakmak için kullanılır.

SPI.beginTransaction(): Tanımlanan SPI ayarlarını kullanarak SPI veri yolunu başlatmak için kullnılır.

SPI.endTransaction(): Diğer kitaplıkların SPI veri yolunu kullanmasına izin vermek amacıyla mevcut kitaplığın SPI veri yolunu kullanmasını durdurmak için kullanılır.

SPI.usingInterrupt(): Program bir kesme işlemi içinde SPI işlemlerini gerçekleştirecekse, kesme numarası veya adını SPI kitaplığına kaydetmek için kullanılır.

SPI.setClockDivider(): SPI veri yolu haberleşmesinin saatini sistem saatine göre ayarlamak için kullanılır. Standart SPI saati "SPI_CLOCK_DIV4" olarak ayarlanmıştır. Fonksiyonun alabileceği diğer değişkenler; SPI_CLOCK_DIV8, SPI_CLOCK_DIV16, SPI_CLOCK_DIV32, SPI_CLOCK_DIV64, SPI_CLOCK_DIV128'dir.

SPI.transfer(): SPI hattına eşzamanlı olarak veri göndermek veya veri almak için kullanılır.

Aşağıdaki örnekte birbirine SPI pinleri ile bağlı 2 Arduino UNO arasından SPI protokolu ile yapılan veri alışverişinin master ve slav kodları yeralmaktadır. Uygulama, dersin sitesinde 7.12.2. numarasıyla yer almaktadır. Uygulamayı indirerek test ediniz.

Master Kodu

```
#include <SPI.h>
void setup (void) {
   Serial.begin(115200); // Baud hızı 115200 olarak ayarlanıyor
   digitalWrite(SS, HIGH); // Slave Select devre dışı bırakılıyor
   SPI.begin ();
   SPI.setClockDivider(SPI_CLOCK_DIV8);// Sistem saati 8'e bölünüyor
}
void loop (void) {
   char c;
   digitalWrite(SS, LOW); // Slave Select etkinleştiriliyor
   // Test dizesi gönderiliyor
   for (const char * p = "Merhaba, Dünya!\r" ; c = *p; p++) {
      SPI.transfer (c);
      Serial.print(c);
   }
   digitalWrite(SS, HIGH); // Slave Select devre dışı bırakılıyor
   delay(2000);
}
```

Resim 7.117: SPI veri yolu uygulaması master örneği

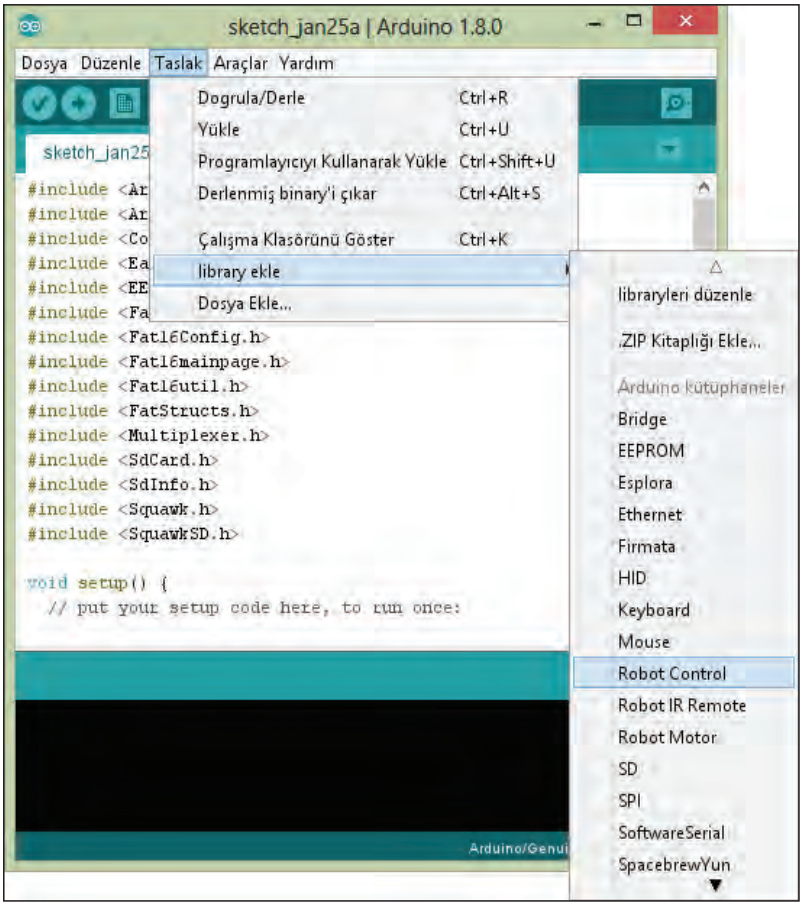
Slave Kodu

```
#include <SPI.h>
char buff [50];
volatile byte indx;
volatile boolean process;
void setup (void) {
   Serial.begin (115200); // Seri iletim Master ile aynı olmalı
   pinMode(MISO, OUTPUT); // Çıktı olarak ayarlanacak şekilde master'ı gönderiliyor
   SPCR |= _BV(SPE); // Köle modunda SPI'yı açılıyor
   indx = 0; // Arabellek boş
   process = false;
   SPI.attachInterrupt(); // Kesme açılıyor
}
ISR (SPI_STC_vect) // SPI kesme rutini
   byte c = SPDR; // SPI Veri Kaydı'ndan bayt okunuyor
   if (indx < sizeof buff) {
      buff [indx++] = c; // Veriler dizinin önbellekteki bir sonraki dizininde saklanıyor
      if (c == '\r') // Sözcüğün sonu kontrol ediliyor
      process = true;
   }
}
void loop (void) {
   if (process) {
      process = false; // İşlem sıfırlanıyor
      Serial.println (buff); // Dizi seri monitörde yazdırılıyor
      indx= 0; // Resetlenerek sıfırlanıyor
   }
}
```

Resim 7.118: SPI veri yolu uygulaması slave örneği

7.13. Kütüphaneler

Arduino kütüphaneleri belirli görevleri yerine getirecek bileşen bilgilerini içeren yapılardır ve bu yapıları sayesinde yapılacak işlemlere daha kısa yoldan ve komut karmaşası olmadan ulaşılmasını sağlarlar. Diğer bir ifadeyle bu kütüphaneler sayesinde mikrodenetleyiciler ve kullanılan bileşenler ayrıntılı olarak bilinmese de kolayca programlanabilmektedir. Arduino projelerini oluşturan elektronik bileşenler, çeşitli sensörler, motorlar, LCD ekranlar, butonlar gibi çok fazla sayıda ek elemana ihtiyaç duymaktadır. Bu elemanlardan bir çoğu ise kendi içerisinde başlı başına bir yapıya, çalışma örgüsüne sahiptirler. Bu elemenların Arduino ile kullanımı, o eleman tarafından yapılacak iş için gerekli kodlamanın da programcı tarafından yapılmasını gerektirmektedir. Bu gerekliliğin sağlanmasını, bu elemanlar ile Arduino programlamasınının birleştirilmesi ve aralarında köprü görevi görecek yapıların oluşturulması görevi kütüphaneler tarafından gerçekleştirilmektedir. Kütüphanesi olmayan bileşenler için kütüphaneler kullanıcı tarafından oluşturabileceği gibi genellikle kullanılan bileşenler, bileşen üretici firmalar tarafından hazırlanırlar veya Arduino IDE ile birlikte hali hazırda yüklü olarak gelirler. Kullanılacak bileşenin ihtiyaç duyduğu kütüphane, hazırlanan programa bir komut aracılığıyla eklenerek bu bileşen kolayca kullanıma hazır hale getirilmektedir. Kütüphaneler içeriğinde belli başlı ek komutlar ve değişkenler içerirler. Programa eklenen bir kütüphane ile bu kütüphane içeriğinde bulunan komutları ve değişkenler hazırlanan program üzerinde kullanıma hazır hale gelmektedir.



Resim 7.119: Arduino IDE'de kütüphane ekleme

Belli başlı güncel kütüphaneler Arduino IDE ile yüklü halde gelmektedir. Bunlar IDE penceresinde

Taslak>library ekle sekmesinde açılan listeden istenilen seçilerek programa eklenmektedir. Burada yüklü gelen tüm kütüphaneler de görülebilmektedir. Ancak kullanılacak kütüphane farklı bir kaynaktan temin ediliyorsa kütüphane dosyası, Arduino'nun kurulu olduğu dosya konumu altında **libraries** klasörü içerisine kopyalanmalıdır. Ayrıca yapılacak programın en başına yazılacak **#include <KütüphaneAdı.h>** şeklinde bir komutla da çalışılan programa eklenmelidir. Burada standart kütüphane Arduino Robot kütüphanesi kısaca açıklanmıştır.

7.13.1. Arduino Standart Kütüphaneleri

EEPROM: Kalıcı hafızaya veri yazmak ve okumak için kullanılmaktadır.

Ethernet / Ethernet 2: Arduino Ethernet Shield, Arduino Ethernet Shield 2 ve Arduino Leonardo ETH kullanarak internete bağlanmak için kullanılmaktadır.

Firmata: Standart bir seri protokol kullanılarak bilgisayardaki uygulamalarla iletişim kurmak için kullanılmaktadır.

GSM: GSM Shield ile bir GSM / GPRS ağına bağlanmak için kullanılmaktadır.

LiquidCrystal: Likit kristal ekranları (LCD) kontrol etmek için kullanılmaktadır.

SD: SD kartları okumak ve yazmak için kullanılmaktadır.

Servo: Servo motorları kontrol etmek için kullanılmaktadır.

SPI: Seri Çevresel Arabirim (SPI) kullanan cihazlar ile iletişim kurmak için kullanılmaktadır.

SoftwareSerial: Herhangi bir dijital pin üzerinden seri haberleşme yapmak için kullanılmaktadır.

Step: Step motorlar kontrol etmek için kullanılmaktadır.

TFT: Arduino TFT ekranda metin, resim ve şekiller çizmek için kullanılmaktadır.

Wi-Fi: Arduino üzerinde Wi-Fi Shield kullanarak internete bağlanmak için kullanılmaktadır.

Wire: İki Tel Arabirimi (TWI / I^2C veri yolu üzerindeki cihaz veya sensörlerden) veri almak ve göndermek için kullanılmaktadır.

7.13.2. Arduino Robot Kütüphanesi

Robot kütüphanesi Arduino IDE 1.0.5 ve sonrası ile birlikte gelmektedir. Kütüphane, Arduino tarafından satışı yapılan Arduino Robot fonksiyonlarına kolayca erişmek için tasarlanmıştır. Robota Stok Kartı Yazılımı yüklendiğinde, robotu oluşturan dâhili algılayıcıları ve aktüatörleri için destek sağlamaktadır. Robotun ve ana kontrol kartı ve motor kontrolünden oluşan iki temel bileşen için gerekli araçları sunar. Her kart ayrı bir programlanabilir işlemciye sahiptir ve robot kütüphanesiyle kolayca kullanılabilir. Örneğin kütüphane; kontrol kartındaki çeşitli algılayıcılar ve çevre birimleri ile arabirim oluşturulmasına, motor hızı ve yönünün kontrol edilmesine, I^2C bağlantı noktasının kontrol edilmesine vb. olanak sağlamaktadır. Bunlar için oluşturulan RobotControl sınıfı fonksiyonları robot kontrol kartına ve kontrol kartındaki tüm giriş çıkışlara (I/O) ve motorlara komut verilmesine olanak sağlamaktadır. RobotMotor sınıfı ise programcının motor kontrolü için kendi bellenimini (firmware) oluşturmasına olanak tanımaktadır.

7.14. Düşünelim / Araştıralım / Uygulayalım

Burada verilen uygulamaların çözümleri dersin web sayfasında uygulama numaralarıyla yer almaktadır. Öncelikle uygulamaları kendiniz yapmaya çalışınız. İlk olarak uygulamaya ait akış diyagramlarını hazırlayarak başlayınız. Sonra uygun bileşenleri biraraya getirerek gerekli bağlantıları hazırlayınız. Son olarak da kod yazmaya geçiniz. Zorlandığınız durumlarda çözüm ve kontrol için örneklere bakmanız daha uygun bir yöntem olacaktır. Hazırlanmış örnekleri de inceleyerek, üzerinde değişiklikler yaparak etkilerini gözlemleyiniz. Farklı çözüm yöntemleri düşünrek araştırma yapınız.

Uygulamalarda kullanılan elektronik devre elemanları ve algılayıcılar ile ilgili ayrıntılı bilgilerin teknik dökümanlarına (datasheet) bakılarak gözden geçirilmesinde yarar bulunmaktadır. Bu dökümanlara İnternet üzerinde yapılacak basit bir aramayla ulaşılabilmektedir. Dökümanları inceleyerek bileşenlerin çalışma yapısı, elektriksel özelikleri ve mikrodenetleyici kartlarla kullanım şekillerini, bağlantı pinleri ve yapılarını öğrenebilirsiniz.

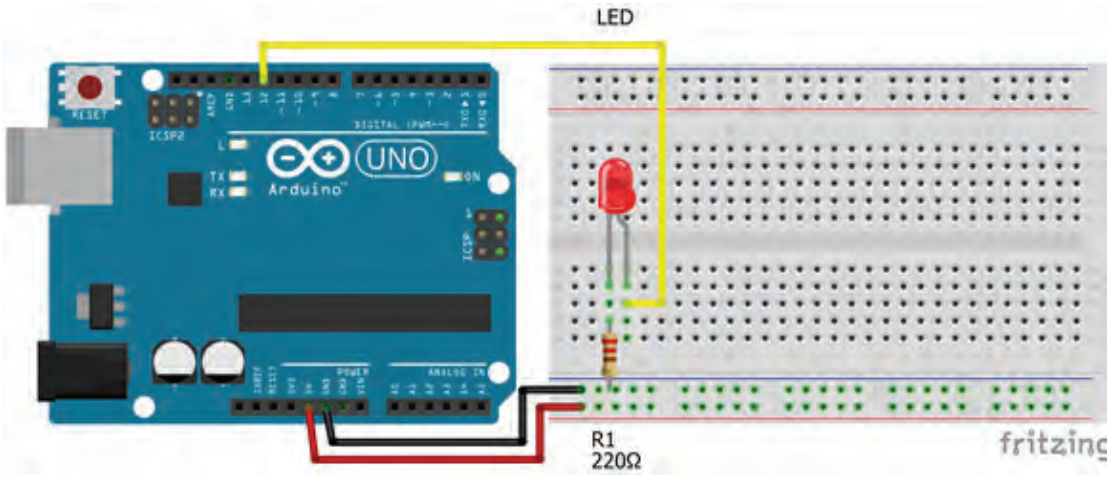
Eğer uygulamalarda kullanılan algılayıcıların arduino ile haberleşmesinde kütüphane kullanılacaksa veya bu zorunluysa uygun kütüphanenin çalışmaya eklenmesi gerektiği unutulmamalıdır. Zira eklenen kütüphane dosyaları algılayıcıdan gelen farklı yapıdaki bilginin okunması görevini üstlenerek kullanımı kolaylaştırmaktadir. Bu kütüphaneler uzmanlar tarafından hazırlanmakta ve algılayıcılardan gelen bilgiyi anlamlı hale getiren kod parçacıklarından meydana gelmektedir. Algılayıcılara ait Arduino kütüphane dosyaslarına da İnternette yapılan kısa bir arama ile ulaşılabilmektedir. İndirilen kütüphane dosyaları Arduino IDE programına ait program dosya kütüphanesine (libraries) kopyalanması gerekmektedir.

Burada yer alan örnek uygulamalarda Arduino UNO R3 versiyonu kullanılmıştır. Aksi belirtilmeyen uygulamalarda besleme voltajları Arduino UNO'nun 5 Volt (+) ve GND (-) pinlerinden bağlanmalıdır. Diğer bağlantılar uygulamada açıklanmış ve Breadboard çizimi üzerinde gösterilmiştir. Temel bileşenler dışında elektronik devre elemanı olarak yalnızca direnç kullanılmıştır. Direnç elektronik devrelerde akımı sınırlayan ve gerilimi bölen, iki uçlu bir elemandır. R harfi ile sembolik olarak gösterilir ve birimi Ohm (Ω)'dur. Örnek uygulamalarda LED'lerin ve gerekli olan diğer bileşenlerin korunması için kullanılmıştır.

Yapılan örnek uygulamalarda; Arduino IDE programı ile gelen hazır örneklerden, İnternette yer alan açık kaynak uygulamalardan ve algılayıcılar için hazırlanan örnek programlardan da yararlanılmıştır. Bütün uygulamalar hazırlanarak test edilmiştir.

7.14.1. LED Yakma Uygulaması

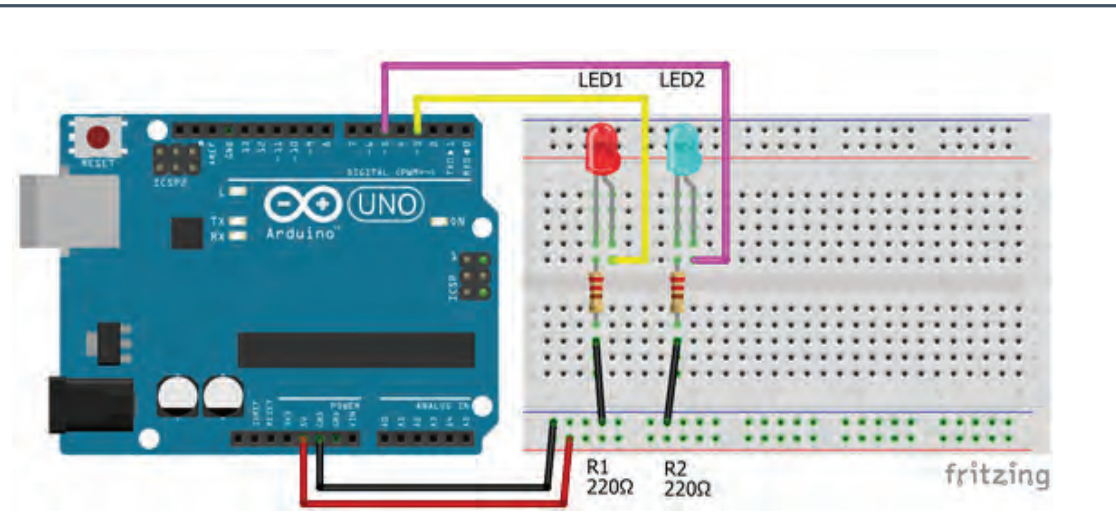
Arduino'nun 12 numaralı dijital pinlerine bağlı bir LED'in 1 sn aralıklarla yanıp sönmesini sağlayacak bir uygulama hazırlayınız. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 220 ohm (Ω) direnç kullanılmıştır.



Resim 7.120: LED yakma uygulaması breadboard çizimi

7.14.2. PWM LED Uygulaması

Arduino'nun 3 ve 5 numaralı pwm pinlerine bağlı 2 LED'in çalıştığı sürece ışık seviyesinin artarak yanması sağlayan bir uygulama hazırlayınız. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 2 adet 220 Ω direnç kullanılmıştır.

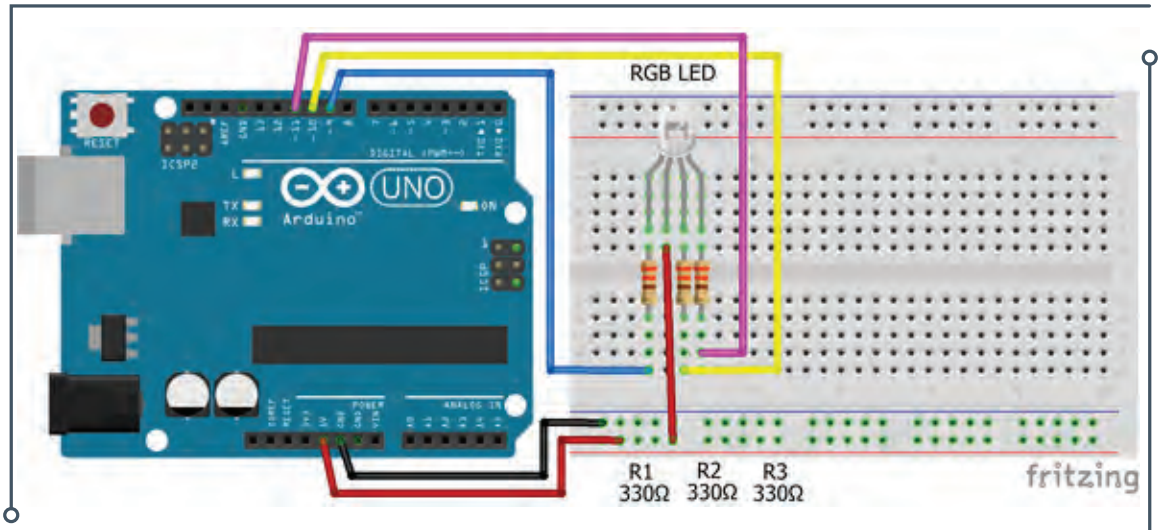


Resim 7.121: PWM uygulaması breadboard çizimi

7.14.3. RGB LED Uygulaması

RGB LED'in kırmızı Led pinini Arduino'nun 9, yeşil Led pininin 10 ve mavi Led pinini 11 numaralı pwm pinlerine bağlayarak RGB LED'in kırmızı, yeşil, mavi, sarı, camgöbeği ve beyaz renklerde veriler

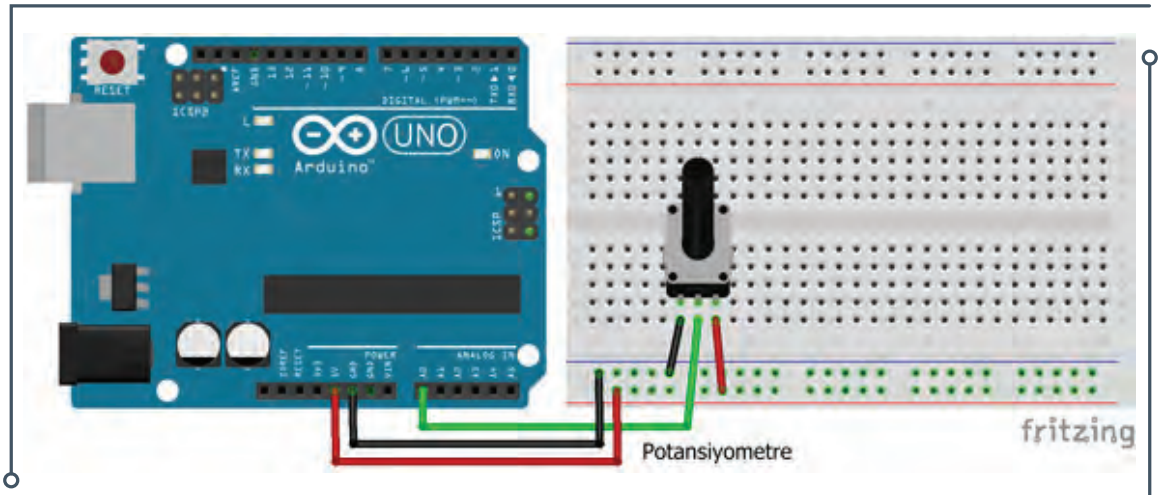
sırayla yanması sağlayacak bir uygulama hazırlayınız. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir RGB LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 3 adet 330 Ω direnç kullanılmıştır.



Resim 7.122: PWM uygulaması breadboard çizimi

7.14.4. Potansiyometre Uygulaması

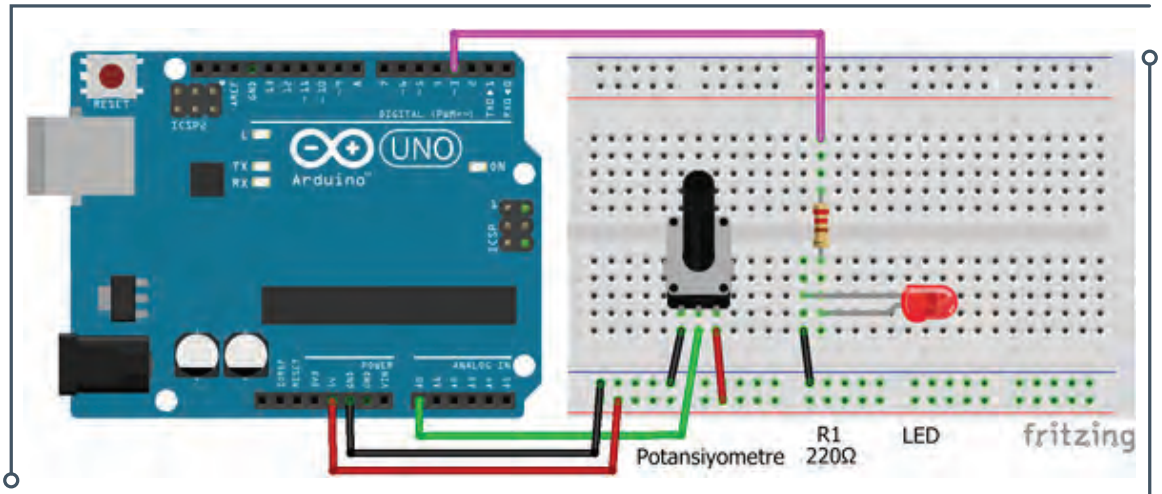
Arduino'nun A1 numaralı analog pinine bağlı bir potansiyometre kullanılarak değer okuma uygulaması hazırlayınız. Potansiyometrenin hareketi ile değişen direnç değerleri Arduino IDE seri port ekranında okunmalıdır. Potansiyometrenin Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. Örnek uygulamada 100 kiloohm (kΩ) potansiyometre kullanılmıştır.



Resim 7.123: Potansiyometre uygulaması breadboard çizimi

7.14.5. Potansiyometre ile PWM LED Uygulaması

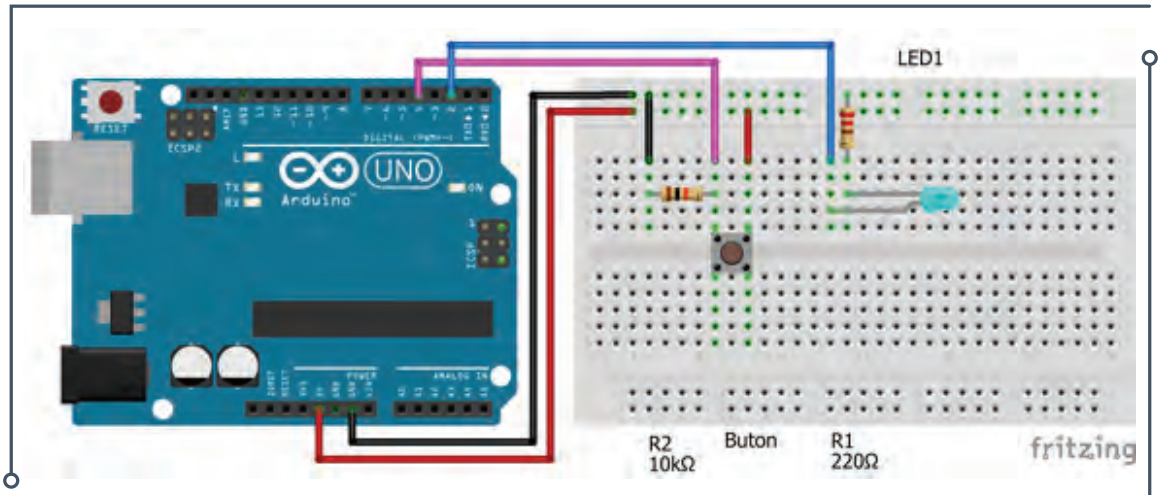
Arduino'nun A0 numaralı analog pinine bağlı bir potansiyometre kullanılarak, Arduino'nun 3 numaralı PWM pinine bağlı bir LED'in ışık seviyesinin kontrolülü sağlayacak bir uygulama hazırlayınız. Potansiyometrenin hareketi ile ışığın parlaklığı ayarlanabilmelidir. Potansiyometre ve LED'in Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 220 Ω direnç ve 100 kΩ potansiyometre kullanılmıştır.



Resim 7.124: Potansiyometre ile PWM LED uygulaması breadboard çizimi

7.14.6. Buton ile LED Yakma Uygulaması

Arduino'nun 4 numaralı dijital pinlerine bağlı bir buton kullanılarak 2 numaralı dijital pinine bağlı LED'in kontrol edilmesini sağlayacak bir uygulama hazırlayınız. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada LED için 220 Ω, buton için 10 kΩ direnç kullanılmıştır.

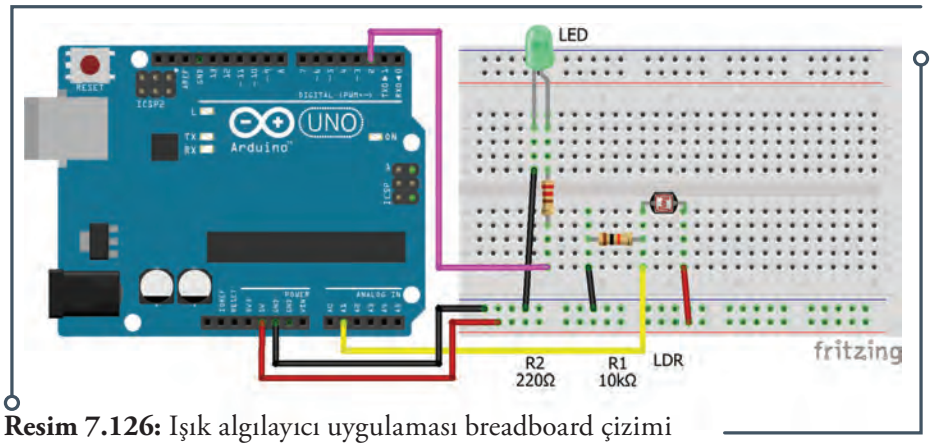


Resim 7.125: Buton uygulaması breadboard çizimi

7.14.7. Işık Algılayıcı Uygulaması

Sinyal çıkış pini Arduino'nun A1 numaralı analog pinine bağlı bir ışık algılayıcısı (LDR) kullanarak ışık algıladığı zaman, ışığın şiddetine göre 2 numaralı dijital pine bağlı LED'i sürekli olarak yakıp söndüren bir uygulama hazırlayınız. Işık şiddetine göre değişen değerler Arduino IDE seri port ekranında okunmalıdır.

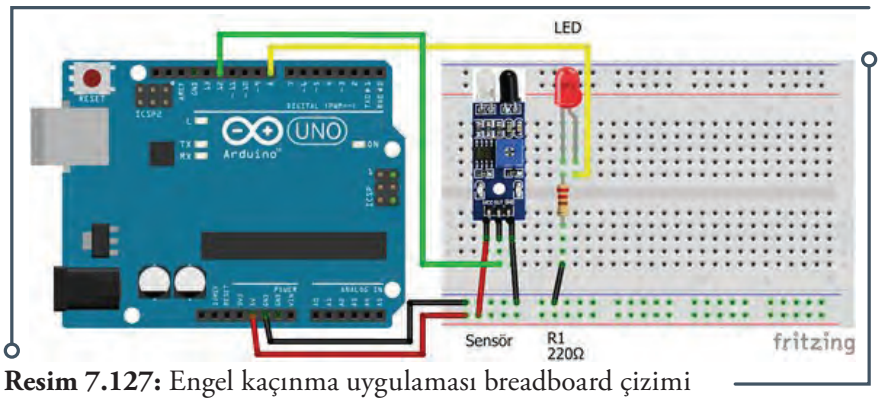
Aynı devre üzerinde yapacağınız ikinci uygulamada ise LED'in karanlıkta yanmasını aydınlıkta sönmesini sağlayınız. LDR algılayıcının algıladığı ışık şiddetini belirleyerek, okunan değer sizin belirleyeceğiniz değerden küçük olunca LED'in yanmasını değilse (büyükse) sönmesini sağlayınız. Işık algılayıcının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada LED için 220 Ω, LDR için 10 kΩ direnç kullanılmıştır.



Resim 7.126: Işık algılayıcı uygulaması breadboard çizimi

7.14.8. Engel Kaçınma Uygulaması

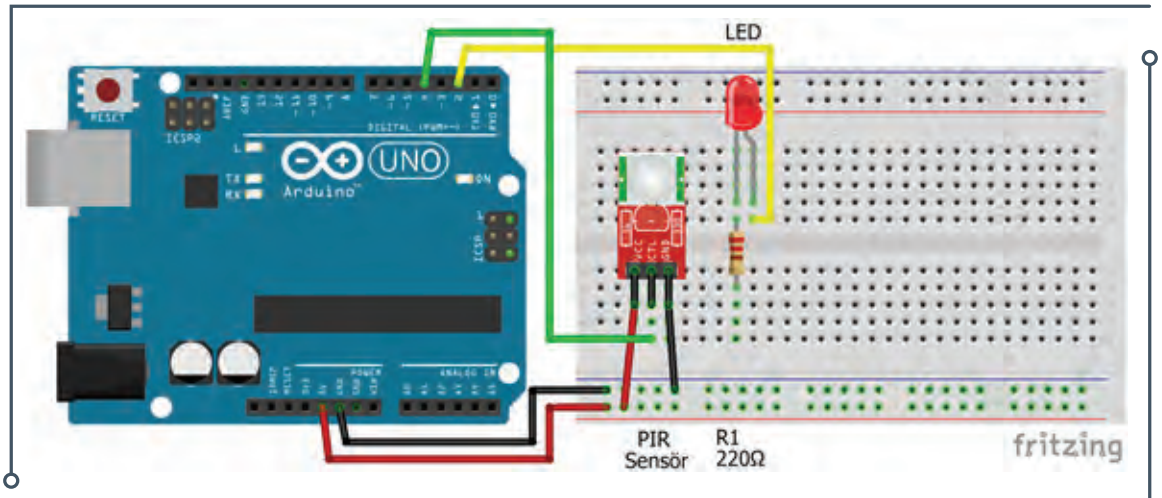
Sinyal çıkışı Arduino'nun 12 numaralı pinine bağlı bir engel kaçınma algılayıcısı kullanılarak engeli algıladığı zaman 4 numaralı dijital pine bağlı LED'i yakan bir uygulama hazırlayınız. Engel kaçınma algılayıcısının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada LED için 220 Ω direnç kullanılmıştır.



Resim 7.127: Engel kaçınma uygulaması breadboard çizimi

7.14.9. PIR Algılayıcı Uygulaması

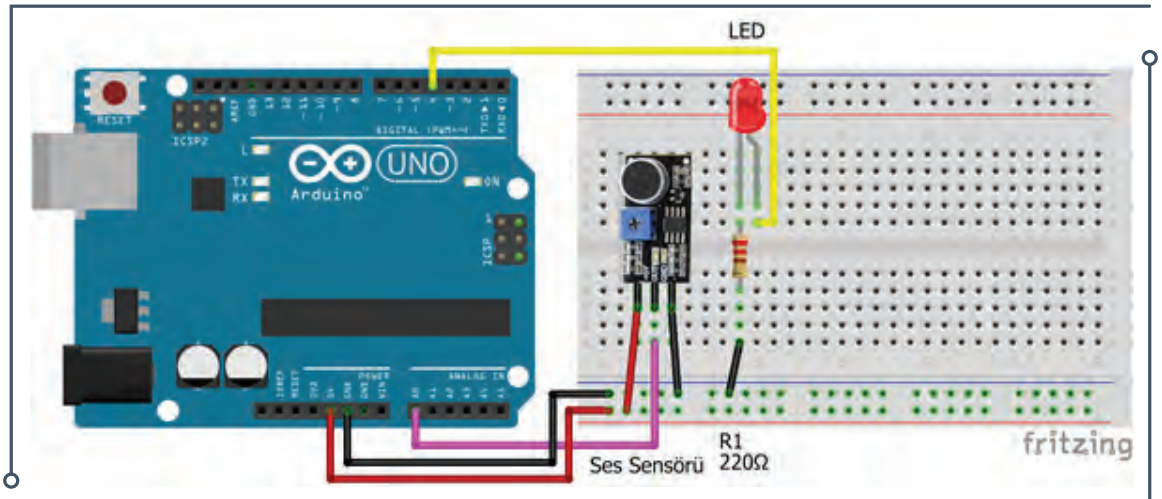
Sinyal çıkış pini Arduino'nun 2 numaralı dijital pinine bağlı bir PIR algılayıcısı kullanarak canlı algıladığı zaman 4 numaralı dijital pine bağlı LED'i yakan bir uygulama hazırlayınız. PIR algılayıcının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 220 Ω direnç kullanılmıştır.



Resim 7.128: PIR algılayıcı uygulaması breadboard çizimi

7.14.10. Sesle LED Kontrol Uygulaması

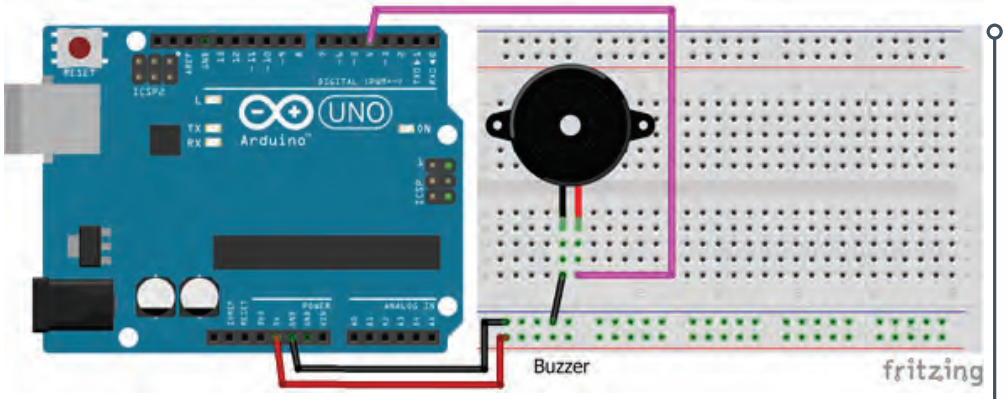
Sınyal çıkışı Arduino'nun A0 numaralı analog pinine bağlı bir ses algılayıcısı kullanarak ses seviyesine göre 4 numaralı dijital pine bağlı LED'i yakıp söndüren ses ile LED kontrol uygulaması hazırlayınız. Ses algılayıcısının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 220 Ω direnç kullanılmıştır.



Resim 7.129: Sesle LED kontrol uygulaması breadboard çizimi

7.14.11. Buzzer Uygulaması

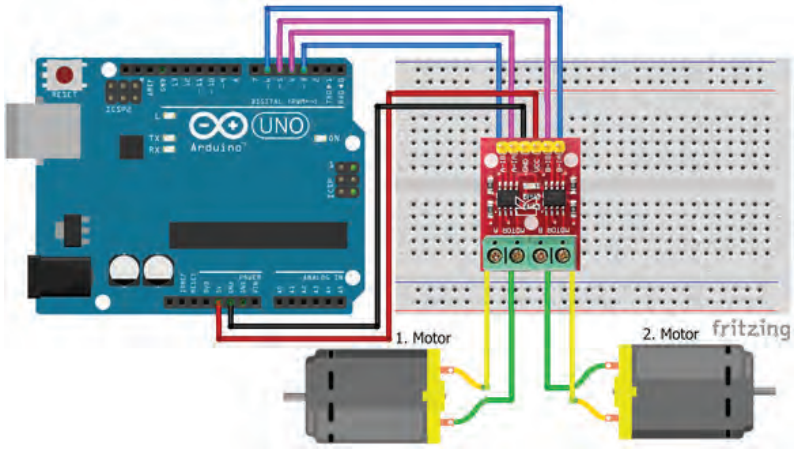
Arduino'nun 4 numaralı dijital pinine bağlı bir buzzer kullanılarak 2 farklı tonla alarm sesi üretmeyi sağlayacak bir uygulama hazırlayınız. Buzzer'in Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmiştir. Örnek uygulamada standart bir buzzer kullanılmıştır.



Resim 7.130: Buzzer uygulaması breadboard çizimi

7.14.12. DC Motor Kontrol Uygulaması

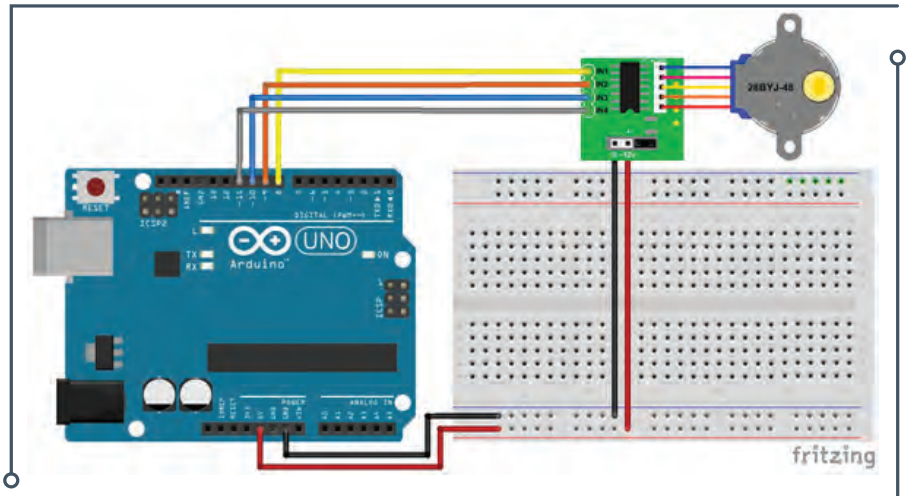
Motor sürücüye bağlı bir çift motoru ileri, geri, sağa ve sola hareket ettirip durduran bir motor kontrol uygulaması hazırlayınız. Bu amaçla farklı motor sürücü kartları kullanılabilir. Uygun voltajda (en fazla 6 volt) herhangi bir dc motor kullanılabilir. Daha güçlü motor kullanılacaksa Arduino UNO'nun zarar görmemesi için harici bir güç kaynağı ile motorların beslenmesini sağlayınız. Bu amaç için hazırlanan örnekte L9110 motor sürücü kartı kullanılmıştır. Sürücünün A-IA girişi Arduino'nun 3 numaralı dijital pinine, sürücünün A-IB girişi Arduino'nun 4 numaralı dijital pinine, sürücünün B-IA girişi Arduino'nun 5 numaralı dijital pinine, sürücünün B-IB girişi Arduino'nun 6 numaralı dijital pinine bağlanmalıdır. Motor sürücünün Arduino ve motorlarla bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir.



Resim 7.131: DC motor kontrol uygulaması breadboard çizimi

7.14.13. Adım (Step) Motor Kontrol Uygulaması

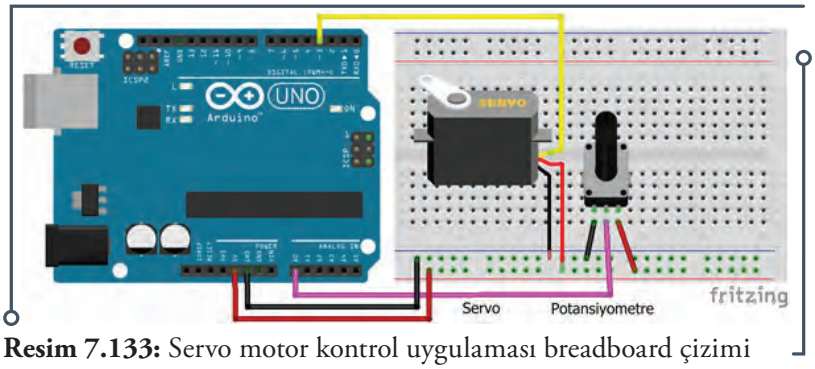
Adım motor sürücüye bağlı bir adım motorun; saat yönünde ve saat yönünün tersinde, verilecek olan adım sayısı kadar adım atmasını sağlayacak adım motor kontrol uygulaması hazırlayınız. Bu amaç için hazırlanan örnekte 28 BYJ-48 redüktörlü adım motor ve ULN2003A adım motor sürücü kartı kullanılmıştır. Daha güçlü adım motor kullanılacaksa Arduino UNO'nun zarar görmemesi için harici bir güç kaynağı ile adım motorun beslenmesini sağlayınız. Motor sürücünün 1N1 pini Arduino'nun 8 numaralı dijital pinine, 1N2 pini Arduino'nun 9 numaralı dijital pinine, 1N3 pini Arduino'nun 10 numaralı dijital pinine ve 1N4 pini Arduino'nun 11 numaralı dijital pinine bağlanmıştır. Adım motor bağlantısı için kart üzerinde konnektör bulunmaktadır. Adım motor sürücünün motor ve Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir.



Resim 7.132: Adım (Step) motor kontrol uygulaması breadboard çizimi

7.14.14. Servo Motor Kontrol Uygulaması

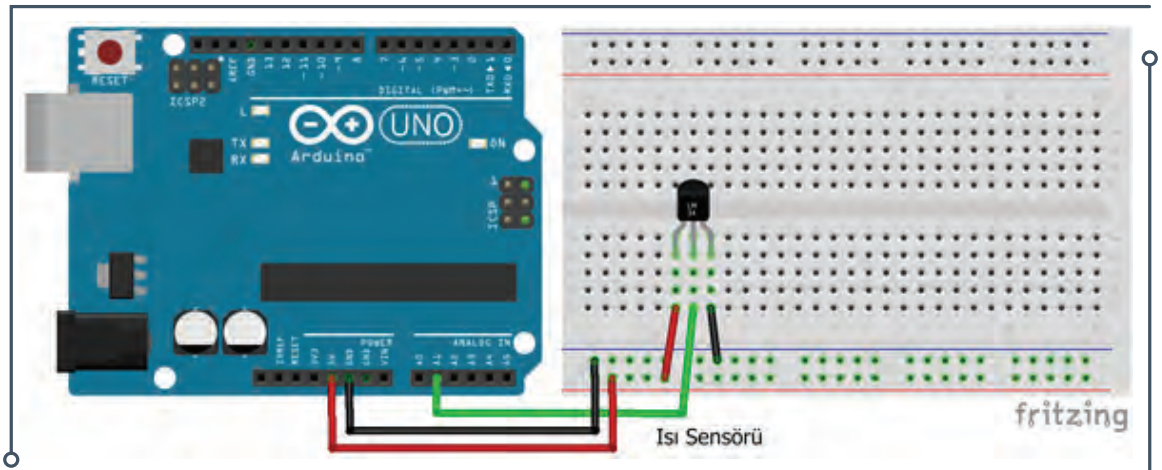
Arduino'nun A0 numaralı analog pinine bağlı bir potansiyometre kullanılarak 0 ile 180 derece arasında hareket eden bir servo motor kontrol uygulaması hazırlayınız. Potansiyometrenin değerini değiştirerek servo motorun 0 ile 180 derece arasında pozisyon alması gerekmektedir. Potansiyometreden okunan analog değerin 0 ile 1023 arasında olduğunu ve map yöntemiyle 0 ile 180 derece arasında sınırlandırılması gerektiğini dikkate alınız. Servo motorun ve potansiyometrenin Arduino ile bağlantısı yukarıdaki breadboard çizimi üzerinde gösterilmektedir. Örnek uygulamada 9 gramlık standart bir servo motor ile 100 kΩ potansiyometre kullanılmıştır.



Resim 7.133: Servo motor kontrol uygulaması breadboard çizimi

7.14.15. Isı Algılayıcısı Uygulaması

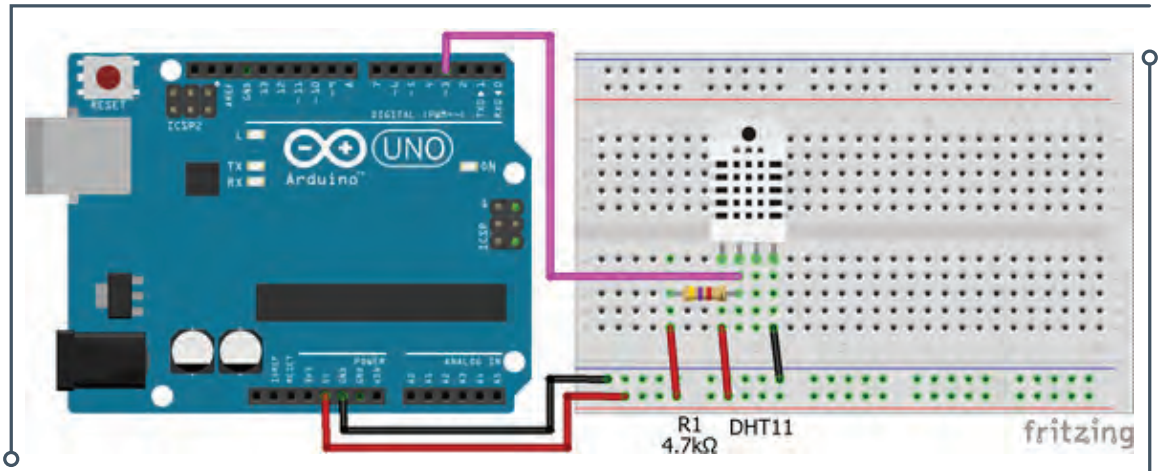
Sensör ucu Arduino'nun A1 numaralı analog pinine bağlı bir ısı algılayıcısı kullanılarak ortam ısısını ölçmeyi sağlayacak bir uygulama hazırlayınız. Ortam ısısı bilgisi Arduino IDE seri port ekranından okunmalıdır. Bu amaç için hazırlanan örnekte LM34 ısı algılayıcısı kullanılmıştır. Isı algılayıcısının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmiştir.



Resim 7.134: Isı algılayıcı uygulaması breadboard çizimi

7.14.16. Nem-Isı Algılayıcısı Uygulaması

Sensör ucu Arduino'nun 3 numaralı dijital pinine bağlı bir nem-ısı algılayıcısı kullanılarak içinde bulunulan ortamın nem ve ısısını ölçmeyi sağlayacak bir uygulama hazırlayınız. Uygulamada algılayıcının doğru şekilde çalışması için 4.7k'lık bir direncin çizimde gösterildiği gibi sensör ucu ile besleme voltajı (5 Volt) arasına bağlanması gerekmektedir. Ortamın nem ve ısı bilgisi Arduino IDE seri port ekranından okunmalıdır. Bu amaç için hazırlanan örnek uygulamada 4.7 kΩ direç ve DHT11 nem-ısı algılayıcısı ile buna ait https://github.com/RobTillaart/Arduino/tree/master/libraries/DHTlib adresinde bulunan kütüphane kullanılmıştır. Nem-ısı algılayıcısının Arduino ile bağlantısı yukarıdaki breadboard çizimi üzerinde gösterilmiştir.

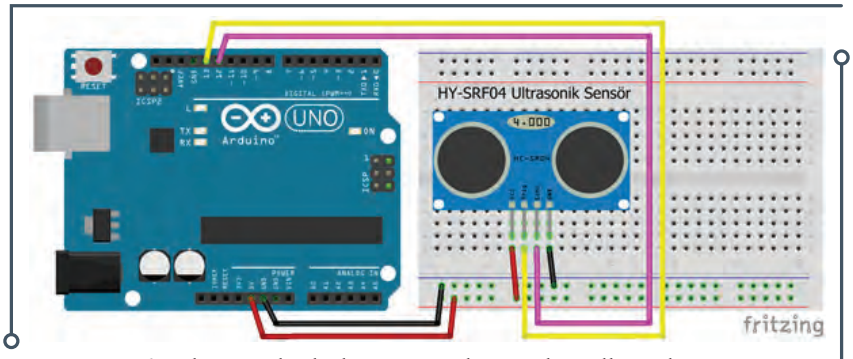


Resim 7.135: Nem-ısı algılayıcı uygulaması breadboard çizimi

7.14.17. Ultrasonik Algılayıcı Uygulaması

HY-SRF04 veya HY-SRF05 ultrasonik algılayıcı kullanılarak bir mesafe ölçme uygulaması hazırlayınız. Bu amaç için hazırlanan örnek uygulamada HY-SRF04 model ultrasonik algılayıcı kullanılmıştır. Ultrasonik algılayıcının tetik (trig) pini Arduino'nun 13 numaralı, okuma (echo) pini 12 numaralı dijital pinlerine bağlanmıştır. Ölçme 2 ile 200 cm aralığı için sınırlandırılmalı ve mesafe bilgisi Arduino IDE seri port ekranından okunmalıdır. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir.

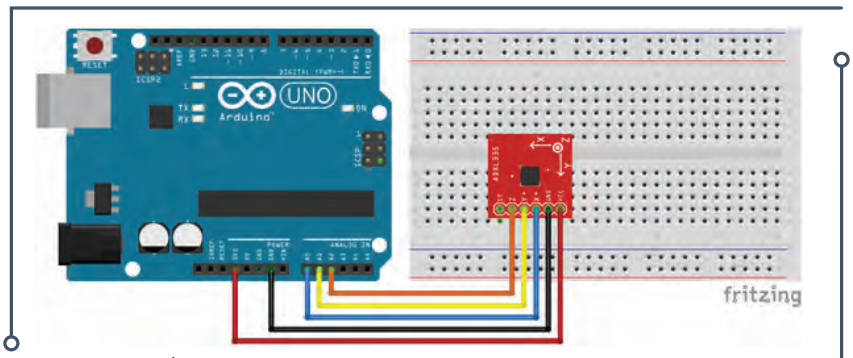
Aynı devre üzerinde yapacağınız ikinci mesafe ölçme uygulamasında ise aynı ultrasonik algılayıcının tetik (trig) ve okuma (echo) pini birleştirilerek, Arduino'nun 12 numaralı dijital pinine bağlanmalıdır. Bunun dışındaki bağlantılar aynı şekilde kalmalıdır. Uygulamada mesafe bilgisi Arduino IDE seri port ekranından okunmalı ve http://playground.arduino.cc/Code/NewPing adresinde bulunan "NewPing" kütüphanesi kullanılmalıdır.



Resim 7.136: Ultrasonik algılayıcı uygulaması breadboard çizimi

7.14.18. İvmeölçer Uygulaması

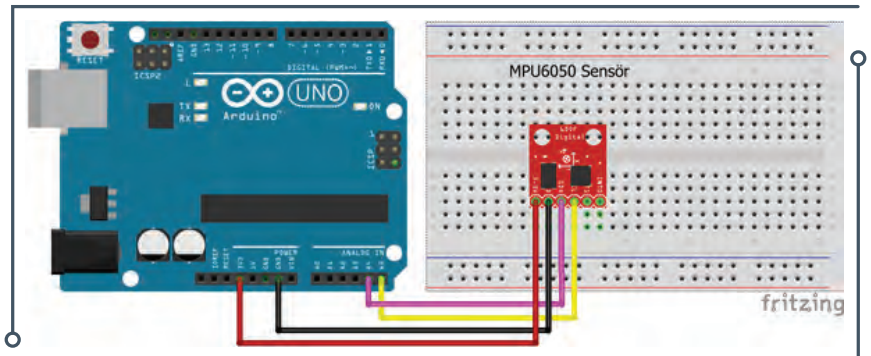
İvme ölçerin X ekseni sınyal çıkışı Arduino'nun A0, Y ekseni sınyal çıkışı Arduino'nun A1 ve Z ekseni sınyal çıkışı Arduino'nun A2 numaralı analog pinlerine bağlı ivmeölçer algılayıcısı kullanarak 3 eksenli ivme ölçme uygulaması hazırlayınız. Bu amaçla farklı ivmeölçer agılayıcıları kullanılabilir. Bu amaç için hazırlanan örnek uygulamada ADXL335 ivmeölçer algılayıcısı kullanılmıştır. Okunan 3 eksen ivme bilgisi Arduino IDE seri port ekranından okunmalıdır. İvmeölçer algılayıcısının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. Kullanılan algılayıcı 1.8 ile 3.6 Volt arasındaki doğru akım ile çalıştığı için algılayıcının Vcc girişi Arduino'nun 3.3 Volt pinine bağlanmalıdır. 5 Volt ile çalışan bir algılayıcı kullanıyorsanız Arduino'nun 5 Volt pinine bağlayınız.



Resim 7.137: İvmeölçer uygulaması breadboard çizimi

7.14.19. Açıölçer Uygulaması

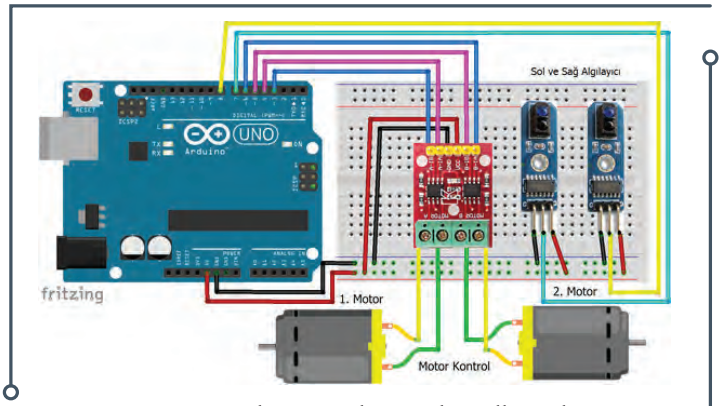
Sinyal çıkışları Arduino'nun A4 (SDA) ve A5 (SCL) numaralı analog pinlerine bağlı gyro ve ivmeölçer algılayıcısı kullanarak bir açıölçer uygulaması hazırlayınız. Bu amaçla farklı gyro algılayıcılar kullanılabilir. Bu amaç için hazırlanan örnekte MPU-6050 algılayıcı kartı kullanılmıştır. Bu üzerinde 3 eksenli bir gyro ve 3 eksenli bir açısal ivmeölçer bulunan 6 eksenli bir IMU algılayıcı kartıdır. Okunan açı bilgisi Arduino IDE seri port ekranında gösterilmelidir. Algılayıcının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. Bu kartın bazı çeşitleri 1.8 ile 3.6 Volt arasındaki doğru akım ile çalışmaktadır. Bu özellikteki kartların güç girişi Arduino'nun 3.3 Volt pinine bağlanmalıdır.



Resim 7.138: Açıölçer uygulaması breadboard çizimi

7.14.20. Çizgi İzleyen Robot Uygulaması

DC Motor kontrol uygulamasında kullandığınız motor ve motor sürücüsüne çizgi takip algılayıcılarını da ekleyerek bir çizgi takip uygulaması hazırlayınız. Bu amaçla farklı motor, motor sürücü kartı ve çizgi takip algılayıcıları kullanılabilir. Uygulama için motorlara tekerlek takarak ve bütün sistemi bir şase üzerine yerleştirerek hareketli bir robot haline dönüştürmeniz gereklidir. Bu uygulamayı gurup halinde hazırlamanız daha uygun olacaktır. Hazırlanan örnekte 6 voltla çalışan redüktörlü iki adet dc motor, L9110 motor sürücü kartı ve iki adet TCRT5000 çizgi takip algılayıcısı kullanılmıştır. Motor sürücünün A-IA girişi Arduino'nun 3 numaralı dijital pinine, sürücünün A-IB girişi Arduino'nun 4 numaralı dijital pinine, sürücünün B-IA girişi Arduino'nun 5 numaralı dijital pinine, sürücünün B-IB girişi Arduino'nun 6 numaralı dijital pinine bağlanmıştır. Sol çizgi algılayıcının sinyal pini 7, sağ çizgi algılayıcının sinyal pini Arduinonun 8 numaralı dijital pinine bağlanmıştır. Algılayıcılar ile motor sürücünün Arduino ve motorlarla bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir.



Resim 7.139: Çizgi takip uygulaması breadboard çizimi

7.15. DEĞERLENDİRME SORULARI

1. **Arduino IDE ortamında hangi programlama dili kullanılır?**
   a) C
   b) C++
   c) C #
   d) Pascal
   e) Python

2. **Arduino IDE yapısı ve temel özellikleri açısından aşağıda verilen ifadelerden hangisi doğrudur?**
   a) Arduino'ya yüklenen programlar elektriği kapatılana kadar Arduino içinde kalır. Yüklemeden sonra bağımsız olarak çalıştırılabilir
   b) Tüm komutlar virgül (,) ile biter. Fakat blok başlatan ifadelerden sonra virgül kullanılmaz
   c) Programda kullanılan bazı değişkenler ve bilgi tipleri bildirilmelidir
   d) Programın başında kullanılacak kütüphaneler aktifleştirilmeli /çağrılmalıdır
   e) Açıklamalar "\\" ve "* *\ " (birden fazla satır için) ile yazılabilir

3. **Arduino uygulamalarında kullanılacak elektronik bileşenler için aşağıdakilerden hangisinin kullanılması uygulamaların hızlı, kolay ve en önemlisi lehim yapmadan yapılmasına olanak tanıyacaktır?**
   a) Breadboard (Devre Tahtası)
   b) Delikli pertinaks
   c) Baskılı devre
   d) Konnektör
   e) Jumper

4. **Arduino IDE için aşağıda verilen ifadelerden hangisi tam olarak doğru değildir?**
   a) Kod yazım editörüdür
   b) Tümleşik bir derleyicidir
   c) Programlama dilidir
   d) Yorumlayıcı ve hata ayıklayıcıdır
   e) Tümleşik geliştirme ortamıdır

5. **Arduino IDE ortamında mikrodenetleyici ya da Arduino'nun beslemesi devam ettiği sürece tekrarlanan komutlar hangi fonksiyon içinde yer alır?**
   a) void setup()
   b) digitalWrite()
   c) serial.begin()
   d) digitalRead()
   e) void loop()

6. Arduino IDE ortamında program yüklenilip enerji verildikten veya tekrar başlatıldıktan sonra 1 defa çalışan fonksiyon aşağıdakilerden hangisidir?

a) digitalRead()
b) digitalWrite()
c) serial.begin()
d) void setup()
e) void loop()

7. Arduino UNO, USB ile bilgisayara bağlı olduğu sürece giriş veya çıkış işlemleri için hangi numaralı pinler kullanılamazlar?

a) A0 ve A1
b) 0 ve 1
c) 3 ve 5
d) A2 ve A3
e) Güç pinleri

8. Arduino UNO'nun belli bir pinine gelen sinyalle belli bir fonksiyonun ya da önceden belirlenmiş bir alt programın çalıştırılmasını sağlamak için hangi numaralı pinler kullanılır?

a) 2 ve 3
b) 4 ve 5
c) 6 ve 7
d) 8 ve 9
e) 10 ve 11

9. I^2C veri yolu kullanan bir algılayıcı Arduino UNO'nun hangi numaralı SDA ve SCL pinlerine bağlanmalıdır?

a) A0 ve A1
b) A2 ve A3
c) A4 ve A5
d) 0 ve 1
e) 2 ve 3

10. SPI veri yolu kullanan bir algılayıcı Arduino UNO'nun hangi numaralı MOSI ve MISO pinlerine bağlanmalıdır?

a) 3 ve 4
b) 5 ve 6
c) 7 ve 8
d) 9 ve 10
e) 11 ve 12

KAYNAKÇA

Adafruit (2017). Sensors. [Çevrim-içi: https://www.adafruit.com/category/35, Erişim tarihi: 03.03.2017].

Adafruit (2017). Sensors. [Çevrim-içi: https://learn.adafruit.com/category/sensors, Erişim tarihi: 03.03.2017].

Akademikport (2017). AkademikPort Arduino Başlangıç Projeleri. [Çevrim-içi: http://kitap.akademikport.com/AkademikPort-Arduino-Baslangic-Projeleri.pdf?, Erişim tarihi: 04.01.2017].

Arduino (2017). What is Arduino? [Çevrim-içi: https://www.arduino.cc/en/Guide/Introduction, Erişim tarihi: 03.01.2017].

Arduino (2017). Arduino IDE. [Çevrim-içi: https://www.arduino.cc/en/main/software, Erişim tarihi: 03.01.2017].

Arduino (2017). Language Reference. [Çevrim-içi: https://www.arduino.cc/en/Reference/HomePage, Erişim tarihi: 04.01.2017].

Arduino (2017). Arduino Libraries. [Çevrim-içi: https://github.com/arduino-libraries, Erişim tarihi: 04.01.2017].

Banzi, M., & Shiloh, M. (2014). Getting started with Arduino: the open source electronics prototyping platform. Maker Media, Inc.

Baykara, M. (2017). Mikroişlemciler ve Programlama Dersi- ARDUINO. [Çevrim-içi: http://muhammetbaykara.com/wp-content/uploads/2017/04/arduino-uygulamalar%C4%B1.pdf, Erişim tarihi: 10.01.2017].

Benson, C. (2012). Basics: What Types of Mobile Robots are There? [Çevrim-içi: http://www.robotshop.com/blog/en/what-types-of-mobile-robots-are-there-3652, Erişim tarihi: 05.01.2017].

Boxall, J. (2013). Arduino Workshop: A Hands-On introduction with 65 projects. No Starch Press.

Çobanoğlu, B. (2016). Arduino Programlama. [Çevrim-içi: http://cobanoglu.wikispaces.com/file/view/Arduino_Cobanoglu.pdf/495731668/Arduino_Cobanoglu.pdf, Erişim tarihi: 12.01.2017].

Demir, U. (2015). Arduino Programlama Kitabı. [Çevrim-içi: https://ugrdmr.wordpress.com/2016/02/01/arduino-programlama-kitabi-cikti-2/, Erişim tarihi: 12.01.2017].

EBA (2017). Arduino Temel Atölye Uygulamaları. [Çevrim-içi: http://img.eba.gov.tr/659/8f6/e5b/21b/932/024/027/881/4f3/997/77f/4e5/eac/ff2/001/6598f6e5b21b9320240278814f399777f4e5eacff2001.pdf, Erişim tarihi: 06.02.2017].

Electronicsteacher (2017). Robotics - Types of Robots. [Çevrim-içi: http://www.electronicsteacher.com/robotics/type-of-robots.php, Erişim tarihi: 09.01.2017].

Evans, B. (2011). Beginning Arduino Programming. Apress.

Firmata (2017). Firmata. [Çevrim-içi: https://github.com/firmata/arduino, Erişim tarihi: 16.01.2017].

Font, D. B., García, C. S., & de Mántaras, R. L. (2003). A Multiagent Approach to Qualitative Navigation in Robotics. Universitat Politècnica de Catalunya. [Çevrim-içi: http://eia.udg.es/~busquets/thesis/thesis_html/node34.html, Erişim tarihi: 06.01.2017].

Fritzing (2017). Fritzing. [Çevrim-içi: http://fritzing.org/home/, Erişim tarihi: 05.04.2017].

Geleceğiyazanlar (2017). Arduino Dünyası. [Çevrim-içi: https://gelecegiyazanlar.turkcell.com.tr/konu/arduino, Erişim tarihi: 18.01.2017].

Harold, T. (2011). Practical Arduino Engineering.Technology In Actıon.

Herrmann, M. (2015). IVR: A Brief Outlook to Robot Architectures. [Çevrim-içi: http://homepages.inf.ed.ac.uk/mherrman/IVRINVERTED/pdfs/architectures.pdf, Erişim tarihi: 11.01.2017].

İskefiyeli, M. (2017). Sakarya Üniversitesi Bilgisayar Mühendisliği Gömülü Sistemler Deney Föyü. [Çevrim-içi: http://content.lms.sabis.sakarya.edu.tr/Uploads/50142/35704/gomsisdeneyler01.pdf,

http://content.lms.sabis.sakarya.edu.tr/Uploads/50142/35706/gomsisdeneyler02.pdf, Erişim tarihi: 14.04.2017].

Karvinen, T., & Karvinen, K. (2011). Make: Arduino Bots and Gadgets Six Embedded Projects with Open Source Hardware and Software (Learning by Discovery). Make Books-Imprint of: O'Reilly Media.

Kelly, J. F., & Timmis, H. (2012). Arduino Adventures: Escape from Gemini Station. Apress.

Makeblock (2017). Robot Kits. [Çevrim-içi: http://www.makeblock.com/robot-kit-series-STEM, Erişim tarihi: 08.03.2017].

Makeblock (2017).mBlock. [Çevrim-içi: http://learn.makeblock.com/en/software/, Erişim tarihi: 08.03.2017].

mBlock (2017). Program Robots / Arduino. [Çevrim-içi: http://www.mblock.cc/, Erişim tarihi: 08.03.2017].

McRoberts, M. (2013). Beginning Arduino. Apress.

Monk, S. (2013). 30 Arduino projects for the evil genius. McGraw-Hill Professional.

Odendahl, R. (2015). Robotics Notes – Paradigms. [Çevrim-içi: http://www.cs.oswego.edu/~odendahl/coursework/csc338/notes/paradigms/overview.html, Erişim tarihi: 05.01.2017].

Olsson, T. (2012). Arduino wearables. Apress.

Oxer, J., & Blemings, H. (2011). Practical Arduino: cool projects for open source hardware. Apress.

Pololu (2017). Elektronics. [Çevrim-içi: https://www.pololu.com/category/6/electronics, Erişim tarihi: 13.02.2017].

Premeaux, E., & Evans, B. (2012). Arduino Projects to Save the World. Apress.

Qureshi, F., Terzopoulos, D., & Gillett, R. (2004). The cognitive controller: a hybrid, deliberative/reactive control architecture for autonomous robots. Innovations in Applied Artificial Intelligence, 1102-1111. [Çevrim-içi: https://www.researchgate.net/profile/Ross_Gillett/publication/221048199_The_Cognitive_Controller_A_Hybrid_DeliberativeReactive_Control_Architecture_for_Autonomous_Robots/links/00b7d5368dd3057ca6000000.pdf, Erişim tarihi: 05.01.2017].

Reis, L., P. (2007). Robot Software Architectures. [Çevrim-içi: http://paginas.fe.up.pt/~lpreis/ir2007_08/documents/IR0708-3-AgentArchitectures.pdf, Erişim tarihi: 05.01.2017].

Robots and Androids (2017). Robots and Androids. [Çevrim-içi: http://www.robots-and-androids.com/, Erişim tarihi: 15.02.2017].

Robotpark (2017). All Types of Robots - By Locomotion. [Çevrim-içi: http://www.robotpark.com/All-Types-Of-Robots, Erişim tarihi: 13.01.2017].

Robot Wiki (2017). Robot Types. [Çevrim-içi: http://robotics.wikia.com/wiki/Robot_types, Erişim tarihi: 12.01.2017].

Sevinç, H. (2017). Temel Elektronik Arduino Eğitimi. [Çevrim-içi: http://www.ardimed.com/upload/post/datasheet/datasheet_2_564773fb1dcd2.pdf, Erişim tarihi: 14.03.2017].

Smith, A. G. (2011). Introduction to Arduino. Reference book, September, 30, 1-172.

SparkFun (2017). Sensors. [Çevrim-içi: https://www.sparkfun.com/categories/23, Erişim tarihi: 14.03.2017].

Thomas, D. (2002). Robot Architectures. [Çevrim-içi: http://cs.brown.edu/~tld/courses/cs148/02/architectures.html, Erişim tarihi: 09.01.2017].

Thrun, S. (2001). Is Robotics Going Statistics? The Field of Probabilistic Robotics. Communications of the ACM, 45 (3), 1-8. [Çevrim-içi: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.79.8332&rep=rep1&type=pdf, Erişim tarihi: 03.01.2017].

TTKB (2016). Ortaöğretim Bilgisayar Bilimi Dersi (Kur 1, Kur 2) Öğretim Programı. [Çevrim-içi: http://ttkb.meb.gov.tr/www/ogretim-programlari/icerik/72#, Erişim tarihi: 10.10.2016].

Warren, J. D., Adams, J., & Molle, H. (2011). Arduino Robotics. Collection: Technology in Action.

Zöhra, B. (2015). Arduino ile Sensor Uygulamaları. [Çevrim-içi: https://www.academia.edu/16909759/ARDUINO_%C4%B0LE_SENSOR_UYGULAMALARI?auto=download, Erişim tarihi: 05.04.2017].

II. BÖLÜM

WEB PROGRAMLAMA

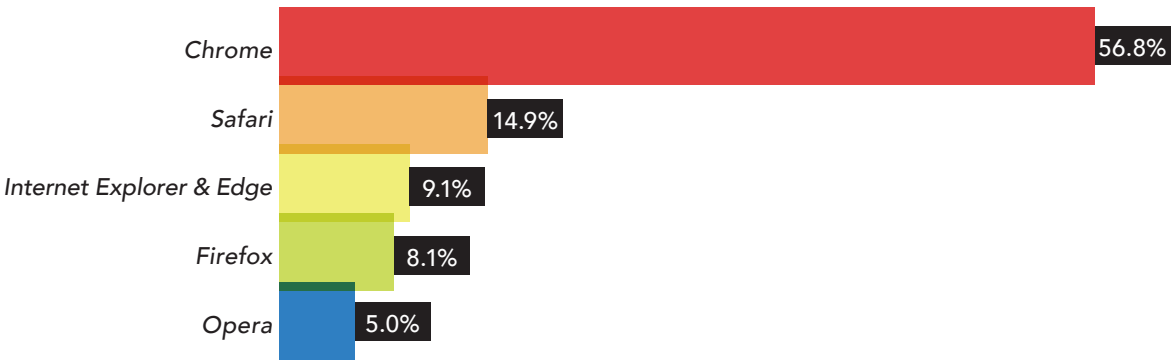
1. İNTERNET VE WEB SERVİSLERİ

1. İnternet ve Web Servisleri

Arkadaşlar biliyorum ki bilgisayarda oyun oynamayı çok seviyorsunuz. Özellikle çevrimiçi oyun oynamak daha heyecan verici sanırım. Akıllı cihazların da hayatımıza hızla girmesiyle bazılarımız için gittiğimiz her yerde oyun oynamak vazgeçilmez bir tutku oldu. Bununla birlikte bilgisayar ya da mobil cihazımızı açtığımızda İnternet'i bilgi aramak, alışveriş yapmak, arkadaşlarımızla sohbet etmek, arkadaşlarımızın paylaşımlarını beğenmek, paylaşımlara yorum yazmak ve daha birçok farklı amaçla kullanıyoruz değil mi? Peki hiç düşündünüz mü? İnternet nasıl çalışır, nasıl geliştirilir? Acaba siz de bir İnternet sitesi yapabilir misiniz? Gelin ilk olarak İnternet'in nasıl çalıştığına yanıt arayalım.

Web Tarayıcıları

İnternet'e bağlanmak için hepimiz birer web tarayıcısı (browser) kullanıyoruz. Fakat kullanmış olduğumuz web tarayıcılar farklılık gösterebiliyor. Ağustos 2017 verilerine göre dünya genelinde en çok kullanılan tarayıcılar Chrome, Safari, Internet Explorer, Firefox, Opera olarak sıralanmaktadır[1] (Bkz. Şekil 1).



Şekil 1: Web tarayıcılarının dünya genelinde yüzde olarak kullanım grafiği

Bir web tarayıcısı bizim görüntülemek istediğimiz siteyi ya da kaynağı, sitenin yüklü olduğu bilgisayardan ya da sunucudan çağırarak (istek yaparak) ekrana yansıtır. Dolayısı ile web tarayıcısının bulunduğu bilgisayar istemci, görüntülenmek istenen sitenin yüklü olduğu bilgisayar ise sunucu olarak adlandırılır.

Şimdi bir örnek üzerinde bu durumu senaryolaştıralım:

- Chrome web tarayıcısını açtınız.
- Tarayıcınızın adres satırına http://www.google.com yazdınız ve site açıldı.
- Bu durumda sizin bilgisayarınız istemci, google web sitesi kaynak oldu.

Peki sunucuya nasıl eriştik ve sunucu nerede?

- Sunucuya http://www.google.com.tr ismi ile erişim sağladık.
- Web tarayıcımıza bu ismi girdiğimizde ismin bağlı olduğu adresteki bilgisayarlar ise istek yaptığımız sunucularıdır. Bu sunucuların İnternet ortamında yerini gösteren adresleri IP adresleridir. Google'nin Türkiye web sitesinin ipsi 216 58 209 195'tir. Akılda daha kalıcı olması için "alan

1 https://www.w3counter.com/globalstats.php?year=2017&month=8

adı sunucuları" http://www.google.com.tr adresi ile 216 58 209 195 ipsini eşleştirilir. Dolayısı ile daha kolay bir erişim sağlamış oluruz.

Gündelik yaşamımızda web tarayıcılarını hangi amaç için kullanırsak kullanalım web tarayıcıları aslında arka planda aynı işlemleri yapar. Yani bizlerin kaynak olarak isimlendirdiğimiz resim, metin, animasyonları görüntüleme ve bu kaynaklar üzerinde ekleme, silme, düzenleme yapmamıza olanak sağlar. Bunun yanı sıra site tarafından izin verilmişse görüntülediğimiz kaynakları kendi bilgisayarımıza indirmemiz için aracı görevi üstlenir. Son olarak siz İnternet'te gezinirken ziyaret etmiş olduğunuz sitelerin ve bu siteleri gezerken girmiş olduğunuz verilerin bilgilerini sonradan size hatırlatmak üzere tutar.

Web Teknolojileri

World Wide Web (www), Tim Berners-Lee tarafından 1989 yılında icat edilmiştir. O günden bugüne İnternet ve ilgili teknolojilerde pek çok gelişme ve değişim meydana gelmiştir. Örneğin geçmişte tarayıcılarda sadece metin görüntüleyebiliyor ve başka herhangi bir işlem gerçekleştiremiyorduk. İşte web teknolojisinin bu ilk versiyonu **Web 1.0** olarak adlandırılmaktadır. Daha sonra web teknolojileri; tarayıcıda görüntülediğimiz site üzerinde yeni kaynaklar ekleme, kaynaklar üzerinde değişiklik yapma gibi olanaklar sağlamaya başladı. Bu İnternet teknolojisi ise **Web 2.0** olarak adlandırılmaktadır. Web 2.0 ile çalışan birçok örnek verebileceğinizi düşünüyorum. Biraz ipucu ister misiniz? "Paylaşım" size ne ifade ediyor? Facebook, Instagram, Youtube, Vikipedia … Bunlara benzer daha birçok web 2.0 ile hazırlanmış Blog, Sosyal Ağ, Viki, Elektronik Portfolyo gibi site örnekleri verebiliriz. Hadi bu türden bir site ortamında neler yapabiliyoruz sıralamaya çalışalım (Bkz. Şekil 2).

Şekil 2: Web 2.0 ile hazırlanmış bir sitede neler yapabiliriz?

Sanki bir şeyi atladık gibi değil mi arkadaşlar? İnternet'te bizim yapabildiklerimizin yanında bir de İnternet'in bizim için bizden habersiz olarak yapabildikleri var. Örneğin Google'da arama yaparken bir kelime yazıyoruz Google tamamlıyor. Aynı şekilde bir alışveriş sitesinden futbol ayakkabısı ya da elbise beğeniyorsunuz. Bir sonraki web tarayıcınızı açtığınızda benzer ürünlerin ekranınızda reklamının ya-

pıldığını görebiliyorsunuz. İşte bu web teknolojisi de akıllı web ya da anlamsal ağ yani Web 3.0 olarak adlandırılmaktadır. Gelin şimdi web 1.0, web 2.0 ve web 3.0 teknolojilerini karşılaştıralım (Bkz. Tablo 1).

Web 1.0	Web 2.0	Web 3.0
Web sayfası sadece okunabilirdir.	Web sayfası hem okunabilir hem de yazılabilirdir.	Web sayfası yazılabilir okunabilir ve programlanabilirdir.
Etkileşim tek yönlüdür. Yani bilgisayar gösterir, kullanıcı okur.	Etkileşim çok yönlüdür. Yani web sayfasındaki tüm kullanıcılar birbirleri ile etkileşim kurabilir.	Web sayfasındaki kullanıcılar ile bilgisayarlar ve bilgisayarlar arasında etkileşim vardır.
Kullanıcı ara yüzleri daha sade ve genellikle metin tabanlı tasarım vardır.	Zenginleştirilmiş, etkileşimli ve dokunmatik arayüz tasarımı vardır.	Yaşamla bütünleşik duyularla şekillenebilen arayüzler ve kullanıcıya göre değişen arayüz tasarımı vardır.

Tablo 1: Web 1.0, Web 2.0 ve Web.3.0 Karşılaştırma

2. İŞARETLEME DİLİNE GİRİŞ (HTML)

2. İşaretleme Diline Giriş (HTML)

İnternet'in nasıl çalıştığı ve web teknolojilerinin neler olduğu hakkında bilgi sahibi olduk. Bu ünite ile örnek bir web sayfasının temellerini oluşturmaya başlıyoruz.

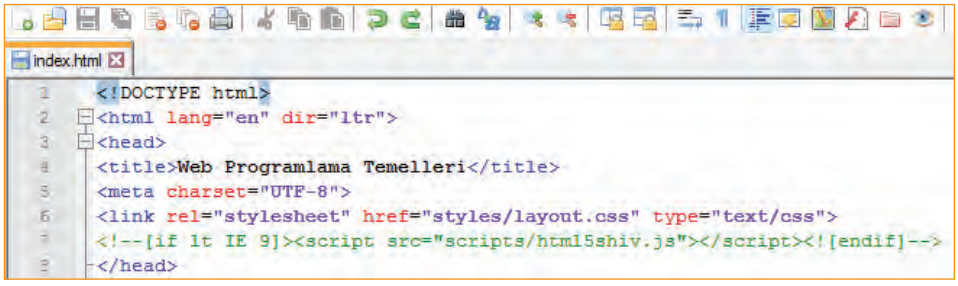
Ön Hazırlık

Kodlama yaparken yararlanabileceğimiz birçok editör var. Biz bu bölümde iki editörü tanıtacağız. Bu editörler dışında öğretmeninizin önerdiği editörleri de kullanabilirsiniz.

I. https://www.w3schools.com/ Adresindeki Web Sitesi: Öncelikle siteye bağlandıktan sonra HTML kodların yazılabileceği ekrana bağlanmak için yeşil butona tıklıyoruz. Ya da QR Kodunu mobil aygıtınızdan okutabilirsiniz (Bkz. Şekil 4).

Şekil 3: www.w3schools.com editörü

II. NotPad++ Editörü: Çeşitli dilleri destekleyen ücretsiz kaynak kodu düzenleyicisidir. https://notepad-plus-plus.org/ adresinden kişisel bilgisayarınıza indirip kolayca kurulum yapabilirsiniz.

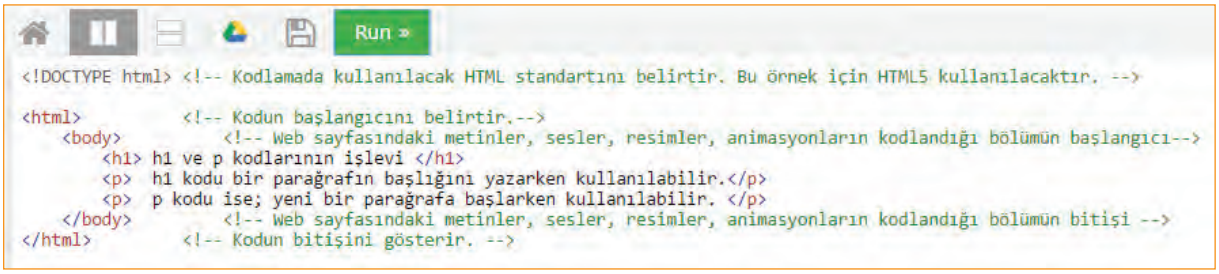


Şekil 4: Notpad ++ editörünün genel görünümü

Editörler hakkında bilgi sahibi olduk. Şimdi çalışma dizinimizi hazırlayalım. Öncelikle, masaüstüne ismimize ait bir dizin oluşturalım. Yapacağımız bütün uygulamaları dosya uzantısı .html olacak şekilde (Ör: "dosyaismi".html) bu dizin içerisine kaydedeceğiz.

HTML

HTML, Hiper Metin İşaretleme ya da Biçimlendirme Dili (Hyper Text Markup Language) olarak Türkçe'ye çevrilmiştir. Web sitesi oluşturmak için kullanılan standart dildir. İlerleyen bölümlerde başka dillerle de tanışacağız. Şimdi, HTML kodlarının nasıl çalıştığını kavramaya çalışalım.



Resimdeki kodu istediğiniz bir editörü kullanarak yazınız.

Dikkat!

- Kodların yazılırken küçük büyük işaretleri <> arasına <html>, <p>, <body> gibi kod yazıldığını fark ettiniz mi?
- Herhangi bir koda başlarken < > işaretlerini, kodu bitirirken </ > işaretlerini kullandığımızı fark ettiniz mi?

Kodları çalıştırınız ve ekran çıktısını inceleyiniz.

Dikkat!

- Sadece siyah renkteki yazıların çıktı olarak sunulduğunu fark ettiniz mi?
- <!-- --> işaretleri arasındaki yazılı metinlerin sayfada görüntülenmediğini fark ettiniz mi?

Resim Ekleme

Şimdi birlikte bir resmin web sayfasına nasıl eklendiğine bakalım. Web sayfamıza <img> kodunu kullanarak resim ekleyebiliriz. Öncelikle İnternet'ten kendi ilgimize göre bir resim bulalım. Konumuz kodlama olduğu için Google'da görsellerden "web programlama" anahtar kelimesi ile arama yapıyoruz. İlgimizi çeken herhangi bir resme sağ tıklayarak resim adresini kopyalıyoruz.

```
<!DOCTYPE html>
<html>
    <body>
        <h2>Resim Ekleme</h2>
        <img src="http://www.lincolnshirewebdesign.net/wp-content/uploads/2012/05/Code-2.jpg"
        alt="Web kodlama" style="width:200px;height:150px;">
        <!--
            img kodu açıklamalar
            ----------------------------------------------------
            src: webden kopyaladığımız resmin adresi
            alt: imleç resmin üzerine geldiğinde görüntülenecek yazı
            style: resmin boyutları
        -->
        <p> Geleceğin kodlayıcıları olarak hepimiz çok çalışmak zorundayız. </p>
    </body>
</html>
```

Dikkat!

- Yeşil renkteki yazıların web sayfasında görüntülenmediğini fark ettiniz mi?
- < > işaretleri arasına birden fazla kod yazılabildiğini fark ettiniz mi?

```
<img src="http://www.lincolnshirewebdesign.net/wp-content/uploads/2012/05/Code-2.jpg"
alt="Web kodlama" style="width:200px;height:150px;">
```

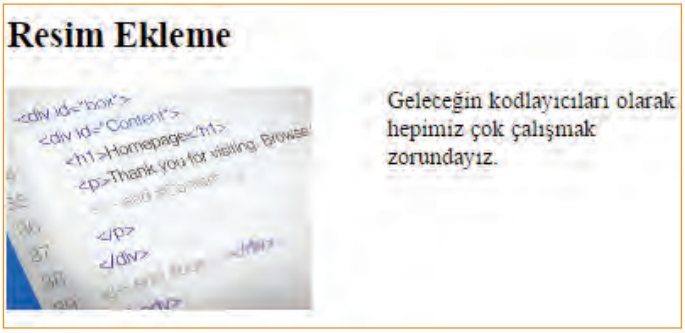
Kodu çalıştırdığınızda resim ve yazının alt alta olduğunu göreceksiniz. Peki bunları yan yana nasıl getirebiliriz? Bunun birçok yolu var. Fakat başlangıç aşamasında olduğumuz için sadece resmi sağa ya da sola yaslamamızı sağlayan float komutunu göreceğiz.

- Resim eklerken resmin genişliği (width) ve yüksekliğini (height) belirtmek için style kodunu kullanmıştık.
- Resmi sağa ya da sola yaslamak için de yine bu kodun içerisine float:left; kodunu ekliyoruz ve kodu test ediyoruz.

Dikkat!

- style kodu içerisinde birden fazla kod yazarken kodlar arasına ";" işareti eklendiğini fark ettiniz mi?
- style="witdh:200px;height:150px;float:left;"

- Eğer kodu çalıştırdıysanız resim ile yazının bitişik durduğunu fark etmiş olmalısınız. Resim ve yazı arasında boşluk bırakmak istediğimizde ise margin:50px; komutunu style kodu içerisine ekliyoruz.
- Fakat bu kez resmin üstünde-sağında-solunda ve altında boşluklar oluştu. Biz sadece yazı ile boşluk olsun istiyorduk. Bunun için margin kodunda bir değişiklik yapmamız gerekiyor.
- margin:50px kodunu margin-right:50px olarak değiştirdiğimizde istediğimiz sonuca ulaşacağız.



Kodlayıcı Görevi

- margin-top, margin-left, margin-right, margin-bottom komutlarının ne işe yaradığını keşfetmeye çalışınız.

Başka Bir Sayfaya Bağlantı (Link) Verme

Öncelikle bu işlem için gerekli olan kodumuzu tanıyalım. HTML ile sayfalar arası bağlantı (link) vermemizi <a> kodu sağlar.

Bu iki kod arasına, üzerine tıkladığımızda bizi farklı bir sayfaya yönlendirecek metni ya da resmi ekliyoruz.

```
<a>Kodlama Öğreniyorum Sayfası</a>
```

Daha sonra href ile yönlendirilecek sayfayı yazıyoruz.

```
<a href="https://www.w3schools.com">Kodlama Öğreniyorum Sayfası</a>
```

Yönlendirilen sayfanın farklı bir sekmede açılmasını istiyorsak target="_blank" kodunu kullanıyoruz.

```
<a href="https://www.w3schools.com" target="_blank">Kodlama Öğreniyorum Sayfası</a>
```

Kodlayıcı Görevi 1

- Şimdi siz de <a target="_blank|_self|_parent|_top|framename"> komutlarının ne işe yarayabileceğini keşfediniz.

Kodlayıcı Görevi 2

- Herhangi bir resmin üzerine tıklandığında okulunuzun web sitesine yönlendirme sağlayan kodu yazınız.

HTML5

Tim Beurners Lee Web'i icat ettiğinden bu güne kadar hem İnternet kullanıcılarının hem de kodlayıcıların ihtiyaçları değişmiştir. Buna yönelik olarak Web'i programlarken kullanmış olduğumuz diller de kendilerini güncellemektedir. Şuan en güncel html sürümü HTML5'tir. Eski HTML sürümlerinde siteyi tasarlamış olduğunuz kodların bazıları farklı web tarayıcıları tarafından desteklenmiyordu. HTML5, tüm modern tarayıcılarda desteklenir. Aynı zamanda tek bir doğru kodlama şekli yoktur. Aşağıdaki kodların hepsi en son güncel tarayıcılar tarafından çalıştırılabilir. Böylece kodlayıcının işi daha kolaylaşmış olur.

```
<input type="text" value="John" disabled>
<input type="text" value=John>
<input type="text" value="John Doe">
<input type="text" value='John Doe'>
```

- HTML5'te, var olan etiketler (kodlar) dışında etiketler yaratabiliriz (Örneğin isminizi bir etiket olarak kodlayabilirsiniz.). Bu özellik, kodlayıcılara kendilerine özgü kodlama yöntemleri geliştirmelerine olanak sağlar.

```
<!DOCTYPE html>
<html>
<head>
  <script>document.createElement("yeninesne")</script> <!--ie9 ve öncesi için gerekli-->

  <style>
  yeninesne {
      display: block;
      background-color: #dddddd;
      padding: 50px;
      font-size: 30px;
  }
  </style>

</head>
<body>

<h1>Bilinmeyen Nesnenin Öğretilmesi?</h1>
<yeninesne>Daha önce tagı olmayan nesnemi yarattım</yeninesne>

</body>
</html>
```

- HTML5, bir web sayfasının farklı bölümlerini tanımlamak için yeni anlamsal etiketler sunar. Anlamsal etiket; <video>, <form> gibi içeriği hakkında fikir sahibi olduğumuz elementlerdir. Yani ne tür bir kaynağı etiketlediği açıktır. Anlamsal olmayan elementler ise , <p> gibi içeriği hakkında fikir sahibi olamadığımız elementlerdir. Yani içerisine birçok farklı türden kaynak alabilir. HTML5 ile <header>, <nav>, <section>, <article>, <footer>, <figure>, <mark>, <details>, <progress>, <dialog> gibi yeni anlamsal elementler sunulmaktadır. Şimdi bu elementleri tanıyalım.

<header> </header>	<section> </section>	<article> </artice>	<footer> </footer>
Bir belge veya bölüm için bir başlık belirtir.	Genellikle başlıklı tematik bir içerik gruplandırmasıdır. Bir anasayfa giriş, içerik ve iletişim bilgileri gibi bölümlere ayrılabilir.	Forum yazısı, blog yazısı, gazete makalesi gibi. Bağımsız, kendine yeten içeriği belirtir.	Bir belge veya bölüm için altbilgiyi belirtir.
<footer> </footer>	**<nav> </nav>**	**<aside> </aside>**	**<figure> </figure> <figcaption></figcaption>**
Bir belge veya bölüm için altbilgiyi belirtir.	Gezinim bağlantılarını belirtir.	İçeriği (yan bar gibi) bir kenara koyar.	Bir resim ve resim yazısı, bir <figure> ögesinde gruplandırılabilir.
<mark></mark>	**<details> </details>**	**<progress> </progress>**	**<dialog> </dialog>**
İşaretli/vurgulanmış metinleri tanımlar.	Kullanıcının görüntüleyebileceği veya gizleyebileceği ek ayrıntıları tanımlar.	Görevin ilerlemesini temsil eder.	Bir iletişim kutusu veya pencere tanımlar.

1. UYGULAMA

Notpad++ editöründe aşağıdaki basamakları takip ederek kodlama yapalım.

1) İlk olarak genel sayfa yapısını oluşturuyoruz.

```
1   <!DOCTYPE html> <!-- Sayfanın HTML5 standartında kodlanacağını belirtir  -->
2   <html>          <!-- HTML etiketini açıyoruz.  -->
3   <body>          <!-- Sayfanın içeriğini bu etiket içerisine kodluyoruz.  -->
4
5   <section> <!-- HTML5 elementlerini tanıtmak için bir bölüm açıyoruz. -->
6
7       <header> <!-- Bölümümüzün başlık etiketini açıyoruz.  -->
14
15      <article> <!-- Bölüm içerisindeki birinci makale içeriği   -->
21
22      <article> <!-- Bölüm içerisindeki ikinci makale içeriği   -->
28  </section> <!-- HTML5 elementlerini tanıttığımız bölümü kapatıyoruz.-->
29
30  <footer> <!-- Sayfanın altında görünecek altbilgi  -->
31          <header>
34          <nav>
40  </footer>
41
42  </body> <!-- body etiketini kapatıyoruz.  -->
43  </html> <!-- HTML etiketini açıyoruz.  -->
```

Yukardaki kodu incelediğimizde sayfamız bir bölüm (<section>) ve altbilgiden (<footer>) oluşuyor. Bölüm içerisinde bir başlık etiketi (<header>), iki makale içeriği etiketi (<article>) var. Altbilgi içerisinde ise yine bir başlık ve gezinim etiketi (<nav>) var.

2) Şimdi bölüm içeriğini kodluyoruz.

```
<section> <!-- HTML5 elementlerini tanıtmak için bir bölüm açıyoruz. -->

    <header> <!-- Bölümümüzün başlık etiketini açıyoruz.  -->
        <figure> <!-- Bölümümüzün başlığı içerisinde bir resim elementi görünsün istiyoruz. -->
            <img src="https://www.w3.org/html/logo/downloads/HTML5_Logo_512.png"
            alt="HTML5" width="150" height="150">
            <figcaption>Fig.1 - HTML5</figcaption> <!-- Resmin açıklaması  -->
        </figure> <!-- Resim etiketini kapatıyoruz.  -->
    </header> <!-- Bölümümüzün başlık etiketini kapatıyoruz.  -->

    <article> <!-- Bölüm içerisindeki birinci makale içeriği   -->
        <header> <!-- Birinci makale başlığı   -->
            <h1>Anlamsal Elementler</h1> <!-- h1,h2,h3 Metnin büyüklüğünü değiştirir. -->
        </header> <!-- Birinci makale başlığını kapatıyoruz.   -->
        <p>form, table, article, audio, video vb.</p> <!-- p bir parağrafa başlarken kullanılır. -->
    </article> <!-- Birinci makale içeriğini kapatıyoruz   -->

    <article> <!-- Bölüm içerisindeki ikinci makale içeriği   -->
        <header> <!-- İkinci makale başlığı   -->
            <h2>Anlamsal Olmayan Elementler</h2> <!-- h1,h2,h3 Metnin büyüklüğünü değiştirir. -->
        </header> <!-- İkinci makale başlığını kapatıyoruz.   -->
        <p> div, p, span vb. </p> <!-- p bir parağrafa başlarken kullanılır. -->
    </article> <!-- Birinci makale içeriğini kapatıyoruz   -->

</section> <!-- HTML5 elementlerini tanıttığımız bölümü kapatıyoruz.-->
```

Bölümün başlık etiketi içerisine bir resim eklemek için ilk olarak <figure> etiketini açtık. Daha sonra, öğrenmiş olduğumuz <img> etiketini kullanarak bir resim ekledik. Resmin başlığı ya da açıklaması için <figcaption> etiketini kullandık. Son olarak bölümün başlık etiketini kapattık.

Bölüm içerisine bir içerik, açıklama metni, bilgi eklemek istediğimizde <article> etiketini kullandık. Daha sonra, yine <header> etiketi içerisine <h1> etiketini kullanarak içeriğin ya da makalenin başlığını ekledik. Son olarak makale metnini <p> etiketi içerisine ekledik ve <article> etiketini kapattık.

Kodlayıcı Görevi

- <h1> <h2> <h3> <p> etiketlerinin ne işe yaradıklarını test ediniz.
- Kodunuzu bu etiketler olmadan tekrar çalıştırınız. Ne tür farklılıklar olduğunu gözlemleyiniz.

3) Şimdi de sayfanın alt bilgisini kodluyoruz.

```
<footer> <!-- Sayfanın altında görünecek altbilgi  -->
        <header>
            <h2>Footer Bağlantıları</h2>
        </header>
        <nav> <!-- Sayfanın altında görünecek bağlantılar (linkler)  -->
            <a href="/html/">HTML</a> |
            <a href="/css/">CSS</a> |
            <a href="/js/">JavaScript</a> |
            <a href="/jquery/">jQuery</a>
        </nav> <!-- Gezinim etiketini kapatıyoruz.  -->
</footer> <!-- Altbilgiyi kapatıyoruz-->
```

Altbilgi (<footer>)etiketi içerisinde başlık ve gezinim bağlantıları yer alıyor. Gezinim bağlantılarını <nav> etiketi içerisine yazdık. Bağlantıları daha önce öğrenmiş olduğumuz <a> etiketi ile verdik. Şimdi sıra sizde kodlamanız sonucunda aşağıdaki gibi bir çıktı elde etmiş olmalısınız.

2. UYGULAMA

```
<!DOCTYPE html>
<html>
<body>

<!-- <mark> ile etiketlenen yazı sarı ile vurgulanır. -->
<p>Bu metinde bir vurgulanmış <mark>yazı</mark> var.</p>

<br/>

<!-- <details> ile açılır kapanır içerikler oluşturabiliriz. -->
<details>
    <summary> <!-- Üzerine tıklandığında açılır metin -->
    Denizer YILDIRIM
    </summary>
    <p>Ankara Üniversitesi, Uzaktan Eğitim Merkezi</p>
    <p>Öğretim Tasarımı & Web Programlama</p>
</details>

<br/><br/>

<!-- Herhangi bir görevin ilerlemesini gösteren etiket
    value: görevin hangi aşamada olduğunu belirten değer
    max: görevin tamamlanma aşamasını gösteren değer
-->
<progress value="22" max="100"></progress>

<br/><br/>

<!--Kullanıcıya bildirim vermek için kullanılan etiket -->
<dialog open> Dialog etiketi ile kullanıcıya uyarı verebiliyoruz. </dialog>

</body>
</html>
```

Kodlayıcı Görevi

- <dialog open> etiketi içerisindeki open kodunun ne işe yaradığını keşfediniz.

ÇOKLU ORTAM İÇERİK EKLEME

Sitenize HTML5 ile eklemiş olduğunuz mp4 video ve mp3 ses biçimleri şu anki mevcut tarayıcılar tarafından çalıştırabilir. Kodlamaya geçmeden önce İnternet'ten sitenize eklemek istediğiniz bir video ve sesi kendi isminizle oluşturmuş olduğunuz dizine indiriniz.

1. UYGULAMA

```
<!DOCTYPE html>
<html>
<body>

<header> <p>Video Ekleme</p> </header>
  <video id="video1" width="420" controls>
  <!-- Video etiketini açıyoruz -->
    <source src="mov_bbb.mp4" type="video/mp4">
    <!-- Video kaynağını ve türünü belirtiyoruz. Siz indirmiş olduğunuz videonun ismini yazınız.-->
    Tarayıcınız mp4 videoyu desteklemiyor.
    <!-- Eğer tarayıcı eski ise videonun (mp4) çalıştırılamayacağı uyarısı veriyoruz. -->
  </video>
  <!-- Video etiketini kapatıyoruz -->

  <br/><br/>
  <!-- Video ve ses arasına iki satır boşluk bırakıyoruz. -->

 <header> <p>Ses Ekleme</p> </header>
  <audio controls>
  <!-- Ses etiketini açıyoruz -->
    <source src="horse.mp3" type="audio/mpeg">
    <!-- Ses kaynağını ve türünü belirtiyoruz. Siz indirmiş olduğunuz ses dosyasının ismini yazınız.-->
    Tarayıcınız mp3 sesi desteklemiyor.
    <!-- Eğer tarayıcı eski ise sesin (mp3) çalıştırılamayacağı uyarısı veriyoruz. -->
  </audio>
  <!-- Ses etiketini kapatıyoruz. -->
```

Kodlayıcı Görevi

- Video ya da ses etiketleri içerisinde çeşitli kodlar yazılabilir. Bu kodların işlevleri aşağıdaki tabloda verilmiştir. Bu kodları test edelim.

Etiket	Özellik	Değer	Açıklama
Video, Audio	autoplay	autoplay	Sayfa yüklendiğinde oynatılmaya başlayacağını belirtir.
Video, Audio	controls	controls	Kontrol butonlarının görüntüleneceğini belirtir.
Video	height	pixels	Yüksekliği ayarlar.
Video, Audio	loop	loop	Her defasında tekrar başlayacağını belirtir.
Video, Audio	muted	muted	Videonun sesinin kapatılmasını belirtir.
Video	poster	URL	Kullanıcı oynat düğmesine basana kadar gösterilecek bir resmi belirtir.
Video, Audio	src	URL	Video dosyasının URL'sini belirtir.
Video	width	pixels	Video oynatıcının genişliğini ayarlar.

<source> kodu içerisine video ya da sesin nerede olduğunu belirtirken dikkat edilmesi gereken önemli bir nokta var. Aşağıdaki kod, kaynağın olduğu yeri belirtir.

```
src="mov_bbb.mp4"
```

Bu durumda kodlamanın yapıldığı dosyanın kayıt edildiği dizin aynı olmalıdır. Dizinler farklı ise aşağıdaki şekilde kodlama yapılmalıdır.

```
src="dizinismi/mov_bbb.mp4"
```

Peki, youtube gibi bir sosyal ağ sitesinden sitemize video eklemek istendiğinde ne yapılabilir? İşte bu durumda HTML5'teki <video> etiketi yerine başka kodlara ihtiyaç duyulur. Çünkü sosyal video paylaşım sitelerinin çalışma mantığı mp4 video biçimlerinden farklıdır. Şimdi ilk olarak herhangi bir sosyal video paylaşım sitesinden ilgi çekici bir video bulunuz ve daha sonra videonun altındaki paylaş (share) butonuna basarak karşınıza çıkan linki kopyalayınız. Bu link videonun kaynağı olacaktır. Şimdi üç farklı yöntemle bu videoyu site sayfasına eklenecektir.

2. UYGULAMA

```
1   <!DOCTYPE html>
2   <html>
3   <body>
4
5   <!--Yöntem 1-->
6   <iframe width="420" height="345" src="https://www.youtube.com/embed/K6S4O-VtZBI">
7   </iframe>
8
9   <!--Yöntem 2-->
10  <object width="420" height="315" data="https://www.youtube.com/embed/K6S4O-VtZBI">
11  </object>
12
13  <!--Yöntem 3-->
14  <embed width="420" height="315" src="https://www.youtube.com/embed/K6S4O-VtZBI">
15
16
17  </body>
18  </html>
```

Bu uygulama ile youtube'daki bir videoyu üç farklı yöntemle sayfaya eklenmiştir. Peki HTML5 ile <video> içerisinde controls, loop kodları eklenebiliyordu. Youtube'dan video eklenen üç yöntemde bunu nasıl yapılabilir?

```
src="https://www.youtube.com/embed/K6S4O-VtZBI?autoplay=0&loop=1&controls=1">
```

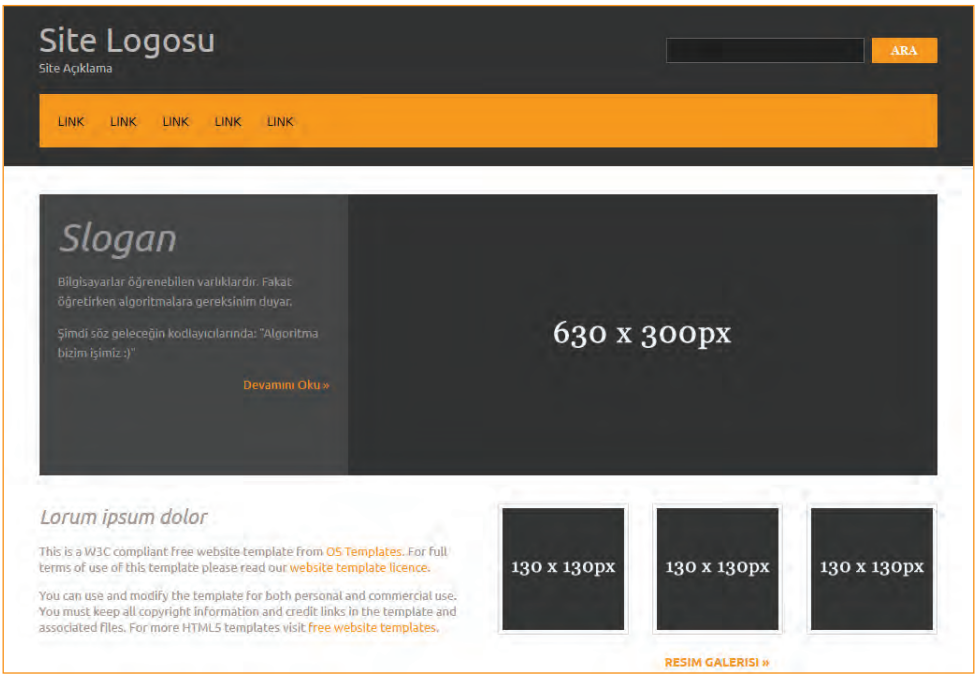
Autoplay=0 : Sayfa yüklendiğinde videoyu otomatik başlatma.

Loop=1 : Video bittiğinde tekrar başlat.

Controls=1 : Video kontrol butonlarını görünür yap.

PROJE ŞABLONUNUN OLUŞTURULMASI

Öğrenilenleri tekrarlamak ve yeni kodlar öğrenmek amacıyla örnek bir site şablonu üzerinde çalışılacaktır. Şablonun ana sayfası Şekil 5'teki gibidir.



Şekil 5: Örnek bir anasayfa görünümü

İlk olarak şimdiye kadar öğrenilenlerden hareketle, yazılabilecek kodlar, Resim 5'teki gibi bir görüntü elde etmek için yeterli olmayacaktır. Bunun için HTML5 kodları ile sitenin ana çerçeveleri oluşturulmaya çalışılacaktır. Daha sonraki ünitede ise CSS kodları kavrandığında site resimdeki gibi bir görüntüye kavuşturulabilecektir.

Site Başlığı Ve Dil Kodlaması

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4       <title>Web Programlama Temelleri</title>
5       <!-- Tarayıcı sekmesinde web sayfasının isminin görüntülenmesini sağlar-->
6       <meta charset="UTF-8">
7       <!-- Web sayfasının içerisindeki Türkçe karakterlerin görüntülenmesini sağlar -->
8       </head>
9
10  <body>
11      <!-- banner  -->
12      <div>
40      <!-- İçerik -->
41      <div>
78      <!-- Footer -->
79      <div>
96  </body>
97
98  </html>
```

Yeni bir kodla karşılaştık. <head> etiketi arasına yazılan kodlar, sayfanın genel ayarlarının yapıldığı bölümdür. Bu etiket içerisinde,

- <title> ile tarayıcı sekmesinde sayfanın başlığını görüntüleyebiliriz.
- <meta> ile hangi dil içeriğinin kullanılacağını belirtebiliriz (UTF-8 Türkçe karakterlerin de desteklenmesini sağlar.).
- <style> ve <script> ile farklı dosyaları sayfaya bağlayabiliriz. Bu konuya ilerleyen ünitelerde değinilecektir.

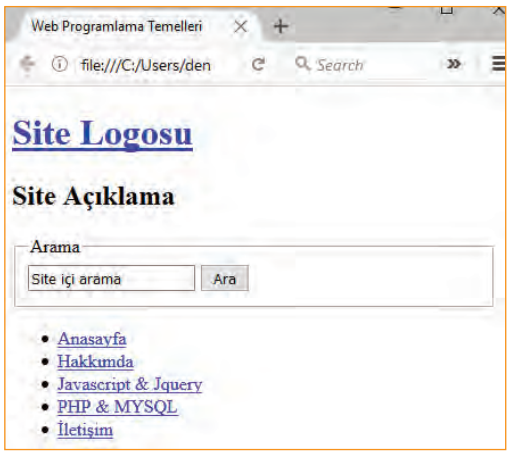
Site Banner'ı ve Gezinim Linklerinin Oluşturulması

```
<!-- banner  -->
<div>
  <header>

    <div id="hgroup">
      <h1><a href="#">Site Logosu</a></h1>
      <h2>Site Açıklama</h2>
    </div>

    <!-- Site içi arama için bir metin girişi-->
    <form action="#" method="post">
      <fieldset>
        <legend>Arama</legend>
        <input type="text" value="Site içi arama">
        <input type="submit" id="sf_submit" value="Ara">
      </fieldset>
    </form>

    <nav>
      <ul>
        <li><a href="#">Anasayfa</a></li>
        <!-- Şuan tasarlamaya çalıştığımız sitenin anasayfa bağlantısı -->
        <li><a href="#">Hakkımda</a></li>
        <!-- Siteyi kodlayanla ilgili kişisel bilgilerin yer alacağı hakkımızda sayfası -->
        <li><a href="#">Javascript & Jquery</a></li>
        <!-- İlerleyen bölümlerde Javascript ya da Jquery ile yapacağımız kodlamaların yer alacağı site sayfası  -->
        <li><a href="#">PHP & MYSQL</a></li>
        <!-- İlerleyen bölümlerde PHP & MYSQL ile yapacağımız kodlamaların yer alacağı site sayfası  -->
        <li><a href="#">İletişim</a></li>
        <!-- İletişim sayfası -->
      </ul>
    </nav>

  </header>
</div>
```

Daha önceden div elementinin anlamsal olmayan yani içeriği her türden element alabilen bir element olduğu bilinmektedir. Burada **id = "hgroup"** olan bir div elementi oluşturulmuştur. Bu bölüm ilerleyen aşamalarda sitenin logo ve açıklamasının bulunduğu yer olacak.

Yeni Kodlar

- **Form etiketi:** Kullanıcıdan veri girişi almamızı sağlayan, tüm elementleri bir arada toplayan çerçevedir.
- **Fieldset:** Birbiri ile ilişkili elementleri bir arada toplayan bir çerçevedir.
- **ul ve li:** Madde işaretli metinler oluşturmamızı sağlayan elementlerdir. Birlikte kullanılan; madde işareti konulacak metin gruplarını, <li> ise her bir maddeyi içerir.



Kodlayıcı Görevi

- Kullanıcının veri girişi yapmasına olanak sağlayan kaç tür <input> nesnesi vardır?
- İpucu: <input type=”?”>
- Madde imlerinin görünümünü nasıl değiştirebiliriz?
- İpucu: <ul type=”?”>

Sitenin Slayt Resmi ve Ana İçerik Bölümünün Oluşturulması

a) Sitenin Slayt Resminin Oluşturulması

```
47      <!-- İçerik -->
48      <div>
49        <!-- Sitenin Slayt Gösterisi ve Ana İçeriğini id' si container olan bir div elementi içerisinde topluyoruz. -->
50        <div id="container">
51
52          <!-- Sitenin slayt gösterisini id' si slider olan div elementi içerisinde topluyoruz.-->
53          <section id="slider">
63
64          <!-- Sitenin Ana İçeriğini id' si intro olan bir div elementi içerisinde topluyoruz. -->
65          <div id="intro">
90
91        </div>
92      </div>
```

```
<!-- Sitenin slayt gösterisini id' si slider olan div elementi içerisinde topluyoruz.-->
<section id="slider">
  <figure><img src="images/demo/630x300.gif" alt=""> <!-- genişliği:630px, yüksekliği:300px bir resim ekliyoruz. -->
    <figcaption>
      <h2>Slogan</h2>
      <p> Bilgisayarlar öğrenebilen varlıklardır. Fakat öğretirken algoritmalara gereksinim duyar. </p>
      <p> Şimdi söz geleceğin kodlayıcılarında: "Algoritma bizim işimiz :)"</p>
      <footer><a href="#">Devamını Oku &raquo;</a></footer>
    </figcaption>
  </figure>
</section>
```

b) Sitenin Ana İçerik Bölümünün Oluşturulması

```
47      <!-- İçerik -->
48      <div>
49        <!-- Sitenin Slayt Gösterisi ve Ana İçeriğini id' si container olan bir div elementi içerisinde topluyoruz. -->
50        <div id="container">
51
52          <!-- Sitenin slayt gösterisini id' si slider olan div elementi içerisinde topluyoruz.-->
53          <section id="slider">
63
64          <!-- Sitenin Ana İçeriğini id' si intro olan bir div elementi içerisinde topluyoruz. -->
65          <div id="intro">
90
91        </div>
92      </div>
```

```
<!-- Sitenin Ana İçeriğini id' si intro olan bir div elementi içerisinde topluyoruz. -->
<div id="intro">
  <section >
    <!-- article 1 -->
    <article>
      <h2> Biz Kodlayıcılar, </h2>
      <p> Web programlama bölümünü bitirdiğimizde örnek bir site tasarlamış olacağız. <br/>
      Siteyi tasarlarken hem kodların ne işe yaradığını öğrenecek hem de kodları örnek tasarımımız için uygulayacağız. <br/>
      Programlama yapmaktan keyif aldıkça kendi kendinize daha ilgi çekici, beğeni oranı yüksek siteler tasarlayabileceksiniz. </p>
    </article>
    <!-- article 2 -->
    <article>
      <figure>
        <ul>
          <li><a href="a1.html"><img src="images/demo/130x130.gif" width="130" height="130" alt=""></a></li>
          <!-- Madde işaretli, a1.html sayfasına bağlantılı genişliği:130px, yüksekliği:130px bir resim ekliyoruz.  -->
          <li><a href="a2.html"><img src="images/demo/130x130.gif" width="130" height="130" alt=""></a></li>
          <!-- Madde işaretli, a2.html sayfasına bağlantılı genişliği:130px, yüksekliği:130px bir resim ekliyoruz.  -->
          <li><a href="a3.html"><img src="images/demo/130x130.gif" width="130" height="130" alt=""></a></li>
          <!-- Madde işaretli, a3.html sayfasına bağlantılı genişliği:130px, yüksekliği:130px bir resim ekliyoruz.  -->
        </ul>
        <figcaption><a href="#">Resim Galerisi &raquo;</a></figcaption> <!-- Buraya dikkat yeni bir kod! -->
      </figure>
    </article>
  </section>
</div>
```

Yeni Kod!

Resim Galerisi » şeklinde bir görüntü nasıl elde edilir? Bunun için kodlara dikkat etmek gerekir.

```
<figcaption><a href="#">Resim Galerisi &raquo;</a></figcaption>
```

Görüldüğü üzere » simgesi » kodu ile elde edilir. Bu türden kodlar varlık isimleri olarak adlandırılır. Benzer şekilde birçok simgenin varlık ismi vardır. Yandaki QR kodlu bağlantıda[2] simgelerin karşılıklarına gelen kodlar bulunabilir.

Sitenin Alt Bilgi Bölümünün Oluşturulması

```
<!-- Footer -->
<div>
    <footer id="footer">
        <section>
            <h2>Hızlı Bağlantılar</h2>
            <nav>
              <ul>
                <li><a href="index.html">Anasayfa</a></li>
                <li><a href="hakkimda.html">Hakkımda</a></li>
                <li><a href="jskodlar.html">Javascript & Jquery</a></li>
                <li><a href="phpmysqlkodlar.php">PHP & MYSQL</a></li>
                <li><a href="iletisim.php">İletişim</a></li>
              </ul>
            </nav>
        </section>
    <p>Copyright &copy; 2017 - <a href="hakkimda.html">Siteyi Kodlayan kişi İsmi</a></p>
    </footer>
</div>
```

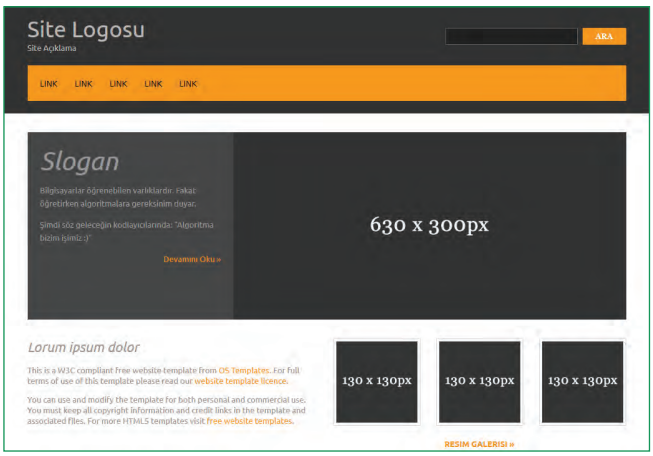
Bu ünite sonunda HTML5 ile site şablonunun ana çerçeveleri oluşturuldu. Böylece ünitedeki kazanımlara yönelik kodlama örneklerini uygulama fırsatı oldu. Şimdi yapılan kodlamaları, index.html şeklinde kendi isminizle oluşturduğunuz dizinin (klasörün) içerisine kaydediniz.

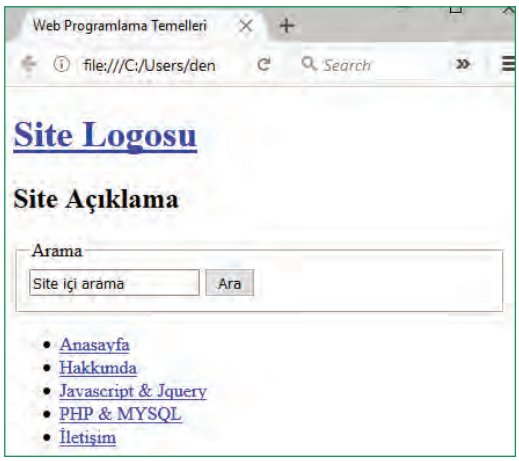
2 https://www.w3schools.com/charsets/ref_html_entities_4.asp

3. STİL SAYFALARINA GİRİŞ

3. STİL SAYFALARINA GİRİŞ

Proje olarak geliştirilen web sitemizde şu ana kadar sadece HTML kodları kullanıldı. Fakat tasarlamaya çalışılan web sayfası ile şuana kadar yapılanlar karşılaştırıldığında yapılacak çok işlemin olduğu görülür.





Bu bölümde site görsel olarak yukarıdaki resme benzetilmeye çalışılacaktır. Bunun için kullanılacak kodlama Cascading Style Sheets (CSS)' dir. CSS, bir HTML belgesinin stilini yani HTML öğelerinin nasıl görüntüleneceğini açıklar. Şu an en yeni standartı CSS3'tür. Her yeni standart kendinden önceki tüm standartları destekler. Şimdi şekildeki gibi bir görüntü elde etmek için ilk CSS kodlaması yapılacaktır. CSS kodlamanın üç yöntemi vardır.

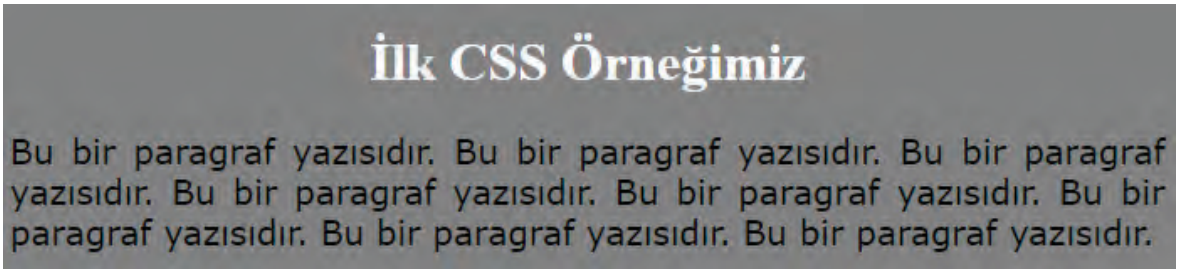
CSS'İN YAZILACAĞI YERE GÖRE KODLAMA YÖNTEMLERİ

1. HTML Etiketleri İçinde CSS Kodlama

```
<!DOCTYPE html>
<html>
<body style="background-color: grey;">
<!-- Sayfanın arkaplan rengini gri yapar -->

<h1 style="color: white; text-align: center;">İlk CSS Örneğimiz</h1>
<!--h1 ile etiketlenmiş başlığımızın yazı rengini beyaz ve ortalanmış yapar. -->

<p style="font-family: verdana; font-size: 20px; text-align: justify;">Bu bir paragraf yazısıdır.
Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır.
Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır.
Bu bir paragraf yazısıdır.</p>
<!-- Paragrafı, yazı tipi verdana, yazı büyüklüğü 20px ve iki yana yaslı olacak şekilde biçimlendirir. -->
</body>
</html>
```

Dikkat!

- Bu yöntemle yapılan biçimlendirme sadece kodu yazılan elemente uygulanır. Şimdi sizde </body> kapatma etiketinden önce bir p etiketi ekleyiniz ve içeriğini doldurunuz. Sonra test ediniz.



2. Bir Stil Bloğu (<style></style>) İçerisinde CSS Kodlama

```
<!DOCTYPE html>
<html>

<head>
<style>     /*Head bölümü içerisinde style etiketi açıyoruz.*/


body {      /*Hangi elementi biçimlendirmek istiyorsak elementten sonra bir parantez açıyoruz. */
    background-color: grey;     /* body elementinin arka plan rengini gri yapıyoruz. */
}           /* Elementin parantezini kapatıyoruz. */

h1 {
    color: white;       /* h1 elementinin yazı rengini beyaz yapıyoruz. */
    text-align: center; /* h1 elementi içerisindeki yazıları ortalanmış yapıyoruz. */
}

p {
    font-family: verdana;   /* p elementinin yazı tipini verdana yapıyoruz. */
    font-size: 20px;        /* p elementinin yazı büyüklüğünü 20px yapıyoruz. */
}
</style>
</head>
<body>

<h1>İlk CSS Örneğimiz</h1>
<p>Bu bir paragraf yazısıdır.
Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır.
Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır.
Bu bir paragraf yazısıdır.</</p>

</body>
</html>
```

Dikkat!

- Bu yöntemle yapılan biçimlendirme, sayfadaki kodu yazılan tüm elementlere uygulanır. Örneğin h1 etiketini ele alalım. Bu yöntemi kullandığınızda sayfadaki tüm h1 elementleri aynı biçimde olacaktır. Şimdi siz de </body> kapatma etiketinden önce bir h1 etiketi ekleyiniz ve içeriğini doldurunuz. Sonra test ediniz.

3. Harici Bir Stil Dosyası İçinde CSS Kodlama

Bu yöntemde sadece stil kodları *.css (ör: temel.css) şeklinde kaydedilir. Daha sonra web sayfasının head bölümü içerisine css uzantılı kaydedilmiş dosya, bağlantı verilir.

```
<!DOCTYPE html>
<html>

<head>
<link rel="stylesheet" href="styles/temel.css" type="text/css">
</head>
<body>

<h1>İlk CSS Örneğimiz</h1>
<p>Bu bir paragraf yazısıdır.
Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır.
Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır.
Bu bir paragraf yazısıdır.</</p>

</body>
</html>
```

Dikkat!

- Bu yöntemle yapmış olduğunuz biçimlendirme 2. yöntemle aynı işlemi yapar. Yani biçim, kodu yazdığınız tüm elementlere uygulanır.
- `<link rel="stylesheet" href="styles/temel.css" type="text/css">` kodundaki href içeriğinde temel.css dosyasından önce styles dizini olduğunuz fark ettiniz mi?

ELEMENTİN ETİKETİNE GÖRE CSS KODLAMA

Sayfamızda biçimlendirme yaparken elementin etiketine (ör: div, body, input, article, header, footer) göre kodlama yapılır. Bu kodlama biçimini CSS kodlamanın 2. Yönteminde görüldü.

```
body {      /*Hangi elementi biçimlendirmek istiyorsak elementten sonra bir parantez açıyoruz. */
    background-color: grey;     /* body elementinin arka plan rengini gri yapıyoruz. */
}           /* Elementin parantezini kapatıyoruz. */

h1 {
    color: white;       /* h1 elementinin yazı rengini beyaz yapıyoruz. */
    text-align: center; /* h1 elementi içerisindeki yazıları ortalanmış yapıyoruz. */
}

p {
    font-family: verdana;   /* p elementinin yazı tipini verdana yapıyoruz. */
    font-size: 20px;        /* p elementinin yazı büyüklüğünü 20px yapıyoruz. */
}
```

Bu kodlama yöntemi tek bir css kodu ile sayfanızdaki benzer etiketlerin aynı şekilde görüntülenmesine olanak sağlar. Yukarıdaki örneğimizi ele alacak olursak, sayfamızdaki h1 etiketli tüm elementler için benzer biçimlendirme uygulanmıştır. Bazı h1 etiketli elementlerin farklı biçimlendirmeye sahip

olması isteniyorsa ilk olarak "HTML etiketleri içinde CSS kodlama" yöntemi kullanılır. Fakat bu yöntem çok kullanışlı değildir.

```
<h1 style="color:green; background-color:grey;">Farklı Biçimlendirmeye sahip olmasını
istediğimiz h1 etiketi</h1>
```

Kullanılacak diğer bir yöntem ise, "Elementin Özelliklerine Göre CSS Kodlama" dır.

ELEMENTİN ÖZELLİKLERİNE GÖRE CSS KODLAMA

HTML elementlerinin özelliklerine göre css kodlama yöntemini açıklamadan önce, "Elementin Özellikleri" bilinmelidir. Aşağıdaki HTML kodlarını incelendiğinde etiket isimleri dışında kırmızı etikete özgü özellikler (kırmızı renkli) olduğu görülecektir. Bu özelliklerden id ve class tüm etiketlerdeki ortak özelliktir.

```
<h1 id="" class="" style="" > ... </h1>

<img id="" class="" alt="" src="" title="" />

<a id="" class="" href="" target="" > ... </a>
```

Elementlerin sahip olduğu özelliklere göre css kodlamada genellikle id ve class özellikleri kullanılır. Örneğin class özelliği "renkli" ya da id özelliği "renkli" olan herhangi bir elementin biçimi değiştirilebilir. Bunun için id'ye göre kodlama yaparken "#" simgesi, class' a göre kodlama yaparken "." simgesi ile başlanır.

```
.renkli { color:orange; }
#renkli { color:green; }
```

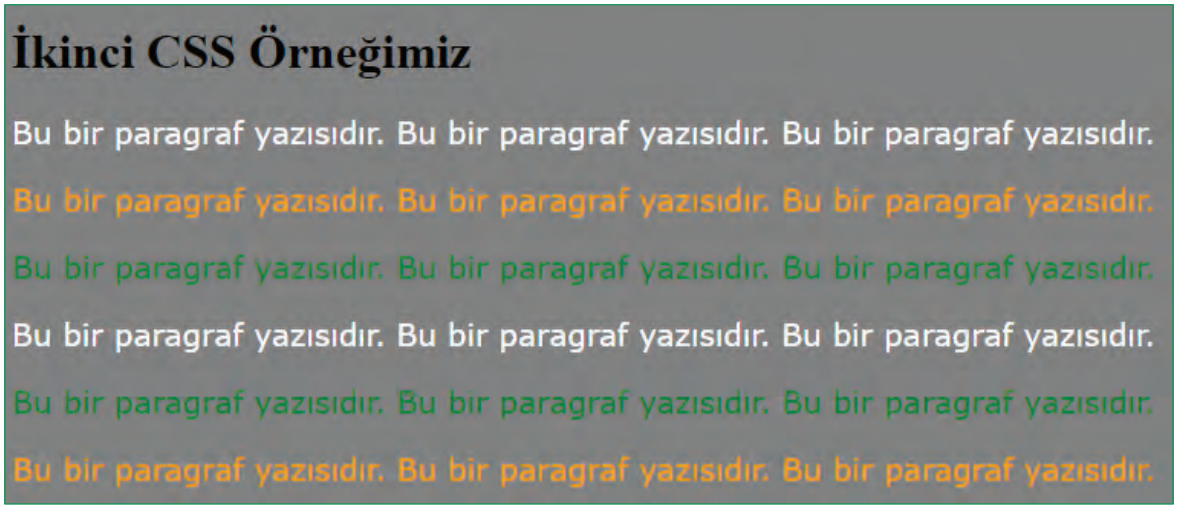
Herhangi bir kodlama yönteminin kullanılabilmesi için önce, biçimlendirme yapmak istenilen etikete id ya da class özellikleri girilir. Fakat bazen buna gereksinim olmadan da kodlama yapılabilir. Örneğin <a> etiketi ile bir bağlantı oluşturulduğunda:

```
<a href="http://okulunuz.com" target="_blank" > ... </a>
```

Bu etiketi href özelliği http://okulunuz.com olacak şekilde bir seçim yaparak biçimlendirilebilir.

```
a[href="http://okulunuz.com"]
{
    ...
}
```

```
<!DOCTYPE html>
<html>

<head>
<style>
/* Bir önceki örnekte kodları altalta yazmıştık. Aşağıda görüldüğü gibi kodları yanyana da yazabiliriz*/
body { background-color: grey;}
p { font-family: verdana;font-size: 20px;color:white; }
.renkli { color:orange; }
#renkli { color:green; }

</style>
</head>
<body>

<h1>İkinci CSS Örneğimiz</h1>
<p>Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. </p>
<!-- Class özelliği renkli olan paragraf-->
<p class="renkli"> Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. </p>
<!-- id özelliği renkli olan paragraf-->
<p id="renkli">Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır.</p>
<!-- Herhangi bir özellik atanmamış standart paragraf-->
<p>Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. </p>
<!-- Class özelliği renkli olan paragraf-->
<p id="renkli"> Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. </p>
<!-- id özelliği renkli olan paragraf-->
<p class="renkli"> Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır. </p>
</body>
</html>
```

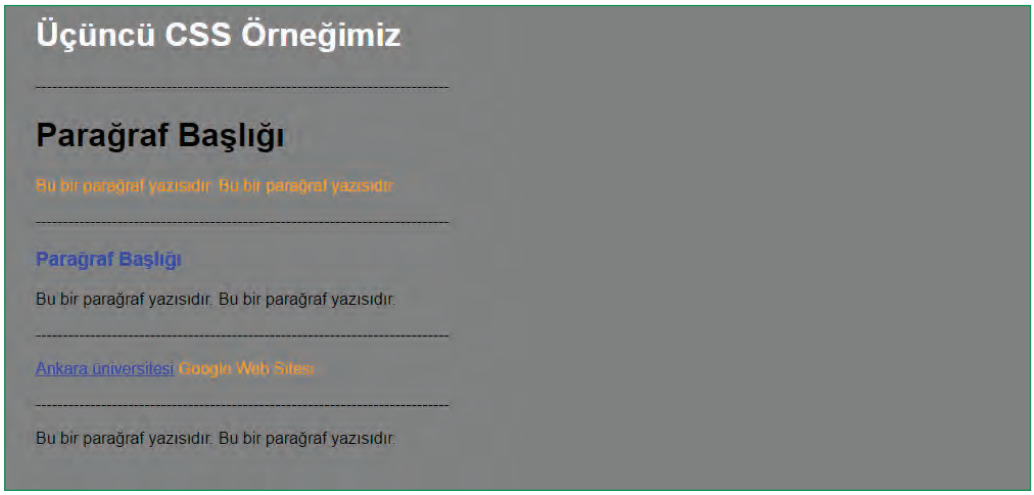


İç İçe Geçmiş CSS Kodlama

Son olarak, içiçe geçmiş css kodlama örneklerini inceleyelim. Örnek:

- Sadece **header** elementi içerisindeki **class özelliği "renkli"** olan elementleri nasıl biçimlendirilir?
- Sadece **section** elementi içerisindeki **article** elementi içerisinde olan h1 elementini nasıl biçimlendirilir?
- Sadece **footer** elementi içerisindeki **google sitesine link verilmiş a** elementleri nasıl biçimlendirilir?

```
<!DOCTYPE html>
<html>

<head>
<style>
body {padding:25px; background-color: grey;font-family:arial;}

header .renkli {color:orange;}
/* header elementi içerisindeki class özelliği "renkli" olan elementlerin seçimi */

section article h1 {color:blue;}
/* section elementi içerisindeki article elementi içerisinde olan h1 elementinin seçimi */

footer [href*=google] {color:orange; text-decoration:none;}
/* footer elementi içerisindeki google sitesine link verilmiş a elementinin seçimi */
</style>
</head>
<body>

<h1 style="color:white;">Üçüncü CSS Örneğimiz</h1>
<p>-------------------------------------------------------------------------------</p>
<header>
    <h1>Parağraf Başlığı </h1>
    <p class="renkli"> Bu bir parağraf yazısıdır. Bu bir parağraf yazısıdır. </p>
</header>
<p>-------------------------------------------------------------------------------</p>
<section>
    <article>
        <h1>Parağraf Başlığı </h1>
        <p class="renkli"> Bu bir parağraf yazısıdır. Bu bir parağraf yazısıdır. </p>
    </article>
</section>
<p>-------------------------------------------------------------------------------</p>
<footer>
    <a href="http://ankara.edu.tr">Ankara üniversitesi</a>
    <a href="http://google.com.tr">Google Web Sitesi</a>
</footer>

<p>-------------------------------------------------------------------------------</p>
<p class="renkli"> Bu bir parağraf yazısıdır. Bu bir parağraf yazısıdır. </p>

</body>
</html>
```



Dikkat!

- Class özelliği "renkli" olan iki p elementinin birbirinden farklı biçime sahip olduğunu farkettiniz mi?
- İki h1 elementinin birbirinden farklı biçime sahip olduğunu farkettiniz mi?
- Aynı footer elementi içerisinde olup farklı biçime sahip iki a elementini farkettiniz mi?

PROJE SİTESİNİN CSS İLE GÖRSELLEŞTİRİLMESİ

Varsayılan Ayarları

Bir web sitesinin CSS ile görselleştirilirken ilk olarak yapılması gereken varsayılan olarak kullanacağımız genel CSS ayarlarını oluşturmaktadır.

```
<style>
/*-----------------------------------------------Genel Ayarlar---------------------------------------*/
    * {             /* Web sayfasındaki tüm etiketlerin seçimi */
        margin:0;   /* Elementlerin kendinden önce ve sonraki elementlerle olan mesafesi 0 pixcel (px) olsun.*/
        padding:0;  /* Elementlerin kendinden önce ve sonraki elementlerle olan mesafesi 0 px olsun.*/
        text-decoration:none; /* Elementlerin içerisinde kalan metinlerin altı çizili olmasın.*/
    }
    body{           /*body elementinin seçimi*/
        font-size:13px;  /* Elementin içerisindeki metinlerin yazı büyüklüğü 13 px olsun.*/
        font-family: Ubuntu;  /* Elementin içerisindeki metinlerin yazı tipi Ubuntu olsun.*/
        color:#919191;  /* Elementin içerisindeki metinler #919191 koduna karşılık gelen yazı renginde olsun. */
        background-color:#232323; /* Elementin arka plan rengi #232323 koduna karşılık gelen renkte olsun*/
        }

    header, container, footer {  /* header, container, footer elementlerinin seçimi*/
        width:960px; /* Elementin Genişliği 960 px olsun.*/
        margin:auto; /* Elementin sağında ve solunda kalan boşluklar eşit oranda (ortalı) olsun. */
        }
    h1, h2, h3, h4, h5, h6{

        font-size:22px;
        font-weight:normal;
        font-style:italic;
        line-height:normal;}
    nav ul{
        list-style:none; /* madde işareti imleri (yuvarlak, kare, sayı) gösterilmesin. */
        }
    form, fieldset, legend{
        border:none;
    }
    input, textarea, select{
        font-size:12px;
        font-family:Georgia,"Times New Roman",Times,serif;
    }
    a {
        color:#ffffff; /* Linkler #ffffff koduna karşılık gelen yazı renginde olsun.*/
    }
</style>
```

Site Şablonundaki Temel Çerçevelerin Biçimlendirilmesi

Şablonumuzda, body etiketi içerisinde **banner, content** ve **footer** olmak üzere üç ana çerçeve oluşturulmuştur. Şimdi bu bölümlerin her biri içerisindeki etiketler için şablona uygun şekilde biçimlendirme yapılacaktır.

Banner Çerçevesinin Biçimlendirilmesi

"banner" çerçevesinin html kodlarını açarak en üsteki div elementinin class özelliği "çerçeve1", sitenin logosunun ve açıklamasının bulunacağı div etiketinin id özelliği "logogrup" olarak değiştirilir.

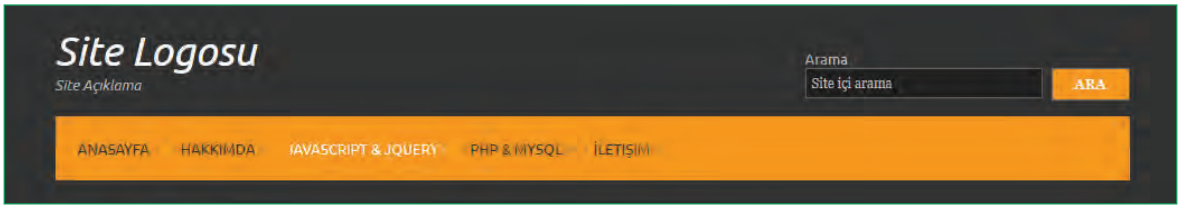
```
<!-- Banner -->
<div class="cerceve1">
  <header>
    <div id="logogrup">
      <h1><a href="index.html">Site Logosu</a></h1>
      <h2>Site Açıklama</h2>
    </div>
    <form>
      <fieldset>
        <legend>Arama</legend>
        <input type="text" value="Site içi arama"
        onFocus="this.value=(this.value=='Site içi arama')? '' : this.value ;">
        <input type="submit" id="ara" value="Ara">
      </fieldset>
    </form>
    <nav>
      <ul>
            <li><a href="#">Anasayfa</a></li>
            <li><a href="#">Hakkımda</a></li>
            <li><a href="#">Javascript & Jquery</a></li>
            <li><a href="#">PHP & MYSQL</a></li>
            <li><a href="#">İletişim</a></li>
      </ul>
    </nav>
  </header>
</div>
```

Şimdi en dıştaki elementten başlayarak en içteki elemente doğru banner çerçevesinin CSS ayarları yapılabilir.

```
/*Çerçeve*/
    .cerceve1 {
        width:100%;
        text-align:left;
        color:#C0BAB6;
        background-color:#333333;
        }

/* Renk kodlarının hangi renge karşılık geldiğini
http://htmlcolorcodes.com/ adresinden bulabilirsiniz. */


/*---Header---*/
    header{ padding:20px; width:960px; }

    /*---Header içindeki logogrup idli element---*/
    header #logogrup { float:left; margin:0 0 20px 0;} /* margin: üst sağ alt sol */
    header #logogrup h1{ font-size:36px;}
    header #logogrup h2{ font-size:13px;}
```

```
/*---Header içindeki form elementi---*/
header form{
    width:290px;
    float:right; /*sağa hizala*/
    margin-top:20px;
    padding:0;
}
header form input{
    float:left; /*sola hizala*/
    width:200px;
    margin:0;
    padding:5px;
    color:#C0BAB6;
    background-color:#232323;
    border:1px solid #666666;
    }
header form #ara{
    float:right;
    width:70px;
    font-size:12px;
    font-weight:bold;
    text-transform:uppercase; /*yazıları büyük harfe dönüştürme*/
    color:#FFFFFF;
    background-color:#FF9900;
    }
```

```
/*---Header içindeki nav elementi---*/
header nav {
    width:100%;
    margin:0;
    padding:20px 0;
    color:#C0BAB6;
    background-color:#FF9900;
    clear:both; /*  Elementin solunda veya sağında herhangi bir başka element bulunamaz. */
    }
header nav ul{padding:0 20px;}
header nav li{
    display:inline; /* Maddeleri alt alta değil de yanyana sıralar*/
    margin-right:25px;
    text-transform:uppercase;
    }
header nav li a{color:#333333;} /*linklerin genel rengi*/
header nav li a:hover{color:#ffffff;} /*linklerin üzerine gelindiğinde rengi*/
```



Şekil 6: Header çerçevesinin görünümü

Content Çerçevesinin Biçimlendirilmesi

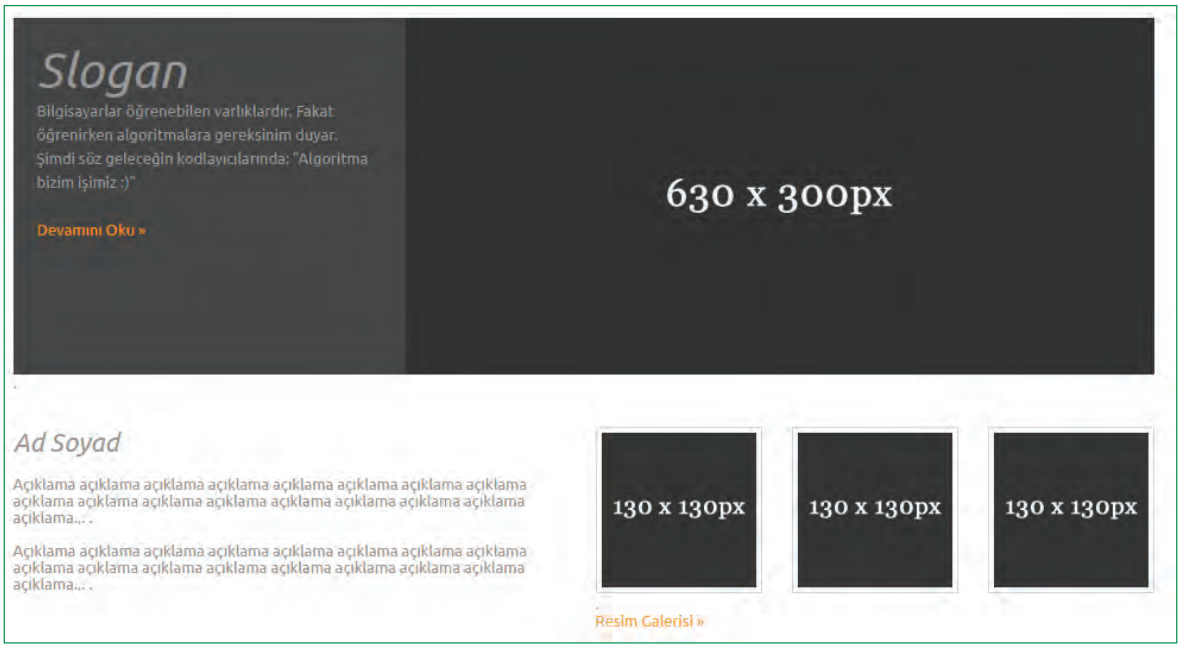
"content" çerçevesinin html kodlarını açarak etiketlere verilen id ve class özellikleri aşağıdaki gibi değiştirilir.

```
<!-- Content -->
<div class="cerceve2">
  <div id="container" class="clear">
    <!-- Slider -->
    <section id="slider" class="clear">
      <figure><img src="images/demo/630x300.gif" alt="">
        <figcaption>
          <h2>Slogan</h2>
          <p> Bilgisayarlar öğrenebilen varlıklardır.
          Fakat öğrenirken algoritmalara gereksinim duyar. </p>
          <p> Şimdi söz geleceğin kodlayıcılarında:
          "Algoritma bizim işimiz :)"</p>
          <footer class="devam"><a href="#">Devamını Oku &raquo;</a></footer>
        </figcaption>
      </figure>
    </section>
    <!-- Aciklama -->
    <div id="intro">
      <section class="clear">
        <!-- article 1 -->
        <article class="aciklama">
          <h2>Ad Soyad</h2>
          <p>Açıklama açıklama açıklama açıklama açıklama açıklama
          açıklama açıklama açıklama açıklama açıklama açıklama
          açıklama açıklama açıklama açıklama açıklama... </a>.</p>
          <br/>
          <p>Açıklama açıklama açıklama açıklama açıklama açıklama
          açıklama açıklama açıklama açıklama açıklama açıklama
          açıklama açıklama açıklama açıklama açıklama... </a>.</p>
        </article>
        <!-- article 2 -->
        <article class="aciklama ek">
          <figure>
            <ul class="clear">
              <li>
                <a href="#">
                    <img src="images/demo/130x130.gif" width="130" height="130">
                </a>
              </li>
               <li>
                <a href="#">
                    <img src="images/demo/130x130.gif" width="130" height="130">
                </a>
              </li>
               <li class="last">
                <a href="#">
                    <img src="images/demo/130x130.gif" width="130" height="130">
                </a>
              </li>
            </ul>
            <figcaption><a href="#">Resim Galerisi &raquo;</a></figcaption>
          </figure>
        </article>
      </section>
      </div>
</div>
</div>
```

Şimdi en dıştaki elementten başlayarak en içteki elemente doğru content çerçevesinin CSS ayarları yapılır.

```
<style>
/* ----------------------------------------------Genel---------------------------------------*/
.cerceve2 { width:100%; margin:0; padding:0; text-align:left; color:#979797; background-color:#FFFFFF;}
.clear:after{       /* class özelliği clear olan elementten sonra */
    content:".";    /* içeriği . olan bir element yaratalım ve */
    display:block;  /* bu elementten sonraki elementin bir alt satırda görüntülenmesini sağlayalım. */
    }

/*----------------------------------------------Content---------------------------------------*/
#container{width:960px;
        margin: 0 auto;
        /* üst dış boşluğu 0px sağında ve solunda kalan boşluklar eşit oranda (ortalı) olsun. */
        padding:30px 0;
        /* üst iç boşluğu 30px diğer iç boşlukları 0px olsun.*/
        }
#container section{
    display:block;
    /* elementin genişliğini bir üstündeki elementin genişliğine (width=100%) eşitleyelim. */
    margin-bottom:30px;
    padding:0;
    }
#container .last{margin:0;}
#container .more{text-align:right;}

/* ----------------------------------------------Slider---------------------------------------*/

#container #slider{
    width:100%; ,
    /* elementin genişliğini bir üstündeki elementin genişliğine (width=100%) eşitleyelim. */
    }

#container #slider figure img{float:right; width:630px; height:300px;}

#container #slider figure figcaption{
    float:left; width:290px; height:260px;
    padding:20px; color:#989898; background-color:#464646;
    line-height:20px; /* satır yüksekliğini belirtir. */
    }

#container #slider figure figcaption a{color:#FF9900; background-color:#464646;}
#container #slider figure h2{font-size:42px;}
```

```
/* ----------------------------------------------Intro---------------------------------------*/

#container #intro{margin-bottom:60px; padding-bottom:20px; }
#container #intro section .aciklama{width:470px; float:left; margin:0 20px 0 0;}
#container #intro section .ek {margin-right:0px; }
#container #intro section article h2 {margin-bottom:15px;}
#container #intro section article figure ul {list-style:none;}
#container #intro section article figure ul li {float:left; margin:0 25px 0 0;}
#container #intro section article figure ul li.last {margin-right:0;}
#container #intro section article figure ul li img { width:130px; height:130px; padding:4px;
    border-width:1px; /* resmin kenarları 1px genişliğinde olsun */
    border-style:solid; /* resmin kenarları solid (çizgi) stilinde olsun */
    border-color:#D6D6D6; /* resmin kenarları #D6D6D6 koduna karşılık gelen renkte olsun */

    }
#container #intro section article figure figcaption a {color:orange;}

</style>
```



Şekil 7: Content çerçevesinin görünümü

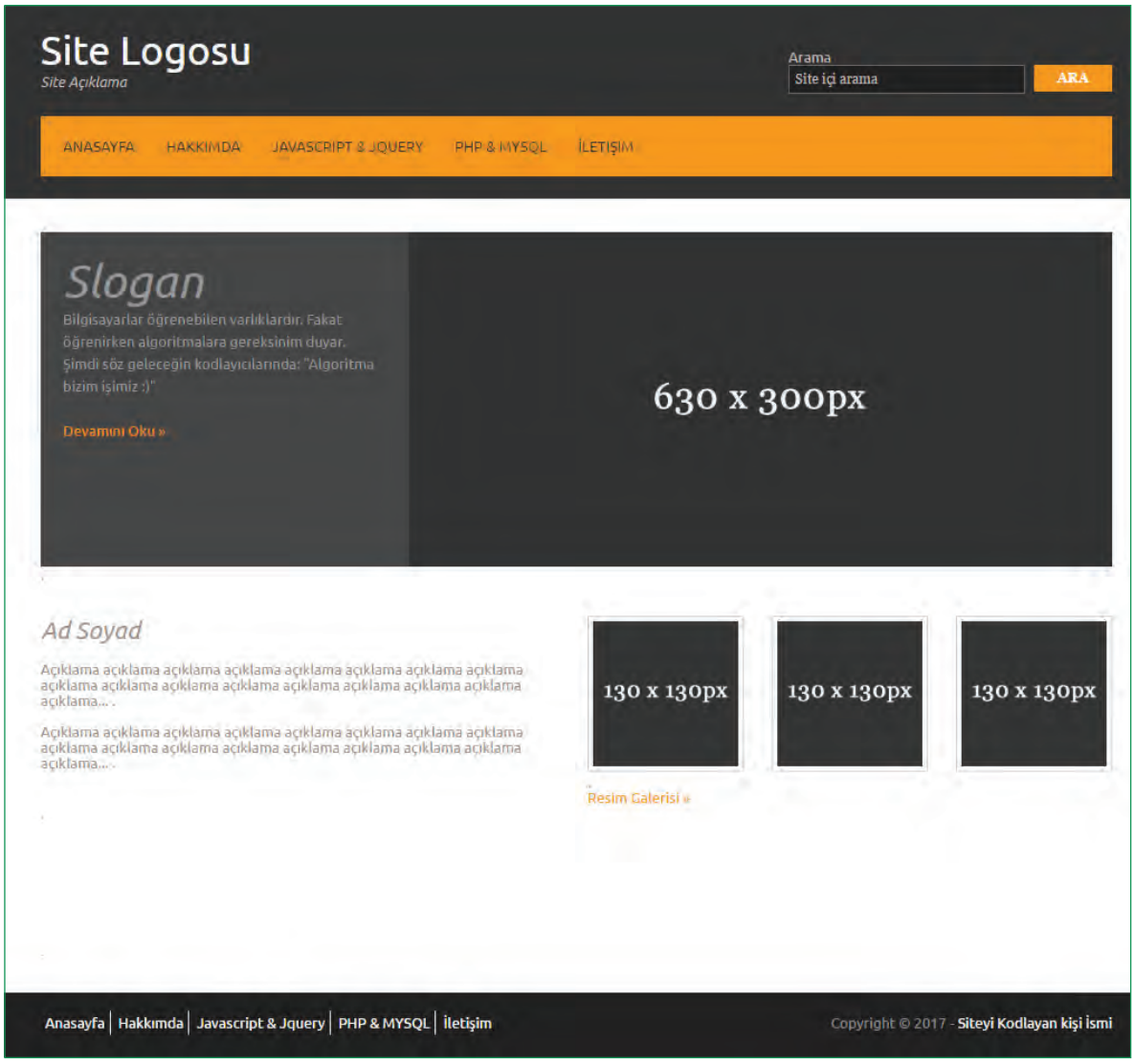
Footer Çerçevesinin Biçimlendirilmesi

Son olarak "footer" çerçevesinin html kodları açılır ve en üsteki div elementinin class özelliği "cerceve3" değiştirilir. Diğer etiketler anlamsal elementler olduğu için özel bir id ya da class belirtmeye gerek yoktur.

```
<!-- Footer -->
<div class="cerceve3">
  <footer id="footer">
    <nav>
      <ul>
            <li><a href="#">Anasayfa</a></li>
            <li><a href="#">Hakkımda</a></li>
            <li><a href="#">Javascript & Jquery</a></li>
            <li><a href="#">PHP & MYSQL</a></li>
            <li><a href="#">İletişim</a></li>
      </ul>
    </nav>
    <p style="float:right">Copyright &copy; 2017 -
        <a href="#"> Siteyi Kodlayan kişi İsmi</a>
    </p>
  </footer>
</div>
```

Şimdi en dıştaki elementten başlayarak en içteki elemente doğru footer çerçevesinin CSS ayarları yapılır.

```
<style>
    .cerceve3 { width:100%; margin:0; padding:0; text-align:left; color:#919191; background-color:#232323;}
    footer  {padding:20px 0;}
    footer nav {
        display:inline;
        /* block kendinden sonraki elementi satır başına atıyordu.
            inline-block ise, kendinden sonraki elementi yan yana koyar.*/
        }
    footer nav li {padding:4px; border-right:1px solid white;
        display:inline;
        /* menüyü oluşturan madde imlerinin alt alta değil yanyana gelmesini sağlar.*/
    }
    footer nav li:last-child{ /* menünün en son elementini seçer */
        border:none;
        }
    footer a{color:white;}
    footer a:hover{color:orange;}
</style>
```



Şekil 8: Sitenin genel görünümü

4. ETKİLEŞİM

4. ETKİLEŞİM

Şimdiye kadar HTML kodları ile oluşan bir site şablonu, CSS kodları ile görsel hâle geldi. Şimdi siteyi ziyaret eden kullanıcıların site içerisinde neler yapabileceği ile ilgili beyin fırtınası yapılacaktır. İçerik okunabilir, linkleri kullanarak sayfalar arası geçiş yapılabilir. Eklenmesi istenilen başka bir madde var mıdır?

Örneğin;

- Her hangi bir HTML elementi içerisinde yazılı olan bir metni değiştirebilir mi?
- Ya da bir div elementinin arka plan rengini değiştirebilir mi?

İşte, kullanıcıların bu türden işlemleri gerçekleştirebilmesi için HTML ve CSS kodları yeterli olmayabilir. Bunun için farklı kod yapılarına gereksinim duyulur. Bu noktada Javascript:

- HTML içeriğini değiştirebilir.
- HTML niteliklerini değiştirebilir.
- CSS stilini değiştirebilir.
- Bu işlemleri mantıksal sınamalara göre de yapabilir. Örneğin siteyi ziyaret eden kullanıcı harf yerine metin girişi yaptıysa, uyarı gösterebilir.

Özetle, HTML web sayfalarının içeriğini tanımlamak, CSS web sayfalarının biçimini belirlemek, Javascript ise web sayfalarının davranışını programlamak için kullanılır. Peki Javascript kodları nereye eklenir. Daha önce görüldüğü üzere, farklı CSS kodlama yöntemleri gibi Javascript kodlama yöntemleri de vardır.

Javascript Kodlama Yöntemleri

İşlevsel Olmayan Kullanımı

```
<!DOCTYPE html>
<html>
    <body>
        <h2>Hızlı ama işlevsel olmayan kullanım</h2>
        <p id="demo">Paragraf</p>
        <button
            onclick='document.getElementById("demo").innerHTML = "Butona tıklayınca parağraf değişti.";'>
            Değiştir
                </button>
    </body>
</html>
```

Hızlı ama işlevsel olmayan kullanım

Butona tıklayınca parağraf değişti.

Değiştir

Head Elementi Arasındaki Kullanımı

```
<!DOCTYPE html>
<html>
    <head>
        <script>
            function degistir() {
                document.getElementById("demo").innerHTML = "Butona tıklayınca parağraf değişti.";
            }
        </script>
    </head>
    <body>
        <h2>head elementi arasındaki kullanımı </h2>
        <p id="demo">Parağraf</p>
        <button type="button" onclick="degistir()">Değiştir</button>
    </body>
</html>
```

head elementi arasındaki kullanımı

Butona tıklayınca parağraf değişti.

Değiştir

Body Elementi Arasındaki Kullanımı

```
<!DOCTYPE html>
<html>
    <body>
        <script>
            function degistir() {
                document.getElementById("demo").innerHTML = "Butona tıklayınca parağraf değişti.";
            }
        </script>
        <h2>body elementi arasındaki kullanımı </h2>
        <p id="demo">Parağraf</p>
        <button type="button" onclick="degistir()">Değiştir</button>
    </body>
</html>
```

body elementi arasındaki kullanımı

Butona tıklayınca parağraf değişti.

Değiştir

Dış Bir Dosyaya Bağlantı Verilerek Kullanımı

```
<!DOCTYPE html>
<html>
    <body>
        <script src="myScript.js"></script>

        <!--Dosyanın içeriği:

            function degistir() {
                document.getElementById("demo").innerHTML = "Butona tıklayınca parağraf değişti.";
            }
        -->

        <!-- ya da
            <script src="https://www.w3schools.com/js/myScript.js"></script>
            <script src="js/myScript.js"></script>
            şeklinde de olabilirdi.
            -->
        <h2>Dış bir dosyaya bağlantı verilmesi </h2>
        <p id="demo">Parağraf</p>
        <button type="button" onclick="degistir()">Değiştir</button>
    </body>
</html>
```

Dış bir dosyaya bağlantı verilmesi

Butona tıklayınca parağraf değişti.

Değiştir

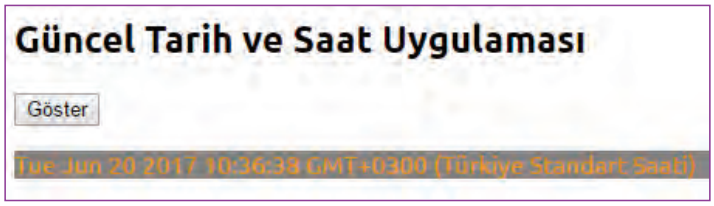
Şimdi bu yöntemlerden bazılarını kullanarak senaryo temelinde kodlama yapılacaktır.

HTML İÇERİĞİ DEĞİŞTİRME

Javascriptle HTML içeriğinin değiştirilebileceğinden bahsedilmiştir. Web sayfamızda bir buton aracılığıyla güncel tarih ve saatin gösterilmesi bu türden uygulamalara bir örnek olarak verilebilir.

```
<!DOCTYPE html>
<html>
<body style="font-family:Ubuntu;">

<h2>Güncel Tarih ve Saat Uygulaması</h2>

<button onclick="document.getElementById('demo').innerHTML = Date()">Göster</button>
<!--
    Kod Açıklaması
    onclick="document.getElementById('demo').innerHTML = Date()"
    onclick: kullanıcı butona tıkladığında
    document: sayfa içerisindeki
    getElementById('demo'): id'si demo olan elementin
    innerHTML: içeriğini
    Date(): güncel tarih ve saat ile değiştir.

-->

<p id="demo" style="background-color:grey;color:orange;"></p>

</body>
</html>
```

Aşağıdaki ekran görüntüsünde "Göster" butonuna her tıklandığında butonun altındaki saat de değişmektedir.

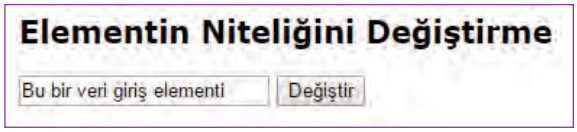


HTML NİTELİKLERİNİ DEĞİŞTİRME

Hatırlanıldığı üzere, Html sayfaları etiketlerden oluşmaktadır. Örneğin div, p, a, button, input, vb. sayfa ile kullanıcının etkileşimi sonucunda, herhangi bir etiketin özelliğini Javascript ile değiştirmek mümkündür. Örneğin kullanıcının herhangi bir butona tıklaması ile div olan bir etiketi butona ya da input olan bir etiketi butona dönüştürmek mümkündür.

```
<!DOCTYPE html>
<html>
<body style="font-family:'Verdana'; font-size:14px;">

<h2>Elementin Niteliğini Değiştirme</h2>

<input type="text" id="degistirilecekElement" value="Bu bir veri giriş elementi">

<button onclick="Degistir();">Değiştir</button>

<script>
    function Degistir() {

        document. // sayfa içerisindeki
        getElementsByTagName('INPUT')['degistirilecekElement'].
        // id'si "degistirilecekElement" olan input elementinin
        setAttribute('type', 'button');
        // type niteliğini button olarak değiştir.

        document.   //sayfa içerisindeki
        getElementsByTagName('INPUT')['degistirilecekElement'].
        //id'si "degistirilecekElement" olan input elementinin
        setAttribute('value', 'Bu bir buton elementi');
        //value niteliğini "Bu bir buton elementi" olarak değiştir.
    }
</script>

</body>
</html>
```

Kullanıcı Butona Tıklamadan Önce

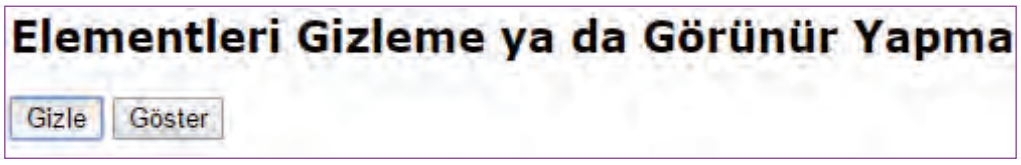


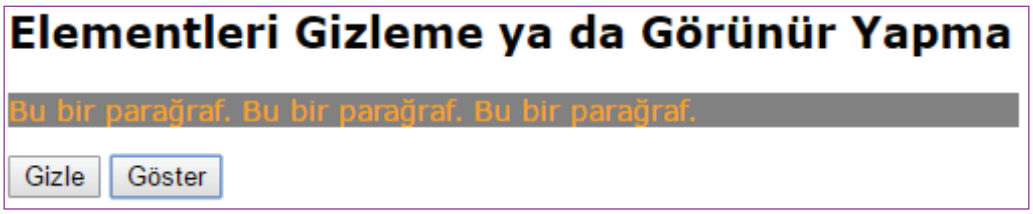
Kullanıcı Butona Tıkladıktan Sonra

CSS DEĞİŞTİRME

HTML sayfaları biçimlendirilirken CSS kullanılmıştır. Bu biçimlendirme özelliklerini de kullanıcıların eylemlerine göre (Örneğin bir butona tıkladığında ya da bir veri girişi yaptığında) Javascriptle değiştirmek mümkündür.

```
<!DOCTYPE html>
<html>
<body style="font-family:Verdana; font-size:14px;">

<h2>Elementleri Gizleme ya da Görünür Yapma</h2>

<p id="p1" style="background-color:grey;color:orange;">
Bu bir parağraf. Bu bir parağraf. Bu bir parağraf.
</p>

<input type="button" value="Gizle"
onclick="document.getElementById('p1').style.display='none'">
<!--
    Elementin Gizlenmesi
    butona tıklandığında, id'si p1 olan elementin stilleri içerisinden
    display özelliğine none yap.
    Not: ...style.visibility='hidden' da kullanılabilir.
-->

<input type="button" value="Göster"
onclick="document.getElementById('p1').style.display='block'">

<!--
    Elementin Görünür Yapılması
    butona tıklandığında, id'si p1 olan elementin stilleri içerisinden
    display özelliğine block yap.
    Not:  ...style.visibility='visible' da kullanılabilir.
-->
</body>
</html>
```





Şimdiye kadar Javascript kullanarak basit etkileşimli kodlamalar yapıldı. Artık hedefi daha yükseklere taşımanın sırası gelmedi mi sizce? Hatırlarsanız, javascript' de mantıksal sınamalara göre işlem yapılabileceğinden bahsedilmişti. Örneğin

- Eğer … olduysa, …. yap.
- Eğer … ise … değiştir.
- Eğer … ise … göster.

Bu şekilde sayısız senaryolar oluşturulabilir. Peki bu sınama işlemini javascript ile nasıl yapabiliriz?

MANTIKSAL SINAMA

Bu konu başlığı altında javascript' de mantıksal sınama yapılmasına olanak sağlayan kontrol yapıları görülecektir.

if-else kontrol yapısı

```
<script>
  if (durum1) {
     // durum1 doğru ise, yapılacak kodlama
  } else if (durum2) {
     // durum2 doğru ise, yapılacak kodlama
  } else {
     // durum1 ve durum2 dışındaki tüm diğer durumlarda yapılacak kodlama
  }
</script>
```

Şimdi, bir senaryo üzerinden if-else kontrol yapısı kullanılacaktır. Örneğin siteye giriş zamanına göre kullanıcıya "Günaydın", "İyi günler" ya da "İyi akşamlar" mesajı veren bir kodlama yapılacaktır.

```
<!DOCTYPE html>
<html>
<body style="font-family:Verdana;font-size:14px;">
    <p style="font-weight:bold;">Saate Göre Mesaj Verme</p>

    <p id="demo"></p>

    <script>
        var mesaj, saat; // var kodu ile değişken tanımlıyoruz.

        saat = new Date().getHours();
        // Date()       : yeni bir date nesnesi yaratır
        // .getHours()  : ve date nesnesinin sadece hours yani saatini alır.

        if (saat < 10) {            // eğer saat 10 dan küçükse
            mesaj = "Günaydın";     // mesaj değişkenini "Günaydın" olarak eşitle
        } else if (saat < 20) {     // eğer saat 20 den küçükse
            mesaj = "İyi Günler";   // mesaj değişkenini "İyi Günler" olarak eşitle
        } else {                    // eğer saat bu iki durumun dışında ise yani 20 ve 20 den büyükse
            mesaj = "İyi akşamlar"; // mesaj değişkenini "İyi Akşamlar" olarak eşitle
        }
        // Şimdi sıra bu mesaj değişkeninin ekranda gösterilmesine geldi.
        // Üç farklı yöntemle kullanıcıya mesaj verilebilir.
        document.getElementById('demo').innerHTML= mesaj;
        // Daha önce görmüş olduğumuz HTML içeriği değiştirme
        window.alert(mesaj);
        // Bir uyarı penceresi içerisinde mesajı verme
        document.write(mesaj);
        // Sayfaya javascriptle yazdırma
    </script>

</body>
</html>
```

SWİTCH KONTROL YAPISI

```
<script>
    switch(degisken) {
        case x:
            // değişkenin x olma durumunda yapılacak kodlama
            break;
        case y:
            // değişkenin y olma durumunda yapılacak kodlama
            break;
        default:
            // değişkenin x ve y olmama durumunda yapılacak kodlama
    }
</script>
```

Şimdi bir senaryo üzerinden switch kontrol yapısı kullanılacaktır. Örneğin siteye hangi günde giriş yapıldığının mesajını veren bir kodlama yapılacaktır.

```
<html>
<body style="font-family:Verdana;font-size:14px;">

<p style="font-weight:bold;"> Hangi gün olduğunu bulan kontrol yapısı</p>
<p style="font-weight:bold;padding:30px;background-color:orange;" id="demo"></p>

<script>
    var gun, mesaj;             // var kodu ile gun ve mesaj değişkenleri oluşturuldu.
    gun = new Date().getDay();  // getDay() ile o andaki tarihin gün numarasını gun değişkenine eşitliyoruz.
    switch (gun) {              //

      case 0:                   // gün değişkeni 0 ise;
          mesaj = "Pazar";      // mesaj değişkenini "Pazar" a eşitle
          break;                // diğer durumları kontrol etmene gerek kalmadı.

      case 1:                   // gün değişkeni 1 ise;
          mesaj = "Pazartesi";  // mesaj değişkenini "Pazartesi" ye eşitle
          break;                // diğer durumları kontrol etmene gerek kalmadı.

      case 2:                   // gün değişkeni 2 ise;
          mesaj = "Salı";// mesaj değişkenini "Salı" ya eşitle
          break;                // diğer durumları kontrol etmene gerek kalmadı.

      case 3:                   // gün değişkeni 3 ise;
          mesaj = "Çarşamba";// mesaj değişkenini "Çarşamba" ya eşitle
          break;                // diğer durumları kontrol etmene gerek kalmadı.

      case 4:                   // gün değişkeni 4 ise;
          mesaj = "Perşembe";// mesaj değişkenini "Perşembe" ye eşitle
          break;                // diğer durumları kontrol etmene gerek kalmadı.

      case 5:                   // gün değişkeni 5 ise;
          mesaj = "Cuma";       // mesaj değişkenini "Cuma" ya eşitle
          break;                // diğer durumları kontrol etmene gerek kalmadı.

      case  6:                  // gün değişkeni 6 ise;
          mesaj = "Cumartesi";// mesaj değişkenini "Cumartesi" ye eşitle
          // başka durum olmadığı için break kullanmadık.
    }
    document.getElementById("demo").innerHTML = "Bugün: " + mesaj;
</script>

</body>
</html>
```

Kodlayıcı Görevi

- Kontrol yapılarında kullanılan kodlar test edildiğinde sayfa açılır açılmaz javascript kodlarının çalıştığı fark edilecektir. Daha önceki örneklerde kullanıcının sayfa ile etkileşimi vardı yani kullanıcı bir butona tıkladığında (ya da imleç herhangi bir nesnenin üzerine getirildiğinde) javascript kodları çalışıyordu.
- Şimdi siz kodlayıcılardan sayfaya bir etkileşim ekleyerek kontrol yapılarının çalıştırılması beklenmektedir.

Tekrarlı (Döngü) Yapılar

Diğer programlama dillerinde olduğu gibi, Javascript' de de aynı kod her seferinde farklı bir değer için çalıştırılmak isteniyorsa tekrarlı yapılar (Döngüler) kullanılır. Örneğin birden 100'e kadar sayı yazdırmak istenirse 100 satır kodlama yapmak yerine bunu 3 satır kodlama ile yazdırabilmek mümkündür. Şimdi, senaryolar temelinde tekrarlı yapılar kullanılacaktır.

For Döngüsü

```
for (ifade 1; ifade 2; ifade 3) {
    // Çalıştırılacak kod bloğu
}

//  ifade1: Kod bloğu başlamadan önce döngünün başlama değerini belirtir.
//  ifade2: Kod bloğunun çalışma koşulunu tanımlar.
//  ifade3: Kod bloğu çalıştırıldıktan sonra her defasında çalıştırılacak kodu tanımlar.
```

1-100 arasındaki sayıların 3'e ya da 5'e bölünenlerini yazdıran javascript kodlaması

```
<!DOCTYPE html>
<html>
<style>
    body    {font-family:Verdana;font-size:14px;}
    #demo   {padding:15px; margin:10px; background-color:#ffcc00;}
</style>
<body>
    <h2>For Döngüsü Örneği</h2>
    <p id="demo"></p>

    <script>
        // var ile bir i değişkeni tanımlıyoruz.
        var i;

        // i=1 den i<=100 (100 dahil) olana kadar her seferinde for içerisindeki kod buloğunu çalıştır.
        // (i++) Kod bloğu her seferinde çalıştırıldığında ise i değişkenini son değerinden 1 artır.
        for(i=1; i<=100; i++)
        {
            // i%3: i değerinin 3 ile bölümünden kalanı verir.
            // i%5: i değerinin 5 ile bölümünden kalanı verir.
            // || (veya işareti) : koşullardan herhangi birinin sağlanması durumunda kod bloğu çalıştırılır.

            if(i%3==0 || i%5==0){
                // (+=)         : bir önceki değerin üzerine yeni değerin eklenmesini sağlar.
                // (i + ", ")   : i değerinden sonra ", " string değerini ekler.
                document.getElementById('demo').innerHTML += i + ", ";

            }
        }

    </script>
</body>
</html>
```

For Döngüsü Örneği

3, 5, 6, 9, 10, 12, 15, 18, 20, 21, 24, 25, 27, 30, 33, 35, 36, 39, 40, 42, 45, 48, 50, 51,
54, 55, 57, 60, 63, 65, 66, 69, 70, 72, 75, 78, 80, 81, 84, 85, 87, 90, 93, 95, 96, 99, 100,

While Döngüsü

```
while (koşul) {
    // Çalıştırılacak kod bloğu
}
//  koşul: belirtilen koşul doğru olduğu sürece kod bloğu çalıştırılır.
```

1 den n' e kadar olan sayıların toplamı

```
<!DOCTYPE html>
<html>
<style>
    body    {font-family:Verdana;font-size:14px;}
    #demo1, #demo2  {padding:15px; margin:10px; background-color:#f2f2f2;}
</style>
<body>
    <h2>While Döngüsü Örneği</h2>
    <p id="demo1"></p>
    <p id="demo2"></p>

    <script>
        // var ile kullanacağımız değişkenleri tanımlıyoruz.
        var i, n, tektoplam, cifttoplam;

        // değişkenlerin başlangıç değerini atıyoruz.
        i=1;    tektoplam=0; cifttoplam=0;

        // kullanıcıdan bir veri girmesini istiyoruz.
        n= prompt("Bir sayı griniz.");

        // i <= n (ya da i<n+1) : 1, n+1 den küçük olana kadar
        //her seferinde kod buloğunu çalıştır.
        while(i<=n)
        {
            // i nin ikiye bölümünden kalanın sıfır olup olmadığını kontrol et.
            if (i%2==0){    // eğer sıfırsa (yani çift sayı)
                // cifttoplam değişkenine i kadar toplam sayı ekle
                cifttoplam += i;
            } else {    // eğer sıfır değilse (yani tek sayı)
                // tektoplam değişkenine i kadar toplam sayı ekle
                tektoplam += i;
                }
            i++; // i yi en son değerinden 1 artır.
        }


        document.getElementById('demo1').innerHTML+=
        "1 den "+ n +" e kadar olan <mark>çift</mark> sayıların toplamı: <mark>"+ cifttoplam + "
</mark>";

        document.getElementById('demo2').innerHTML+="1 den "+ n +" e kadar olan <mark
style=background-color:orange;>tek</mark> sayıların toplamı: <mark style=background-
color:orange;>"+ tektoplam + "</mark>";

    </script>
</body>
</html>
```

Dikkat!

```
// ----String Birleştirme İşlemi---------

document.getElementById('demo1').innerHTML
+=
"1 den "+ n +" e kadar olan <mark>çift</mark> sayıların toplamı: <mark>"+ cifttoplam + "</mark>";
```

Javascript' de herhangi bir değişkenin değerini sadece metin, sadece sayı atanabileceği gibi HTML ya da CSS kodları olarak da atanabilir. Yukarıdaki kodlama bir string birleştirme işlemidir. Burada kullanıcının girmiş olduğu n sayısı ile toplam sayıları bir anlam ifade edebilecek şekilde birleştirilmiştir. Javascript, bu kodları web sayfası içerisine sadece yazmakla görevlidir. Bu kodun nasıl görüntüleneceği ise daha önceden de değinildiği üzere tarayıcının görevidir.

Do While Döngüsü

```
do {
    // Çalıştırılacak kod bloğu
} while (koşul)
//  koşul: belirtilen koşul doğru olduğu sürece kod bloğu çalıştırılır.

//  "do while" döngüsünün while döngüsünden farkı
//  ---------------------------------------------
//  koşulun doğru olup olmadığını kontrol etmeden önce kod bloğunu bir kez çalıştırmasıdır.
```

Kodlayıcı Görevi

- do while döngüsünü kullanarak kullanıcının girmiş olduğu sayı kadar rastgele sayı üreten bir javascirpt kodlama yapınız.
- İpucu
  - Math.random()
  - Math.floor()

FONKSİYON YAZIMI

Fonksiyonlar belirli bir görevi gerçekleştirmek için tasarlanmış kod bloklarıdır. Herhangi bir eylem sonucunda çağrıldıklarında yürütülmeye başlarlar.

Fonksiyonlarla ilgili daha önceden siteyi ziyaret eden bir kullanıcının bir butona tıklamasıyla elementlerin niteliğini değiştirebilen bir kodlama yapılmıştı. Bu senaryo üzerindeki kodlama iki bölüme ayrılabilir.

I. Elementlerin niteliğini değiştiren kod bloğu

```
<script>
    function Degistir() {

        document. // sayfa içerisindeki
        getElementsByTagName('INPUT')['degistirilecekElement'].
        // id'si "degistirilecekElement" olan input elementinin
        setAttribute('type', 'button');
        // type niteliğini button olarak değiştir.

        document.   //sayfa içerisindeki
        getElementsByTagName('INPUT')['degistirilecekElement'].
        //id'si "degistirilecekElement" olan input elementinin
        setAttribute('value', 'Bu bir buton elementi');
        //value niteliğini "Bu bir buton elementi" olarak değiştir.
    }
</script>
```

II. Kullanıcı bir butona tıkladığında elementlerin niteliğini değiştiren kod bloğunun çağrılması

```
<button onclick="Degistir();">Değiştir</button>
```

Şimdi fonksiyon kullanım yöntemlerini karşılaştırmalı olarak inceleyelim.

```
// Yöntem 1
// Basit Fonksiyon Kullanımı
function fonksiyonAdi1() {
    // fonksiyon çağrıldığında çalıştırılacak kod bloğu
}

// Yöntem 2
// Fonksiyonun kod bloğunun çalıştırılabilmesi için
// dışarıdan değişken gönderilmesi gereken
// fonksiyon kullanımı
function fonksiyonAdi2(a, b, ...) {
    // fonksiyon çağrıldığında çalıştırılacak kod bloğu
    // örneğin a ile b nin toplamını hesaplama
}

// Yöntem 3 (Bu yöntem yukarıdaki iki yöntem için de kullanılabilir)
// Fonksiyonun kod bloğunu çalıştırıldıktan sonra
// bir değer döndüren fonksiyon kullanımı
function fonksiyonAdi3(...){
    // fonksiyon çağrıldığında çalıştırılacak kod bloğu
    return geriDondurulecekDeger;
}
```

Uygulama

Aşağıda kullanıcının doğum tarihine göre yaşını hesaplamayan bir kodlama örneği paylaşılmıştır.

1- Kullanıcının doğum tarihini girebileceği bir sayfa tasarımının yapılması

2- Kullanıcının doğum tarihine göre yaşını hesaplayan fonksiyonun yazılması

```
// Yöntem 1 ile kullanıcının yaşını hesaplama
function Hesapla(){

    // id'si yil olan bir input nesnesinden kullanıcının doğum tarihini alma
    var dogumtarihi = document.getElementById('yil').value;

    // güncel tarihi alma
    var gunceltarih = new Date().getFullYear();

    // yaşı hesaplama
    var yas = gunceltarih-dogumtarihi;

    //yaşı hesapladıktan sonra kullanıcıya gösteren fonksiyonu çağırma
    Goster(yas);
}
```

3- Kullanıcının yaşını gösteren fonksiyonun yazılması

```
// Yöntem 2 ile kullanıcının yaşını gösterme
function Goster(a){
    // id'si mesaj olan elementin içerisinde yaşı kullanıcıya gösterme
    document.getElementById('mesaj').innerHTML= "Yaşınız: "+ a;
}
```

4- Javascirpt Kodları ile HTML kodlarının birleştirilmesi

```
<!DOCTYPE html>
<html>
<style>
    body {font-family:Verdana;font-size:14px;}
    h2 {padding:5px;background-color:orange;}
    dogumtarihi {
        background-color:#f3f3f3;padding:30px;display:block;
        }
    mesaj {
        background-color:#f4f4f4;padding:30px;display:block;
        margin-top:30px;color: orange;visibility:hidden;
        }
</style>
<body>
  <h2>Fonksiyon Kullanım Örneği</h2>
  <div>
      <dogumtarihi>
          <input id=yil type=text value='Doğum Tarihini Giriniz'>
          <input type=submit value='Hesapla' onclick="Hesapla();">
      </dogumtarihi>
      <mesaj id="mesaj">

      </mesaj>
  </div>

  <script>
      // Yöntem 1 ile kullanıcının yaşını hesaplama
      function Hesapla(){

          // id'si yil olan bir input nesnesinden kullanıcının doğum tarihini alma
          var dogumtarihi = document.getElementById('yil').value;

          // güncel tarihi alma
          var gunceltarih = new Date().getFullYear();

          // yaşı hesaplama
          var yas = gunceltarih-dogumtarihi;

          //yaşı hesapladıktan sonra kullanıcıya gösteren fonksiyonu çağırma
          Goster(yas);
      }

      // Yöntem 2 ile kullanıcının yaşını gösterme
      function Goster(a){
          // id'si mesaj olan elementin içerisinde yaşı kullanıcıya gösterme
          document.getElementById('mesaj').innerHTML= "Yaşınız: "+ a;
          document.getElementById('mesaj').style.visibility='visible';
      }
  </script>
</body>
</html>
```

Bu örnekte, Yöntem 1 ve Yöntem 2' ye göre fonksiyon kullanımları birleştirilmiştir. Şimdi, örnek Yöntem 3 ' e göre değiştirilecektir.

```
<script>
  // Yöntem 1 ile Kullanıcının Yaşını Gösterme
    function Goster(){

        // id'si yil olan elementin değerini alma.
        var dogumtarihi = document.getElementById('yil').value;

        // id'si mesaj olan elementin içerisinde yaşı kullanıcıya gösterme
        document.getElementById('mesaj').innerHTML= "Yaşınız: "+ Hesapla(dogumtarihi);
        document.getElementById('mesaj').style.visibility='visible';
    }
   // Yöntem 3 ile Kullanıcının yaşını hesaplayıp geri döndürme
    function Hesapla(a){

        // güncel tarihi alma
        var gunceltarih = new Date().getFullYear();

        // fonksiyon içerisine gönderilen değeri kullanarak yaşı hesaplama
        var yas = gunceltarih-a;

        //yaşı hesapladıktan sonra, fonksiyonun çağrıldığı yere gönderme
        return yas;
    }
</script>
```

Kodlayıcı Görevi

- Yukarıdaki örnekte verilen senaryoya göre Yöntem 1 ve Yöntem 3 kullanıldı. Peki sizce kullanıcı ile etkileşim sırasında ilk olarak hangi fonksiyonu çağırılmalıdır?

```
<dogumtarihi>
      <input id=yil type=text value='Doğum Tarihini Giriniz'>
      <input type=submit value='Hesapla' onclick=" ? ">
</dogumtarihi>
```

DİZİLER

Diziler, birden çok değeri tek bir değişkende depolamak için kullanılır. Önceki örneklerde, her bir değer için yeni değişken tanımlamaları yapıldı. Örneğin:

```
var oto1=   "Volvo";
var oto2=   "Opel";
var oto3=   "Ford";
var takim1= "Çorumspor";
var takim2= "Muğlaspor";
var takim3= "Gençlerbirliği";
```

Bu türden bir değişken tanımlama yerine, benzer niteliklere sahip olan değişkenler bir dizi değişkende toplanabilir ve daha kolay yönetilebilir. Örneğin:

```
var otolar= ["Volvo", "Opel", "Ford"];
var takimlar=   ["Çorumspor", "Muğlaspor", "Gençlerbirliği"];
```

ya da

```
var otolar  =   new Array("Volvo", "Opel", "Ford");
var takimlar=   new Array("Çorumspor", "Muğlaspor", "Gençlerbirliği");
```

Yukarıdaki iki örnek de aynı şeyi yapmaktadır. Basitlik, okunabilirlik ve kodun çalıştırılma hızı için birincisinin kullanılması tavsiye edilmektedir. Peki, dizi içerisindeki bir öğeye (ör: Muğlaspor) nasıl ulaşılabilir?

Diziler, içerisindeki öğelere erişmek için sayıları kullanır. Bu sayılar 0' dan başlayarak dizinin içerisindeki öğe sayısından 1 eksik değer arasında olabilir. Örneğin takımlar dizisi 3 öğe içermektedir. Burada "Muğlaspor" değerini ekrana yazdırmak için;

```
document.getElementById("demo").innerHTML = takimlar[1];
```

kodlaması kullanılabilir. Dizinin tüm öğelerini ekrana yazdırmak için ise,

```
document.getElementById('mesaj').innerHTML = takimlar;
```

Uygulama

Şimdi bir senaryo temelinde yukardaki örnekler kodlanacaktır. Kullanıcının girmiş olduğu değere göre dizinin elemanını ekranda gösteren bir kodlama aşağıdaki şekilde olabilir.

1- Kullanıcının veri giriş yapabilmesi için HTML kodları

```
<dizi>
    <input id=eleman type=text value='Dizinin kaçıncı elemanı'>
    <input type=submit value='Göster' onclick="Goster(document.getElementById('eleman').value);">
</dizi>
```

2- Kullanıcının girmiş olduğu değere göre dizinin öğesini gösterme

```
<script>
    function Goster(a){
          var otolar= ["Volvo", "Opel", "Ford"];
          var takimlar=   ["Çorumspor", "Muğlaspor", "Gençlerbirliği"];
          document.getElementById('mesaj').innerHTML = takimlar[a];
      }
</script>
```

Ekran Çıktısı

Kodlayıcı Görevi

- Kullanıcı herhangi bir değer girmez ya da dizinin öğe sayısı aralığında olmayan (0' dan küçük ya da dizi uzunluğundan 1 eksik değerden büyük) bir değer girerse ne olur? Test ediniz?

Hatırlanacağı üzere, bu gibi durumlar için bir mantıksal sınama yapılması gerektiği daha önceden öğrenilmişti. Buna göre kullanıcı herhangi bir değer girmez ya da dizinin öğe sayısı aralığında olmayan bir değer girdiğinde dizinin tüm öğelerini ekrana yazan bir kodlama yapalım.

```
<script>
   function Goster(a){
      var takimlar=   ["Çorumspor", "Muğlaspor", "Gençlerbirliği"];
      if (a==0 || a==1 || a==2)
      {
          document.getElementById('mesaj').innerHTML = takimlar[a];
          } else {
                  document.getElementById('mesaj').innerHTML = takimlar;
              }
      }
</script>
```

Ekran Çıktısı

BASİT DİZİ İŞLEMLERİ

Önceki örneklerde sabit bir dizi tanımlayarak, dizi içerisindeki öğelere erişilmeye çalışıldı. Şimdi ise, diziye öğe eklemek, silmek, değiştirmek gibi işlemler görülecektir.

1- Dizinin öğelerini değiştirme

En basit şekliyle dizinin herhangi bir öğesi aşağıdaki gibi değiştirilebilir.

```
<script>
    var meyveler = ["Muz", "Portakal", "Elma"];

    meyveler[0] = "Kiwi";
    meyveler[1] = "Kiraz";
    meyveler[2] = "Vişne";

</script>
```

2- Kullanıcının seçmiş olduğu değeri dizinin başına ya da sonuna ekleme ve dizinin başından ya da sonundan veri silme

HTML kodları

```
<basitdizi>
  <select id="oto">
      <option value="Anadol">Anadol</option>
      <option value="Mercedes">Mercedes</option>
      <option value="Saab">Saab</option>
      <option value="Audi">Audi</option>
      <option value="Tata">Tata</option>
      <option value="Subaru">Subaru</option>
  </select>
  <button id="Ekle" onclick="Ekle(document.getElementById('oto').value);">Ekle</button>
  <button id="Sil" onclick="Sil();">Sil</button>
</basitdizi>
```

Javascript Kodları

```
<script>
  var otolar =  ["Volvo", "Opel", "Ford"];
  function diziGoster(){
      document.getElementById('demo').innerHTML= otolar;
    }
  function Ekle(a) {
      otolar.push(a); //gönderilen değeri dizinin sonuna ekler.
      otolar.unshift(a); //gönderilen değeri dizinin başına ekler.
      diziGoster();
    }
  function Sil(){
      otolar.pop();   //dizinin sonuncu öğesini siler.
      otolar.shift(); //dizinin birinci ([0]) öğesini siler.
      diziGoster();
    }
</script>
```

3- Dizinin Herhangi Bir Yerine (ör: 3. Ögesinden sonra) Yeni Bir Öğe Ekleme ya da Herhangi Bir Öğesini Değiştirme

HTML kodları

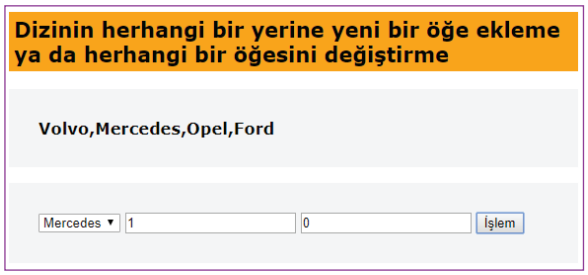
```
<basitdizi>
  <select id="oto">
      <option value="Anadol">Anadol</option>
      <option value="Mercedes">Mercedes</option>
      <option value="Saab">Saab</option>
      <option value="Audi">Audi</option>
      <option value="Tata">Tata</option>
      <option value="Subaru">Subaru</option>
  </select>
  <input id="kacinci" value="Kaçıncıdan itibaren">
  <input id="degistirilecek" value="Değiştirilecek Öğe">
  <button id="islem" onclick="diziIslem();">İşlem</button>
</basitdizi>
```

Javascript Kodları

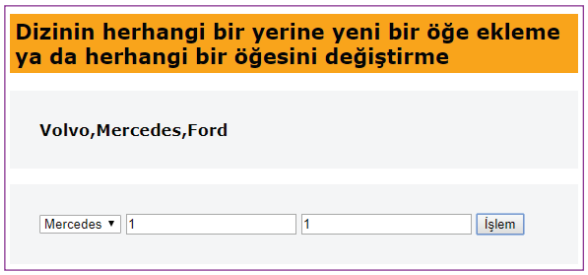
```
<script>
  var otolar =  ["Volvo", "Opel", "Ford"];
  function Ekle(a, b, c) {
      otolar.splice(a,b,c);
      // b=0 geleceği için bu bir ekleme işlemidir.
      document.getElementById('demo').innerHTML= otolar;
    }
  function Degistir(a, b, c){
      otolar.splice(a,b,c);
      // b 0 dan farklı bir değer geleceği için bu bir değiştirme işlemidir.
      document.getElementById('demo').innerHTML= otolar;
    }
  function diziIslem(){
      var yenioge = document.getElementById('oto').value;
      var kacinci = document.getElementById('kacinci').value;
      var degistirilecek = document.getElementById('degistirilecek').value;
      if (degistirilecek == 0) {
          Ekle(kacinci, degistirilecek, yenioge);
      } else {
          Degistir(kacinci, degistirilecek, yenioge);
      }
    }

</script>
```

Ekran Çıktısı

Ekleme



Değiştirme



4. Dizinin Ögelerini Sıralama

Programlama dilleri ile iki tür sıralama yapılabilir. İlk olarak harf, metin ya da sayıları alfabetik sıralayabilir (Ör: 0, 1, 10, 2, 20, a, ab, b, bd). İkinci olarak ise sayıları numerik olarak sıralayabilir (Ör: 0, 1, 2, 10, 20). Javascriptle de dizi içerisindeki öğeleri alfabetik ya da numerik şekilde sıralayabilirsiniz. Bu işlem için .sort() komutu kullanılır. sort(), varsayılan olarak değerleri metin olarak sıralar. "Ankara", "10" ve "?" öğelerini kıyaslandığında, rakamlar harflerden daha önce geldiği için 10 ilk sırada olmalıdır. "a" harfi "m" den önce geldiği için de dizinin sıralaması 10, Ankara, Muğla olur. Sayılar metin hâlinde sıralanırsa "25", "100"den büyüktür, çünkü "2", "1"den büyüktür.

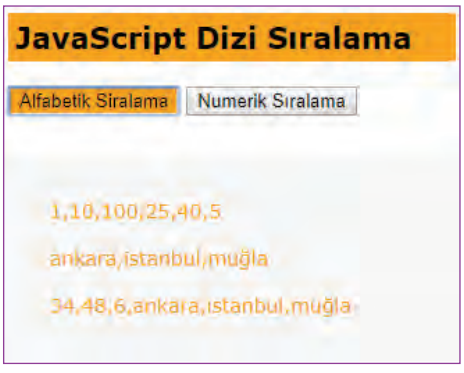
HTML Kodları

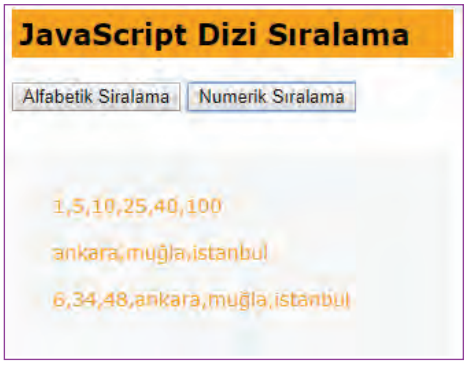
```
<style>
    body {font-family:Verdana;font-size:14px;}
    h2 {padding:5px;background-color:orange;}
    h3, basitdizi {
        background-color:#f3f3f3;padding:30px;display:block;
        }
    .demo {
        background-color:#f4f4f4;padding:30px;display:block;
        margin-top:30px;color: orange;
        }
    button:hover{background-color:orange;
    }
</style>

<h2>JavaScript Dizi Sıralama</h2>

<button onclick="AlfabetikSiralama()">Alfabetik Siralama</button>
<button onclick="NumerikSıralama()">Numerik Sıralama</button>

<p id="demo" class="demo"></p>
```

Javascript Kodları

```
<script>
    //Sadece sayılardan (int) oluşan dizi
    var sayiIcerenDizi = [40, 100, 1, 5, 25, 10];
    //Sadece metinden (string) oluşan dizi
    var metinIcerenDizi = ["ankara", "muğla", "istanbul"];
    //Hem sayı hem de metinden oluşan dizi
    var harfMetinVeSayiIcerenDizi = [06, 48, 34, "ankara", "muğla", "istanbul"];
    //Dizileri ekrana yazdıran fonksiyon
    function Goster() {
        document.getElementById("demo").innerHTML = sayiIcerenDizi + "<br/><br/>";
        document.getElementById("demo").innerHTML += metinIcerenDizi + "<br/><br/>";
        document.getElementById("demo").innerHTML += harfMetinVeSayiIcerenDizi;
    }

    //sayfa ilk yüklendiğinde dizilerin ilk halini göstermek için fonksiyonu çağırıyoruz.
    Goster();

    function AlfabetikSiralama() {
        //sort() dizi öğelerinin alfabetik sıralanmasını sağlar.
        sayiIcerenDizi.sort();
        metinIcerenDizi.sort();
        harfMetinVeSayiIcerenDizi.sort();
        //dizi öğelerinin sıralnmış hallerini göstermek için fonksiyonu çağırıyoruz.
        Goster();
    }
    function NumerikSıralama() {
        // dizi öğelerinin numerik sıralanması için
        // sort() kodunu farklı bir biçimde kullanıyoruz.

        sayiIcerenDizi.sort(function(a, b){return a - b});
        metinIcerenDizi.sort(function(a, b){return a - b});
        harfMetinVeSayiIcerenDizi.sort(function(a, b){return a - b});

        // sort(function(a, b){return a - b}) kodu
        // her dizi öğesini iikili olarak karşılaştırır.
        // eğer küçükten büyüğe sıralamak istiyorsak a-b
        // büyükten küçüğe sıralamak istiyorsak b-a döndürülür.

        Goster();
    }
</script>
```

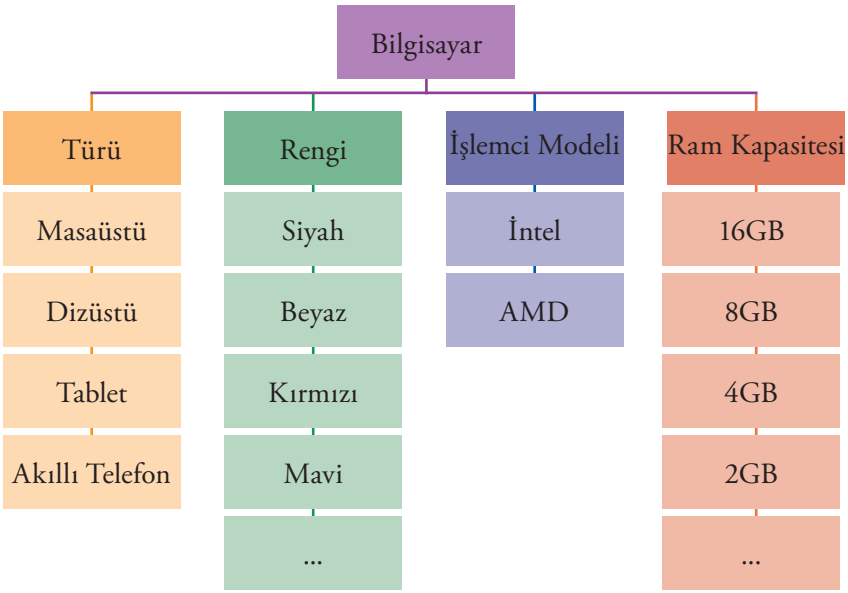
Ekran Çıktıları





NESNELER

Gerçek yaşam içerisinde herhangi bir nesne düşününüz. Örneğin bir bilgisayarı ele alalım. Genel bir nesne ismini içeriyor değil mi? Bir diğer ifadeyle bir bilgisayar tür, renk, işlemci, ram gibi özelliklerin alt özelliklerinin birleşiminden doğan bir varlığı simgeliyor.



Şekil 9: Bilgisayar Nesnesinin Özellikleri

Benzer şekilde kişileri, arabaları nesneleştirmek mümkündür. Peki bu nesne yapısı kodlama yapmak ne işe yarar?

Dizileri kullanarak benzer niteliklere sahip olan değişkenler tek bir değişkende toplanabilir.

```
var bilgisayarlar = ["Asus","Casper","Dell","IBM","Lenova","Vestel"]
```

Peki, şekil 9 'daki bilgisayar özellikleri bir değişkende nasıl tutulabilir? İşte bu durumda nesne temelli değişken tanımlanabilir. Javascript'te bu işlem aşağıdaki gibidir:

```
var bilgisayar = { turu:"Dizüstü", marka:"Asus", rengi:"siyah", islemci:"AMD", ram:"4GB" };
```

Dizi içerisindeki herhangi bir öğeye erişirken dizi sıra numarası (Ör: dizi [0]) kullanılırdı. Nesne içerisindeki bir öğeye erişirken de, "nesne" . "özelliği" şeklinde erişilebilir.

```
var bilgisayar = { turu:"Dizüstü", marka:"Asus", rengi:"siyah", islemci:"AMD", ram:"4GB" };

document.getElementById("demo").innerHTML = bilgisayar.turu + "|" + bilgisayar.marka;
```

Uygulama

Kullanıcının girmiş olduğu bilgisayar özelliklerini bir nesneye aktaran ve bu nesneleri bir dizi içerisinde tutup girilen bilgisayar özelliklerini bir tabloda gösteren sayfa kodlamasını yapınız.

1. Kullanıcının Girmiş Olduğu Bilgisayar Özelliklerini Bir Nesneye Aktarma

HTML Kodu

```
<h2>Dizi ve Nesne Kullanımı</h2>
<div>
  <nesne>
    Türü:
    <select id="tur">
        <option value="Dizüstü">Dizüstü</option>
        <option value="Masaüstü">Masaüstü</option>
        <option value="Tablet">Tablet</option>
        <option value="Akıllı Telefon">Akıllı Telefon</option>
    </select>
    <br/><br/>
    Markası:
    <select id="marka">
        <option value="Asus">Asus</option>
        <option value="Casper">Casper</option>
        <option value="Dell">Dell</option>
        <option value="IBM">IBM</option>
        <option value="Lenova">Lenova</option>
        <option value="Vestel">Vestel</option>
    </select>
    <br/><br/>
    Rengi:
    <input id="renk" type=text>
    <br/><br/>
    İşlemci Modeli:
    <select id="islemci">
        <option value="Intel">Anadol</option>
        <option value="AMD">Mercedes</option>
    </select>
    <br/><br/>
    RAM Kapasitesi:
    <input id="RAM"type=text>
    <br/><br/>
    <button id="Ekle" onclick="Ekle();">Ekle</button>
    <button id="Sil" onclick="Sil();">Sil</button>
  </nesne>
</div>
<div id="tablo"><!-- Nesneleri burada tablo şeklinde göstereceğiz. --></div>
```

Javascript Kodu

```
function Ekle() {
    // Özellikleri atanmamış bir nesne tanımladık
    var bilgisayar = { tur:" ", marka:" ", islemci:" ", ram:" ", fiyat:" "};

    // Daha sonra kullanıcının girmiş olduğu verileri atıyoruz.
    bilgisayar.tur = document.getElementById('tur').value;
    bilgisayar.marka = document.getElementById('marka').value;
    bilgisayar.islemci = document.getElementById('islemci').value;
    bilgisayar.ram = document.getElementById('ram').value;
    bilgisayar.fiyat = document.getElementById('fiyat').value;
    }
```

2. Oluşturulan Nesneleri Bir Dizi İçerisinde Tutma

```
var dizi = []; //içi boş bir dizi tanımladık.
function Ekle() {
    // Özellikleri atanmamış bir nesne tanımladık
    var bilgisayar = { tur:" ", marka:" ", islemci:" ", ram:" ", fiyat:" "};

    // Daha sonra kullanıcının girmiş olduğu verileri atıyoruz.
    bilgisayar.tur = document.getElementById('tur').value;
    bilgisayar.marka = document.getElementById('marka').value;
    bilgisayar.islemci = document.getElementById('islemci').value;
    bilgisayar.ram = document.getElementById('ram').value;
    bilgisayar.fiyat = document.getElementById('fiyat').value;

    // Kullanıcının girmiş olduğu verileri tutan nesneyi diziye ekliyoruz.
    dizi.push(bilgisayar);
    }
```

3. Dizi İçerisindeki Nesneleri Bir Tabloda Gösterme

Yeni CSS Kodları

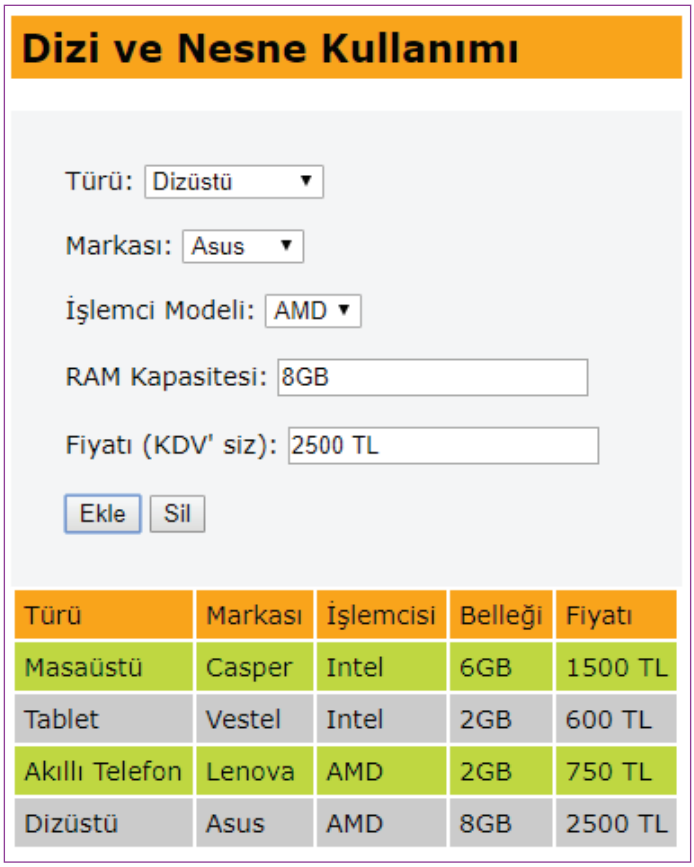
```
/*tablonun tek (1,3,5,7) satırlarının arkaplan rengini ayarladık*/
tr:nth-child(odd) {background: #CCC;}

/*tablonun çift (2,4,6,8) satırlarının arkaplan rengini ayarladık*/
tr:nth-child(even) {background: #bff442;}

/*tablonun ilk satırının (başlık) arkaplan rengini ayarladık*/
tr:first-child {background: #ffa500;}

/*tablonun satırları içerisindeki hücrelerin iç boşluğunu ayarladık*/
td {padding:5px; }
```

Javascript Kodu

```
var dizi = []; // içi boş bir dizi tanımladık.
// tablonun HTML kodlarını tutacağımız bir değişken tanımladık.
var tabloKodu;

function diziGoster(){
    // ilk olarak bir tablo etiketi açtık.
    tabloKodu = "<table>";

    // tablonun ilk satırını (başlıklar) oluşturduk.
    tabloKodu += "<tr>";
    tabloKodu += "<td>Türü</td>";
    tabloKodu += "<td>Markası</td>";
    tabloKodu += "<td>İşlemcisi</td>";
    tabloKodu += "<td>Belleği</td>";
    tabloKodu += "<td>Fiyatı</td>";
    tabloKodu += "</tr>";

    // dizinin içerisinde ne kadar nesne öğesi varsa,
    for (var i = 0; i< dizi.length; i++)
    {
        // o kadar yeni satır ekliyoruz.
        tabloKodu += "<tr>";

        // her yeni satırda
        // dizi içerisindeki nesnenin özellikleri
        // ayrı ayrı sütunlarda gösterilecek.
        tabloKodu += "<td>"+dizi[i].tur+"</td>";
        tabloKodu += "<td>"+dizi[i].marka+"</td>";
        tabloKodu += "<td>"+dizi[i].islemci+"</td>";
        tabloKodu += "<td>"+dizi[i].ram+"</td>";
        tabloKodu += "<td>"+dizi[i].fiyat+"</td>";

        // açtığımız yeni satır etiketini kapatıyoruz.
        tabloKodu += "</tr>";
    }

    // kodlamanın başında açtığımız tablo etiketini kapattık.
    tabloKodu += "</table>";

    // son olarak, oluşturmuş olduğumuz HTML' yi kullanıcıya göstermek için
    // id'si tablo olan elementin içerisine tabloKodu değişkenini ekledik.
    document.getElementById('tablo').innerHTML = tabloKodu;
}
```

```
function Ekle() {
    // Özellikleri atanmamış bir nesne tanımladık
    var bilgisayar = { tur:" ", marka:" ", islemci:" ", ram:" ", fiyat:" "};

    // Daha sonra kullanıcının girmiş olduğu verileri atıyoruz.
    bilgisayar.tur = document.getElementById('tur').value;
    bilgisayar.marka = document.getElementById('marka').value;
    bilgisayar.islemci = document.getElementById('islemci').value;
    bilgisayar.ram = document.getElementById('ram').value;
    bilgisayar.fiyat = document.getElementById('fiyat').value;

    // Kullanıcının girmiş olduğu verileri tutan nesneyi diziye ekliyoruz.
    dizi.push(bilgisayar);

    // Her yeni bir nesne eklendiğinde, dizi içerisindeki nesneleri
    // ekrana yazdıracak fonksiyonu çağırıyoruz.
    diziGoster();
    }
function Sil() {
    dizi.pop();
    diziGoster();
}
```

Ekran Çıktısı



5. VERİ TABANI YÖNETİMİ

5. VERİ TABANI YÖNETİMİ

Veri tabanı yönetimine geçmeden önce, bir önceki ünitede Javascript ile kullanıcının girmiş olduğu bilgisayar özelliklerini ekranda gösteren uygulamayı biraz tartışalım.

Türü	Markası	İşlemcisi	Belleği	Fiyatı
Masaüstü	Casper	Intel	6GB	1500 TL
Tablet	Vestel	Intel	2GB	600 TL
Akıllı Telefon	Lenova	AMD	2GB	750 TL
Dizüstü	Asus	AMD	8GB	2500 TL

Kullanıcı sayfaya bağlandığında, yukarıdaki gibi bilgisayarlar özelliklerini girmiş olsun. Sonrasında tarayıcısını açıp kapattığında bu girilen bilgilere ulaşılabilir mi?

Şu ana kadar yazılan kodlar buna izin vermez. Peki girilen bilgilere sonradan ulaşabilmek için ne yapılabilir? Bunun web programlamada temel olarak iki yöntemi var:

- Birincisi istemci tarafında cookie (çerez) yönetimi
- İkincisi ise sunucu tarafında veri tabanı yönetimi

Bu bölümde, kullanıcı tarafından girilen verilerin tutulması için sunucu tarafında veritabanı yönetim sistemlerinin kullanımı üzerinde durulacak. Eğer Microsoft Excel ve Microsoft Access'in kullanımı konusunda bilgi sahibiyseniz bu ünite kazanımlarını daha kolay kavrayıp uygulayabilirsiniz. Bununla birlikte, daha önce bu programları hiç kullanmadıysanız bile kitaptaki uygulama örnekleri ile veri tabanı yönetiminin nasıl yapıldığı ile ilgili bilgi düzeyinizi artırabilirsiniz.

VERİ TABANI

Veri tabanı; kolayca erişilebilecek, yönetilebilecek ve güncellenebilecek şekilde düzenlenmiş bir bilgi topluluğudur. Veriler, satırlar, sütunlar ve tablolar hâlinde organize edilir ve ilgili bilgileri bulmayı kolaylaştırmak için kayıt hâlinde eklenir. Örneğin okullarınızda kullanılan e-Okul Web Sitesi sizinle ilgili tüm bilgileri bir veri tabanında tutmaktadır. Bunun yanı sıra hastanelerdeki hasta, doktor, tedavi, tetkik bilgileri, bankalardaki; müşteri, mevduat, kredi bilgilerinin hepsi bir veri tabanında tutulmaktadır.

Veri tabanı sistemleri, veri tekrarlarını ortadan kaldırarak veri bütünlüğünün sağlanmasına ve verilerin bir düzen içinde tutulmasına olanak sağlar. Kodlayıcıların kullanabileceği birçok veri tabanı sistemi vardır. Bunlar yandaki şekilde gösterilmektedir. Bu ünitede, ücretsiz bir veri tabanı yönetim sistemi olan MariaDB yönetimi ele alınacaktır.

Veritabanı Sistemleri

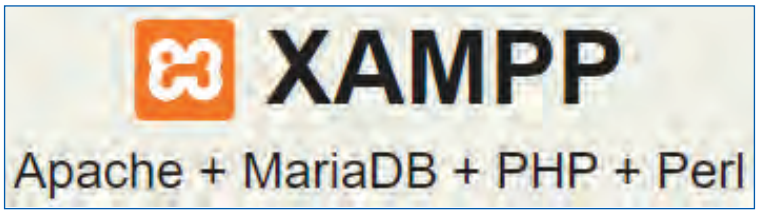
- MySQL
- MariaDB
- Microsoft SQl
- Microsoft Access
- PostgreSQL
- Oracle
- Sybase
- Berkeley

MariaDB, MySQL'in kaynak kodundan türemiş, ücretsiz olarak kullanılabilen ilişkisel veri tabanı sistemidir. İlişkisel veri tabanı ise veri tabanı içerisinde birçok tablo kullanıldığı ve tablolar arasında ilişkiler kurulduğu bir yapıdır.

Tablolar arasındaki bir ilişki bir tabloya, başka bir tablodaki kaydı bağlamayı sağlar. Bu şekilde veriler daha az yer kaplar ve işlemleri kolaylaştırır.

XAMPP KURULUMU

XAMPP ücretsizdir, yüklenmesi kolaydır. İçerisindeki PhpMyAdmin paneli MariaDB veri tabanının yönetimini sağlar.



PhpMyAdmin, MySQL veri tabanı ve türevi olan MariaDB yönetimi için en popüler uygulamalardan biridir. PHP ile yazılmış ücretsiz bir araçtır. Yandaki QR Kodu ya da dipnottaki adresi kullanarak kullanmış olduğunuz işletim sistemine uygun olan yazılımı indirebilirsiniz.

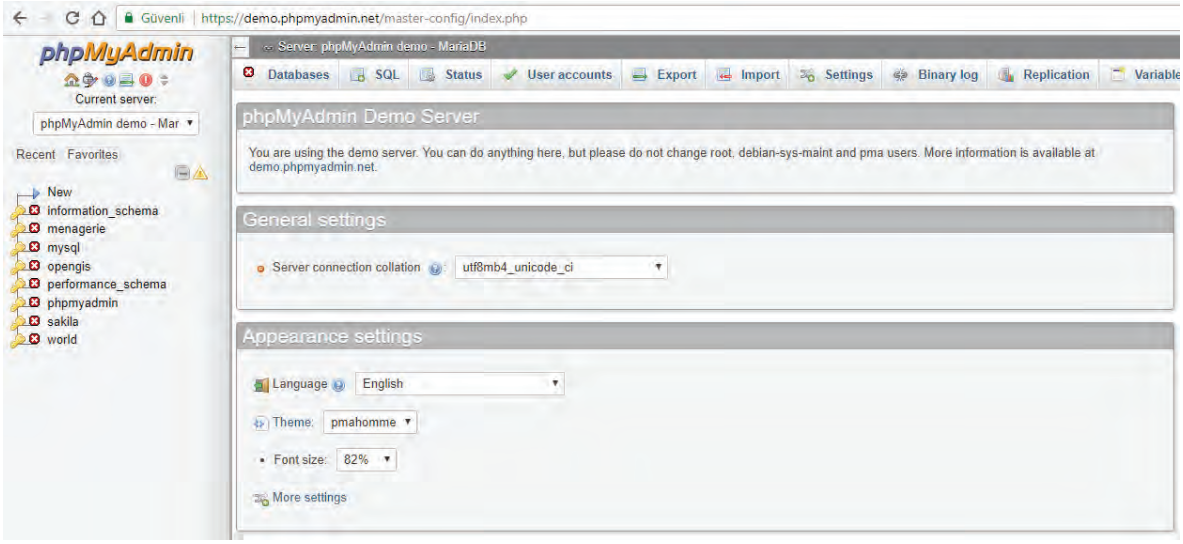
XAMPP kurulumunun nasıl yapılacağı ile ilgili internette, "XAMPP Kurulumu" anahtar kelimeleri ile bir arama yaptığınızda, birçok görsel anlatımlı metin ya da video bulabilirsiniz. Bunun dışında, yandaki QR kodunu kullanarak ya da dipnottaki web adresini ziyaret ederek yazar tarafından seçilmiş video anlatımını kullanabilirsiniz.

Bilgisayarınıza XAMPP kurduğunuzda, kurulum yapmış olduğunuz bilgisayarı basit bir web sunucusuna dönüştürmüş olursunuz. Daha önceden bilgisayarınız bir web sitesine bağlanırken ve istemci olarak çalışırken artık sunucu olarak da çalışabilir. Daha açık bir ifadeyle kendi bilgisayarınızda oluşturmuş olduğunuz bir web sitesinin kaynaklarına yine kendi bilgisayarınızın tarayıcısını kullanarak erişebilirsiniz. Bir sonraki ünitede bu konu daha detaylı incelenecektir.

MARİADB VERİ TABANI YÖNETİMİ

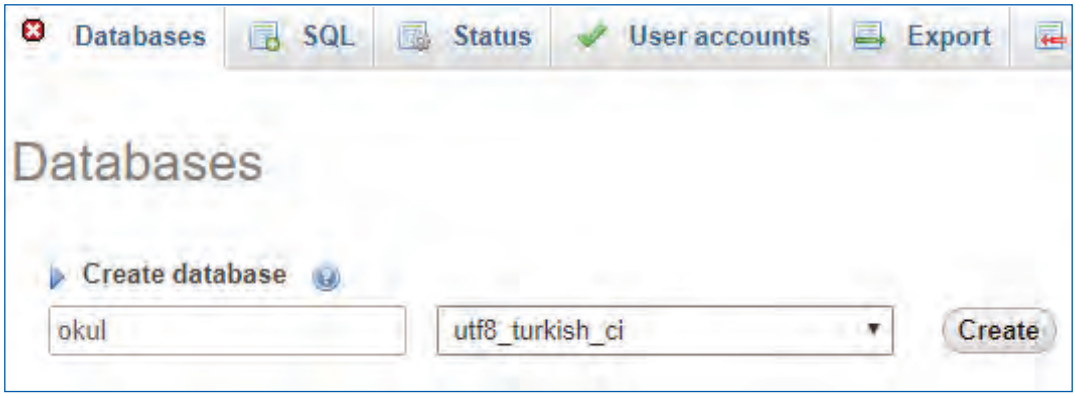
Bilgisayarınızda herhangi bir tarayıcı açılıp ve http://localhost/phpmyadmin/ adresine bağlanılır. Burada localhost, web sitesi kaynaklarının tarayıcının açık olduğu bilgisayarda olduğu belirtilmektedir. Eğer XAMPP kurulu olmayan bir bilgisayarda çalışıyorsanız dipnottaki adresini kullanarak, phpmyadmin sitesinin kullanıcıların testine açtığı demo veri tabanı yönetimini kullanabilirsiniz.

Veri tabanı Oluşturma

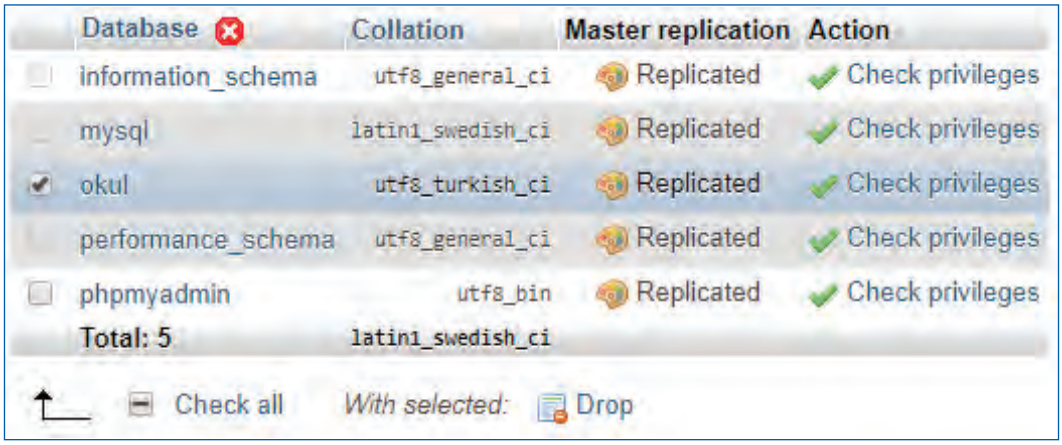


phpMyAdmin yönetim panelini kullanarak yeni bir veri tabanı oluşturabilir ya da oluşturmuş olduğunuz veri tabanları içerisinden yönetmek istediklerinizi seçebilirsiniz.

Yeni bir veri tabanı oluştururken, şekildeki gibi veri tabanı ismi (okul) ve veri tabanının dil desteği seçilir (utf8_turkish_ci). Daha sonra create (oluştur) butonuna tıklanır. Böylece Türkçe karakterleri (ı, ö, ğ vb.) destekleyen okul isimli bir veri tabanı oluşturulmuş olur.



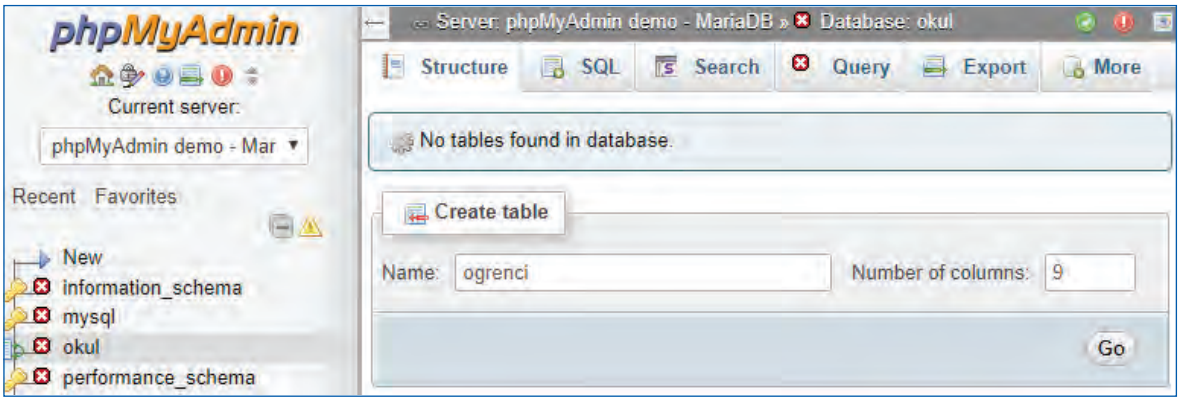
Oluşturulan veri tabanına daha sonrandan erişilmek istendiğinde tekrar Databases sekmesine tıklanarak ulaşılabilir. Eğer veri tabanını silmek isterseniz veri tabanının en solundaki kutucuğu seçerek Drop (Çıkar) butonuna tıklayabilirsiniz.



Tablo Oluşturma

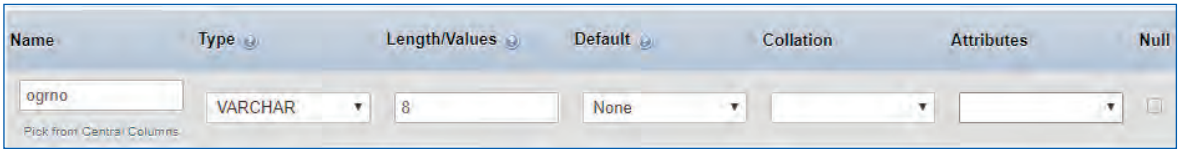
Bir veri tabanı tablolardan, tablolar ise satır ve sütunlardan oluşur. Şimdi oluşturduğunuz veri tabanına tablo ekleyiniz. Bunun için öncelikle tablo içeriğimiz üzerinde tartışalım. Bir okulu ele aldığınızda, okul içerisinde; öğretmen, öğrenci, alt yapı, mali işler vb. alt varlıklar vardır. İlişkisel bir veri tabanı, bu varlıklarla ilgili bilgileri ayrı ayrı tablolarda tutar. Örneğin öğrenci varlığı ile ilgili bilgiler neler olabilir? Öğrenci Numarası, Adı, Soyadı, Yaşı, Doğum Tarihi, Adresi, Dersi, 1. Yazılı Notu, 2. Yazılı Notu, 3. Yazılı Notu, Ortalama, Dersten Geçti ya da Kaldı Durumu vb… Bu örnekler daha da çoğaltılabilir. Şimdilik yukarıdaki bilgileri içeren 12 sütundan oluşan bir öğrenci tablosu oluşturunuz

- Databeses sekmesine tıkla → okul veri tabanına tıkla
- Structure sekmesi altında tablo oluşturacağınız ekranla karşılaştınız. Şekildeki gibi tablo ismi (ogrenci) ve tablonun sütun sayısı (12) yazılarak daha sonra Go butonuna tıklanır.



Not: "utf8_turkish_ci" dil desteği veri tabanımızın içeriğinin Türkçe karakterleri desteklemesini sağlar fakat veri tabanı ismi ve altındaki tablo isimleri Türkçe karakter içermeyecek şekilde yazılır.

Tabloyu oluşturduktan sonra tablo sütunlarının özelliklerini ayarlayacağınız ekranla karşılaşırsınız. Şu an bizim için önemli olan 7 özellik üzerinde duracağız.



Name (Sütun İsmi)

Öğrenci varlığı ile ile ilgili, Öğrenci Numarası, Adı, Soyadı, Dersi, 1. Yazılı Notu, 2. Yazılı Notu, 3. Yazılı Notu, Ortalama Dersten Geçti ya da kaldı durumu vb bilgiler tutulabileceğinden bahsedilmişti. Sütun ismi verilirken bir sonraki ünitede sütunlara kolay erişilebilmesi için aşağıdaki maddelere göre kodlama yapılması önerilir.

- ✓ Türkçe karakter içermemelidir.
- ✓ İki kelime arasında boşluk kullanılmamalı bunun yerine kelimelerin ilk harfleri büyük harfle yazılmalıdır (Ör: OgrNo).
- ✓ Uzun sütün isimleri anlamlı şekilde kısaltılmalıdır (Ör: YazNot1, YazNot2).

Type (Veri Türü)

Sütunlar farklı türde veri içerebilir. Örneğin bazı tür bilgiler sadece numerik değer (10250) içerirken bazı tür bilgiler ise sadece metin (Gazi Mustafa Kemal Atatürk) içerebilir. Bununla birlikte tarih (12.12.2017) de bir veri türüdür. Bu yüzden sütunlarda ne tür bilgi tutulacağının önceden tanımlanması gerekir. MySQL ya da MariaDB'de bir sütun için tanımlanabilecek birçok veri türü vardır. Şu an Numerik, String ve Tarih-Saat türleri üzerinde durulacaktır.

1. Numerik Türler

Bu türler tam sayısal veri türlerini ve ondalık sayı veri türlerini içerir.

a. BOOLEAN: Doğru ya da yanlış bir durumu belirten, sadece 0 ya da 1 numerik değerlerini alabilen bir veri türüdür. Sıfır değeri yanlış kabul edilir. 1 ise doğru sayılır. Örneğin

i. Öğrenciye tablet dağıtıldı (1) ya da dağıtılmadı (0).

ii. Öğrencinin evinde bilgisayar var (1) ya da yok (0).

iii. Öğrenci dersi geçti (1) ya da geçmedi (0).

b. TINYINT: Çok küçük tamsayı değerlerini ifade eder. Eğer negatif değerler kullanılacaksa -128 ile 127 arasındaki değerleri, sadece pozitif değerler kullanılacaksa 0 - 255 arasındaki değerleri kapsar. Örneğin yaş için, not için kullanılabilir.

c. SMALLINT: Eğer negatif değerler kullanılacaksa -32768 ile 32767 arasındaki değerleri, sadece pozitif değerler kullanılacaksa 0 - 65535 arasındaki değerleri kapsar. Örneğin yaş için, not için kullanılabilir.

d. INT: Eğer negatif değerler kullanılacaksa -2147483648 ile 2147483647 arasındaki değerleri, sadece pozitif değerler kullanılacaksa 0 – 4294967295 arasındaki değerleri kapsar. Eğer ihtiyaç duyulan veri bu aralık dışında ise BIGINT kullanılabilir.

e. DECIMAL: Ondalık sayıların istenen formatta biçimlendirmesini sağlayan veri türüdür. Örneğin DECIMAL (10, 2) şeklindeki bir veri tanımlaması toplam 10 numerik değer ve 2 ondalık basamak içeren bir veri türünü işaret eder. Örneğin parasal işlemlerde kullanılabilir. En yüksek 1 000 000 00 TL iş hacmine sahip olan bir şirketi düşünelim. Buna göre DECIMAL (9,2) veri tanımlaması uygundur. DECIMAL için enyüksek basamak sayısı 65'tir. Desteklenen en yüksek ondalık sayıların sayısı (D) 30'dur. Dolayısı ile en yüksek DECIMAL (65, 30) tanımlaması yapılabilir. Bu tanımlamanın yeterli olmadığı durumlarda, INT' da olduğu gibi daha geniş aralıklı veri türleri kullanılabilir. Bunlar FLOAT ve DOUBLE' dir. Bu veri türlerinin tanımlanması da benzer şekildedir FLOAT(x, y) ya da DOUBLE (x,y).

2. String Türler

String; içerisinde harf, rakam, simge barındırabilir. Kitabın Javascript bölümünde, dizi sıralama işleminde numerik değerler içerisinde kullanılan rakamlarla, string değişken içerisinde kullanılan rakamlar arasındaki farklılıktan bahsedilmişti. Bununla birlikte string ve numerik verilerin bir ayrımını da ha ortaya koymamız gerekiyor. Örneğin 150 (string) ve 200 (int) verisini içeren iki veri birbiri ile toplanamaz. Çünkü string sayısal bir değişkeni ifade etmez. 150, 1, 5 ve 0'ın yan yana gelmesinden oluşan bir metni ifade eder. Bu doğrultuda, veri tabanı yönetimi için de veri türlerini tanımlarken bu durumlar göz önünde bulundurulmalıdır. Veri tabanı içerisinde string bir veri tanımlamanın yolları:

a. VARCHAR (X): En yüksek 255 karakter içerisinde barındırabilen bir string veri tanımlar. X yerine gireceğimiz rakamla bu 255 karakter düşürülebilir. Örneğin VARCHAR(1) sadece içerisinde 1 karakter barındırabilir. Bu karakter harf, rakam ya da bir simge olabilir.

b. TEXT (x): En yüksek 65,535 karakter içerisinde barındırabilen bir string veri tanımlar.

3. Tarih ve Saat Veri Türleri

a. DATETIME: Tarih ve Saati birlikte tutan veri türüdür. Desteklenen aralık '1000-01-01 00: 00: 00.000000' ile '9999-12-31 23: 59: 59.999999' arasındadır. MySQL ve türevi MariaDB, 'YYYY-AA-GG HH: MM: SS [.fraction]' formatında DATETIME değerlerini görüntüler.

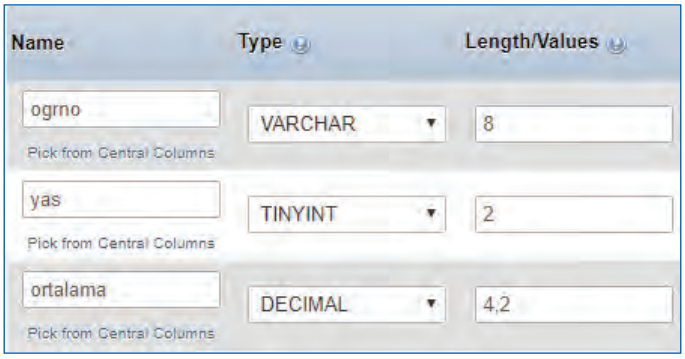
b. DATE: Sadece tarihi tutan veri türüdür. Desteklenen aralık '1000-01-01' ile '9999-12-31' arasındadır. MySQL ve türevi MariaDB, 'YYYY-AA-GG' biçiminde TARİH değerlerini görüntüler. Bu format değiştirilebilir.

c. TIME: Sadece saati tutan veri türüdür. Aralığı '-838: 59: 59.000000' ile '838: 59: 59.000000' arasındadır. MySQL, TIME değerlerini 'HH: MM: SS [.fraction]' formatında görüntüler ancak dizelerin veya sayıların kullanılmasıyla TIME sütunlarına değer atamasına izin verir.

d. YEAR: Dört haneli bir biçimde bir yılı tutan veri türüdür. MySQL ve türevi MariaDB yıl değerlerini YYYY biçiminde görüntüler. Değerler 1901 - 2155 ve 0000 olarak görüntülenir.

Lenght/Values

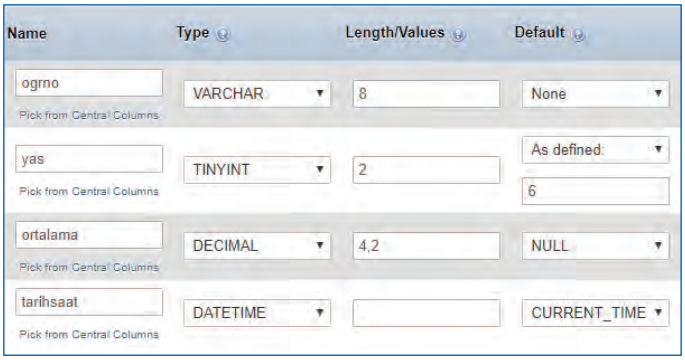
- ✓ Tanımlanan numerik veri türleri için toplam digit veya ondalık değerlerin belirtilmesi
- ✓ String veri türleri için toplam karakter uzunluğunun belirtilmesi



Default

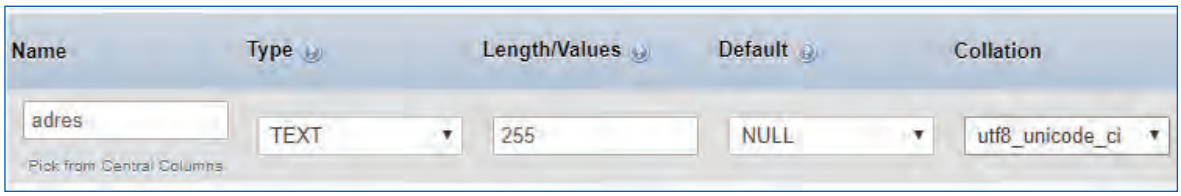
Veri tanımlaması yapıldıktan sonra bazı sütunlara herhangi bir değer girilmemesi durumunda varsayılan olarak bu sütunlarda hangi değerin görüntüleneceğinin ayarlanmasını sağlar. None, As defined, NULL ve CURRENT_TIMESTAMP seçenekleri vardır. Eğer varsayılan olarak

- ✓ Kendinizin tanımlandığı bir değerin görüntülenmesini istiyorsanız, "As defined",
- ✓ Daha önceden hiçbir değer girilip ya da silinmediğini göstermek için "NULL",
- ✓ Sadece tarih zaman türleri için eğer bir değer girilmedi ise o anın güncel tarih ve saatinin otomatik girilmesini istiyorsanız CURRENT_TIMESTAMP seçeneklerini kullanabiliriz.



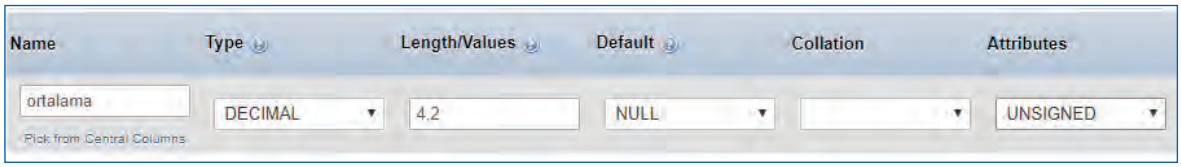
Collation

Veri tabanını oluştururken dil desteği "utf8-turkish-ci" olarak belirtilmişti. Fakat sütunlara ayrı ayrı dil desteği atamanız da mümkündür. Örneğin adres tutan bir string veri kümesi için dil desteğini "utf8-unicode-ci" seçebilirsiniz. Böylece birden fazla dil desteği sağlamış olursunuz.



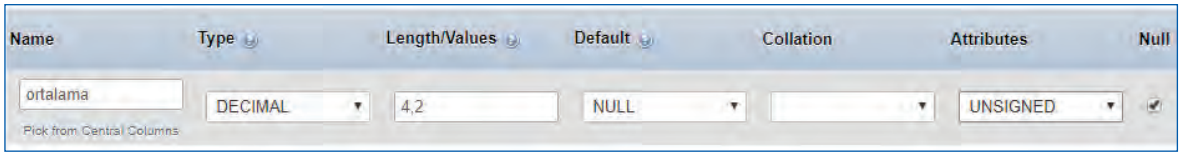
Attributes

Veri tanımlaması yaparken integer(int) negatif değer alınabilir. Sadece pozitif değerli veri girişi yapılması isteniyorsa Attributes özelliği "UNSIGNED" olarak değiştirilmelidir.

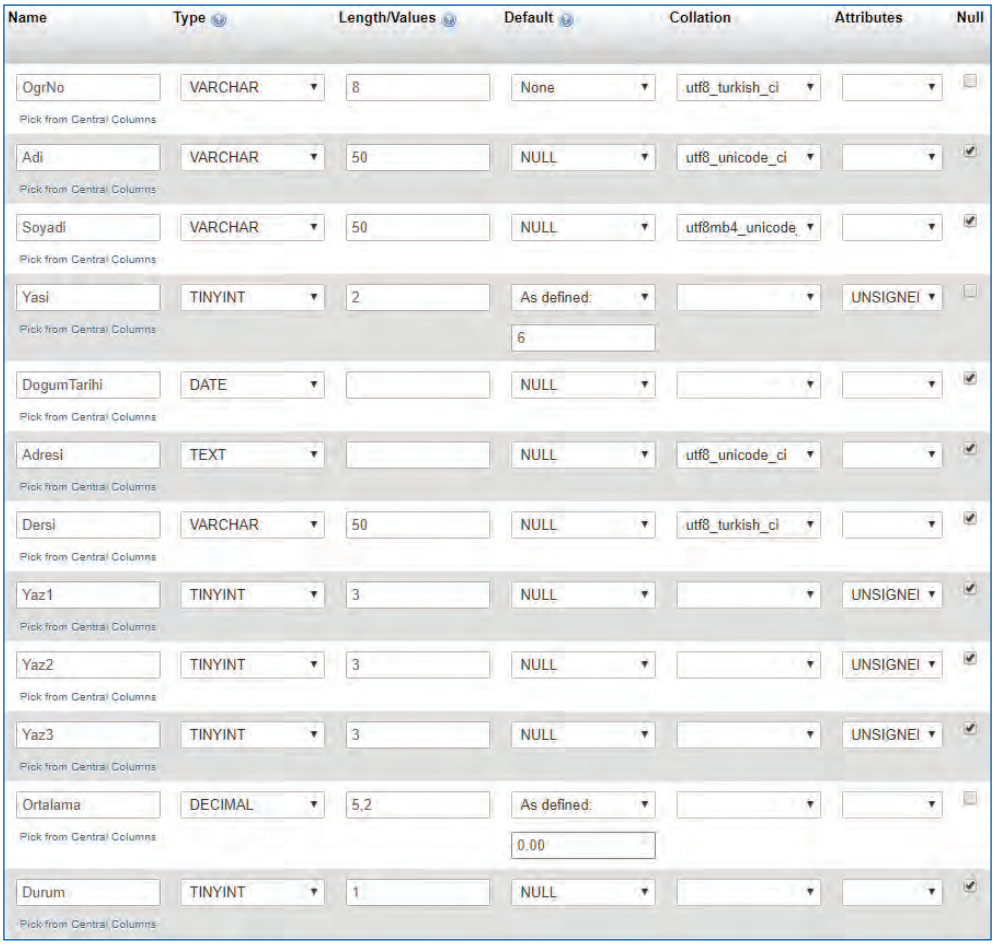


NULL

Eğer sütuna veri girişi zorunlu tutulmak istenilmiyorsa null seçeneğine tik atılması gerekir. Böylece ilk veri girişinde sütun doldurulmadan boş geçilebilir.



Bir veri tabanı içerisinde tablo oluştururken önemli olan 7 özellik ve veri türleri görüldü. Okul veritabanı içerisinde öğrenci varlığı ile ilgili oluşturulacak tablo şekildeki gibidir.



Name	Type	Length/Values	Default	Collation	Attributes	Null
OgrNo	VARCHAR	8	None	utf8_turkish_ci		
Adi	VARCHAR	50	NULL	utf8_unicode_ci		✓
Soyadi	VARCHAR	50	NULL	utf8mb4_unicode		✓
Yasi	TINYINT	2	As defined: 6		UNSIGNEI	
DogumTarihi	DATE		NULL			✓
Adresi	TEXT		NULL	utf8_unicode_ci		✓
Dersi	VARCHAR	50	NULL	utf8_turkish_ci		✓
Yaz1	TINYINT	3	NULL		UNSIGNEI	✓
Yaz2	TINYINT	3	NULL		UNSIGNEI	✓
Yaz3	TINYINT	3	NULL		UNSIGNEI	✓
Ortalama	DECIMAL	5,2	As defined: 0.00			
Durum	TINYINT	1	NULL			✓

Tablonun sütunlarını resimdeki gibi ayarladıktan sonra Save (Kaydet) butonu tabloya kaydedilir. Eğer tabloya yeni sütun eklemek istenirse öğrenci tablosu seçiliyken Structure sekmesine tıklanır. Karşınıza daha önce oluşturmuş olduğunuz sütunlar gelir. Bu ekranda Add bölümünü kullanarak istediğiniz sayıda ve var olan sütunlar arasında istediğiniz bir yere yeni sütun ya da sütunlar ekleyebilirsiniz.

#	Name	Type	Collation	Attributes	Null	Default
1	OgrNo	varchar(8)			No	None
2	Adi	varchar(50)			Yes	NULL
3	Soyadi	varchar(50)			Yes	NULL
4	Yasi	tinyint(2)		UNSIGNED	No	6
5	DogumTarihi	date			Yes	NULL
6	Adresi	text			Yes	NULL
7	Dersi	varchar(50)			No	None
8	Yaz1	tinyint(3)		UNSIGNED	Yes	NULL
9	Yaz2	tinyint(3)		UNSIGNED	Yes	NULL
10	Yaz3	tinyint(3)		UNSIGNED	Yes	NULL
11	Ortalama	decimal(5,2)			No	0.00
12	Durum	tinyint(1)			Yes	NULL

Bir öğrencinin birden çok dersi olabilir. Fakat aynı öğrencinin veri tabanı içerisinde aynı iki dersi içerecek şekilde kaydı olmamalıdır. Çünkü veri tabanı düzenli, tekrar etmeyecek şekilde verilerin tutulmasını sağlamak için kullanılır. Dolayısı ile bu tablo üzerinde aynı öğrenci ve aynı ders olacak şekilde tekrar eden bir kayıt istenmez. Bu durumda birincil anahtar (primary) eklememiz gerekir.

Birincil Anahtar (Primary)

Tablo içerisinde tekrar etmesini istemediğiniz kayıtlar için kullanılabilir. Örneğin OgrNo sütununa birincil anahtar eklediğinizde, tüm tablo içerisinde öğrenci numarası tekrar edemez. Fakat tabloda, OgrNo ve Dersi sütunlarının aynı olması durumda kayıt girişlerinin yapılamaması istenir. Bu durumda, her iki sütunun seçilerek tablonun altındaki Primary butonuna tıklanması gerekmektedir.

	#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
☑	1	OgrNo 🔑	varchar(8)			No	None		Change
☐	2	Adi	varchar(50)			Yes	NULL		Change
☐	3	Soyadi	varchar(50)			Yes	NULL		Change
☐	4	Yasi	tinyint(2)		UNSIGNED	No	6		Change
☐	5	DogumTarihi	date			Yes	NULL		Change
☐	6	Adresi	text			Yes	NULL		Change
☑	7	Dersi 🔑	varchar(50)			No	None		Change
☐	8	Yaz1	tinyint(3)		UNSIGNED	Yes	NULL		Change
☐	9	Yaz2	tinyint(3)		UNSIGNED	Yes	NULL		Change
☐	10	Yaz3	tinyint(3)		UNSIGNED	Yes	NULL		Change
☐	11	Ortalama	decimal(5,2)			No	0.00		Change
☐	12	Durum	tinyint(1)			Yes	NULL		Change

Check all *With selected:* Browse Change Drop Primary

Böylece şekildeki gibi aynı OgrNo' lu öğrencinin farklı derslerle ilgili kaydı tutulurken bir sonraki kayıtta eğer yine 01012323 OgrNo'lu ve matematik dersini içeren bir kayıt girilmesine izin verilmeyecektir.

OgrNo	Adi	Soyadi	Yasi	DogumTarihi	Adresi	Dersi	Yaz1	Yaz2	Yaz3	Ortalama	Durum
01012323	Mete	Yıldırım	6	2011-01-23	Park Eymir Toki	Matematik	90	80	100	90.00	1
01012323	Mete	Yıldırım	6	2011-01-23	Park Eymir Toki	Türkçe	85	95	95	91.66	1

Kodlayıcı Görevi

- Structure Sekmesi altındaki diğer butonların ne işe yaradığını test ediniz.
- Okul veri tabanı içerisinde, öğretmen varlığı ile ilgili bir tablo oluşturunuz.

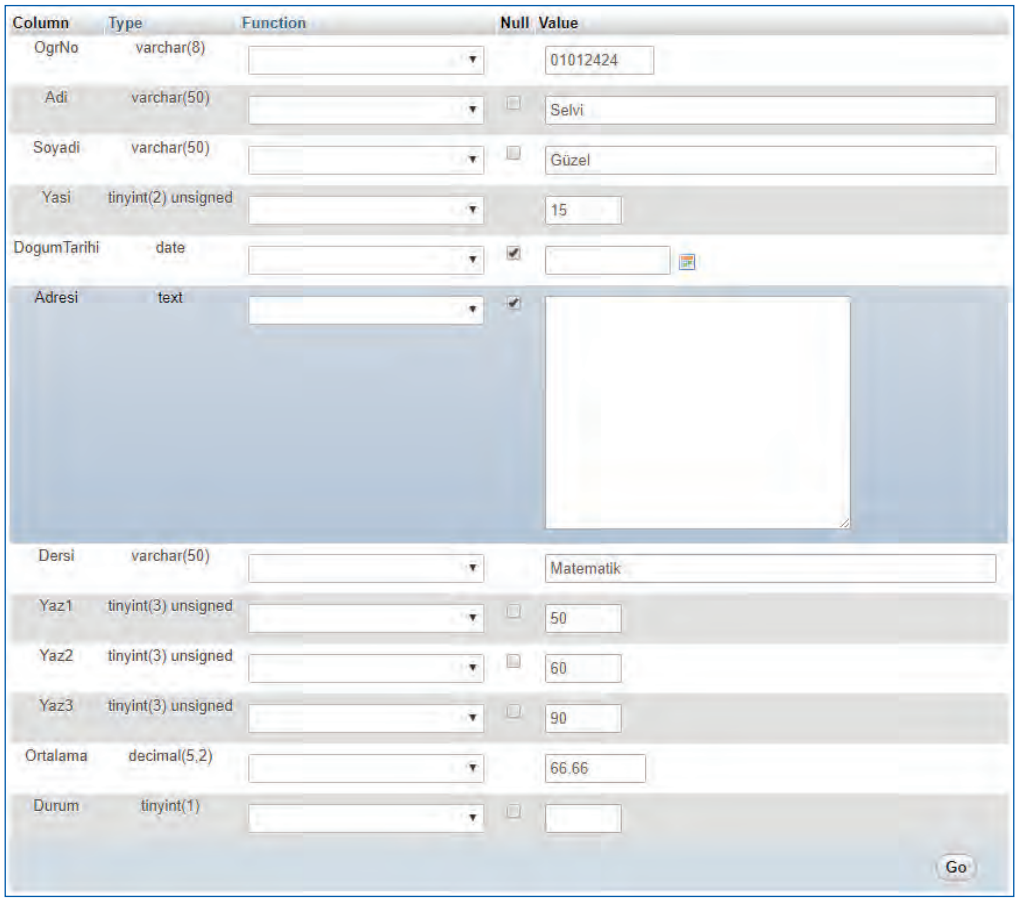
Kayıt İşlemleri

Oluşturmuş olduğunuz tabloların içerisindeki kayıtlarla ilgili Seçme, Ekleme, Güncelleme ve Silme işlemleri yapılabilir. Bu işlemler yapılırken iki yöntem kullanılabilir:

Arayüz (phpMyAdmin) Kullanımı

Kayıt Ekleme

Kayıt girmek isteğiniz tabloyu seçikten sonra, Insert Sekmesine tıklayınız. Daha sonra alanlara değerleri yazınız ve Go butonuna basınız.

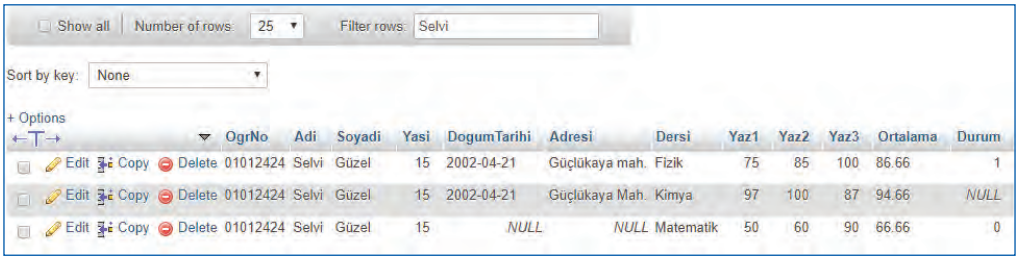


Kodlayıcı Görevi

- Oluşturmuş olduğunuz Okul veri tabanı içerisindeki, ogrenci ve ogretmen tabloları içerisine en az 10' ar kayıt giriniz.

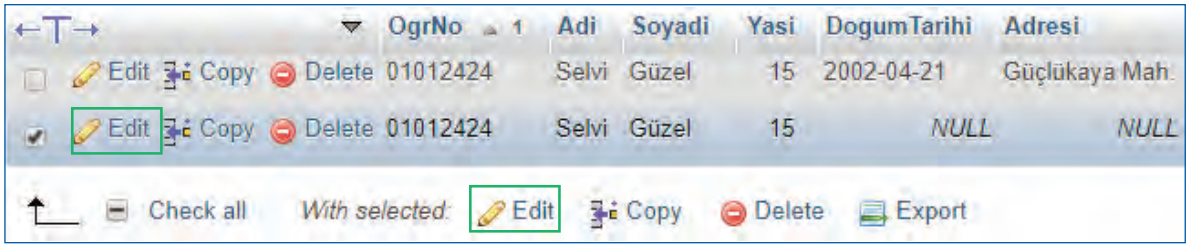
Kayıt Seçme

Herhangi bir tabloyu (ör: ogrenci) seçtiğinizde, tablonun içerisinde var olan kayıtlar "Number of rows" (Kayıt Sayısı)' da belirtilen sayı kadar (Ör: 25 kayıt) sayfalar hâlinde getirilecektir. Eğer isterseniz "Filter rows" (kayıtları filtreleme)' den istenen metin, tarih ya da sayıyı içeren kayıtları getirebilirsiniz. Örneğin şekilde Selvi metnini içeren tüm kayıtlar getirildi. Burada dikkat edilmesi gereken bir nokta, filtreleme bölümüne girmiş olduğunuz anahtar kelimenin tüm sütunlarda aranmasıdır. Anahtar kelime "100" olarak girilmiş olsaydı, yazılı notlarından herhangi biri 100 olan kayıtların hepsini getirilebilecekti.



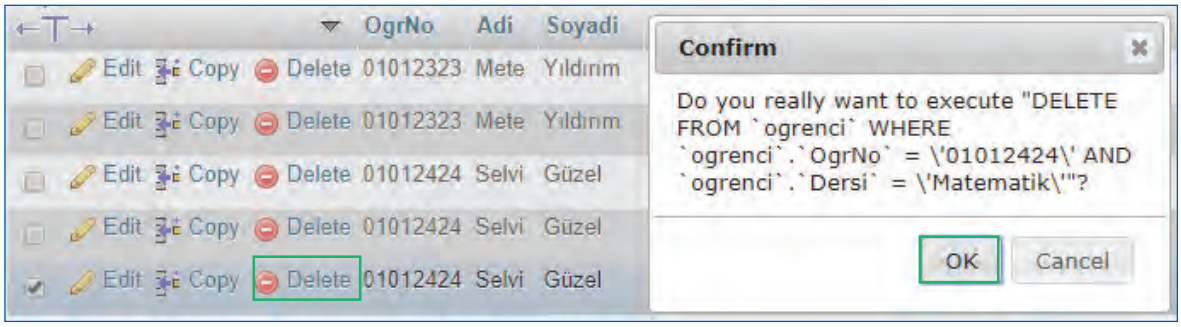
Kayıt Güncelleme

Herhangi bir kaydı güncellemek istediğinizde, kaydın solundaki Edit butonuna tıkladıktan sonra kayıt eklemede yaptığınız gibi yeni değer girebilir ve güncelleme işlemi yapabilirsiniz.



Kayıt Silme

Herhangi bir kaydı silmek istediğinizde, kaydın solundaki Delete butonuna tıkladıktan sonra karşınıza gelen ekrandaki uyarıyı onaylayarak silme işlemi yapabilirsiniz.



SQL Kodları

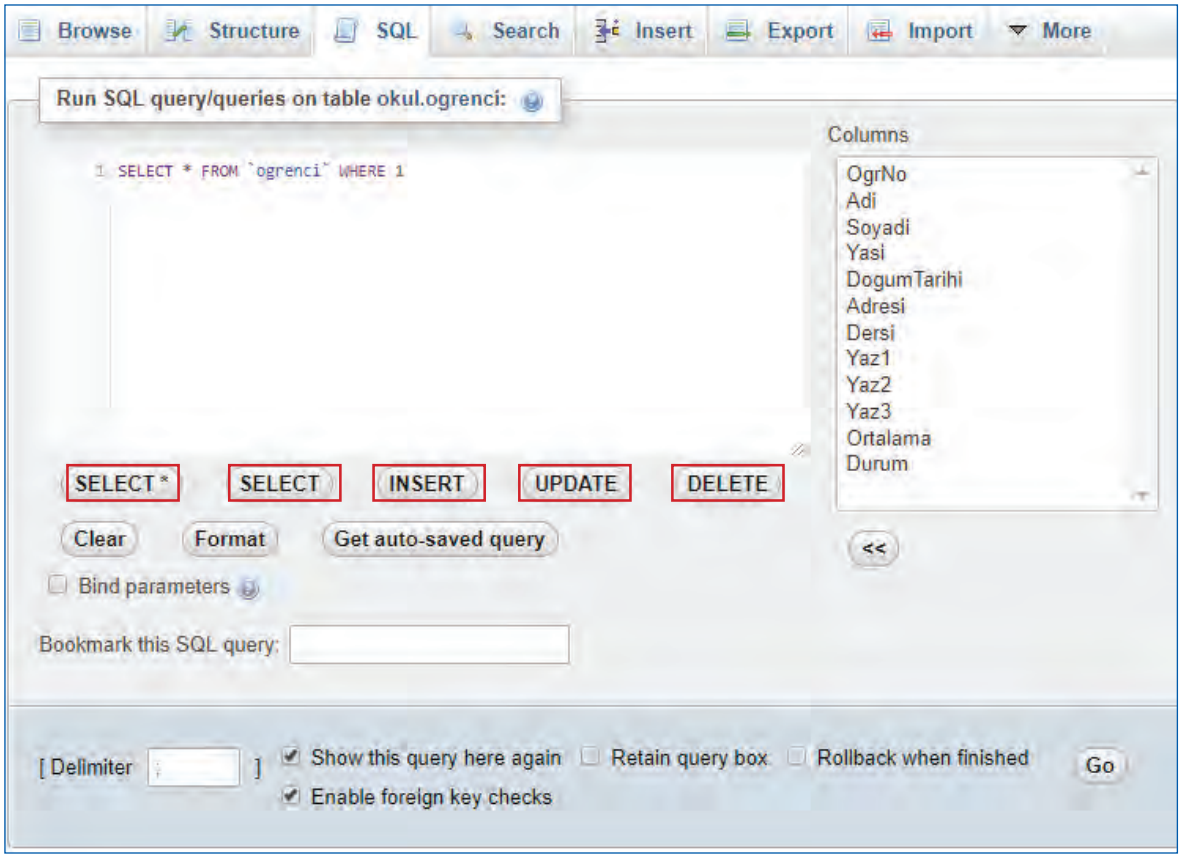
phpMyAdmin kullanıcılara kod yazımı olmadan kolay bir veri yönetimi sunar. Peki, arka planda çalışan kod nasıldır?

Veri tabanı yönetimi aslında SQL kodları ile yapılır. phpMyAdmin panelini kullanarak yapmış olduğumuz veri ekleme, seçme, güncelleme ve silme işlemleri için sql kodlarının nasıl çalıştığını kavrayıp uygulamamız gerekir? Çünkü, bir sonraki bölümde bu kodlara ihtiyaç duyacağız.

Web programlama yaparken web sitesi kaynakları ile veri tabanı yönetimi birbirinden ayrılır. Web sitesi programlarken bizim kaynak dosyalarımız ve veri tabanı arasında ilişki kuracak kodlara ihtiyaç vardır. Örneğin web siteleri sizden iletişim için mail adresinizi, isminizi girmenizi ister ve daha sonra bunu veri tabanlarına yazar. Böylece sizle ilgili iletişim bilgilerini tutmuş olur. Gerektiğinde size bu veri tabanındaki bilgileri kullanarak mail atar ya da bir sosyal ağda yazmış, silmiş yada güncellemiş olduğunuz yorumlarda da aynı tür işlemler yapılır. İşte web tabanlı programlamada bu tür işlemler (veri ekleme, veri getirme, veri güncelleme, veri silme) için phpMyAdmin panelinin arka planda çalıştırdığı SQL kodları kullanılır.

Phpmyadmin panelinde bu kodları çalıştırarak nasıl yazıldıklarını kavramaya çalışılacak. Bir sonraki ünitede, bu kodları yazarak web sayfası üzerinden veri ekleme, seçme, güncelleme ve silme işlemlerini yapılabilecek.

Öğrenci tablosu seçili iken SQL sekmesine tıklayınız. Karşınıza bir tablo üzerinde yapabileceğiniz, veri ekleme, seçme, güncelleme ve silme işlemlerini sql kodları ile yapabileceğiniz ekran geliyor. Burada SELECT, INSERT, UPDATE ve DELETE' e ayrı ayrı tıklayınız ve ekranda değişen kodları inceleyiniz.



SELECT

Veri seçimi için kullanılan SQL komutudur.

```
SELECT sutun1, sutun2,                                    ...
FROM  tablo_adi
WHERE koşul
ORDER BY sutun1, sutun2, ... ASC|DESC;
```

Bir tablo içerisinden veri seçimi nasıl olabilir biraz üzerinde düşününüz.

I. Tablodaki tüm sütunları içerecek şekilde tüm kayıtları seçebilirsiniz.
SELECT * FROM tablo_adi

II. Tablodaki bazı sütunları içerecek şekilde tüm kayıtları seçebilirsiniz.
SELECT sutun1, sutun2, sutun3 FROM tablo_adi

III. Tablodaki tüm sütunları içerecek şekilde bazı kayıtları seçebilirsiniz.

```
SELECT *  FROM tablo_adi WHERE sutun1 = 'filtre'
SELECT *  FROM tablo_adi WHERE sutun1 = 'filtre'  AND sutun2 = 'filtre'
SELECT *  FROM tablo_adi WHERE sutun1 = 'filtre'  OR sutun2 = 'filtre'
```

IV. Tablodaki bazı sütunları içerecek şekilde bazı kayıtları seçebilirsiniz.

```
SELECT sutun1, sutun2, sutun3  FROM tablo_adi  WHERE sutun1 = 'filtre'
```

V. Seçilen kayıtları istenen bir sütuna göre sıralayabiliriz.

```
SELECT *  FROM tablo_adi ORDER BY sutun1 ASC                /* A dan Z ye */
SELECT *  FROM tablo_adi ORDER BY sutun1 DESC               /* Z den A ya */
SELECT sutun1, sutun2, sutun3 FROM tablo_adi  ORDER BY sutun1 DESC
SELECT *  FROM tablo_adi WHERE sutun1 = 'filtre' ORDER BY sutun1 ASC
SELECT *  FROM tablo_adi WHERE sutun1 = 'filtre'  OR sutun2 = 'filtre'  ORDER BY su-
    tun1 ASC, sutun2 DESC
```

INSERT

Veri ekleme için kullanılan SQL komutudur.

```
INSERT INTO tablo_adi (sutun1, sutun2,sutun3, …)
VALUES (sutun1_deger, sutun2_deger, sutun3_deger, ...);
```

Bir tablo içerisinden veri ekleme nasıl olabilir.

I. Tüm sütunlara değerlerinin girilerek kayıt eklenmesi

```
INSERT INTO tablo_adi VALUES ('sutun1', 'sutun2', 'sutun3', …)
/*Tüm sütunların değerleri girilir.*/
INSERT INTO ogrenci VALUES ('01012525','Sezgin','Ardıç','10','2007-05-20','Kalaba Mah.','Mü-
zik','50','75','100','75.00','1')
```

II. Sadece istenen sütunlara değerlerinin girilerek kayıt eklenmesi

```
INSERT INTO tablo_adi ('sutun1', 'sutun2', 'sutun3' …)
/*Null izni olmayan sütunlarla birlikte kayıt eklenmek istenen sütunlar yazılır.*/
VALUES ('sutun1_deger', 'sutun2_deger', 'sutun3_deger' …)
/*İsmi yazılan sütunların değerleri girilir. */
INSERT INTO ogrenci (OgrNo, Dersi) VALUES ('01012626','Beden Eğitimi')
```

UPDATE

Veri güncelleme için kullanılan SQL komutudur.

```
UPDATE tablo_adi
SET sutun1 = sutun1_yenideger, sutun2 = sutun2_yenideger, ...
WHERE koşul;
```

```
UPDATE ogrenci SET Adi=`Deniz Mete` WHERE OgrNo=`01012323`;
/* Öğrenci numarası 01012323 olan tüm kayıtlardaki öğrenci ismini Deniz Mete olarak değiştirir */
UPDATE ogrenci SET Durum=`0` WHERE OgrNo=`01012323` AND Ortalama<`70.00`;
/* Öğrenci numarası 01012323 ve ortalama puanı 70.00 altında olan tüm kayıtlardaki öğrencinin geçti kaldı durumunu 0 (kaldı) olarak değiştirir. */
UPDATE ogrenci SET Adress=`Gölbaşı Mah. Oğuzlar ÇORUM`;
/* Koşulsuz bir update kodu (Where olmadan) yazarsanız, tablonuz içerisindeki bütün kayıtların, belirtmiş olduğunuz sütun yada sütunların (Ör: Adress) değeri aynı olur. Update kodunu bu şekilde kullanırken çok dikkatli olmalısınız!*/
```

DELETE

Veri silme için kullanılan SQL komutudur.

```
DELETE * FROM tablo_adi
WHERE koşul;
```

```
DELETE * FROM ogrenci WHERE OgrNo=`01012323`;
/* Öğrenci numarası 01012323 olan tüm kayıtları tablodan siler*/
DELETE * FROM ogrenci WHERE OgrNo=`01012323` OR OgrNo=`01012424`;
/* Öğrenci numarası 01012323 veya 01012424 olan tüm kayıtları tablodan siler*/
DELETE * FROM ogrenci WHERE OgrNo=`01012323` AND Ders=`Matematik`;
/* Öğrenci numarası 01012323 ve dersi Matematik olan kaydı tablodan siler
Tek kayıt olacağını nasıl bilebiliyoruz? İpucu: Primary (Birincil Anahtar)*/
DELETE * FROM öğrenci
/* Koşulsuz bir delete kodu (Where olmadan) yazarsanız, tablonuz içerisindeki bütün kayıtları silersiniz. Delete kodunu bu şekilde kullanırken çok dikkatli olmalısınız!*/
```

Kodlayıcı Görevi

- En az 10 kayıt girişi yapmış olduğunuz öğretmen tablosu üzerinde insert, select, update ve delete sql cümleciklerini kullanarak, veri ekleme, veri seçme, veri güncelleme ve veri silme işlemlerini yapınız.

6. ETKİLEŞİM VE VERİ YÖNETİMİ

6. ETKİLEŞİM VE VERİ YÖNETİMİ

Etkileşim denildiğinde, web programlama ile ilgili Javascript kodlama üzerinde durulmuştu. Veri yönetimi denildiğinde ise sunucu üzerinde çalışan (sunucu tabanlı) araçlar içerisinden Mysql' in türevi olan MariaDB yönetimi ve sql kodlama üzerinde durmuştuk.

Kodlayıcı Görevi

- Javascript ile veri yönetimi yapılabilir mi? Daha açık bir ifadeyle Javascript kullanarak Mysql ya da MariaDB içerisindeki veriler üzerinde ekleme, seçme, güncelleme ve silme işlemleri yapılabilir mi? Neden?

Web programlama yaparken web sitesi kaynakları ile veri tabanı yönetiminin birbirinden ayrıldığını hatırlayınız. Web sitemiz şu ana kadar öğrendiğiniz kodlar üzerinden hareketle HTML, CSS ve Javascript kodlarının bir birleşiminden meydana geliyordu. Bunun dışında veri yönetimi için phpMyAdmin paneli ile tanıtılmıştı. Fakat bu panel site kodlarından ayrılıyordu. Bu panel ile site kodlarını konuşturmak içinse sql kodlarının kullanılma gereksiniminden bahsedilmişti.

HTML, CSS ve Javascript kodları veri yönetimi yapmak yani sql kodlarını çalıştırmak için yeterli değildir. Bunun için sunucu tabanlı çalışan bir başka kod yapısına ihtiyaç vardır. Bu ihtiyacı PHP, ASP, CSharp vb. programlama dilleri karşılayabilir. Biz bu ünitede PHP üzerinde durulacak.

PHP (HYPERTEXT PREPROCESSOR)

PHP dinamik ve etkileşimli web sayfaları yapmak için sunucu tabanlı bir kodlama dilidir. "Hypertext Preprocessor" kelimelerinin kısaltılmasıdır. Yaygın olarak kullanılan, açık kaynak kodlama dilidir.

PHP dosyaları

- Metin, HTML, CSS, JavaScript ve PHP kodu içerebilir.
- PHP kodu sunucuda yürütülür ve sonuç tarayıcıya düz HTML olarak döndürülür.
- PHP dosyalarının uzantısı ".php" dir.

PHP,

- Dinamik sayfa içeriği üretebilir.
- Sunucudaki dosyaları oluşturabilir, açabilir, okuyabilir, yazabilir, silebilir ve kapatabilir.
- Form verilerini toplayabilir.
- Çerezleri gönderebilir ve alabilir.
- Veri tabanınızdaki verileri ekleyebilir, silebilir, verileri değiştirebilir.
- Veri şifreleyebilir.

PHP kodlarını tarayıcılar HTML, CSS ya da Javascriptte olduğu gibi tek başına çalıştıramaz. Bu yüzden PHP kodlarını yorumlayabilecek bir ara birime ihtiyaç vardır. Hatırlarsanız veri yönetimi için bir ara birim kurulumu (XAMPP) yapılmıştı. Bu ara birim, içerisinde hem veri yönetimi yapabileceğimiz phpMyAdmin panelini hem de PHP kodlarını çalıştırabileceğimiz Apache sunucusunu içerisinde barındırmaktadır. Eğer XAMPP kurulumunu bilgisayara yaptıysanız yeni bir kurulum yapmanıza gereksinim yoktur.

Php kodlama yaparken site kaynak dosyalarınızı, XAMPP kurulumunu yapmış olduğunuz dizin içerisinde (Varsayılan olarak C:\xampp\htdocs) saklamanız gerekir. Böylece Apache tarafından php kodlarınız yorumlanabilir. Basit bir örnek uygulama üzerinde testini yapalım. Notpad++ üzerinde aşağıdaki kodu yazın ve dosya uzantısı .php olacak şekilde kaydediniz.

```
<!DOCTYPE html>
<html>
<style>
    body    {font-family:Verdana;font-size:14px;}
    .baslik     {padding:5px; background-color:green;}
</style>
<body>
<p class="baslik"> Örnek 1 <p>

<?php // Bir PHP kodu <?php ile başlar.

    // php kodları büyük küçük harfle yazılabilir.
    ECHO "Adınız ... <br>";
    echo "Soyadınız ... <br>";
    EcHo "Düşünceleriniz ...<br>";
    // echo sayfaya yazdırma işlemi için kullanılabilir.

?>  <!-- Bir PHP kodu  ile biter. -->

</body>
</html>
```

Daha sonra bu dosyayı C:\xampp\htdocs dizinine kopyalayınız. Aynı şekilde htdocs altında yeni bir klasör (Ör: phpsite) oluşturup bu klasör içerisine de kaydedebilirsiniz. Şimdi sıra tarayıcıdan bu dizine erişmeye geldi. http://localhost/test.php ya da http://localhost/phpsite/test.php adreslerini kullanarak ilk php örneğimizin çıktısına erişilebilir.

Php kodlarını nasıl çalıştırabileceğinizi gördük şimdi php üzerinde daha detaylı duracağız.

DEĞİŞKENLER

PHP'de bir değişken tanımlanırken, $ işaretiyle başlanır ve sonrasında değişken adı yazılır.

- Değişken adı ise, kısa (x ve y gibi) veya daha açıklayıcı (yaş, arabaadi, toplam) olabilir.
- Değişken adı bir harf veya altçizgi karakteriyle başlar. Bir sayı ile başlayamaz.
- Değişken adı yalnızca alfasayısal karakterler (A-z, 0-9) ve altçizgi (_) içerebilir.
- Değişken adları büyük küçük harflere duyarlıdır. ($yas ve $YAS iki farklı değişkendir)

Şimdi bu ölçütlere göre tanımlanmış bazı değişkenleri bir uygulama üzerinde inceleyiniz.

```
<html>
<style>
    body    {font-family:Verdana;font-size:14px;}
    .baslik     {padding:5px; background-color:green;}
    vurgu { font-weight:bold;}
</style>
<body>

<p class="baslik"> Örnek 2 <p>
<?php

    //php' de değişkenin türünü (int, string, decimal vs.) belirtmeye gerek yoktur.
    $ad = "Deniz Mete";
    $soyad = "YILDIRIM";
    $_yas= 2;
    $_agirlik= 12.20;
    $boy1 = 80.50;

    echo "$ad $soyad' ın $_yas yaşındaki fiziksel özellikleri <br/>";
    echo "-------------------------------------------------------------<br/>";
    echo "Kilosu: <vurgu> $_agirlik </vurgu> <br/> Boyu: <vurgu> $boy1 </vurgu> ";

?>

</body>
</html>
```

Örnek 2

Deniz Mete YILDIRIM' ın 2 yaşındaki fiziksel özellikleri

Kilosu: **12.2**
Boyu: **80.5**

Bu örnekte herhangi bir etkileşim olmadığını fark etmiş olmalısınız. Web tarayıcısını açtığınızda doğrudan yukarıdaki ekranla karşılaştınız. Etkileşimli bir örnek tasarlayınız. Örneğin site kullanıcısından kendi doğum tarihini girmesini isteyiniz ve buna göre kullanıcının yaşını hesaplayınız. Bu ve benzer örnekler için kodlamaya geçmeden önce PHP Form Kullanımı ile ilgili bilgi sahibi olmanız gerekiyor.

FORM KULLANIMI

Daha önceden PHP nin sunucu tabanlı çalıştığını dile getirmiştik. Bunun anlamı, kullanıcı herhangi bir eyleme geçtiğinde diğer bir ifadeyle sayfa ile etkileşim kurduğunda (örneğin veri girdi ve butona bastı) her seferinde öncelikle sunucuya bizim bilgisayarımızdan bir istek gönderilir daha sonra kodlar sunucuda yorumlanır, HTML'ye dönüştürülür ve son olarak bizim bilgisayarımıza geri gönderilir. Bilgisayarımızdan sunucuya veri gönderilirken daha önceden de görmüş olduğumuz form elementleri kullanılır. Form elementlerine girilen değerler ise POST ya da GET yöntemleri ile taşınır. İlk olarak POST yöntemini ele alınız.

POST YÖNTEMİ

```
<!-- test.php sayfasındayız. -->
<p class="baslik"> Örnek 3 <p>

<form action="test.php" method="post">

<!-- action:    formdaki butona tıklandıktan sonra,
                hangi sayfaya yönlendirmek istiyorsak
                o sayfanın dosya ismi ve uzantısını yazıyoruz.
                Bu örneğimizde, aynı sayfaya (test.php) yönlendirme
                yapacağız.
                Başka bir sayfaya da yönlendirme yapabiliriz.

     method:    verilerin hangi yöntemle taşınacak (POST ya da GET)
-->
    İsim: <input type="text" name="ad"><br>
    E-posta: <input type="text" name="eposta"><br>
    <input type="submit">
</form>
```

```
<!-- Butona tıklandıktan sonraki işlemler:
Veriler POST yöntemi ile sunucuya aktarılır.
Şimdi bu verileri kullanabilmek için gerekli php kodunu yazalım.
 -->
<?php
    //İlk olarak istediğimiz veriler gönderilmiş mi kontrol ediyoruz.

    // isset: veri var mı?
    // $_POST[".."]: post edilen verileri ulaşma
    if ( isset($_POST["ad"]) && isset($_POST["eposta"])){
        // name' i ad ve eposta olan post edilmiş veri var olduğunda
        yapılacaklar...

        echo "Sistem <vurgu>" . $_POST["ad"] . "</vurgu> adlı kişiye
        <vurgu>" . $_POST["eposta"] . "</vurgu> eposta adresini kullanarak
        mail atacak." ;
        // php de . işareti kullanılarak string ifadeler ve değişkenler
        birleştirilebilir.
    } else {
        // name' i ad ve eposta olan post edilmiş veri var olmadığında
        yapılacaklar...
        echo "<p style=background-color:grey;margin:10px;padding:5px;>Sayfa
        POST edilmedi. Bir diğer ifadeyle methodu post olan bir form öğesi
        içerisindeki butona tıklanmadı.</p>";
    }
?>
```

Sayfa İlk Açıldığında

Örnek 3

İsim:

E-posta:

Gönder

Sayfa POST edilmedi. Bir diğer ifadeyle methodu post olan bir form öğesi içerisindeki butona tıklanmadı.

Veri Girişi Yapıp Göndere Basıldığında

Örnek 3

İsim: Mutlu

E-posta: mutlu@test.com

Gönder

Sistem **Mutlu** adlı kişiye **mutlu@test.com** eposta adresini kullanarak mail atacak.

GET YÖNTEMİ

Bu yöntemle çalışırken POST yöntemi ile yazmış olduğunuz kodlar üzerinde aşağıdaki değişiklikleri yapmanız yeterli olacaktır.

```
<form action="test.php" method="get">
```

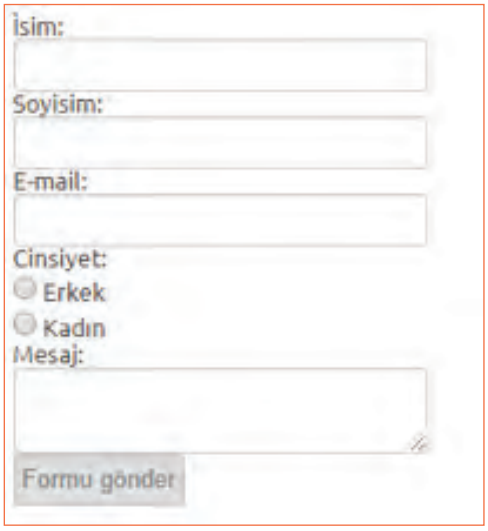
```
<?php
    //İlk olarak istediğimiz veriler gönderilmiş mi kontrol ediyoruz.

    // isset: veri var mı?
    // $_GET[".."]: get ile taşınan verilere ulaşma
    if ( isset($_GET["ad"]) & isset($_GET["eposta"])){
        // name' i ad ve eposta olan get ile taşınan veri var olduğunda
        yapılacaklar...

        echo "<p style=background-color:grey;margin:10px;padding:5px;>
        Sistem <vurgu>" . $_GET["ad"] . "</vurgu> adlı kişiye <vurgu>" .
        $_GET["eposta"] . "</vurgu> eposta adresini kullanarak mail
        atacak.</p>" ;
        // php de . işareti kullanılarak string ifadeler ve değişkenler
        birleştirilebilir.
    } else {
        // name' i ad ve eposta olan post edilmiş veri var olmadığında
        yapılacaklar...
        echo "<p style=background-color:grey;margin:10px;padding:5px;>Sayfa
        POST edilmedi. Bir diğer ifadeyle methodu post olan bir form öğesi
        içerisindeki butona tıklanmadı.</p>";
    }
?>
```

Dikkat!

- Taşınan verilerin ismi ve değeri ile birlikte tarayıcının URL satırında görüntülendiği dikkatinizi çekti mi?

Kodlayıcı Görevi

- Kullanıcının bir sayfada, ismini, soy ismini, şifresini, e-postasını, cinsiyetini ve mesajını girmesini sağlayınız. Daha sonra bu verileri başka bir sayfada (ör: test2.php) gösteriniz.



Uygulama

Şimdi kullanıcının dosya göndermesini sağlayacak bir uygulama yapınız. Bu uygulama için dosya adları dosyayukle.php ve dosyaal.php olan iki sayfa için kodlama yapınız.

Senaryo

- ✓ Kullanıcı dosyayukle.php sayfasını ziyaret eder.
- ✓ Form elementlerini kullanarak herhangi bir dosya seçer ve gönder butonuna basar.
- ✓ dosyaal.php sayfasında post edilen bir dosya olup olmadığı, dosyanın istenen özelliklere uyup uymadığı kontrol edilir.
- ✓ Eğer post edilen bir dosya yoksa ya da istenen özelliklere uymuyorsa GET yöntemi kullanılarak dosyayukle.php sayfasına bir uyarı gönderilir.
- ✓ Eğer post edilen bir dosya varsa ve istenen özelliklere uyuyorsa, post edilen dosya site dizini altında dosyalar klasörüne kaydedilir. Son olarak dosyaal.php sayfasında kullanıcıya dosyanın başarılı şekilde gönderildiği ile ilgili uyarı mesajı verilir.

HTML Kodları (dosyayukle.php)

```
<!--
    action: butona tıkladıktan sonra dosyaal.php
    dosyasına yönlendirilen

    method: veri taşıma yöntemi post olan
    bir form oluşturuyoruz.
-->
<form action="dosyaal.php" method="post">
    <!--Dosya yükleme için type özelliği file olan bir
        input nesnesi oluşturuyoruz.-->
    <input type="file" name="dosya" />
    <input type="submit" value="Gönder" />
</form>
```

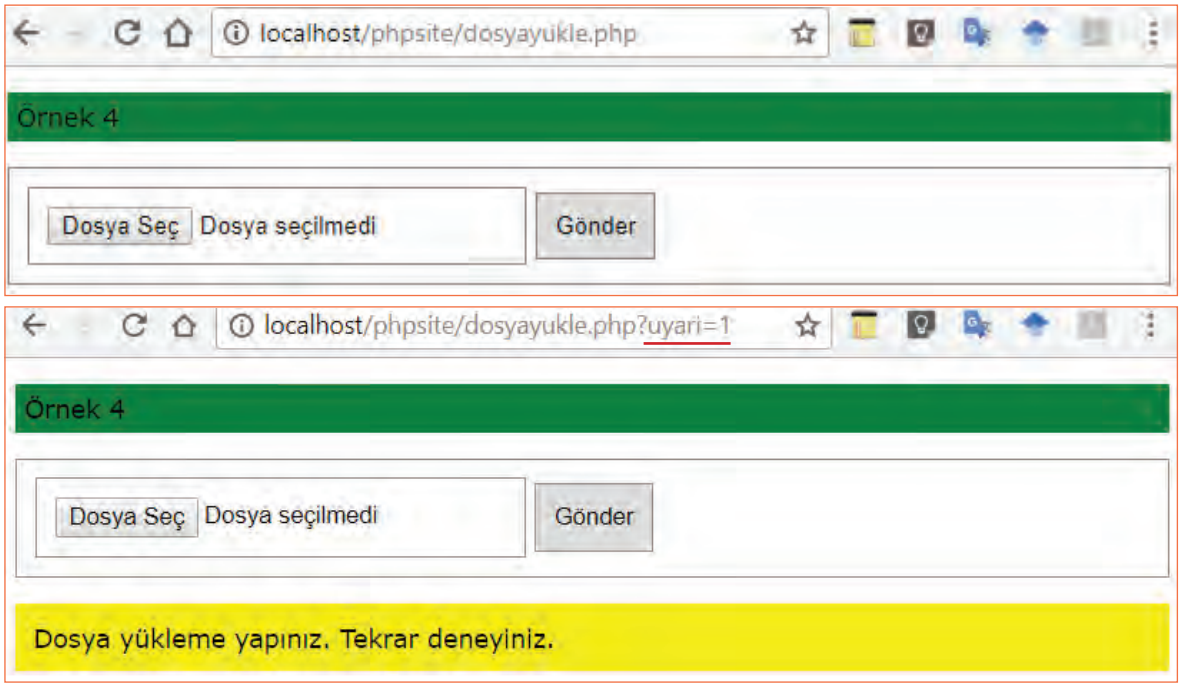
CSS Kodları (dosyayukle.php)

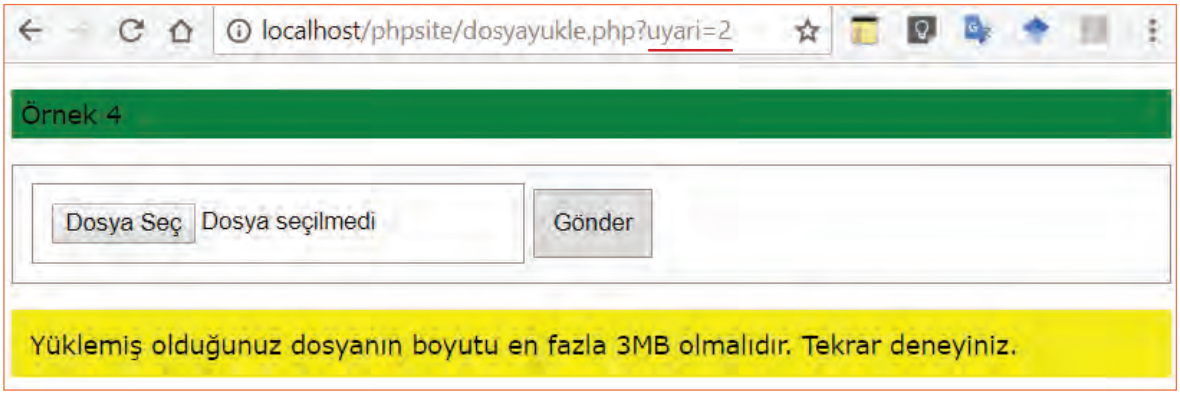
```
<style>
    body     {font-family:Verdana;font-size:14px;}
    .baslik      {padding:5px; background-color:green;}
    form { border:1px solid grey; padding:10px;}
    input { border:1px solid grey; padding:10px;}
    .uyari {padding:10px; background-color:yellow;}
</style>
```

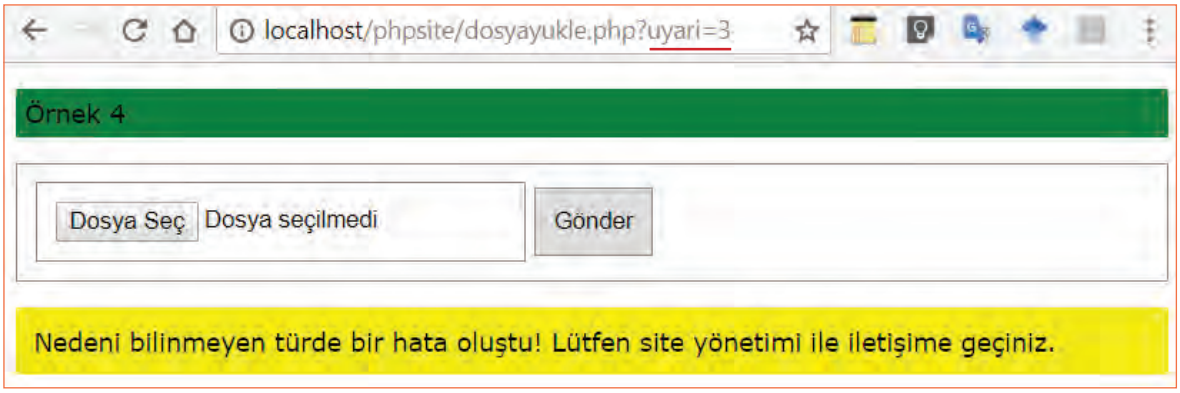
PHP Kodları (dosyayukle.php)

```
<?php
    if (isset($_GET["uyari"])){
    // Eğer get yöntemi ile gönderilmiş bir uyarı varsa ve değeri 1 ise;
        //Uyarı numarasına göre (1 veya 2) hata mesajını ekrana yazdır.
        if ($_GET["uyari"] == 1) {
            echo "<p class=uyari> Dosya yükleme yapınız.
            Tekrar deneyiniz. </p>";
            } else if ($_GET["uyari"] == 2){
                echo "<p class=uyari> Yüklemiş olduğunuz dosyanın boyutu
                en fazla 3MB olmalıdır. Tekrar deneyiniz. </p>";
            } else {
                echo "<p class=uyari> Nedeni bilinmeyen türde bir hata oluştu!
                Lütfen site yönetimi ile iletişime geçiniz.  </p>";
            }
    }
?>
```

Ekran Görüntüsü (dosyayukle.php)







CSS Kodları (dosyaal.php)

```
<style>
    body     {font-family:Verdana;font-size:14px;}
    .baslik      {padding:5px; background-color:green;}
    .uyari {padding:10px; background-color:yellow;}
    a { padding:10px; border-radius:4px; background-color:orange;
        color: white; text-decoration:none;}
    a:hover { padding:10px; border-radius:4px; background-color:grey;
              color: white; text-decoration:none;}
</style>
```

PHP Kodları (dosyaal.php)

Senaryoya göre dosyanın gönderilip gönderilmediği, istenen özelliklere (ör: 3MB dan küçük olması) uygun olup olmadığı kontrol edilecekti. Bu doğrultuda, genel kod algoritması aşağıdaki şekilde oluşturulabilir.

```php
<?php
    // isset(?): ? ile belirtilen nesne var mı yok mu diye kontrol etmek
    // $_FILES["?"]: ? ile belirtilen adda post edilen dosyayı çekmek
    if(isset($_FILES["dosya"])) {
        // eğer adı "dosya" olan bir dosya post edilmişse;
        // dosyanın özelliklerini kontrol et
        // ---------------------------------

        $boyut = $_FILES["dosya"]["size"];
        //dosyanın boyutunu $boyut değişkenine eşitle.
        if($boyut > (1024*1024*3)){
            //eğer dosya boyutu 3MB dan büyükse;
            // dosyayukle.php sayfasına git ve uyarı 2'yi get yöntemi ile gönder.
        } else {
            // eğer dosya boyutu 3MB dan küçükse;
            // dosyayı sunucuya kaydet ve dosyaal.php sayfasında uyarı göster.
            // ................................
            }
   } else {
            // eğer adı "dosya" olan bir dosya post edilmemişse;
            // dosyayukle.php sayfasına git ve uyarı 1 i get yöntemi ile gönder.

        }
?>
```

Bu algoritmada, dosyayukle.php sayfasına geri nasıl yönlendirme sağlanacağı ve dosyanın sunucuya kaydedilmesi ile ilgili işlemler gösterilmemiştir. Şimdi bu iki kodlama üzerinde duralım.

Php ile Sayfa Yönlendirme Kodu

header ("Location:yonlendirileceksayfa.php");

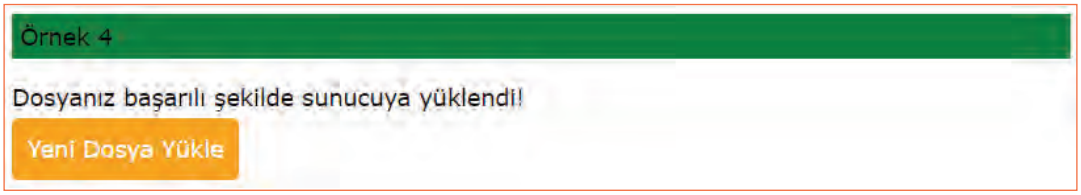
Get Yöntemi ile Sayfa Yönlendirme

header ("Location:yonlendirileceksayfa.php?degiskenismi=deger");

Dosyanın Sunucuya Kaydedilmesi

```php
$dosya = $_FILES['dosya']['tmp_name'];
// ($_FILES['dosya']['tmp_name']) :
// Yüklenen dosyanın sunucuda saklandığı sıradaki geçici dosyayı
// $dosya değişkenine aktarma

copy($dosya, 'dosyalar/' . $_FILES['dosya']['name']);
// $dosya 'yı
// site ana dizininde oluşturmuş olduğunuz dosyalar klasörü içerisinde
// $_FILES['dosya']['name'] ismine göre kaydetme.

// Kullanıcıya dosyanın yükleme durumu ile ilgili uyarı verme
echo "<p class=baslik>Örnek 4<p>";
echo "<p style=uyari>Dosyanız başarılı şekilde sunucuya yüklendi!<p>";
echo "<a href=dosyayukle.php>Yeni Dosya Yükle</a>";
```

Şimdi bu kodları algoritmadaki yerlerine uygun şekilde yazalım. Daha sonra sayfamızı test edelim.



Kodlayıcı Görevi

- Tarayıcıyı ilk açtığınızda dosyayukle.php sayfasını değil de, dosyaal.php sayfasına bağlanmak istediğimizde ne olur? Neden?
- Post edilen dosyanın boyutu ile birlikte başka hangi özellikleri kontrol edilebilir? Araştırınız ve kodlamasını yaparak özelliklere uymayan bir dosya için yeni bir uyarı mesajı oluşturunuz.

PHP İLE VERİ YÖNETİMİ

PHP ile veri tabanına veri ekleyebileceğimizden, veri tabanından veri silebileceğimizden ve veri tabanındaki verileri değiştirebileceğimizden bahsedilmişti. Bir uygulama üzerinden php ile veri yönetiminin nasıl yapılabileceği üzerinde durunuz.

Uygulama

- Kullanıcı uyeol.php sayfasını ziyaret eder. Karşısına çıkan formu doldurur ve veri tabanına yazdırılması için gönderir.
- Site yöneticisi olan bir kişi, yönetim panelinden giriş yapar ve bu verileri görüntüler. Yanlış veri girişi yapmış olan kullanıcıların bilgilerini siler ya da günceller.

Bu senaryo için

- Veri tabanında kullanici isimli bir tabloya,
- Kullanıcılar bilgilerini girebileceği bir uyeol.php sayfasına,
- Yöneticilerin giriş yapabileceği bir giriş.php sayfasına,
- Yöneticilerin giriş yaptıktan sonra verileri görebileceği, güncelleyebileceği ya da silebileceği bir yönetim.php sayfasına ihtiyaç var.

1- Veri tabanı Tablosunun Hazırlanması

I. Öncelikle "site" isimli bir veri tabanı oluşturunuz.

II. Daha sonra, kullanıcı ile ilgili "kullanıcı adı", "şifresi", "adı soyadı", "e-postası", "öz geçmişi", "rolü" bilgilerini tutacak bir tablo tasarımı yapınız.

#	Name	Type	Collation	Attributes	Null	Default
1	kullaniciAdi	varchar(8)			No	None
2	sifre	varchar(8)			No	None
3	adiSoyadi	varchar(25)			No	None
4	ePosta	varchar(25)			No	None
5	ozgecmisi	text			No	None
6	rolu	int(1)			No	0

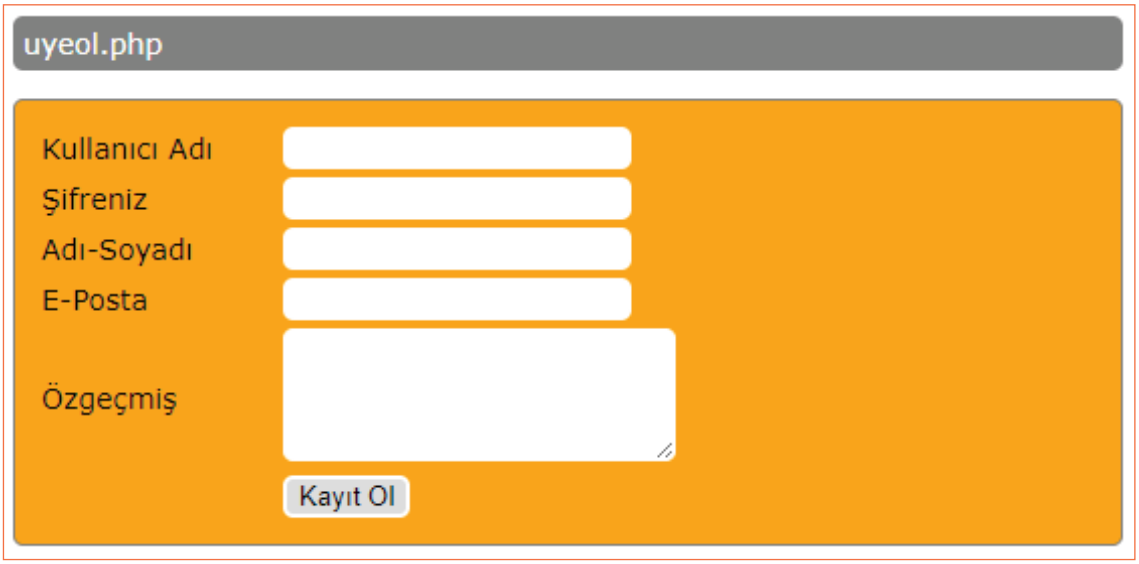
"rolü" sütunu kullanıcının sistemdeki yetkilendirmesi ile ilgili bilgi tutan alandır. Bu örneğimizde, eğer rolü 1 ise yönetici 0 ise yetkisiz kullanıcıdır. Dolayısı ile yeni kayıt yapan bir kullanıcının rolü 0 olacaktır. Bu yüzden bu alanın varsayılan değerini 0 olarak göstermelisiniz.

Dikkat!

- Tablonuzu oluşturduktan sonra yönetici rolünde bir kullanıcı kaydı girişi yapmayı unutmayınız!

2-Kullanıcıların Bilgilerini Girebileceği uyeol.php Sayfasının Hazırlanması

Veri tabanında kullanıcı ile ilgili bilgi tutabileceğimiz tablomuzu oluşturduk. Şimdi senaryomuzdaki sırayı takip ederek, resimdeki gibi bir uyeol.php sayfası oluşturalım. Öncelikle sayfanın tasarımını daha sonra ise, veritabanı bağlantısı ve veri ekleme işlemlerini yapacağız.



CSS Kodu

```
<style>
    body     { font-family:Verdana;font-size:14px;}
    p {padding:5px;background-color:grey;color:white; border-radius:5px;}
    form     { border:1px solid grey; padding:10px;
               background-color:orange; border-radius:5px;}
    input    { border-radius:5px; border-style:solid;border-color:white;}
    textarea { border-radius:5px; border-style:solid;border-color:white;}
</style>
```

Kullanılan Form Elementleri

```
<form action="uyeol.php" method="post">

<table width="400" border="0">
  <tr>
    <td width="115">Kullanıcı Adı</td>
    <td width="269"><input name="kadi" type="text" /></td>
  </tr>
  <tr>
    <td>Şifreniz</td>
    <td><input name="sifre" type="password" /></td>
  </tr>
  <tr>
    <td>Adı-Soyadı</td>
    <td><input name="adsoyad" type="text" /></td>
  </tr>
  <tr>
    <td>E-Posta</td>
    <td><input name="eposta" type="text" /></td>
  </tr>
  <tr>
    <td>Özgeçmiş</td>
    <td><textarea rows="4" cols="25"></textarea></td>
  </tr>
   <tr>
    <td></td>
    <td><input type="submit" value="Kayıt Ol" /></td>
  </tr>
</table>

</form>
```

Php Kodu

Daha önceden de bahsetmiş olduğum gibi, kodlamış olduğumuz web sayfası ile veri tabanı birbirinden farklı platformlardır. Biz kodlayıcı olarak önce bu platformaları birbiri ile konuşturmalıyız. Bunu yaparken php kodlarını kullanacağız. İşlem adımlarımız şekildeki gibidir.

- mysql_connect
- mysql_select_db
- mysql_query
- mysql_close

Veritabanına Bağlan

```php
<?php

$veritabanisunucusu = "localhost";
// veritabanı sunucumuz kendi bilgisayarımızda olduğu için
// sunucumuzun adı localhost ya da ip adresi 127.0.0.1 dir.
// her ikisini de kullanabiliriz.

$vt_kullanicisi = "root";
$vt_kullanicisi_sifre = "1234";
// phpmyadmin kurulumundan sonra güvenlik için veritabanı sunucunuzda
// bir yönetici kullanıcı adı ve şifresi tanımlamanız gerekiyor.
// Php den veritabanına bağlantı kurarken de bu kullanıcı adi ve şifresi ile
// sunucuya bağlanabilirsiniz.

$baglanti = mysql_connect($veritabanisunucusu, $vt_kullanicisi, $vt_kullanicisi_sifre);
// Veritabanımıza mysql_connect kodu ile bağlanıyoruz.
// Kodla beraber, veri tabani sunucusunun adı ya da ip adresi
// veritabanı kullanıcısı ve veritabanı kullanıcısının şifresi
// belirtilmelidir.
// Kod çalıştığında bağlantı gerçekleşirse true,
// gerçekleşmezse false değeri döndürür.
// döndürülen değer $baglanti değişkenine aktarılır.

// Son olarak bağlantı gerçekleşti mi gerçekleşmedi mi kontrol edelim.
if ($baglanti==true) {
    echo "Bağlantı Başarılı";
} else { die("Bağlantı Başarısız! Hata: " . mysql_connect_error());}

?>
```

Eğer bağlantı başarılı şekilde gerçekleştirildi ise, veri tabanı seçme işlemine geçilir. Senaryomuz gereği "site" isminde veri tabanı oluşturmuştuk.

Veritabanı Seç

```php
if ($baglanti==true) {

    // Açıklayıcı Yöntem
    if (mysql_select_db("site") == true) {

        // Veritabanı seçimini mysql_select_db kodu ile yapıyoruz.
        // eğer kod çalıştırıldığında herhangi bir sorunla karşılaşılmazsa;
        // true değer döndürülür.

        echo "Veritabanı seçimi başarılı!";

    } else {
        die("Veritabanı Bulunamadı. Hata: ". mysql_connect_error());
        // Girmiş olduğunuz isimli bir veritabanı olmayabilir.
        // Bu gibi bir durumda, mysql_select_db sonucu false döner.
        // die kodu ile hata gösterilebilir.
    }

    // Daha Kısa Yöntem
    mysql_select_db("site") or die("Veritabanı Bulunamadı. Hata: ". mysql_connect_error());



} else { die("Bağlantı Başarısız! Hata: " . mysql_connect_error());}
```

Eğer veri tabanı seçme işleminde de herhangi bir sorunla karşılaşılmadı ise, artık sorgumuzu gönderebiliriz. Senaryomuz gereği kayıt ekleme işlemini yapacağız. Bu işlem için, kullanici tablosuna ekleyeceğimiz kayıtların değerlerini form elementlerinden gelen bilgilerle dolduracağız. Öncelikle bir veri ekleme sql cümleciğinin nasıl olduğunu hatırlayalım.

INSERT INTO *tablo_adi (sutun1, sutun2,sutun3, …)*
VALUES *('sutun1_deger', 'sutun2_deger', 'sutun3_deger', ...);*

Php ile bir sql cümleciğinin veritabanında çalıştırılmasını sağlayan kod ise mysql_query' dir.

Sorgu Gönder

```
// --Form verilerini çekme--
$kadi = "'".$_POST["kadi"]."'";
$sifre = "'".$_POST["sifre"]."'";
$adsoyad = "'".$_POST["adsoyad"]."'";
$eposta = "'".$_POST["eposta"]."'";
$ozgecmis = "'".$_POST["ozgecmis"]."'";
// Form verilerinin öncesinde ve sonrasında ' işareti eklediğimiz dikkatinizi çekmiştir.
// Veritabanına sql sorgusu göndereceğimiz için değerleri sorgu biçimine uygun hale getirdik.

$sqldegerler = $kadi .",". $sifre.",".$adsoyad.",".$eposta.",".$ozgecmis;
// Daha sonra tüm değerleri aralarına virgül koyarak birleştirdik.

$insert_sql= "insert into kullanici (kullaniciAdi,sifre,adiSoyadi,ePosta,ozgecmisi)
values ($sqldegerler)";
// Değerleri atanmış formata uygun bir insert sql cümlesi oluşturduk.

$eklemeislemi = mysql_query($insert_sql);
// Son olarak oluşturmuş olduğumuz insert sql cümleciğinin
// mysql_query ile veritabanında çalıştırılmasını sağladık.
// Eğer kayıt başarılı şekilde eklendi ise true,
// tersi ise false değeri döndürülür.

if ($eklemeislemi==true){
    echo "Kayıt başarılı şekilde eklendi.<br/>";
}else {
    die("Kayıt Eklenemedi. Hata: ". mysql_connect_error());
}
// Kayıt ekleme işlemininin gerçekleşip gerçekleşmediğinini kontrol ediyoruz.
```

Bağlantıyı Kapat

Kayıt ekleme işlemimiz de bittiğine göre artık veri tabanı bağlantımızı kapatalım.

```
mysql_close($baglanti);
// vt sunucusu (MySQL/mariaDB) ile bağlantımızı koparttık.
```

uyeol.php sayfanızı ilk açtığınızda şekildeki gibi hatalar göreceksiniz. Bu hataları sayfayı ilk açtığımızda post edilen bir form elementi olmadığı için alıyoruz. Hatırlarsanız form elementlerinin verilerini çekerken sayfanın post edilme durumunu kontrol ediyorduk. Şimdi php kodunuzun başladığı yere sayfada post edilen bir form elementi olup olmadığını kontrol eden bir kontrol yapısı ekleyiniz. Sadece bir tane form elementini (Ör: "kadi" isimli form elementi) kontrol etmeniz yeterli olacaktır.

Notice: Undefined index: kadi in **C:\xampp\htdocs\phpsite\uyeol.php** on line **91**

Notice: Undefined index: sifre in **C:\xampp\htdocs\phpsite\uyeol.php** on line **92**

Notice: Undefined index: adsoyad in **C:\xampp\htdocs\phpsite\uyeol.php** on line **93**

Notice: Undefined index: eposta in **C:\xampp\htdocs\phpsite\uyeol.php** on line **94**

Notice: Undefined index: ozgecmis in **C:\xampp\htdocs\phpsite\uyeol.php** on line **95**

```php
if (isset($_POST["kadi"]))
{
    // Yapmış olduğumuz tüm veritabanı işlemleri
}
```

Bunun dışında bir küçük problemle daha karşılaşmanız olasıdır. Kayıt ekleme işlemlerini yaptıktan sonra phpmyadmin panelinden kullanıcı tablonuza baktığınızda, Türkçe karakterlerin (ç,ğ,ı,ö,ş) gösteriminde hatalar olduğunu göreceksiniz. Bu türden bir hata olmaması için Php kodlarında mysql ile veri işlemleri yapmadan önce karakter kümesini UTF-8 olarak ayarlamalısınız. Bu işlemi veritabanı bağlantısını kurduktan sonra yapabilirsiniz.

```php
if ($baglanti==true) {
    echo "Veritabanı bağlantısı sağlandı!<br/>";

    mysql_query("SET NAMES UTF8");
    // Karakter setinin UTF- olarak ayarlanması

    // Kodun devamı ...
```

Üye olma ekranının görünümü

Phpmyadmin panelindeki kullanıcı tablosunun görünümü

kullaniciAdi	sifre	adiSoyadi	ePosta	ozgecmisi	rolu
dnzyil	12345	Denizer Yıldırım	test@test.com	Doğum tarihim ... . Eğitim durumum ... . gö...	0

Kodlayıcı Görevi

- Şimdi aynı şekilde siz de 10 kayıt giriniz?
- Bu kayıtlardan bir tanesinin rolünü phpmyadmin panelini kullanarak 1 yapınız.

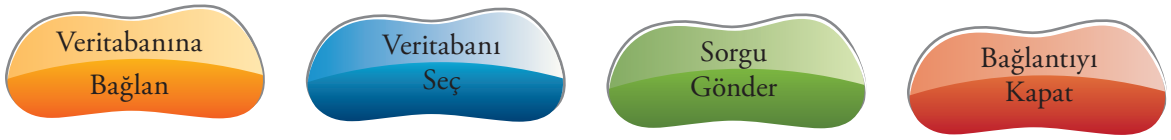
3. Yöneticilerin giriş yapabileceği giriş.php sayfasının hazırlanması

Bu sayfada yönetici olan bir kişinin kullanıcı adı ve şifresini girmesi istenecektir. Daha sonra, bu bilgiler veri tabanındaki bilgilerle karşılaştırılacaktır. Eğer bilgiler birbiri ile tutarlı ise, oturum başlatılıp yönetim.php sayfasına yönlendirme yapılacaktır.

Form Elementlerinin HTML Kodu

```
<form name="giris" action="giris.php" method="post">

 <table cellpadding="8" cellspacing="0" >
   <tr>
     <td width="100">Kullanıcı Adı</td>
     <td><input type="text" name="kullaniciadi"></td>
   </tr>
   <tr>
     <td width="100">Şifre</td>
     <td><input type="password" name="sifre"></td>
   </tr>
   <tr>
     <td colspan="2" align="right">
       <input type="submit" value="Giriş">
     </td>
   </tr>
 </table>

</form>
```

PHP kodu



Veri ekleme işleminde olduğu gibi Veritabanına Bağlanma, Veri tabanı Seçme ve bağlantı kapama işlemlerini aynı şekilde yapıyoruz.

```php
// --Form verilerini çekme--
$kadi = "'".$_POST["kadi"]."'";
// Form verilerinin öncesinde ve sonrasında
// ' işareti eklediğimiz dikkatinizi çekmiştir.
// Veritabanına sql sorgusu göndereceğimiz için
// değerleri sorgu biçimine uygun hale getirdik.

$sql = "select * from kullanici where kullaniciAdi= $kadi";
// Kullanıcı adı, giris.php sayfasında
// kadi isimli form elementine girilen değer olan bir kaydı
// getirecek sql cümlesi...


$sorgulamaislemi = mysql_query($sql);
// Son olarak oluşturmuş olduğumuz select cümleciğinin
// mysql_query ile veritabanında çalıştırılmasını sağladık.
// Eğer kayıt başarılı şekilde getirildi ise
// bir kaynak verisi döndürülür.
// Bu kaynak verisi boş kayıt olabileceği gibi,
// birden fazla kayıtta içerebilir.
// Tersi  durumda ise false döndürülür.
```

```php
if ($sorgulamaislemi != false) {
    // sorgu sonrasında bir hata ile karşılaşılmamışsa,
    if (mysql_num_rows($sorgulamaislemi) > 0) {

} else {
    // sorgu sonrasında bir hata ile karşılaşıldıysa,
    echo "Sql Cümlesinde Hata Var!";
}
```

```php
if (mysql_num_rows($sorgulamaislemi) > 0) {
    // Eğer kaynak verisindeki kayıt sayısı 0 dan büyükse;
    // Kullanıcının girmiş olduğu kadi veritabanında var demektir.
    // varsa şifresi doğrumu ona bakalım!!!

    $satir = mysql_fetch_assoc($sorgulamaislemi);
    // ilk olarak kaynağımız içerisindeki verilere,
    // kolayca ulaşabilmemiz için
    // kaynağımızı mysql_fetch_assoc() ile bir diziye dönüştürüyoruz.

    if ($satir['sifre'] == $_POST["sifre"]) {
} else {
    // Kullanıcının girmiş olduğu kadi veritabanında yok demektir.
    echo "Kullanıcı Adı Bulunamadı!";
}
```

```
    if ($satir['sifre'] == $_POST["sifre"]) {
        // Veritabanındaki şifre ile kullanıcının girmiş olduğu sifre eşleşiyorsa;
        // kullanicinin rolü kontrol edilir.
        if ($satir["rolu"] == 1) {
    } else {
        // Kullanıcının girmiş olduğu sifre ile veritabanındaki şifre eşleşmiyorsa;
        echo "Şifrenizi Yanlış Girdiniz!";
    }
```

```
if ($satir["rolu"] == 1) {
    // kullanıcının rolü 1 (yönetici) ise;
    //oturum başlatılıp yonetim.php sayfasına yönlendirme sağlanabilir.
    session_start(); // oturumu başlat
    $_SESSION["adsoyad"] = $satir["adiSoyadi"];
    header("Location: yonetim.php");
}else{
    // kullanıcının rolü 0 (yönetici değil) ise;
    echo "Yönetici Rolünüz Atanmamıştır. Giriş Yapamazsınız.";
}
```

Sayfayı test ettiğinizde, eğer tüm kodlarınız içerisinde herhangi bir sorunla karşılaşılırsa, giriş.php sayfasında size hata verilecek, karşılaşılmazsa yönetim.php sayfasına yönlendirileceksiniz.

4- Yöneticilerin giriş yaptıktan sonra verileri görebileceği, güncelleyebileceği ya da silebileceği yönetim.php sayfasının hazırlanması

Sayfamızın senaryomuzdaki gibi tüm işlemleri yapabilecek hale gelmesi için kodlama sırası aşağıdaki gibidir.

```
<!DOCTYPE html>
<?php ### 1. Oturum kontrolü

<html>
<style>
    body    { font-family:Verdana;font-size:14px;}
    p {padding:5px;background-color:grey;color:white; border-radius:5px;}
    input   { border-radius:5px; border: 1px solid #42bff4; padding: 5px;}
    table {table-layout:fixed;}
    td {padding:5px; border: 1px solid green;}
    tr:first-child td{font-weight:bold; background-color:#42bff4;}
</style>
<body>
<p>yonetim.php</p>

<?php ### 2. Veritananı bağlantısı ve veritabanı seçimi
<?php ### 5. Veri güncelleme ve silme işlemleri
<?php ### 4. Veritabanındaki kullanici tablosundan tüm kayıtların getirilmesi
<?php ### 3. Veritabanı bağlantısının kapatılması


</body>
</html>
```

1. Oturum kontrolü

İlk olarak sayfayı sadece adı soyadı için oturum açılmış bir kullanıcı görebilir. Dolayısı ile oturum açılıp açılmadığı ile ilgili bir kontrolümüz olmalıdır.

```php
<?php
    session_start();
    // session kullanımını başlatıyoruz.

    // giris.php sayfasında;
    // kullanici adı ve şifresi doğrulanan kişinin
    // adsoyad için bir session açmıştık.
    // Not: kullanıcı idsi için de açabilirdik.

    if (isset($_SESSION["adsoyad"]) == false)
    {
        // eğer daha önce asdoyad isimli bir oturum açılmamışsa
        // yani $_SESSION["adsoyad"]) false ise;
        // giriş sayfasına yönlendiriyoruz.

        header ("Location: giris.php");
    }
?>
```

2. Veri tabanı bağlantısı ve veri tabanı seçimi

```php
<?php ### Veritananı bağlantısı ve veritabanı seçimi

$veritabanisunucusu = "localhost";

$vt_kullanicisi = "root";
$vt_kullanicisi_sifre = "1234";

$baglanti = mysql_connect($veritabanisunucusu,$vt_kullanicisi,$vt_kullanicisi_sifre);

if ($baglanti==true) {

    mysql_query("SET NAMES UTF8");

    mysql_select_db("site")
    or die("Veritabanı Bulunamadı. Hata: ". mysql_error());
} else {die("Veritabanı bağlantısı kurulamaı! Hata:".mysql_error());}

?>
```

3. Veri tabanı bağlantısının kapatılması

```php
<?php
    mysql_close($baglanti) or die("VT bağlantısı kapatılırken bir hata
    oluştu!. Hata:".mysql_error());
?>
```

4.Veri tabanındaki en güncel kayıtların sayfada gösterimi

Bu bölümde veri tabanındaki kullanıcı tablosundan tüm kayıtları çekerek ve bu kayıtları sayfada gösterebilecek bir kodlama yapacağız. Bunun için iki basamaklı bir kodlama yapacağız.

1. Veri tabanındaki kullanıcı tablosundan tüm kayıtların getirilmesi

```php
<?php ### Veritabanındaki kullanici tablosundan tüm kayıtların getirilmesi

    $sql = "select * from kullanici";
    // Veritabanındaki kullanici tablosundaki tüm kayıtları getiren sql cümlesi...

    $sorgulamaislemi = mysql_query($sql);
    // select cümleciğinin mysql_query ile veritabanında çalıştırılmasını
    // ve geriye bir veri kaynağı döndürdük.

    if ($sorgulamaislemi != false) {
        // sorgu sonrasında bir hata ile karşılaşılmamışsa,
        // artık verileri tablo halinde yöneticiye gösterebiliriz.

        ### Kayıtların sayfada gösterimi

    }
?>
```

2. Kayıtların sayfada gösterimi

```php
### Kayıtların sayfada gösterimi
### Tablonun üst bilgilerinin yazdırılması

echo "<table>";
// --- php ile table html kodlarının sayfaya yazdırılması ---

echo "<tr>";
// table etiketi altında <tr> etiketi

echo "<td>Kullanıcı Adı</td>";
echo "<td>Şifre</td>";
echo "<td>Adı Soyadı</td>";
echo "<td>e-posta</td>";
echo "<td>Özgeçmişi</td>";
echo "<td>Rolü</td>";
echo "<td>Düzenleme</td>";
echo "<td>Silme</td>";
// tr etiketinin altında td etiketleri
// ilk satır sütünların başlıklarını içermektedir.

echo "</tr>";
// ilk satırın kapatılması
```

```php
### Kayıtların sayfada gösterimi
### Tablo içeriğinin veritabanından döndürülen veri kaynağı içerisindeki
### ($sorgulamaislemi) verilerle doldurulması
while($satir = mysql_fetch_assoc($sorgulamaislemi)) {
    // $sorgulamaislemi kaynağındaki veriler bitene kadar
    // her seferinde $satir değişkenine
    // bir kaydın bilgileri dizi olarak getirilir.
    // Kaynaktaki veriler bittiğinde while döngüsü sona erer.

    echo "<tr>";
    echo "<td>" . $satir["kullaniciAdi"]. "</td>";
    echo "<td>" . $satir["sifre"]. "</td>";
    echo "<td>" . $satir["adiSoyadi"]. "</td>";
    echo "<td>" . $satir["ePosta"]. "</td>";
    echo "<td>" . $satir["ozgecmisi"]. "</td>";
    echo "<td>" . $satir["rolu"]. "</td>";
    // while döngüsü sonlandırılana kadar,
    // tablonun ilk satırını oluştururken yapmış olduğumuz gibi
    // tabloya bir kayıt eklenir ve sütunların içeriği
    // $satir dizi elemanları ile doldurulur.

    echo '<td><a href="yonetim.php?eylem=duzenle&kadi='
    .$satir['kullaniciAdi'].'"> Düzenle </a></td>';
    // Düzenleme isimli sütun için bir link oluşturulur.
    // Bu linkte, eylem ve kadi bilgileri get yöntemi ile birlikte taşınır.

    echo '<td><a href="yonetim.php?eylem=sil&kadi='
    .$satir['kullaniciAdi'].'"> Sil </a></td>';
    // Silme isimli sütun için bir link oluşturulur.
    // Bu linkte, eylem ve kadi bilgileri get yöntemi ile birlikte taşınır.

    echo "</tr>";
    // Satır etiketi kapatılır.
}
echo "</table>";
// Tablo etiketi kapatılır
```

Ekran Görüntüsü

yonetim.php

Kullanıcı Adı	Şifre	Adı Soyadı	e-posta	Özgeçmişi	Rolü	Düzenleme	Silme
dnzyil	123456	Denizer Yıldırım	test@test.com	Doğum tarihim ... . Eğitim durumum ... . görevlerde bulundum.	1	Düzenle	Sil
Myildiri	09876	Mutlu Yıldırım	happy@test.com	Ankara Üniversitesi Siyasal Bilimler ve kamu Yönetimi mezunuyum.	0	Düzenle	Sil
Sguzel	345987	Selvi Güzel Yıldırım	selvi@test.com	MEB Fen ve Teknoloji Öğretmeni...	0	Düzenle	Sil

5. Veri güncelleme ve silme işlemleri

Bir önceki aşamada tablo içerisine verilerle birlikte düzenle ve silme linkleri oluşturmuştuk. Kodu daha detaylı incelediğimizde get yöntemi ile eylem ve kadi değişkenlerinin taşındığını göreceğiz. Dolayısı ile ilk olarak sayfamızda get yöntemi ile taşınan bu isimlerde veriler olup olmadığını kontrol etmemiz gerekmektedir. Bu bölümdeki kodlamamızın genel algoritması aşağıda gösterilmektedir.

```php
<?php ### 5. Veri güncelleme ve silme işlemleri

    if (isset($_GET["kadi"])&&isset($_GET["eylem"]))
    {
        if ($_GET["eylem"]=="duzenle"){
            ### 1.Veri Güncelleme Formunun Hazırlanması
        }
    }else if ($_GET["eylem"]=="sil"){
        ### 3. Veri tabanından veri silme işlem
    }
    ### 2. Veri Güncelleme İşlemi
?>
```

Veri Güncelleme Formunun Hazırlanması

İlk olarak bir kaydın sütun verilerinde güncelleştirme yapabilmemiz için veri eklemede olduğu gibi, form öğelerini kullanmamız gerekmektedir. Fakat burada önemli bir ayrıntıdan bahsetmemiz gerekiyor. Veri ekleme için boş form ögelerine değerler yazıyorduk. Şimdi ise değerleri girilmiş verimiz üzerinde değişiklik yapmak istiyoruz. Dolayısı ile, güncelleme yapmak istediğimiz kaydın sütun değerlerini veri tabanından çekerek, form elemetleri içerisine yazmamız gerekmektedir.

```php
### Veri Güncelleme İşlemleri
$sql = "select * from kullanici where kullaniciAdi='".$_GET["kadi"]."'";
// Veritabanındaki kullanici tablosundaki tüm kayıtları getiren sql cümlesi...

$sorgulamaislemi = mysql_query($sql);
// select cümleciğinin mysql_query ile veritabanında çalıştırılmasını sağladık.

if ($sorgulamaislemi != false) {
    // sorgu sonrasında bir hata ile karşılaşılmamışsa,
    // artık verileri form halinde yöneticiye gösterebiliriz.
    echo "<form action=yonetim.php method=post>";
    echo "<table>";

    // Tablo içeriğinin kullanici tablosundaki verilerle doldurulması
    $satir = mysql_fetch_assoc($sorgulamaislemi);
    // $sogulamaislemi kaynağında aynı kullanıcı adı ile birden fazla kayıt olamayacağı
    için while döngüsü açmadık.
    // kullanici tablomuzda birincil anahtar olarak kullaniciAdi sütununu atamıştık.

    echo "<tr>";
    echo "<td>Kullanıcı Adı: <input name=kadi type=text value='".$satir['kullaniciAdi'].
    "'/></td>";
    echo "<td>Şifresi:<input name=sifre type=text value='".$satir['sifre']."'/></td>";
    echo "<td>Adı Soyadı: <input name=adsoyad type=text value='".$satir['adiSoyadi'].
    "'/></td>";
    echo "<td>E-postası:<input name=eposta type=text value='".$satir['ePosta']."'/></td>";
    echo "<td>Özgeçmişi:<textarea name=ozgecmis>".$satir['ozgecmisi']."</textarea></td>";
    echo "<td>Rolü:<input name=rol type=text value='".$satir['rolu']."'/></td>";
    // form öğeleri yazdırılarak, form değerleri ilgili dizi elemanlarına eşitlendi.

    echo '<td><input type="submit" value="Kaydet" /></td>';
    // son sütunda tipi submit olan bir kaydet butonu oluşturulur.

    echo "</tr>";
    // Satır etiketi kapatılır.

    echo "</table>";
    echo "</form>";
    // Tablo ve form etiketleri kapatılır
```

Ekran Görüntüsü

yonetim.php

Kullanıcı Adı:	Şifresi:	Adı Soyadı:	E-postası:	Özgeçmişi:	Rolü:	
Myildiri	09876	Mutlu Yıldırım	happy@test.com	Ankara Üniversitesi Siyasal Bilimler ve kamu Yönetimi	0	Kaydet

Kullanıcı Adı	Şifre	Adı Soyadı	e-posta	Özgeçmişi	Rolü	Düzenleme	Silme
dnzyil	123456	Denizer Yıldırım	test@test.com	Doğum tarihim ... . Eğitim durumum ... . görevlerde bulundum.	1	Düzenle	Sil
Myildiri	09876	Mutlu Yıldırım	happy@test.com	Ankara Üniversitesi Siyasal Bilimler ve kamu Yönetimi mezunuyum.	0	Düzenle	Sil
Sguzel	345987	Selvi Güzel Yıldırım	selvi@test.com	MEB Fen ve Teknoloji Öğretmeni...	0	Düzenle	Sil

Veri Güncelleme İşlemi

Oluşturmuş olduğumuz form üzerinde yönetici değişiklik yaptıktan sonra kaydet butonuna bastığında sayfa tekrar yönetim.php sayfasına post edilecektir. Dolayısı ile herhangi bir form elementinin post edilip edilmediğinin kontrolünden sonra artık veri güncelleme için gerekli olan sorguyu oluşturabiliriz.

```
if (isset($_POST["kadi"]) == true)
{
    // kadi isminde bir form elementi post edilmişse anlıyoruz ki,
    // bizden güncelleştirme yapmamız isteniyor.

    $kadi = "'".$_POST["kadi"]."'";
    $sifre = "'".$_POST["sifre"]."'";
    $adsoyad = "'".$_POST["adsoyad"]."'";
    $eposta = "'".$_POST["eposta"]."'";
    $ozgecmis = "'".$_POST["ozgecmis"]."'";
    $rol= "'".$_POST["rol"]."'";
    // update sql cümlesi için gerekli olan değerleri
    // uygun biçimde yeni değişkenlere aktarıyoruz.

    $updatesql = "Update  kullanici set kullaniciAdi=$kadi, sifre=$sifre
    , adiSoyadi=$adsoyad, ePosta=$eposta, ozgecmisi=$ozgecmis,rolu=$rol
    where kullaniciAdi=$kadi";
    // daha sonra yeni değişkenleri update cümleciğinde uygun yere
    ekliyoruz.

    mysql_query($updatesql) or die ("Veri güncelleme işlemi başarısız
    oldu. Hata: ". mysql_error());
    // Son olarak oluşturmuş olduğumuz update cümleciğini veritabanına
    çalıştırılması için gönderiyoruz.
}
```

Veri güncelleme işlemi yapıldıktan sonra veri tabanındaki verilerin güncel hâli ekranda gösterilecektir. Örneğin "Myildiri" kullanıcı isimli kaydın mail adresi ve rolünü değiştirelim.

yonetim.php

Kullanıcı Adı	Şifre	Adı Soyadı	e-posta	Özgeçmişi	Rolü	Düzenleme	Silme
dnzyil	123456	Denizer Yıldırım	test@test.com	Doğum tarihim ... . Eğitim durumum ... . görevlerde bulundum.	1	Düzenle	Sil
Myildiri	09876	Mutlu Yıldırım	mutluyildirim@test.com	Ankara Üniversitesi Siyasal Bilimler ve kamu Yönetimi mezunuyum.	1	Düzenle	Sil
Sguzel	345987	Selvi Güzel Yıldırım	selvi@test.com	MEB Fen ve Teknoloji Öğretmeni...	0	Düzenle	Sil

Veri Silme İşlemi

```
### Veri Silme İşlemleri
$kadi= "'". $_GET["kadi"]. "'";
// delete sql cümlesi için gerekli olan değeri
// uygun biçimde yeni değişkenlere aktarıyoruz.

$silmesql ="Delete from kullanici where kullaniciAdi='". $_GET["kadi"].
"'";
// daha sonra yeni değişkenleri update cümleciğinde uygun yere ekliyoruz.

mysql_query($silmesql) or ("Veri silme işlemi başarısız oldu. Hata: ".
mysql_error());
// Son olarak oluşturmuş delete cümleciğini veritabanına çalıştırılması
için gönderiyoruz.
```

Veri silme işlemi yapıldıktan sonra veri tabanınındaki verilerin güncel hali ekranda gösterilecektir. Örneğin üç kaydımız vardı. "Myildiri" kullanıcı isimli veriyi sildiğimizde ekranda sadece iki kayıt gösterilmelidir.

Ekran Görüntüsü

yonetim.php

Kullanıcı Adı	Şifre	Adı Soyadı	e-posta	Özgeçmişi	Rolü	Düzenleme	Silme
dnzyil	123456	Denizer Yıldırım	test@test.com	Doğum tarihim ... . Eğitim durumum ... . görevlerde bulundum.	1	Düzenle	Sil
Sguzel	345987	Selvi Güzel Yıldırım	selvi@test.com	MEB Fen ve Teknoloji Öğretmeni...	0	Düzenle	Sil

7. WEB TABANLI PROJE GELİŞTİRME

7. WEB TABANLI PROJE GELİŞTİRME

Şimdiye kadar web temelli programlamada kullanılan HTML5, CSS3, Javascript, PHP ve Mysql hakkında bilgi sahibi olduk. Kitabın bu bölümüne geçmeden önce sizlerden bu dillerin temel yazım kurallarını, arasındaki farklılıkları kavramış olmanız ve temel ve orta seviyedeki örnekleri uygulayabiliyor olmanız beklenmektedir. Bu bölümde ise bu dillerin hepsini kullanarak bir proje geliştirmeniz beklenmektedir. Böylece, kitabın bu bölümüne kadar öğrenmiş olduklarınızı seçmiş olduğunuz bir proje temelinde daha detaylı ve bütünsel bir şekilde uygulama fırsatı bulacaksınız.

Bir web sitesi geliştirme projesinde, sürecin başından sonuna kadar size yol haritası olabilecek aşağıdaki gibi bir yöntem izlemeniz önerilir.

PLANLAMA

İlk olarak web sitesinin ne amaçla kullanacağınızı açıkça ortaya koymanız gerekmektedir. Bir web sitesinin amaçlarının önceden belirlenmesi zamanınızı daha verimli kullanmanızı sağlayacaktır. İyi bir web sitesinden beklenen kullanıcının her şeye ulaşabilmesi değildir. Aksine kullanıcının amaçladığı eylemi kolay ve çabuk bir şekilde yapabilmesidir.

Bununla birlikte, web sitesini kimlerin kullanacağı ve hangi bilgisayar ortamlarında (akıllı telefon, tablet, düşük çözünürlüklü kişisel bilgisayar ya da yüksek çözünürlüklü kişisel bilgisayar) kullanılacağı ile karar vermelisiniz. Böylece web sitesinin tasarımını, kullanıcının yaş, bilgisayar kullanma becerilerine uygun ya da hangi cihazların kullanımı daha ön plana çıkıyorsa ona uygun yapmanız mümkün olabilir.

Web tasarımcısının web sitesi içeriği hakkında karar vermesi gerekir. Web sitesinin metin ve resimleri dışında video, ses içerecek mi? İçerecekse bu video ve ses dosyaları nasıl temin edilecek? İnternetten indirilecekse, telif hakları ile ilgili sorun olur mu? Bununla birlikte "Web sitesinde yetkilendirilmiş kullanıcılar olacak mı ya da yetkilendirilmiş kullanıcılar web sitesinde neler yapabilecek?" sorularına da cevap vermiş olmanız gerekmektedir. Örneğin hangi site içeriklerine yetkilendirilmemiş kullanıcılar erişebilecek? Böylece web sitenizin veri tabanının gereksinimleri ile ilgili de fikir sahibi olabilirsiniz.

TASARIM

İnternet ortamında daha önceden benzer siteleri araştırınız. Amacınıza göre hangi tür bir web sitesinin tasarımının daha uygun olacağına karar veriniz. Bu sitelerdeki gibi bir tasarım sizin için yeterli midir yoksa siz yeni bir tasarım ortaya koyabilir misiniz?

Siteniz menü yapısı nasıl olacak? Örneğin site menüsü tüm sayfalarda görünecek mi ya da her sayfa için ayrı bir menü mü olacak? Hangi menü yapısını kullandığınızda, web sitenizin amacına ve kullanıcısına daha uygun olur?

En basit ve sade şekilde sitenin amacına uygun bir tasarımı kağıt üzerinde çiziniz. Sitenizde kaç sayfa olmasını bekliyorsanız bu sayfalar arasındaki bağlantıları da çiziniz. Eğer bu sayfalar arasında veri taşınma gereksinimi varsa kağıt üzerinde hangi yöntemi kullanacağınızı da belirtiniz.

Tüm ayrıntıları ile web sitenizin tasarımını ve sayfalar arasındaki bağlantıları çizdikten sonra başka bir arkadaşınızla, bilişim öğretmeni ve web sitesini kullanacak en az 3 kişi ile tasarım üzerinde konuşunuz. Böylece gözden kaçırmış olduğunuz ayrıntıları yakalama ve sorunlara çözüm bulma olanağını bulabilirsiniz.

Tasarımınızı kağıt üzerinde tamamladıktan sonra web sitenizin hangi bölümlerinde hangi programlama dillerini kullanacağınıza karar vermiş olmalısınız. Daha sonra "Web sitesinde gereksinim duyduğunuz kodlama ile ilgili daha önceden bir örnek yaptınız mı? Kodun nasıl yazılacağını hatırlıyor musunuz? Ek kaynaklara gereksinim duyuyor musunuz? Kimlerden yardım alabilirsiniz? " sorularına yanıt veriniz.

GELİŞTİRME

Tasarım sürecinden sonra, web sitenizi geliştirme sürecine geçebilirsiniz. Bu aşamada kağıt üzerindeki taslağınız doğrultusunda kitabın daha önceki bölümlerinde öğrenmiş olduğumuz kodlama becerilerini uygulayacaksınız. Bu süreç sizler için hem öğrendiklerinizi pekiştirme hem de yeni kodlama becerileri geliştirmenize olanak sağlayacaktır. Bu süreçte, öncelikle yedekli çalışmanız tavsiye edilir. Kodlama yaparken en basit ve kolay yollardan hedefe ulaşmayı deneyiniz. Çözüm bulamadığınız kodlama bölümlerinde, motivasyonunuzu düşürmeden hemen İnternette araştırmaya yöneliniz. İnternette neyi nasıl arayacağınız konusunda fikriniz yoksa bilişim öğretmeninden yardım isteyiniz.

TEST ETME

Web sitesi, tasarım ve geliştirme aşamaları içerisinde ve son hali ile test edilmelidir.

Web sitesinin içeriği ve işlevselliği ile ilgili test işlemleri

- Sayfa başlıklarını da içerecek şekilde metinleri yazım kurallarına uygun şekilde düzeltmek,
- Sayfalar arası yanlış ya da çalışmayan bağlantıları düzeltmek,
- Görüntülenmeyen ya da metinle tutarsız grafiklerin kaynağını (src) düzeltmek
- Sayfalar arası formlarla taşınan verileri kontrol etmek. Eğer veriler doğru şekilde taşınıyorsa, veritabanına bu verilerin doğru şekilde yazdırılıp yazdırılmadığını kontrol etmek.
- Diğer etkileşimli sayfa öğelerinin kullanıcı eylemine göre doğru şekilde çalışıp çalışmadığını kontrol etmek,
- Düşük hızlı bağlantıda yükleme hızını kontrol etmek için sayfaları test etmek.

Web sitesinin kullanılabilirliği ile ilgili test işlemleri

Kullanılabilirlik web sitesinin tasarımının ne kadar iyi olduğunun ölçüsüdür ve kullanılabilirliği yüksek olan bir site, kullanıcıların amaçlarını gerçekleştirmelerine olanak tanır.

Kullanılabilirlik testi için sitenin amacına uygun görevler belirlenir. Daha sonra kullanıcısı olacak kişilerden bu görevleri gerçekleştirmesi istenir. Her görevin gerçekleştirilip gerçekleştirilemediği ve gerçekleştirildi ise gerçekleştirilme süresi not edilir. Daha sonra test sonuçlarına göre web sitesinde düzeltmeler yapılır. Bu düzeltmeler, gerçekleştirilemeyen ya da uzun sürede gerçekleştirilebilen görevler için kullanıcı görüşleri alınarak sayfa tasarımındaki küçük değişiklikler renklerin belirginleştirilmesi, yazı puntolarının büyütülmesi, menü yapısının daha anlaşılır yapılması vb.) olabilir. Görevlerden hiçbiri gerçekleştirilemiyorsa amacına uygun bir site tasarımı yapılamamıştır. Bu durumda tasarımınızı yenilemeniz gerekli olabilir.

WEB TABANLI PROGRAMLAMA İÇİN PROJE ÖRNEKLERİ

Kitabın bu bölümünde sizlere üzerinde çalışabileceğiniz proje örnekleri tanıtılacaktır. Eminim ki siz kodlayıcılar da bu örnekler dışında birçok yaratıcı proje fikirleri ortaya atabilirsiniz. Bununla birlikte, projenin kodlanması için gerekli planlamaları yaparak öğrenmiş olduğumuz dilleri bu projeleri geliştirme amacıyla uygulayabilirsiniz.

Kişisel Gelişim Sitesi

Hatırlarsanız kitabın ilk iki bölümü sonunda bir site şablonu oluşturmuştuk. Bu şablon üzerinden devam

ederek kitap içerisinde uygulamış olduğumuz Javascript, Mysql ve Php örneklerini şablon içerisine yerleştirebilirsiniz. Bu site sizin Bilgisayar Bilimi dersindeki kişisel gelişim siteniz olabilir. Bununla birlikte, bu siteyi okulunuzdaki Bilgisayar Bilimi dersinin bir web sitesi haline getirebilirsiniz. Böylece sizden daha sonra bu dersi alacak alt sınıflardaki arkadaşlarınız için yol gösterici bilgiler içeren bir site kodlamış olursunuz. Öte yandan, her yıl ortaya çıkan özgün projeleri bu siteden duyurarak projelerinizin geliştirilmesini sağlayabilirsiniz.

Fotoğraf Paylaşım Sitesi

Her yaz döneminde sınıf arkadaşlarınızın neler yaptıkları ile fikir sahibi olabileceğiniz resimlerin paylaşılabileceği basit bir sosyal ağ sitesi geliştirebilirsiniz.

- Bu site basitçe üç sayfadan oluşabilir. Birincisi, site açıldığında paylaşılanların görülebileceği ana sayfa, ikincisi arkadaşlarınızın paylaşım yapmak için oturum açma sayfası ve sonuncusu arkadaşlarınızın oturum açtıktan sonra resim yükleme sayfası.
- Bu proje kapsamında paylaşım yapan kullanıcıyı, paylaşım tarihini ve paylaştığı resmin dosya ismini tutan bir veri tabanı oluşturabilirsiniz.
- Site ana sayfasını ilk ziyarette bu tablodaki verileri çekerek tasarımınızda (Ör: facebook sayfanızdaki gibi) uygun yerlere yazdırarak, resimlerin paylaşım yapan kişi ile birlikte alt alta görüntülenmesini sağlayabilirsiniz.
- Eğer çok fazla kullanıcı tarafından paylaşım yapılırsa ve dolayısı ile veri tabanında çok fazla kayıt varsa veri tabanından gelen verileri sayfalandırarak gösterebilirsiniz.
- Bu projeyi daha da ilerletmek isterseniz facebook ve youtube gibi siteleri inceleyerek sayfalandırma yerine sayfanın altına ilerledikçe veri gelmesini sağlayan Javascript kütüphaneleri kullanabilirsiniz.

Ödev Notlandırma Sistemi

Bilişim öğretmeni için sizlere vermiş olduğu ödevleri kolayca takip ederek değerlendirebileceği bir web sitesi geliştirebilirsiniz. Bu siteyi basitçe senaryolaştıracak olursak;

- Siteye ilk erişildiğinde karşımıza bir oturum açma sayfası gelebilir.
- Oturum açan kullanıcının rolüne göre (öğrenci, öğretmen) kullanıcıyı öğrenci ya da öğretmen sayfasına yönlendirebiliriz.
- Öğrenci sayfasında kullanıcının haftalara göre gönderim yapabileceği bir form tasarımı yapabiliriz. Bu sayfada aynı zamanda daha önce gönderim yapılan ve öğretmen tarafından değerlendirilmiş ödevlerin notu ile birlikte listelenmesi sağlanabilir.
- Öğretmen sayfasında öğrenciler tarafından gönderim yapılmış ödevlerin listelenmesini sağlayabiliriz. Bununla birlikte, listelenen ödevlerden herhangi birine tıklandığında ödevle ilgili not ve açıklama girilmesini sağlayacak bir form tasarımı gerekir. Bu işlemi farklı sayfalarda yapabileceğiniz gibi aynı sayfa içerisinde de kodlamanız mümkündür.

Bu senaryoya göre veri tabanımızda kullanıcılar ve ödevlerle ilgili iki tablo tutulması yeterli olabilir. Fakat ödev tablosunu oluştururken öğrenci rolündeki bir kullanıcının birden fazla ödev gönderiminin olabileceğini göz önünde bulundurunuz ve birincil anahtar olacak sütunları buna göre belirtiniz.

Kodlama Öğretiyorum Sitesi

"En iyi öğrenme anlatma yoludur." varsayımından hareket ederek iyi bir kodlayıcı, yazmış olduğu

kodu sade ve anlaşılır şekilde anlatabilendir. Böyle bir bakış açısı temelinde, siz de kod yazarken anlaşılabilir şekilde videolarınızı çekebilir ve bu videoları paylaşabileceğiniz basit bir blog sayfası tasarlayabilirsiniz. Böylece, akranlarınızla, aynı işi farklı kodlama yöntemleri ile yapabilen çözümleri paylaşabilir ve kodlama becerilerinizi geliştirebilirsiniz.

Şimdi basit bir şekilde bu sitenin senaryosunu yazalım. Siteye ilk giriş yapıldığında, karşımıza HTML5, CSS3, MYSQL Yönetimi ve PHP başlıklarını içeren bir menü yapısı geleceğini hayal edin. Bu menüler alt başlıklar da içerebilir. Başlıklardan herhangi birine tıklandığında, sizlerin videoya çekmiş oldukları kodlama örnekleri, tıklanan başlığa göre veri tabanından çağrılsın.

Peki, bu şekilde bir senaryo için veri tabanımızda neler olmalı? Öncelikle temel kullanıcı bilgileri ile birlikte, sadece yetkilendirilen kişilerin video ekleyebilmesi için kullanıcı rolünü tutan bir tablo gereklidir. Bununla birlikte kullanıcıların eklemiş oldukları videonun yolu ve dosya ismini tutan videolar tablosu gereklidir. Video tablosunu oluştururken bir kullanıcının birden fazla video gönderiminin olabileceğini göz önünde bulundurunuz ve birincil anahtar olacak sütunları buna göre belirtiniz.

Soru Cevap Sitesi

ÖSYM sınavlarına yönelik okulunuz öğrencilerinin soru sorabilecekleri, yine okulunuz öğretmenleri tarafından bu soruların yanıtlanabileceği bir soru cevap sitesi tasarlayabilirsiniz. Öğrencilerin sitede soru sormalarını teşvik etmek için haftanın en değerli sorusunun seçimine yönelik bir kodlama yapabilirsiniz. Siteyi ziyaret eden kullanıcılar sorunun önemine göre bir değerlendirme notu verebilir. Bu notların ortalamasına göre haftalık önemli soruları listeleyebilir ve en önemli soruyu soran kullanıcının sitenin ana sayfasında görüntülenmesini sağlayabilirsiniz.

Anket Sitesi

Herhangi bir konuda okulunuz öğrencileri ya da velilerinin görüşlerini toplamak için bir anket sitesi tasarlayabilirsiniz. Bu site için okulunuz yöneticileri ya da öğretmenleri ile görüşerek önceden belirlenmiş anket sorularını internet ortamına geçirebilir, daha sonra web sitesinde hazırlamış olduğunuz anketi okulunuz öğrencileri ya da velilerine duyurabilirsiniz. Öğrencileri ve velilerden almış olduğunuz yanıtları bir veri tabanında tutabilir ve istenildiğinde bu verilere ilgili birimlerin erişebilmesini sağlayabilirsiniz. Bununla birlikte projenizi biraz daha geliştirmek isterseniz, yetkilendirilmiş kullanıcıların site içerisinde anket oluşturmasını sağlayarak sitenize daha dinamik bir işlev kazandırabilirsiniz.

Kıyaslama Sitesi

Popüler ürün ya da servisleri kıyasladığınız bir site tasarlayabilirsiniz. İnsanlara hangisinin daha iyi olduğunu gözlemlerinize ve araştırmalarınıza dayalı olarak bu web sitesinde gösterebilirsiniz.

Deney Sitesi

Fen laboratuvarındaki deney araçları ile yapılan deney videolarının paylaşıldığı bir web sitesi oluşturabilirsiniz. Bu web sitesini laboratuvarda bulunan araçları listeleyebilecek, kolayca güncelleştirilmesini sağlayabilecek ve istenildiğinde bu araçlar ile ilgili tanıtıcı bilgilere de ulaşılabilecek şekilde genişletebilirsiniz.

Ortak Wiki Alanı

Herhangi bir derse (Fizik, Kimya vb.) ilişkin terimlerin tanımlarının paylaşıldığı sınıflar arasında ortak kullanılabilen bir web sitesi oluşturabilirsiniz.

Kantin Sitesi

Okulunuzda işletilen kantinde satılan ürünlerin fiyatlarının takip edilebileceği, kantin işleticisi tarafından fiyat güncelleştirmelerinin yapılabileceği bir web sitesi tasarlayabilirsiniz.

Tasarruf Sitesi

Para tasarrufunun nasıl yapılması gerektiği hakkında bir site tasarlayabilirsiniz. Bunun için yaşlılar, üniversite öğrencileri, ev kadınları ile konuşarak onların deneyimlerini sitenizde paylaşabilirsiniz. Sitenize kullanıcılar üye olabilir ve kendi deneyimlerini paylaşabillir.

Özet

Yukarıdaki proje örnekleri, projenizi tasarlayıp geliştirmeden önce örnekler üzerinde tartışarak ufkunuzu genişletmek amacıyla paylaşılmıştır. Sadece bu site örnekleri ile sınırlı kalmayabilirsiniz. Bilgi çağı toplumundaki sorunlara çözüm olabilecek yeni proje fikirleri ortaya koyabilir ve bu projeleri web tabanlı programlama dilleri ile geliştirebilirsiniz.

III. BÖLÜM

MOBİL PROGRAMLAMA

1. MOBİL UYGULAMA GELİŞTİRMEYE GİRİŞ

1.1. Mobil Uygulama Geliştirmeye Giriş

İnternet bilgiye erişim süreçleri üzerinde çok önemli gelişime yol açmıştır. Bilgi kaynaklarına erişim ağ üzerinden gerçekleşmeye başladığı andan itibaren; mekandan bağımsız olarak bilgiye erişim de mümkün hâle gelmiştir. İnternet'e bağlı bir bilgisayardan çok farklı hizmetlere (e-devlet, e-finans, e-ticaret vb.) ve kaynaklara erişilmesi ve bunun sağlamış olduğu avantajlar, kullanıcıların bu sürece mobil ortamda da devam etme taleplerini beraberinde getirmiştir. Önceleri bu taleplerin platform bağımsız uygulamalar ile çözlmesi düşünülmüş ancak uygulamanın mobil ortamlarda çalışma sıkıntısı yaşamamasına rağmen görsel uyumluluk problemleri ortaya çıkmıştır. Bulut teknolojisinin kullanımı ve internet tarayıcıların önem kazanması da yine kullanıcıların bu mobil platform tercih taleplerinden kaynaklanmaktadır. Bu talepleri mobil platformlarda karşılayabilmek için farklı yazılım çözümleri hayata geçirilmiştir. Bu çözümlerden mobil cihazlardaki işletim sistemleri için özel olarak geliştirilen yazılımlar "Mobil Uygulama" olarak adlandırılır.

1.2. Mobil Programlamadaki Temel Kavramlar

Mobil programlama, uygulamanın çalışacağı platformun yapısına uygun olarak özellikle çalışma performansı açısından dikkatli bir geliştirme sürecine ihtiyaç duymaktadır. Başta mobil cihazların sınırlı donanım özellikleri ve kısıtlı enerji kaynakları nedeniyle geleneksel program geliştirme yaklaşımının dışına çıkılması gerekmektedir. Ayrıca, programcılıktaki en önemli bileşenlerden olan "görsel tasarım" da mobil programlamada yepyeni dinamiklere sahip olmuştur. Örneğin, mobil cihazlarda çözünürlük ve görüntü kalitesi bilgisayar kontrollü diğer görüntü cihazlarından çok farklıdır.

Bunların yanında, mobil cihazlarda uygulamaların çalışma öncelikleri ve oluşabilecek olay analizleri, geleneksel programcılıktaki yapıdan çok farklıdır. Örnek verecek olursak, mobil cihazlarda standart olarak kimi uygulamalar diğerlerine göre her daim çalışma önceliğine sahiptir. Bunu cep telefonunuzda oyun oynarken bir arama gerçekleştiğinde oyununun kapanması ve arama ekranının belirmesi durumunda görebilirsiniz. Çünkü "arama" uygulaması "oyun" uygulamanızdan önceliklidir. Mobil programlama yaparken bu tür durumlar önceden düşünülmeli ve kullanıcının mobil cihazında bizim gerçekleştirdiğimiz uygulamyı çalıştırırken oluşabilecek mobil platforma özgü durumlar (arama gelmesi, mesaj gelmesi vb.) göz önünde bulundurulmalıdır.

1.3. Uygulama Geliştirirken Kullanılan Tasarım Yapıları

Mobil uygulama geliştirirken kullanılabilecek farklı tasarım yapıları bulunmaktadır. Bu tasarım yapılarının seçiminde uygulamanın kullanım alanı, hedefi, mobil cihazların donanım özellikleri, proje süresi ve maliyeti gibi çok farklı değişkenlerin devreye girdiği rahatlıkla söylenebilir. Mobil programlamada yerel geliştirme (native) olarak ifade edilen mobil işletim sistemine özel olarak kodlanan uygulamalar olduğu gibi, tamamen web tabanlı olarak geliştirilen uygulamalar ya da karma (hibrit) olarak adlandırılan yerel ve web tabanlı uygulamaların iç içe kullanıldığı yapılar da bulunmaktadır.

Native uygulamalar ile hibrit uygulamalar arasındaki en temel fark; biri (native) cihazın kendi ana işletim sistemi tarafından desteklenen bir programlama dili ile yazılmış ve derlenmiş olması, diğerinin (hibrit) ise her türlü sistem için kodlanmış olmasıdır. Her ikisinin olumlu ve olumsuz yönleri vardır. Örneğin, Java ile geliştirilen Android uygulamaları ve Objective C yada Swift ile geliştirilen IOS uygulamaları nativ uygulamalardır ve bu cihazların kaynaklarına doğrudan ulaşabilirler. Bunun yanında HTML5 ve JavaScript gibi hibrit uygulamalar ise her iki tür platformda da çalışabilmekle birlikte cihazların sistem kaynaklarına ulaşabilmek için ara bir katman kullanmak zorundadır.

2. MOBİL DONANIM

2.1. Mobil Donanım

Bilişim cihazları iki temel unsurundan oluşur. Bunlar donanım ve yazılımdır. Donanım, en temel tanım olarak, bir elektronik cihazın fiziksel parçalarıdır. Mobil, diğer bir adıyla taşınabilir, cihazların donanımı ise bilgisayar donanımından faklı değildir. Temelde mobil cihazlar da bilgisayarın küçültülmüş birer örnekleridir, ancak içerisinde taşınabilirliklerine özgü çeşitli sensörler de bulundururlar.

Bu bölümde mobil donanım bileşenleri ve sensörler ile bu donanımların çalışma prensipleri işlenmektedir. Sonrasında mobil işletim sistemleri, farklılıkları sunulmuş; ardından mobil uygulama geliştirme araçları ve kullanım alanları açıklanmıştır. Son olarak, emülatörlerin çalışma mantıkları belirtilmiştir.

2.2. Donanım Bileşenleri

Mobil cihazlar da bilgisayarda bulunan temel donanımlara sahiptir. Bu temel bileşenler aşağıda kısaca anlatılmaktadır.

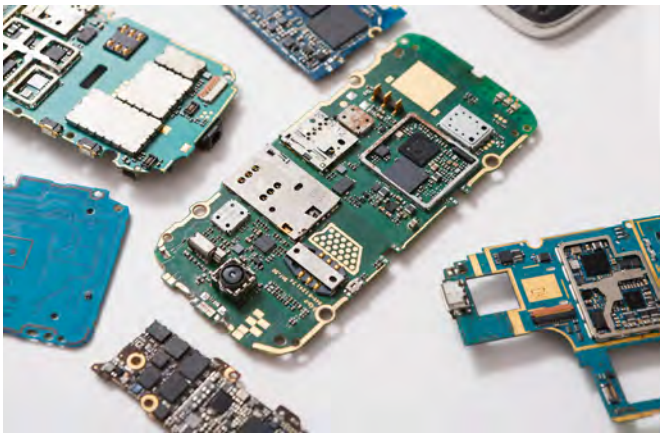
a. Kasa: Bilgisayarda olduğu gibi mobil cihazın tüm iç donanımlarını bir arada tutan birimdir. Cihazın dış görünümünü oluşturur. Yakın zamana kadar boyutlarının küçültülmesi en önemli nokta iken günümüzde boyuttan çok ergonomisi üzerinde çalışmalar yürütülmektedir.

Resim 2.1: Telefon kasası

b. İşlemci: Matematiksel ve mantıksal işlemlerin yapıldığı Merkezi İşlem Birimi (Central Processing Unit-CPU) olarak da adlandırılan en önemli bileşendir. Bilgisayardaki işlemcilerden görev olarak farkı yoktur. Ancak akıllı cihazların en fazla enerji tüketen ve en çok ısının bileşeni işlemcidir ve mobil cihazlarda bilgisayardakinden farklı olarak, soğutma ve enerji tüketimi önem arz etmektedir. Bundan dolayı karmaşık ve üst düzey işlemlere hakim, daha kararlı çalışan X86 mimarisi akıllı telefonlar ve tabletlerde kullanılamaz olmuştur. Bu aşamada ise ARM işlemci mimarisi oluşturulmuş ve mobil cihazlar için vazgeçilmez olmuştur. Günümüzde Apple, Samsung, HiSilicon ve Qualcomm gibi en büyük mobil işlemci üreticileri ARM mimarisini kullanmaktadır.

Resim 2.2: İşlemci

c. Anakart: Mobil cihazın işlemci ve ana belleği başta olmak üzere birçok temel elektronik bileşenini üzerinde bulunduran bileşendir. Bilgisayar anakartından farklı olarak küçüktür ve enerji tasarrufu göz önünde tutularak tasarlanır. Ayrıca sim kart ve GSM iletişimi elektronik bileşenleri de içermektedir.



Resim 2.3: Anakart

d. Ana Bellek (RAM): Ana bellek bilgisayarlarda olduğu gibi programların üzerinde çalıştığı geçici hafıza alanıdır. Mobil RAM, cihazın performansını etkileyen en önemli unsurlardan biridir. Ancak bilgisayardan farklı olarak, mobil cihazlara RAM ekleme veya açık olan uygulamaların RAM kullanımını sınırlandırma gibi seçenekler yoktur. Bundan dolayı yüksek RAM kullanımı gerektiren mobil uygulamalar eski cihazlarda çalışmaz ya da çalışma sırasında kilitlenmeler oluşur. Dolayısıyla mobil yazılım geliştirirken uygulamanın RAM kullanımına çok dikkat etmek gerekir.

e. Ekran: Yakın geçmişe kadar sadece görüntü sunma işi yapan bir çıkış birimi olan ekranlar, günümüzde dokunmatik özellikleri ile hem giriş hem çıkış birimi olarak kullanılmaktadır. Bu durumda mobil cihazların ekranları için; "görüntü kalitesi", "çözünürlüğü" ve "renk derinliği" özelliklerinin yanında "dokunma duyarlılığı", "eş zamanlı dokunma sayısı" ve hatta yeni model mobil cihazlarda "3 boyutlu dokunmatik" (3D-Touch) özelliği diye adlandırılan sert ve yavaş dokunma özellikleri de kalite değerlendirme kriterleri arasında yer alır. En popüler mobil ekran teknolojileri TFT-LCD (Thin Film Transistor Liquid Crystal Display) ve AMOLED (Active Matrix Organic Light Emitting Diode) ve Retina olarak sayılabilir.

Resim 2.6: Telefon ekranları

f. Depolama alanı: Mobil cihazlarda depolama dahili ve harici depolama birimleri tarafından yapılmaktadır. Yalnız bu depolama bileşenleri bilgisayarlardaki gibi farklı kaydetme mimarileri (manyetik ve optik) kullanamaz. Mobil cihazlarda bulunan depolama birimleri flash disk özelliğine sahiplerdir ve elektrik ile kayıt yapma mimarisi kullanırlar. Birçok mobil cihaz dahili depolama birimlerine ek olarak SD, Mini SD ve Micro SD Kart ile harici depolama olanağı da sunmaktadır.

Resim 2.4:

Bu temel bileşenlere ek olarak çeşitli çoklu ortam bileşenleri de mobil cihazların olmazsa olmaz donanımları arasına girmiştir. Bu bileşenlerden bazıları (kameralar, mikrofonlar vb.) aynı zaman daalgılayıcı (sensör) olarak da kullanılmaktadır:

a. Arka kamera: Mobil cihazların en çok kullanılan özelliklerinden birisi resim ve video çekmedir. Ancak arka kamera sadece resim ve video çekmek için değil aynı zamanda Arttırılmış Gerçeklik (Augmented Reality) gibi günümüzde yeni yeni popüler olan uygulamalarda da etkin bir şekilde kullanılmaktadır.

Resim 2.5: Arka kamera

b. Ön kamera: Mobil cihazların resim ve video çekme özelliklerinin hayatımıza kattığı "Özçekim" kavramının olmazsa olmaz donanımıdır. Günümüzde mobil cihazların en sık kullanılan ve kalite değerlendirmesinde en ön planda tutulan donanım bileşeni ön kameradır. Mobil uygulama geliştirirken sıklıkla kullanılmasından dolayı mobil cihaz üreticileri bu donanıma özellikle önem vermekte ve her geçen gün daha kaliteli kamera içeren cihazlar piyasaya sürülmektedir.

c. Hoparlör: Mobil cihazlar için en önemli donanımlardan biridir. GSM görüşmelerini sağlamak için kulaklık çıkışı düzeyinde ses gücüne sahip bir tane olmasının yanında, uygulamalardaki dışarı ses verme özelliklerinin kullanımı amacıyla büyük bir hoparlör daha mevcuttur.

d. Mikrofon: Hoparlör gibi mobil cihazların diğer bir vazgeçilmez donanımıdır. Sadece GSM görüşmeleri ve sesli kayıt oluşturmak için değil günümüzde giderek popülerleşen sesli komut özellikli uygulamalar için de mikrofon önemli bir mobil donanımdır.

Resim 2.7: Telefon mikrofonu

e. Batarya: Temel donanım bileşenlerinden mobil cihazlara özgü olan tek donanım batarya (pil) denilebilir. Lityum-İyon (Li-Ion) özellikli bataryalar günümüzde en popüler bataryalardır. Bu bataryaların tercih edilme nedenleri kullanılmadıkları zaman (boşta beklediklerinde) kayıplarının az olmasıdır. Ancak ömürlerinin kısa olması bir dezavantajdır. Bunun yanında Nikel-Metal Hidrat (Ni-MH) ve Nikel-Kadmiyum (Ni-Cd) bataryalar da vardır. Ancak bu bataryaların en büyük dezavantajı kısa sürede boşalmaları ve dolayısıyla sık sık şarj gerektirmeleridir.

Resim 2.8: Batarya

Mobil cihazların dış dünya ile iletişim kurabilmesi açısından gerekli olan bağlantı modülleri de önemli mobil donanım bileşenleri olarak öne çıkmaktadır:

a. Kablosuz ağ bağlantısı: Bilinen kısa adıyla Wi-Fi, en sık kullanılan ağ ve internet bağlantı teknolojisidir. Bu bağlantı türü için mobil cihazlarda Wi-Fi ağ adaptörü ve anteni mevcuttur. Elektronik cihazlarda bağlantı türleri ve standartlarını belirleyen Uluslararası Elektrik ve Elektronik Enstitüsü'nün (IEEE) belirlediği 802.11 protokolü kablosuz ağlar ve bu ağlar için kullanılan cihazların standartlarından oluşur. Şu anda mobil cihazlarda 802.11g standardı ile 54 Mb saniye hızına ulaşılmaktadır. 802.11g standardında bir mobil cihaz 100 metreye kadar Wi-Fi anten-alan genişliğine sahiptir. Ancak geliştirilmekte olan 802.16 standardındaki cihazlar ile 80 Mb saniye hızı ve 50 km anten-alan genişliğine varılması planlanmaktadır.

b. Bluetooth: Bluetooth iki cihazın yakın mesafede birbirleriyle veri alış-verişi yapabilmeleri için kullanılan bir kablosuz standardıdır. Mobil cihazlarda bluetooth, bu kablosuz teknolojiyi kullanan aksesuarların (kulaklık vb.) sayısının artmasıyla olmazsa olmaz bir donanıma dönüşmüştür. Günümüzdeki mobil cihazlarda 4.0 versiyonu sunulan Bluetooth ile neredeyse Wi-Fi kalitesinde veri alışverişi sağlanabilmektedir ve bu durum uygulama geliştiricilerin mobil cihazlardaki bu teknolojiyi kullanmayı tercih etmelerine neden olmaktadır.

c. GPS: GPS, 24 tane uydunun dünya etrafında kendine özgü bir yörüngeye yerleştirilmesi ve konumlarını koruyarak yeryüzüne sabit konum ve zaman sinyali göndermesi ile oluşturulan bir yön bulma sistemidir. Daha detaylı bilgi sensörler bölümünde açıklanmıştır.

d. NFC: Türkçesiyle Yakın Alan İletişimi (Near Field Communication), diğer iletişim teknolojilerine göre en yeni teknolojidir. Çok yakın mesafede cihazlar arası iletişim kurulmasını sağlamaktadır. Cihazlar arasında iletişim kurulması için Bluetooth ve Wi-Fi'de olduğu gibi eşleştirme veya kayıt yapma gibi ayarlara gerek duyulmaması bu teknolojiyi hızlı iletişim başlatma ve bitirme uygulamalarında ön plana getirmektedir. Örneğin, bankaların cep telefonu ile temazsız ödeme uygulamaları NFC teknolojisi ile yapılmaktadır. Yakın zamanda bazı illerimizde ulaşımda kullanılan kartların özellikleri geliştirilen bir mobil uygulama ile NFC uyumlu cep telefonlarına da kazandırılmış ve yolculara toplu taşıma araçlarında cep telefonu ile ödeme yapma şansı tanınmıştır.

2.3. Donanım Bileşenlerinin Çalışma Mantıkları

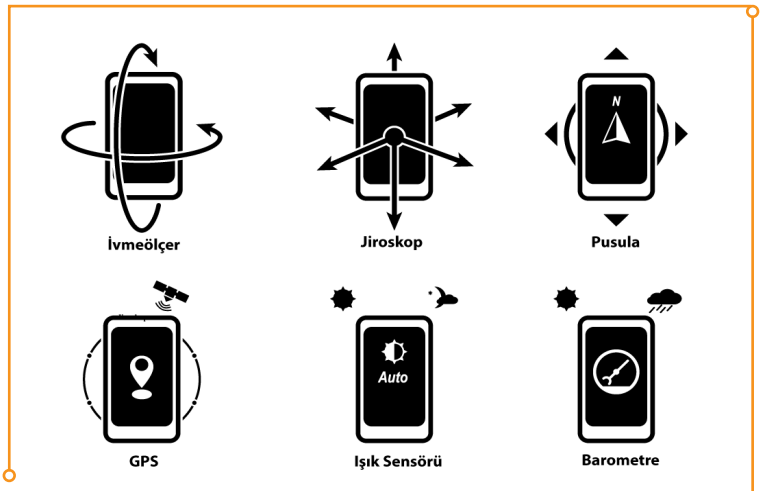
Mobil cihazlardaki donanım bileşenlerinin çalışma mantığı, temel bilgisayar çalışma mantığı ile aynıdır. Sadece mobil cihazların güç yönetimi, ısı dağılımı ve diğer sınırlılıklarını dikkate almak gerekir. Tüm elektronik sistemlerde olduğu gibi, tüm işlemlerin gerçekleştirildiği ve genel denetim işlemlerinin yapıldığı bir merkezi işlem birimi (CPU) bulunmaktadır. Bu merkezi işlem birimi günümüz teknolojisinin izin verdiği sınırlar çerçevesinde çoğunlukla çok çekirdeğe sahiptir ve ek olarak görüntü işleyen ek bir işlem birimi de artık çoğu mobil cihaz için standart hâle gelmiştir. İşlemlerin geçici olarak işlendiği bellek, yapılan çalışmaların kaydedildiği depolama alanı ve tüm donanımın birbiri ile bağlantısının kurulduğu anakart genel yapıyı oluşturur.

2.4. Donanım Bileşenlerinin Programlanması

Donanım bileşenlerinin programlanması, genel olarak ilgili donanım bileşeni için oluşturulmuş kütüphanenin uygulama içerisinden çağrılarak kullanılması şeklinde gerçekleştirilir. Mobil cihazlarda çalışacak işletim sistemlerinin geliştirme ortamlarında donanıma özel olarak hazırlanmış fonksiyonlar donanım kütüphanesi olarak adlandırılır.

2.5. Mobil Cihazlarda Yer Alan Sensörler

Mobil cihazlarda birçok farklı görev ve işlemi gerçekleştirmek için sensörler (algılayıcılar) bulunmaktadır. Mobil donanımı belki de diğer bilgisayar sistemlerinden ayıran en önemli özellik sahip oldukları bu sensörlerdir. Bu algılayıcıları insanlarda bulunan duyu organlarına benzetmek mümkündür. Bir mobil cihazın kendi bulunduğu konumu, yeryüzüne göre aldığı açıyı, ışık koşullarını ve benzeri diğer birçok bilgiyi toplayabilmesini sensörler sağlamaktadır. Mobil donanımlar içerisinde en çok yer verilen ve artık neredeyse standart hâle dönüşen sensörler aşağıda anlatılmıştır.



Resim 2.9: Mobil cihaz sensörleri

a. İvmeölçer (İvme Sensörü) –Accelerometer

İvmeölçer sensörü mobil cihaza etki eden hızlanma kuvvetini ölçmektedir. Yatay (X), dikey (Y) ve sırt üstü durma (Z) eksenleri olmak üzere mobil cihazın bu pozisyonlardaki ivmelenmesi ölçülmektedir. Bu ölçümde yerçekimi kuvveti de dâhildir.

Bu sensör daha çok hareket uygulamalarında, telefon pozisyon bilgisi gerektiren uygulamalarda, cihaz hareketlerini izlemede, adım izlemede, oyun sürecini kontrol etmek için harekete duyarlı mini oyunlarda, çeşitli sağlık uygulamalarında kullanılmaktadır.

b. Doğrusal İvme Sensörü (Hızlanma Sensörü) –Linear Accelerometer

Doğrusal ivme sensörü mobil cihaza etki eden hızlanma kuvvetini yerçekimi kuvveti olmadan ölçmektedir.

Doğrusal ivme = İvme - Yerçekimine bağlı ivme

Aracınızın ne kadar hızlı gittiğini görmek için, bazı oyunlarda telefonu ya da tableti öne eğerek gaz vermeyi, arkaya doğru eğerek fren yapmayı sağlamak için kullanılmaktadır.

c. Jiroskop –Gyroscope

Jiroskop sensörü mobil cihazın 3 eksende gerçekleştirdiği açısal hız bilgisini (dönme hızı) ölçmektedir. Bu eksenler ivmeölçerdeki eksenlerin aynısıdır. (X, Y ve Z eksenleri)

Bu sensör ivmeölçer ile birlikte kullanıldığında verilerde netlik miktarı artmaktadır. Telefonun yön ayarlamasında, ekran döndürme özelliğinde, hareket tanıma uygulamalarında, sanal gerçeklik uygulamalarında kullanılmaktadır.

d. Yerçekimi Sensörü –Gravity Sensor

Yerçekimi sensörü mobil cihaza uygulanan yerçekimi kuvvetini üç eksende ölçmektedir (X,Y ve Z eksenleri). Çıktı olarak yerçekiminin yönünü ve büyüklüğünü gösteren üç boyutlu bir vektör verir. Mobil cihazın eğimini hesaplamak için kullanılmaktadır.

e. Basınç Sensörü –Pressure Sensor

Basınç sensörü mobil cihazın bulunduğu ortamdaki hava basıncını ölçmektedir. Bu sensör ışık, nem ve sıcaklık sensörleri gibi ortam sensörlerindendir.

Basınç sensörü GPS'e yardımcı olarak kullanılan bir sensör olması dışında, birçok sağlıklı yaşam uygulamalarına da katkıda bulunmaktadır.

f. Sıcaklık Sensörü –Temperature Sensor

Sıcaklık sensörü mobil cihazın bulunduğu ortamdaki sıcaklığı ölçmektedir. Bu sensör ışık, nem ve basınç sensörleri gibi ortam sensörlerindendir.

g. Işık Sensörü –Light Sensor

Işık sensörü mobil cihazın bulunduğu ortamdaki ışık seviyesini ölçmektedir. Bu sensör sıcaklık, nem ve basınç sensörleri gibi ortam sensörlerindendir. Ortam sensörlerinde herhangi bir veri işleme ve filtreleme işlemine ihtiyaç duyulmaz. Bu yüzden en kolay kullanılan sensörler olarak bilinir.

Işık sensöründen alınan veriler sayesinde mobil cihaz ekran parlaklığını ayarlar. Bu sayede ortam değişikliklerinde gözlerimizin etkilenmemesi ve mobil cihazın şarjının tasarruflu kullanılması sağlanmaktadır.

h. Nem Sensörü –Humidity Sensor

Nem sensörü mobil cihazın bulunduğu ortamdaki nem miktarını ölçmektedir. Bu sensör sıcaklık, basınç ve ısı sensörleri gibi ortam sensörlerindendir.

i. Yakınlık Sensörü –Proximity Sensor

Yakınlık sensörü bir nesnenin mobil cihaza olan yakınlık-uzaklık durumunu ölçmektedir. Bu sensör kullanılarak kullanıcıların telefon görüşmesi yaptığında mobil cihaz ekranının sönmesi ve diğer özelliklerinin engellenmesi sağlanır. İşlevi nedeniyle en aktif çalışan sensörlerden biridir.

j. Parmak İzi Sensörü –Fingerprint Sensor

Parmak izi sensörü dokunmaya dayalı bir sensördür. Parmağa gönderilen elektrik akımı sayesinde parmak izindeki girinti çıkıntılardan bir veri oluşturulur ve bu veri hafızaya kaydedilir. Kullanıcı tanımlama, ekran kilidi açma gibi durumlarda kullanılmaktadır.

k. GPS –Global Positioning System

GPS mobil cihazın uydulardan gelen sinyaller doğrultusunda bulunduğu konumu hesaplamaktadır. Birden fazla uyduya bağlantı kurulduğundan mobil cihazın dünya üzerindeki yerini bulmada çok başarılı bir sensördür. Bu bağlantılar için internete gerek duymaz. Bilgi alışverişini radyo sinyalleri ile yapar. Bu sensör sadece navigasyon ve yön bulma uygulamalarında değil, sosyal medya uygulamalarından oyunlara kadar birçok mobil platformda sıklıkla kullanılmaktadır. Ayrıca GPS'den gelen ivme verisi verileri ve konum tahminleri kombinasyonu, bir kullanıcının bisiklet ya da araba kullanması ya da bir otobüse ya da metroya binmesi gibi ulaşım şeklini algılayabilmektedir. GPS sensörü kullanımının tek dezavantajı çok fazla batarya tüketmesidir.

l. Manyetik Ölçer (Manyetometre)

Manyetik ölçer mobil cihazın bulunduğu ortamdaki manyetik alanı üç eksen üzerinde ölçmektedir (X, Y ve Z eksenleri). Bu sensör mobil cihazın manyetik kuzey kutbuna göre konumunu belirlemek, dünyanın manyetik alanı üzerindeki değişiklikleri gözlemlemek, GPS ve navigasyon işlemlerine yardım etmek amaçlı kullanılmaktadır.

Yukarıda bahsedilen sensörler dışında mobil cihazların bazı modellerinde adım sayacı sensörü, göz tarayıcı sensör, kalp ritim ölçer vb. farklı sensörler de bulunmaktadır. Ayrıca mobil cihazlarda bulunan mikrofon ve kamerada sensör görevi görmektedir. Bu donanımlar mobil donanım bölümünde anlatıldığı için burada anlatılmamıştır. Tüm sensörler mobil cihazımızı daha etkili kullanma, dış çevreyle etkileşim halinde olma ve bu verileri kullanarak birçok uygulamayı kullanma imkânı sunduğu için çok önemlidir.

2.6. Mobil Cihazlarda Yer Alan Sensörlerin Çalışma Mantıkları

Mobil cihazlarda yer alana sensörlerin çalışma mantıkları, donanım bileşenlerinin çalışma mantıkları ile paralellik gösterir. Mobil işletim sistemlerinin geliştirme paketleri içerisinde sensörlere özel olarak geliştirilmiş kütüphaneler bulunur. Bu kütüphanelerin uygulama içerisinden çağırılması ile birlikte ilgili sensöre erişim sağlanır ve kontrol edilebilir bir yapı ortaya çıkar.

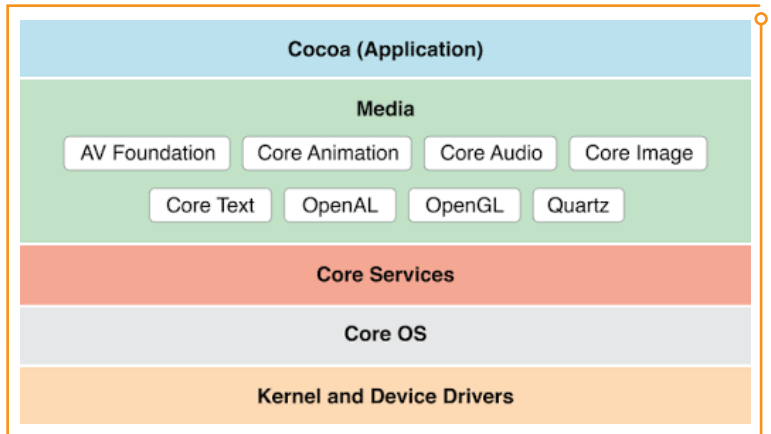
2.7. Mobil İşletim Sistemleri

Akıllı telefonlar, tabletler, PDA'lar ve diğer akıllı taşınabilir cihazların üretilmesi ile beraber mobil işletim sistemleri de ortaya çıkmıştır. Mobil işletim sistemleri kullanıldığı cihazların donanımını doğrudan denetleyen ve yöneten, cihazların içine yüklenecek uygulamaların kullanımını sağlayan sistem yazılımlarıdır. Cihazların üretimini yapan şirketler tarafından geliştirilen, farklı felsefelerle/yöntemlerle, açık veya kapalı kaynak kodlu olarak çalışan çeşitli mobil işletim sistemleri bulunmaktadır. Bunlardan sıklıkla kullanılanları ise Apple tarafından üretilen IOS, Google ve Open Handset Allience tarafından üretilen Android ve Microsoft tarafından üretilen Windows Phone işletim sistemleridir.

2.7.1. IOS

IOS ilk olarak iPhone OS ismiyle Mac OS (Macintosh Operating System)'tan türetilerek 2008 yılında Apple şirketi tarafından piyasaya sürüldü. Şirket 2010 yılında işletim sistemine IOS ismini vererek, cihazların donanım teknolojisinin gelişimine paralel olarak pek çok yenilikler içeren yeni sürümlerini günümüze kadar geliştirmiştir. IOS'un özelliği sadece Apple tarafından üretilen cihazlarda kullanılabilmesi ve kullanıldığı cihaza AppStore ve iTunes gibi platformlar dışından uygulama yüklemeye izin vermemesidir.

IOS ilk zamanlarda iPhone'lar için geliştirilmiş fakat daha sonra iPod Touch ve iPad'in ortaya çıkması ile bu cihazlarda da kullanılmaya başlanmıştır. IOS Linux tabanlı bir işletim sistemi olup, 5 katmandan oluşmaktadır.



Tablo 2.1: IOS işletim sistemi katmanları

Kerneland Device Drivers (Çekirdek ve Aygıt Sürücüleri): Bu katman cihazın sürücü yazılımları, BSD kütüphaneleri, düşük seviye bileşenleri, dosya sistemi güvenliği için destek birimini, işlemler arası iletişimi ve benzeri işlevleri içermektedir.

Core OS (Çekirdek İşletim Sistemi): Donanım ve ağ ile ilgilidir. CPU (Merkezi İşlem Birimi) veya GPU (Grafik İşlem Birimi) üzerinde çalışan yüksek performanslı görevler için arayüzler içermektedir.

Core Services (Çekirdek Servisler): Temel hizmetler (düşük seviye ağ iletişimi gibi), otomatik referans sıralaması, veri formatlama ve (string) manipulasyonu sağlamaktadır.

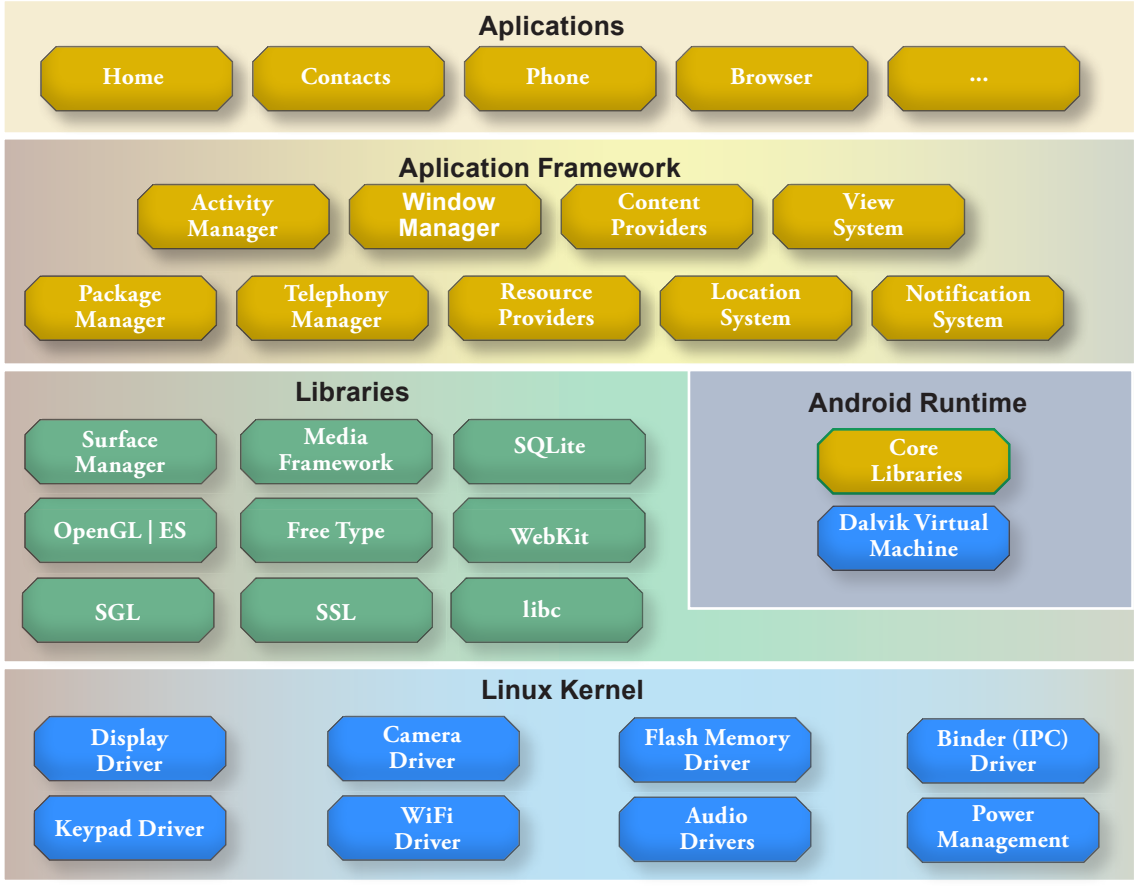
Media: Medyayı oynatma, çekme (recording), görsel-işitsel medyaları düzenleme, 2D ve 3D grafiklerin sunulması gibi işlevleri yerine getirmektedir.

Cocoa Touch: Uygulama kullanıcı arayüzlerini içerir. Kullanıcı olaylarına cevap verir ve uygulama davranışlarını yönetir. Bir uygulama geliştirici çeşitli geliştirme araçları ve mobil programlama dillerini kullanarak bu katmanda çalışır.

2.7.2. Android

Android işletim sistemi, 2003 yılında Android inc adlı şirket tarafından geliştirilmeye başlanmıştır. 2005 yılında Android şirketi Google tarafından satın alınmış ve işletim sisteminin geliştirilmesi, bu tarihten sonra hız kazanmıştır. 2008 yılında Google kaynak kodları piyasaya sürmüş ve Android 1.0 işletim sistemini kullanan ilk telefon HTC Dream üretilmiştir. Android 1.0'dan günümüze pek çok sürüm piyasaya sürülmüş, 2018 itibariyle son olarak Android P kullanıcılara sunulmuştur. Linux tabanlı ve açık kaynak kodlu ücretsiz bir işletim sistemidir.

Android işletim sistemi 5 ana katmandan oluşmaktadır. En alt katmandan başlayarak bu katmanları açıklayalım:



Tablo 2.2: Android işletim sistemi katmanları

Linux Kernel (Linux Çekirdeği): Çekirdek, Android yazılım yığınının altında yer almaktadır. Bu katman, donanım ile yazılım arasında haberleşmeyi sağlamakla görevlidir. Örneğin mikrofon, kamera gibi donanımlar için bir ağ yığını sağlamaktadır. Linux çekirdeği başlangıçta masaüstü bilgisayarlarda çalışan geleneksel işletim sistemleri için geliştirilmiş olsa da, günümüzde Android işletim sisteminin kalbi olarak nitelendirilmektedir.

Libraries (Kütüphaneler): Kütüphane, genel olarak yazılım geliştirirken kullanılan bir takım kodlar şeklinde tanımlanabilir. Android kütüphaneleri ücretsiz ve açık kaynak kodlu olduğu için, yazılım geliştiriciler tarafından sıklıkla tercih edilmektedir.

Android Runtime: Java dili ile kütüphanelerin birleştiği yerdir. Çekirdek kütüphaneleri ve Dalvik sanal makinesi bu katmanda yer alır. Dalvik sanal makinesi, Java sanal makinesinin Android işletim sistemine optimize edilmiş halidir. Android'in ilk sürümlerinde oldukça sık kullanılan Dalvik sanal makinesi, son sürümlerde Android Run Time (ART) olarak kullanılmaktadır.

Application Framework (Uygulama Çerçevesi): Android uygulamaların geliştirildiği ortamları toplu olarak oluşturan çerçevelerdir. İçerik sağlayıcıları, kaynak ve etkinlik yöneticisi, paket yöneticisi gibi bir takım temel hizmetlerin bulunduğu yer, uygulama çerçevesidir.

Application (Uygulama): İşletim sisteminde bulunan temel uygulamalar ve sonradan yüklenen üçüncü parti yazılımlardır.

2.7.3. Windows Phone

Windows Phone, 2010 yılında Microsoft ve Nokia ortaklığı ile piyasaya sürülmüş mobil işletim sistemidir. C ve C++ dillerini temel alan işletim sisteminin son sürümü Windows Phone 8.1, Windows 8'e uyumlu olarak 2014'te piyasaya sürülmüştür. Her ne kadar 2015 yılında Windows 10 ile birlikte Windows Phone 10'a güncellenme imkanı sunsa da, uygulama marketinde az sayıda uygulama barındırması, az sayıda cihazda çalışması ve Microsoft'un mobil işletim sistemi pazarına Apple ve Google'dan yaklaşık on yıl sonra girmiş olması nedenleriyle, mobil pazarda Android ve Ios'un gerisinde kalmıştır. Microsoft, 2017 yazında Windows Phone'u artık geliştirmeye devam etmeyeceğini açıklamıştır.

2.8. Uygulama Geliştirme Araçları ve Kullanım Alanları

Mobil uygulamalar kendi içinde Web, Hybrid ve Native olarak çeşitlere ayrılmaktadır ve uygulama türlerine bağlı olarak geliştirme araçları da farklılık göstermektedir. Örneğin SmartFace App Studio, Xamarin gibi bazı araçlar çoklu platformda çalışma imkanı sunan uygulamalar geliştirmenizi sağlayabilmektedir. Bu kitapta Native uygulamalar geliştirmek üzere Android ve IOS programlama anlatılacağı için bu işletim sistemlerinin desteklediği uygulama programlama araçları ve dilleri incelenecektir.

2.8.1. IOS

IOS tabanlı uygulamalar geliştirmek için çeşitli araçlar (tool) vardır. Bu araçlardan en çok kullanılanları Xcode, Objective C ve Swift'tir.

Xcode

IOS platformunda çalışan mobil bir uygulama yazmak için Xcode uygulaması kullanılmaktadır. Xcode yalnızca Mac işletim sistemine sahip bilgisayarlarda çalışabilmektedir. Windows kurulu bir bilgisayarda virtual machine (sanal makina) yaratarak Mac işletim sistemi kurulabilse de Xcode kullanımı açısından fazla verim alınamamaktadır. Xcode uygulaması AppStore'dan veya Developer Apple web sitesinden ücretsiz olarak indirilebilmektedir. Son sürümünü yükleyebilmek için uygulama hard diskte yeterli boş alan ve güncel işletim sisteminin yüklü olmasını istemektedir.

Ekran Görüntüsü 2.1

Xcode yalnızca iPhone, iPad, Apple Watch gibi mobil cihazlar için değil aynı zamanda Mac ve Apple TV için de uygulama yazmaya olanak tanıyan bir platformdur. Xcode'da yaygın olarak Objective C programlama dili kullanıldığı gibi yeni geliştirilmekte olan Swift programlama dili de son yıllarda kullanılmaya başlanmıştır. Objective C dilinin yanı sıra C ve C++ da yardımcı olarak kullanılabilmektedir. Mobil uygulamalarını farklı işletim sistemlerine de uyarlamak isteyen geliştiriciler IOS için yazılan uygulamayı sil baştan tekrar yazmamak için Xcode ortamında C++ diline de yönelebilmektedirler.

Objective C

Objective C, OS X ve IOS için yazılım geliştirmede öncelikli olarak kullanılan programlama dilidir. C programlama dilinin bir süpersetidir ve nesne tabanlı, dinamik çalışma zamanlı bir programlama dilidir. Objective C, C programlama dilinden sözdizimi kuralları (syntax), ilkel tipleri ve akış kontrol ifadelerini miras almıştır ve üzerine sınıfları (class), yöntemleri tanımlayan syntax'ları da ekleyerek günümüz kullanımına sunulmuştur. Ayrıca güçlü grafik yönetim (method) desteği sağlayarak dinamik çalışma imkanı da yaratmaktadır.

```
58          imgView_storyPages = [[UIImageView alloc] initWithFrame:CGRectMake(i*1024, 0, 1024,
                768)];
59          UIImage *img_storyPages = (UIImage *)[self.presentedStoryPagesMutArray objectAtIndex:i];
60          [imgView_storyPages setImage:img_storyPages];
```

Ekran Görüntüsü 2.2

Swift

Apple şirketinde çalışan bir yazılımcı tarafından ilk adımları atılan Swift programlama dili Objective C'nin zor sözdizim kurallarına bir alternatif olması için ortaya çıkmıştır. Günümüzde ise şirket tarafından "daha yenilikçi, üstün performanslı, hızlı ve gelecek vaadeden" bir programlama dili olarak nitelenmektedir. Kullandığı paralel programlama teknolojisi ile bir işi küçük parçalara bölerek farklı işlemcilerde aynı anda çözümleyerek uygulamanın çalışma hızı ve performansını arttırmaktadır. Böylece geleceğin teknolojileri bulut bilişimde, çoklu işlemcili makinalarda çalışabilecek yeteneğe sahip olduğu düşülmektedir. Fakat bunun yanında yazılımcılar için yeni bir dil olduğundan dolayı öğrenme aşamasında, kullanım deneyimlerinin paylaşımı açısından kitap ve döküman gibi yeterli kaynaklara sahip değildir.

```
extension String {
    var banana : String {
        let shortName = self.dropFirst()
        return "\(self) \(self) Bo B\(shortName) Banana Fana Fo F\(shortName)"
    }
}

let bananaName = "Jimmy".banana      // "Jimmy Jimmy Bo Bimmy Banana Fana Fo Fimmy"
```

Ekran Görüntüsü 2.3

2.8.2. Android

Android tabanlı uygulamalar geliştirmek için çeşitli araçlar (tool) vardır. Bu araçlardan en çok kullanılanları App Inventor ve Android Studio'dur.

App Inventor

Sürükle bırak yöntemiyle kolayca Android tabanlı uygulamalar geliştirilebilen bir platformdur. http://appinventor.mit.edu adresi üzerinden Google hesabı ile giriş yapılarak online çalışan App Inventor, ilk olarak Google tarafından ortaya konulmuş, 2011 yılında açık kaynak kodlu hale getirildikten sonra MIT tarafından geliştirilmiştir. Profesyonel olmayan ve kod yazmaya yeni başlayan kullanıcılar için ideal bir uygulama geliştirme aracı olan App Inventor, mobil cihazlarda sensör kullanımına da olanak tanımaktadır.

Android Studio

Google tarafından desteklenen ve üst seviye özellikler içeren temel ve ücretsiz bir android uygulama geliştirme aracıdır. C++ programlama dili desteği sayesinde profesyonel uygulamalar geliştirilmesine imkan tanır. Android Studio, kendi içinde hızlı bir emülatör barındırır ve tablet, telefon, saat gibi farklı mobil cihazlara uygun arayüzleri vardır.

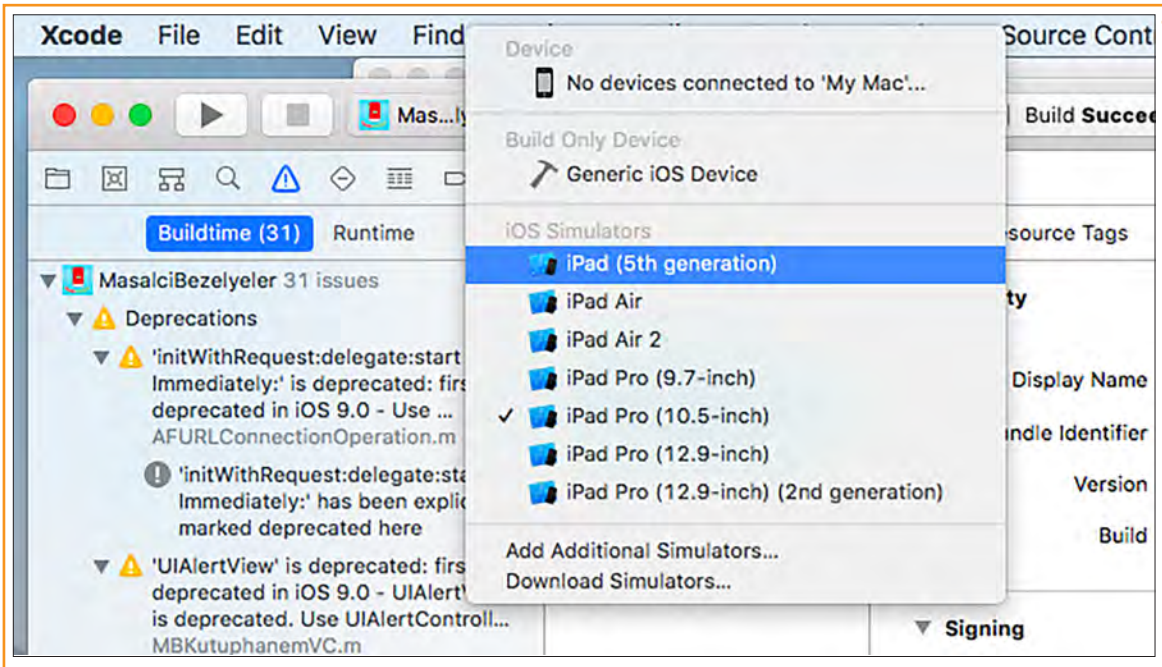
2.9. Sanal Cihaz Kullanımı

Mobil bir uygulama geliştirilirken programın test edilmesi sürecinde mobil cihaz simülasyonu yapan yazılımlara sanal cihaz denmektedir. Bir markaya –örneğin Apple'a- ait tüm modellerin yazılımcı tarafından temin edilip kullanılması hem ekonomik açıdan hem de zaman açısından güçtür. Her cihazın ekran çözünürlük ve boyutu, işlemci ve bellek özellikleri birbirinden farklı olmaktadır. Geliştirilen uygulamanın, farklı kullanıcılar tarafından farklı model cihazlarda da kullanımı düşünüldüğünde tüm cihazlarda sorunsuz çalışabiliyor olması gerekmektedir. Bu sebeple geliştirilen "sanal cihaz yazılımları" aracılığı ile yazılımcı gerçek bir cihaza yakın bir platformda yazılımını test edebilmektedir. Sanal cihazların tek dezavantajı ise eğer geliştirilen uygulama mikrofon, kamera, sensörler gibi dahili donanımları kullanıyor ise bu özelliklerin test edilmesine olanak vermemesidir. Bu durumda gerçek cihazın fiziksel olarak, programlama yapılan bilgisayara bağlanarak uygulamanın bu şekilde test edilmesi gerekmektedir.

Farklı işletim sistemleri mobil uygulama yazmak için farklı geliştirme araçlarını kullanır. Benzer şekilde de farklı geliştirme araçları farklı sanal cihaz yazılımlarını kullanır. Bir uygulama geliştirme aracı bilgisayara indirilip kurulduğunda içine entegre edilmiş sanal cihaz yazılımı da beraberinde iner. Örneğin Android tabanlı uygulama geliştirme araçları Android Emülatör ve Sanal Cihazı (AVD) ismi verilen ortamı kullanırken IOS, Simülatör isimli ortamı kullanmaktadır.

2.9.1. IOS Simulator

IOS tabanlı uygulamalar programlamak için kullanılan Xcode yazılımı makinaya indirildiğinde ve kurulduğunda beraberinde IOS Simulator da yüklenmektedir. Bu program aracılığı ile Xcode yazılım arayüzünden istenilen cihaz türü seçilerek uygulama çalıştırılır (Run).



Ekran Görüntüsü 2.4

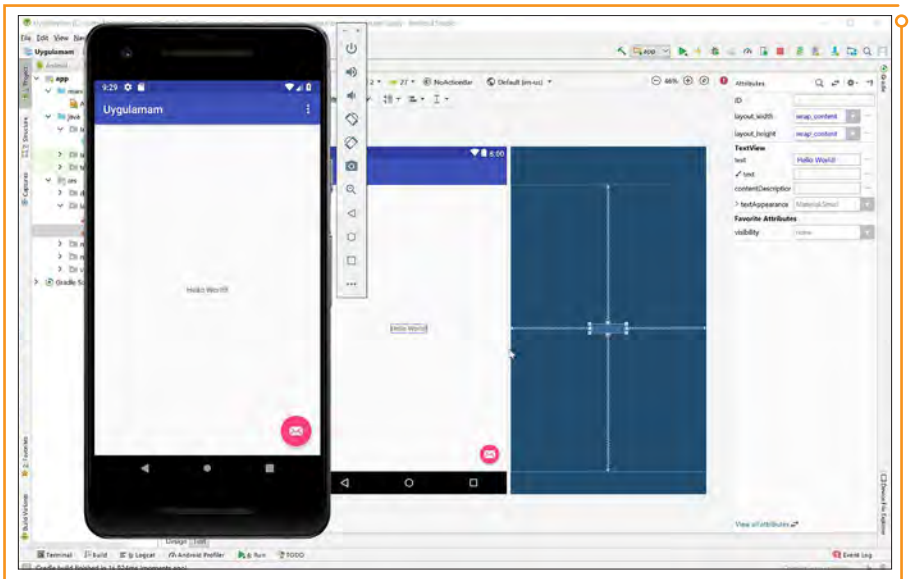
IOS Simulator üzerinde çalışan uygulama fare/touchpad ile kullanılarak test edilebilmekte ve hata ayıklama (debug) işlemi yapılabilmektedir.

Ekran Görüntüsü 2.5

2.9.2. Android Emülatörler

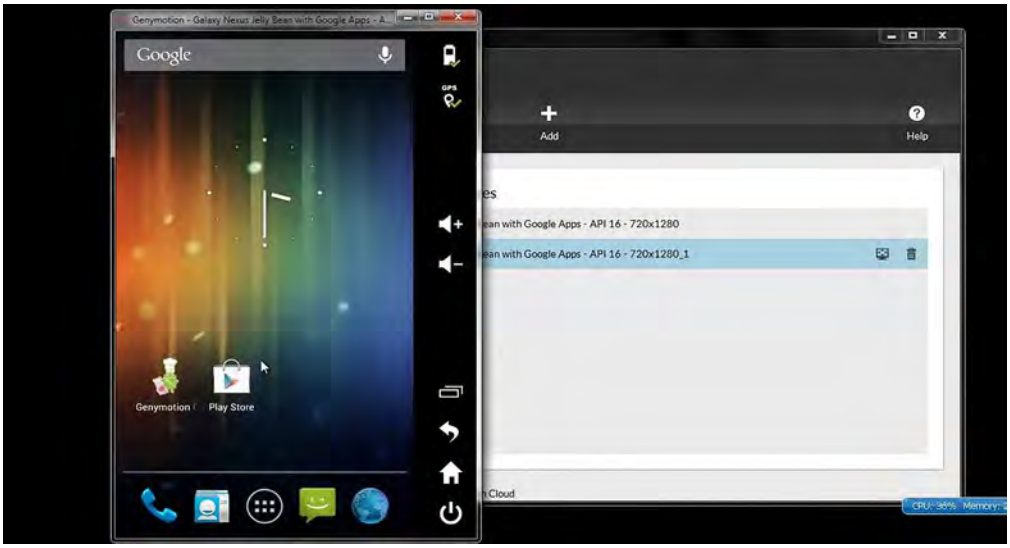
Android tabanlı uygulama geliştirme araçları, 'emülatör' adı verilen sanal cihaz yazılımlarını kullanırlar. En sık kullanılan ve tercih edilen emülatörler şunlardır:

Resmi android emülatörü, Google tarafından desteklenen uygulama geliştirme aracının (Android Studio) içinde ücretsiz olarak sunulmaktadır ve uygulama geliştiricilere yönelik olarak tasarlanmıştır.



Ekran Görüntüsü 2.6

Popüler bir emülatör olan Genmotion, birçok cihazın ortamına uyum sağlayabilmekte ve kısıtlı da olsa ücretsiz kullanıma imkan vermektedir. Görsellik bakımından son derece iyi olsa da, ek olarak virtualbox kurulumu gerektirmektedir.



Ekran Görüntüsü 2.7

Bir diğer emülatör BlueStacks, Windows ortamında çalışmakta ve virtualbox kurulumu gerektirmemektedir. Ayrıca APK kurulumları da kolay olan BlueStack, Windows kullanıcıları tarafından sıkça tercih edilmektedir.

Ekran Görüntüsü 2.8

NoxxApp Player, yine BlueStacks ile benzer özelliklerdedir. Yerleşik GPS kontrol cihazı seçeneği vardır.

AndyRoid, Gooogle tabanlı uygulamaları iyi çalıştırmakta, teknik olarak ve görsel bakımdan yeterli bulunmakta ve sık kullanılmaktadır.

3. UYGULAMA GELİŞTİRME

3.1. Uygulama Geliştirme

Uygulama geliştirme, belirli kuramsal temeller üzerine oturtulmuş yol haritaları ile gerçekleştirilebileceği gibi yazılım dünyasında yer almak isteyen her geliştiricinin kendine göre oluşturduğu farklı metodolojilerden de oluşabilir. Birçok yazılımcının birlikte çalıştığı yazılım projeleri ile tek başınıza geliştirdiğiniz küçük uygulamaların geliştirilme süreçleri arasında önemli farklılıklar da bulunur. Yukarıda belirtilenlere ek olarak aynı hedef platform için geliştirilen bir uygulamanın mimarisi de çok farklı şekillerde kurgulanabilir. Günümüzde bir mobil uygulama geliştirirken kullanabileceğiniz farklı uygulama geliştirme mimarileri bulunmaktadır. Bunlar içerisinde en çok kabul görenler aşağıda sıralanmıştır:

- Tamamen Web tarayıcıda çalışanlar
- Yerel (native) uygulama olarak geliştirilenler
- Web ve yerel bileşenleri içinde barındıran karma (hibrid) uygulamalar

3.2. Uygulamaların Yaşam Döngüleri

Uygulama geliştirme yaşam döngüleri, uygulama geliştirme mimarisinde tercih edilen yönteme göre şekillenir ve bu tercih mobil uygulama geliştirme araçlarının seçimini de etkiler.

3.2.1. Tamamen Web Tarayıcıda Çalışan Uygulamalar İçin Yaşam Döngüsü

Bu uygulamaların yaşam döngüsü, standart bir Web sitesi geliştirme yaşam döngüsü ile paralellik gösterir ve buna ek olarak mobil ortamlar için sunulmuş ek yazılım kütüphanelerinin kullanımlarını da içerir. HTML5 (HTML, CSS3, JavaScript) kullanılarak geliştirilen bu mobil uygulamalar mobil cihazlar için geliştirilen tarayıcılar tarafından yorumlanır ve kullanıcılara sunulur. Geliştirme ortamı olarak herhangi bir metin editörü kullanılabilir. Geliştirilen uygulamanın test edilmesi için ise güncel bir tarayıcı yeterli olacaktır.

3.2.2. Yerel (Native) Uygulamalar İçin Yaşam Döngüsü

Bu uygulamaların yaşam döngüsü, tamamen hedef platforma yönelik olarak uygulamanın geliştirilmesi süreçlerini içerir. Seçilen hedef platforma (ör.:Androd, IOS, Windows Phone) has geliştirme araçları, yazılım dili ve test ortamları kullanılır. Hedef platformun tüm özelliklerini kullanabilmesi açısından önemli avantajlar sağlamasına rağmen her bir platforma özel olarak uygulamanın geliştirilmesi (kodlanması) gerekmektedir. Geliştirme ortamları çoğunlukla mobil cihazın işletim sistemine özel olarak geliştirilmiş ortamlardır ve test ortamları da yine işletim sistemine özel olarak geliştirilmiş emülatörlerden oluşur. Bu ders kapsamında kullanılacak yerel geliştirme platformu Android olarak seçilmiştir.

3.2.3. Web ve Yerel Bileşenleri İçinde Barındıran Karma (Hibrit) Uygulamalar İçin Yaşam Döngüsü

Bu uygulamaların yaşam döngüsü içerisinde hem web tarayıcılar için geliştirilen uygulamaların yaşam döngüsü, hem de yerel uygulama geliştirme yaşam döngüsü yer alır. Bu yapının en büyük avantajı, geliştirme sürecinin temel web teknikleri ile gerçekleştirilebilmesi ve aynı zamanda yerel özelliklerden de faydalanılabilmesidir. Bu aşamada web geliştirme sürecine hakim olan bir kullanıcının ek bir programlama dili öğrenmesine gerek olmamaktadır. Bununla birlikte sağlanan ek bir avantaj da hedef platform için (ör.: Android ya da IOS) yapılan bir kodlamanın bir diğer platform için de yeniden kodlama yapılmaksızın kullanılabilmesidir.

4. MOBİL UYGULAMA GELİŞTİRME

4.1 Mobil Uygulama Geliştirme

Bu ders kapsamında uygulama geliştirme bölümünde listelenen farklı mimariler ayrı ayrı kullanılarak mobil uygulama geliştirme süreci tüm adımları içerecek şekilde anlatılmıştır. Her bir uygulama geliştirme mimarisinde, kullanılan geliştirme araçlarının kurulumundan itibaren uygulamanın mobil cihaz üzerinde çalıştırılma sürecine kadar uygulama yaşam döngüsü detaylandırılmıştır.

Uygulama geliştirme aracı olarak Android işletim sistemi için Android Studio, IOS işletim sistemi için Xcode kullanılmıştır. Bütün örnek uygulamalar her iki programla da yapılmıştır. Kitap içerisinde önce Xcode kurulum ve uygulamalar daha sonra Android Studio kurulum ve uygulamalar yer almaktadır.

Resim 4.1

4.2. Xcode

Xcode masaüstü uygulamasını bilgisayarınıza indirebilmeniz için öncelikle Mac OS X kullanıyor olmanız gerekmektedir. İndireceğiniz Xcode versiyonunun/sürümünün de işletim sisteminizin güncel sürümünü destekliyor olması gerekmektedir. Xcode'un Linux ve Windows işletim sistemleri için ayrı bir sürümü bulunmamaktadır. Bazı kullanıcılar Windows üzerinde sanal makina kurarak Mac OS X işletim sistemini kullanabilmektedir. Fakat donanım uyum sorunlarından ve sanal makine üzerinde MAC OS X'in ve Xcode'un yavaş çalışmasından ötürü sanal makina üzerinde Xcode kurulumu önerilmemektedir.

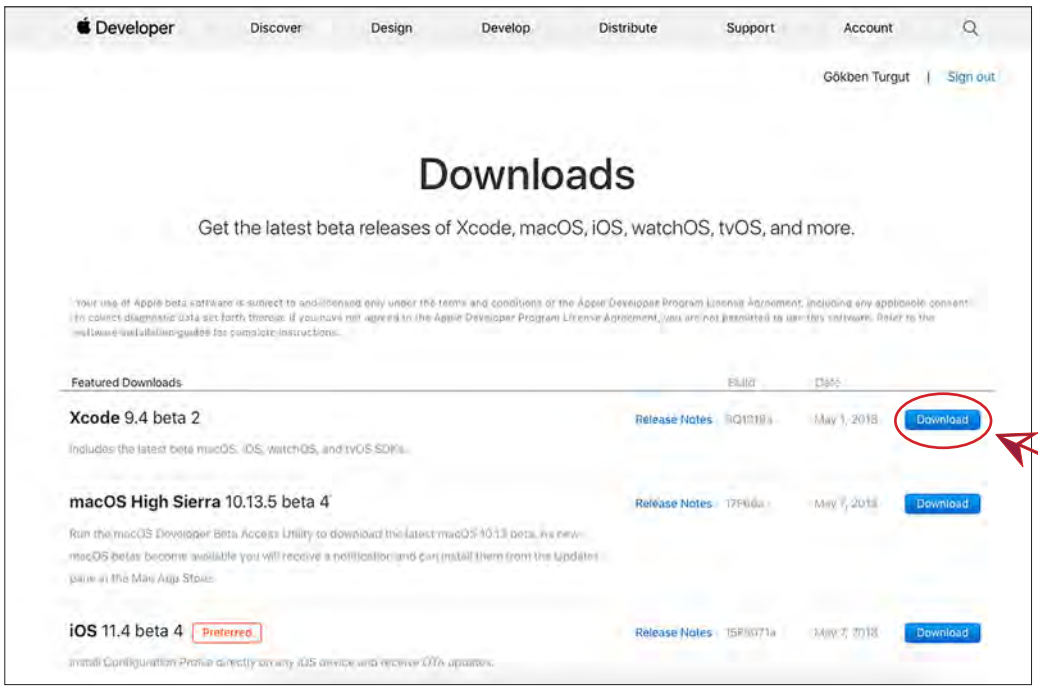
4.2.1. Xcode Programının Kurulumu

Xcode Apple'ın kendi resmi web sayfasından indirildiği gibi App Store uygulama dükkanından da indirilebilmektedir. Eski sürümlerini de indirip kullanabilmek isteyen geliştiriciler http://developer.apple.com adresinden Xcode'u indirebilirler. App Store uygulama dükkanından sadece son sürümüne erişilebilmektedir. Her iki platformda da uygulama ücretsiz olarak indirilebilir. Xcode uygulamasının bilgisayarınıza yüklenme adımları aşağıdaki gibidir:

(1) Öncelikle internet tarayıcısının adres kısmına http://developer.apple.com adresi yazılarak Apple'ın geliştiriciler için çeşitli hizmet ve bilgiler verdiği resmi sitesine bağlanılır.

Ekran Görüntüsü 4.1

(2) Sol üst köşede bulunan download düğmesine basılarak açılan sayfada istenilen Xcode sürümü listeden seçilir. Daha eski sürümleri için sayfanın altına doğru ilerlenir.

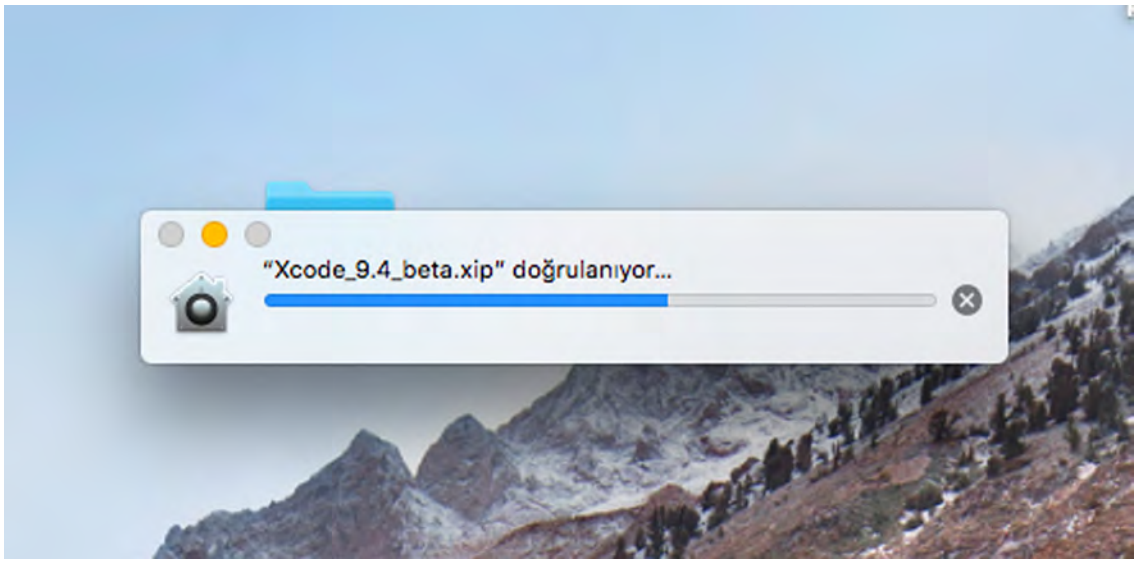


Ekran Görüntüsü 4.2

(3) Seçilen Xcode sürümü download düğmesine tıklanarak bilgisayara indirilir. Aynı zamanda kurulum dosyasının bilgisayara indirilebilmesi için disk üzerinde 5-6 GB kadar yeterli alana ihtiyaç duyulmaktadır.

Ekran Görüntüsü 4.3

(4) İndirme işlemi tamamlandıktan sonra kurulum dosyası indirilenler klasöründen çift tıklanarak, kurulum başlatılır.

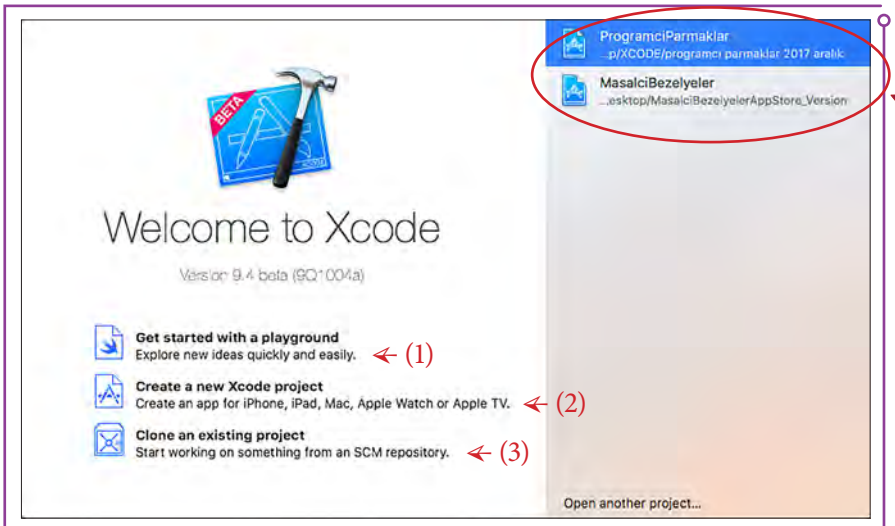


Ekran Görüntüsü 4.4

(5) Kurulum kendi kendine yürütülmektedir ve tamamlandıktan sonra masaüstünden Launchpad veya Finder'dan Uygulamalar klasörü tıklanarak Xcode uygulaması ikonu görülebilir.

Ekran Görüntüsü 4.5

(6) Xcode ikonu tıklanarak uygulama başlatılır. Açılan ilk pencerede sol bölümde yeni açılacak proje ile ilgili seçenekler sunulurken, sağ tarafta üzerinde daha önceden çalıştığınız proje dosyaları listelenecektir.



Ekran Görüntüsü 4.6

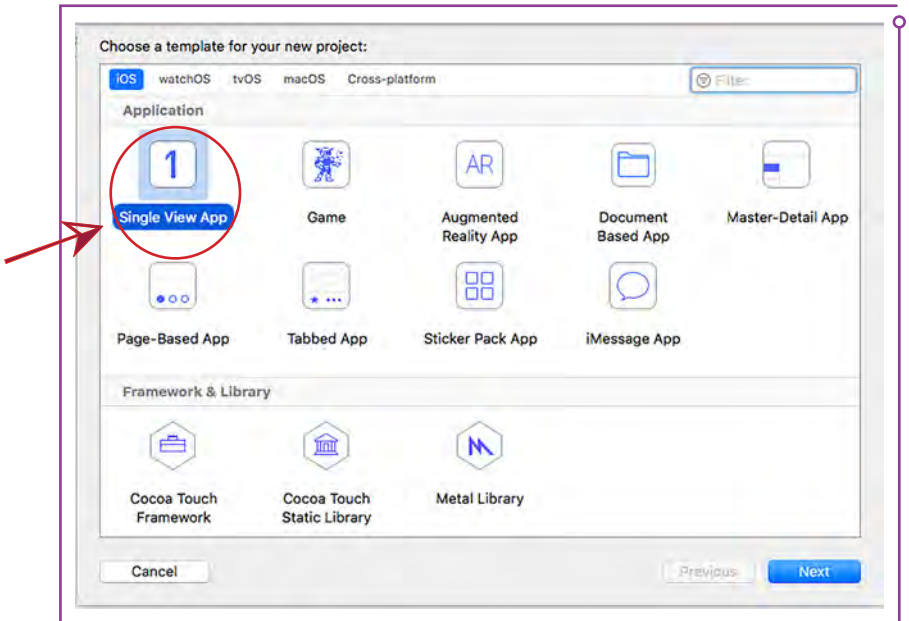
1. Hazır bir şablon ile yeni bir projeye başlamak için,

2. Boş bir proje ile uygulama yazmak için,

3. Var olan bir projenin kopyası üzerinden başlamak için,

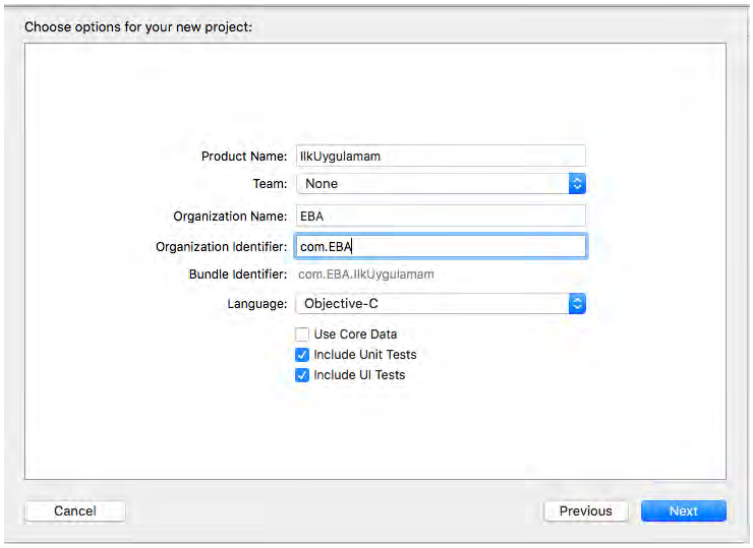
4. Daha önceden çalıştığınız proje dosyalarını açmak için tıklanır.

(7) Yeni bir uygulama geliştirmek için boş ve yeni bir proje açmak üzere (2) Create a new Xcode project seçeneğine tıklanır. Karşımıza gelen pencereden tek view'den oluşacak Single View App seçeneği tıklanır.



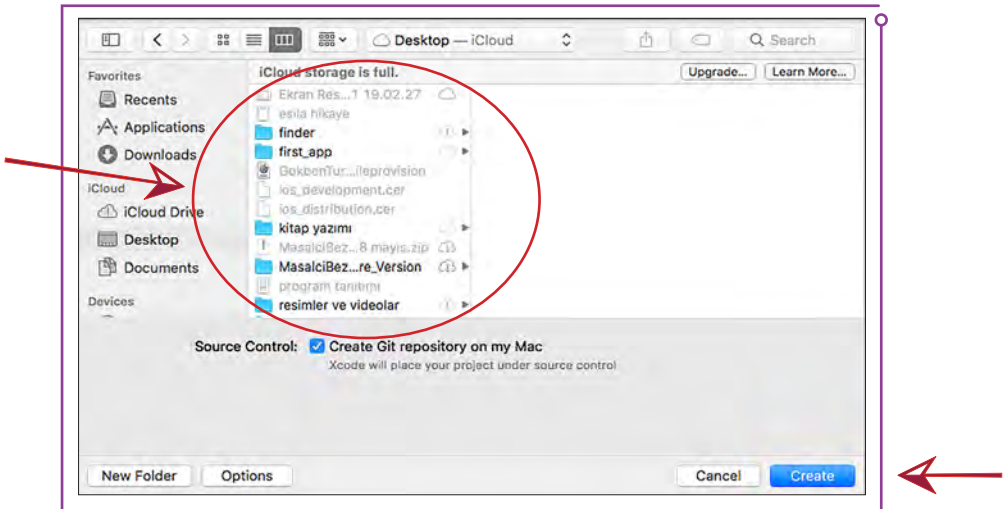
Ekran Görüntüsü 4.7

(8) Açılan pencerede uygulamanızın adı, takım adı, organizasyon ismi, organizasyon kimliği, programlama dili, veri tabanı kullanımı uygulama projenize ait bazı bilgilerini belirlemeniz istenmektedir. Türkçe karakterler, özel karakterler ve boşluk kullanılmaktadır. Product Name alanına projenizin ismini yazınız. Eğer Apple'dan alınan bir developer hesabınız varsa, hesabın ait olduğu şirket veya kişi adını Team alanından seçebilirsiniz. Developer hesabının alınması ve sertifikalarının bilgisayara yüklenerek Xcode projelerine tanımlanması ilerideki bölümlerde anlatılacaktır. Bu bölüm şimdilik None olarak seçilebilir. Organization Name bölümüne uygulamanız için bir ticari kimlik oluşturmak adına isminiz veya organizasyon adı yazılabilir. Organization Identifier alanına ise başına "com." ifadesi koyularak devamında organizasyon ismi veya kendi isminiz yazılabilir. Kullanılacak programlama dili Language alanından Objective-C seçilerek tamamlanır. Eğer uygulamanızda veri tabanı kullanmak istiyorsanız Use Core Data seçeneği işaretlenerek Next düğmesine tıklanır.



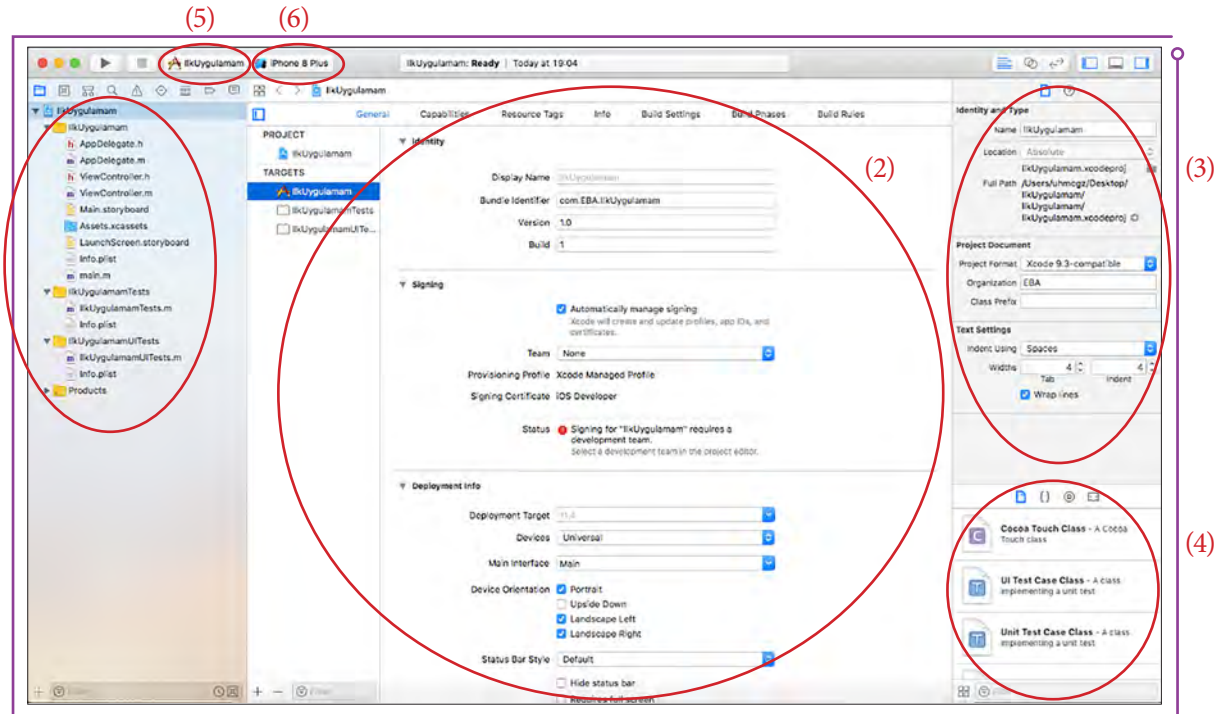
Ekran Görüntüsü 4.8

(9) Son olarak projenin kaydedileceği dizin sol bölümden seçilerek proje klasörü Create düğmesine tıklanarak oluşturulur.



Ekran Görüntüsü 4.9

4.2.2. Xcode Uygulama Ekranı



Ekran Görüntüsü 4.10

(1) Projenize dahil olan tüm uygulama dosyalarının, indirilen kütüphane klasörlerinin, resim/video/ses dosyalarının, plist, storyboard (uygulama arayüz) dosyalarının listelendiği alandır. Bu alandan uygulama içeriğinde bulunan dosyalara ulaşılabilmektedir.

(2) Sol taraftan seçilen dosya içeriğinin görüntülendiği alandır.

(3) Uygulama dosyalarının ve içindeki öğelerin özelliklerinin görüntülendiği ve ayarlandığı alandır.

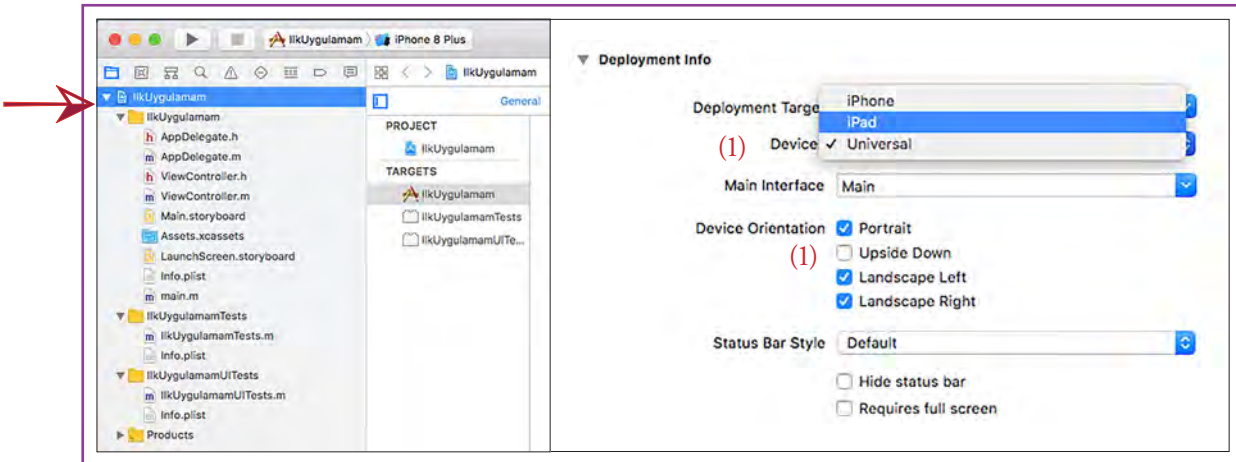
(4) Storyboard'a (uygulama arayüzüne) veya projeye çeşitli sınıflar (class) veya nesneler (object) eklemek için kullanılan alandır.

(5) Uygulamanın simülatör veya cihaz üzerinde çalıştırılması için Çalıştır ve Durdur düğmelerinin bulunduğu alandır.

(6) Uygulama test edilmek üzere çalıştırıldığında kullanılacak simülatör cihaz özelliklerinin (iPadAir, iPadAir 2, iPhone 6, iPhone 6 Plus vb.) veya bilgisayara bağlı gerçek cihazı seçmeye yarayan alandır.

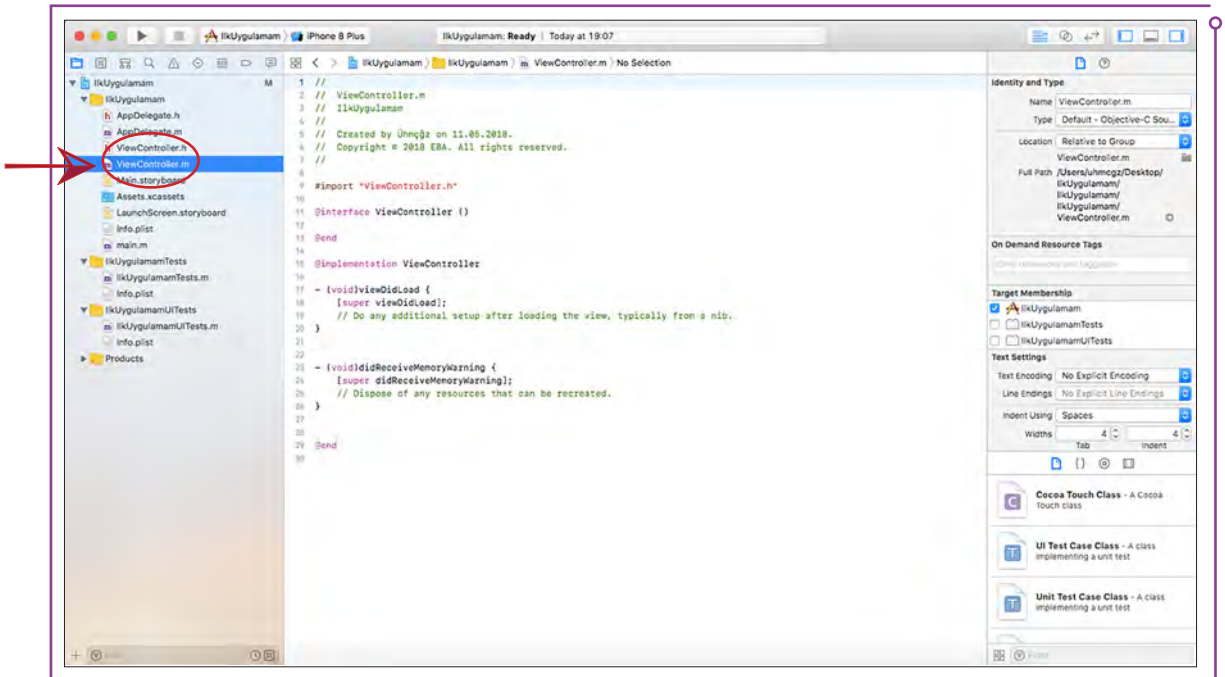
Uygulama Genel Ayarlar Ekranı:

Xcode ekranında sol bölümden uygulama isminizle gösterilen mavi ikon tıklanır. Orta bölümde açılan alanda Identity ve Signing bölümlerinde uygulamanızın versiyon(version), build, bundle identifier, developer hesap (team) bilgilerini belirleyebilirsiniz. Bu ilk iki bölüm özellikle uygulamanızın cihaz üzerinde kullanımı ve App Store'da yayınlanması safhasında önem kazanmaktadır. Deployment Target bölümünden hangi IOS sürümünden itibaren çalışacağı, uygulamanın hangi cihaz için geliştirildiği (iPad, iPhone, iWatch vb.), uygulamanın hangi cihaz yönlerinde (sırasıyla dikey, dikey ters, sol yatay, sağ yatay) çalışabilmesini istediğinizi, cihazın ekran üst kısmında bulunan durum çubuğunun (status bar) stilini ve görünürlüğünü ayarlayabilirsiniz.



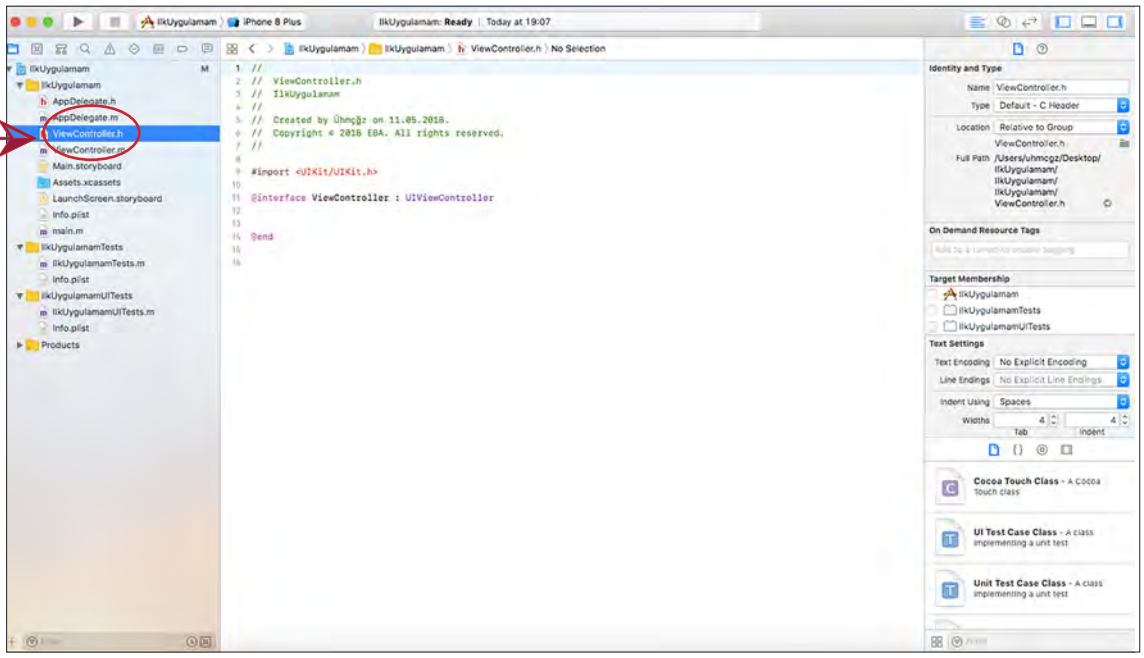
Ekran Görüntüsü 4.11

İlk uygulamamızı iPad için geliştireceğimiz için Device (1) bölümünden iPad seçeneğini tıklıyoruz. Cihaz yönlendirmesi (2) için dikey, sağ yatay, sol yatay seçenekleri işaretli olarak bırakıyoruz.



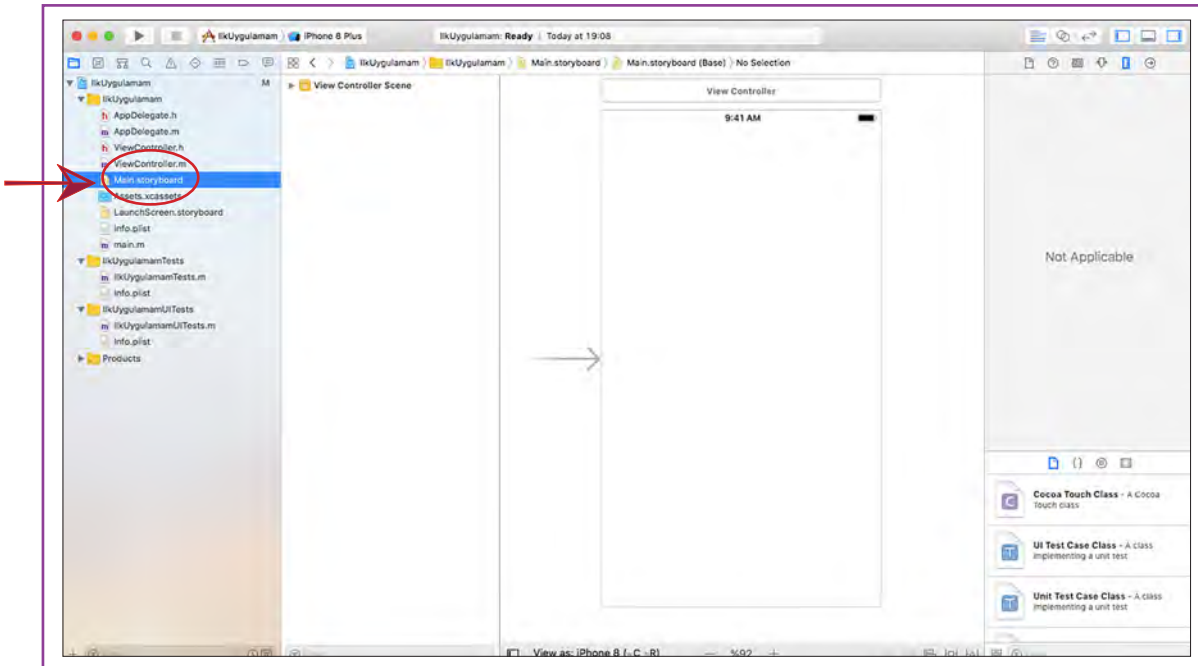
Ekran Görüntüsü 4.12

ViewController.m dosyası uygulamanızda bulunan bir pencerenin (View) programlama olaylarının, fonksiyonlarının yazıldığı ana dosyadır.



Ekran Görüntüsü 4.13

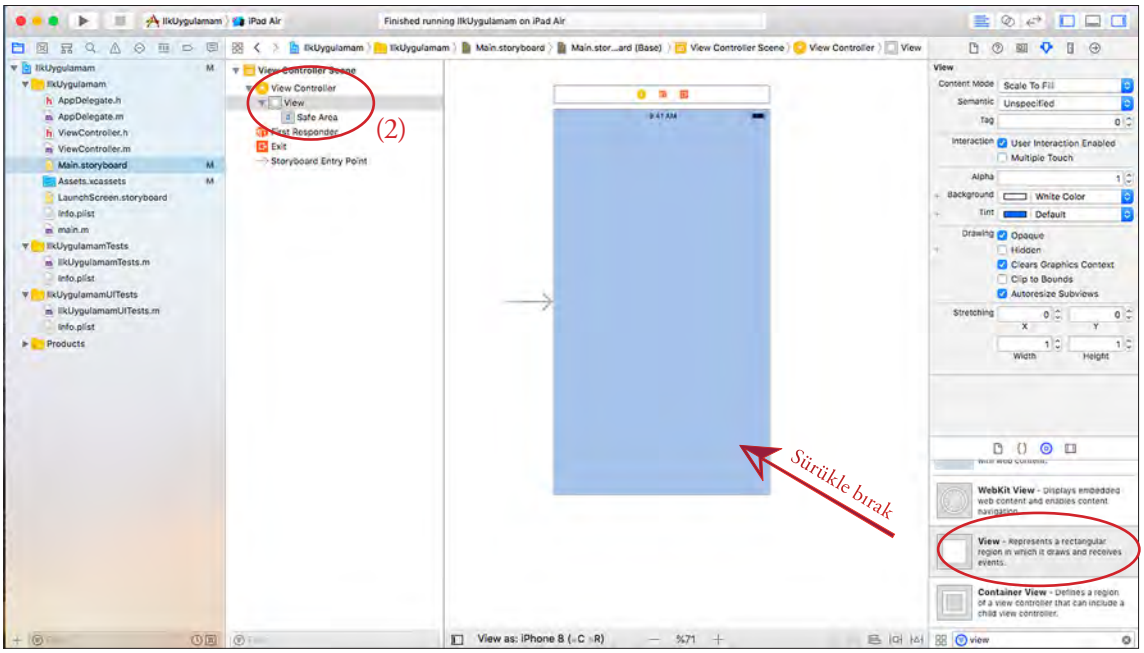
ViewController.h dosyası ise ViewController.m dosyasında kullanacağınız değişkenlerin, nesnelerin (imageview, textfield, label vb.), kütüphanelerin tanımlandığı uygulama dosyasıdır.



Ekran Görüntüsü 4.14

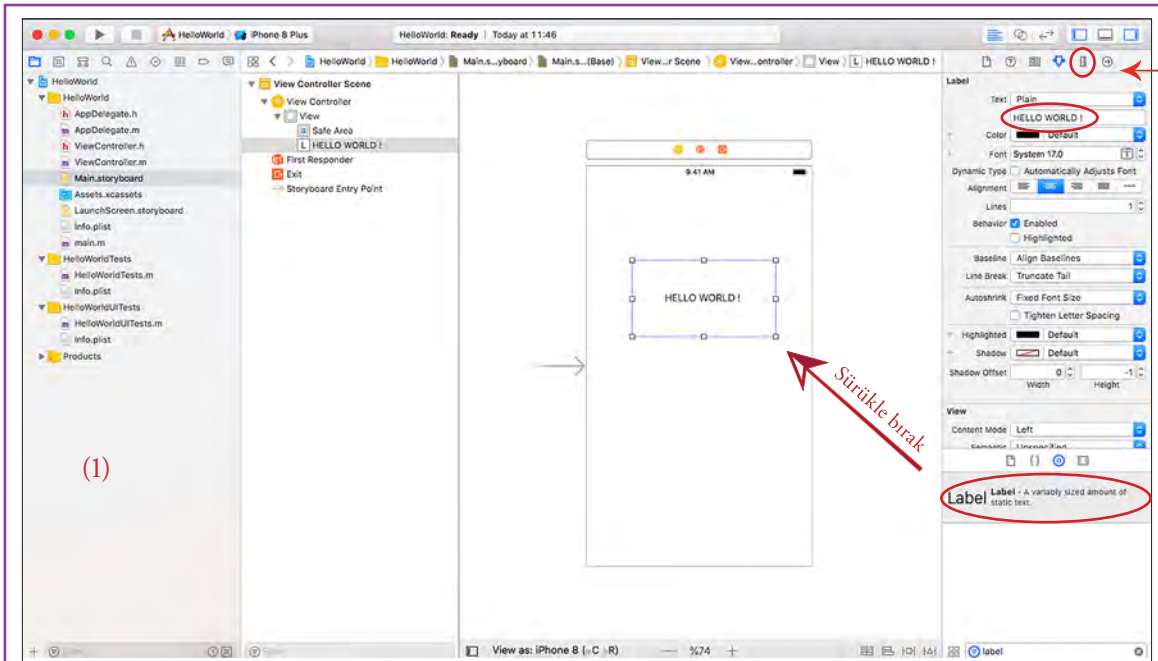
Main.storyboard dosyası uygulama pencerelerinin (View), pencere içerisinde yer alan nesnelerin (imageview, textfield, label vb.) gösterildiği ve düzenlendiği dosyadır.

4.2.3. Hello World Uygulaması



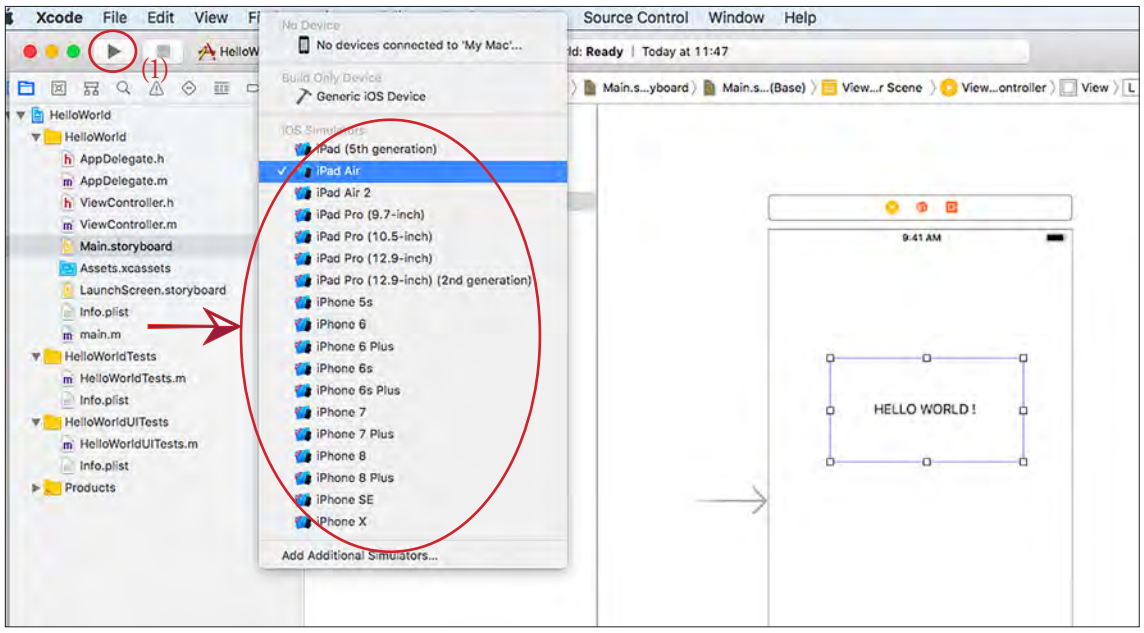
Ekran Görüntüsü 4.15

Storyboard dosyasında, ViewController penceresinin üzerine öncelikle bir View (1) eklenir. Sürükle bırak yöntemi ile ViewController üzerinde bıraktığınızda View otomatik olarak tüm uygulama ekranına genişleyecektir. Storyboard'un sol tarafından (2) sahne nesnelerini hiyerarşik bir sırada ikonik simgelerle görebilirsiniz. Bu bölümden ilgili nesneyi seçme ve isim değişikliği işlemlerini yapabilirsiniz.



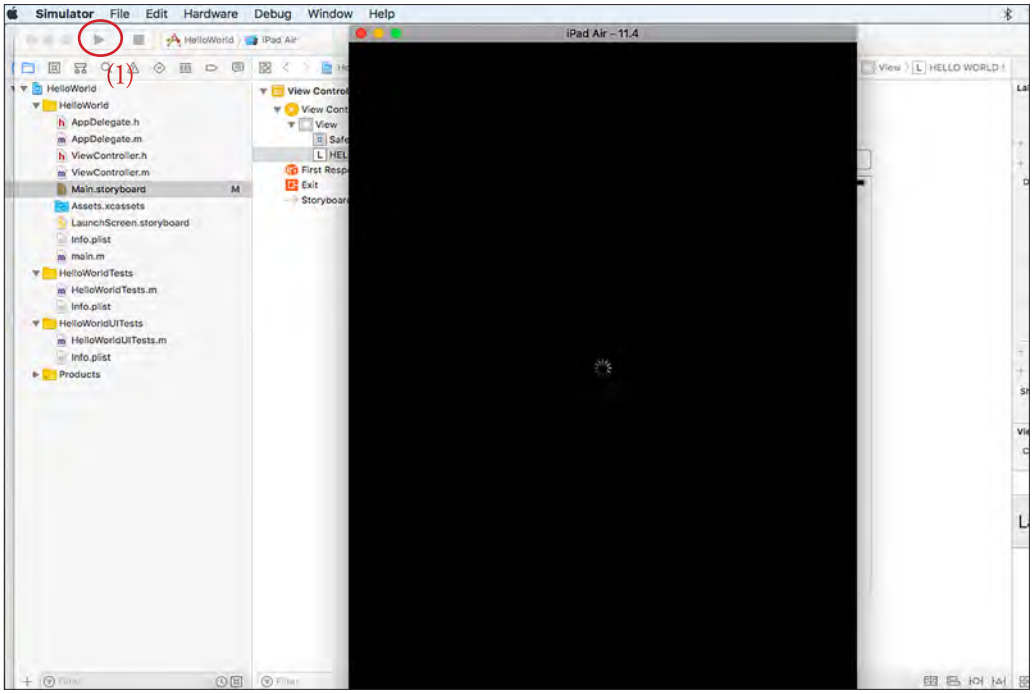
Ekran Görüntüsü 4.16

Sürükle bırak yöntemi ile View'ın üzerine Label nesnesi (1) sürüklenip bırakılır. Label içerisindeki metni ve metnin özelliklerini değiştirmek için özellik düğmesi (2) tıklanır. Label'ın metin içeriği (3) değiştirilir. Alignment bölümünden hizalama, Font bölümünden yazı biçimi, Color bölümünden yazı rengi, Lines bölümünden satır sayısı belirlenebilmektedir.



Ekran Görüntüsü 4.17

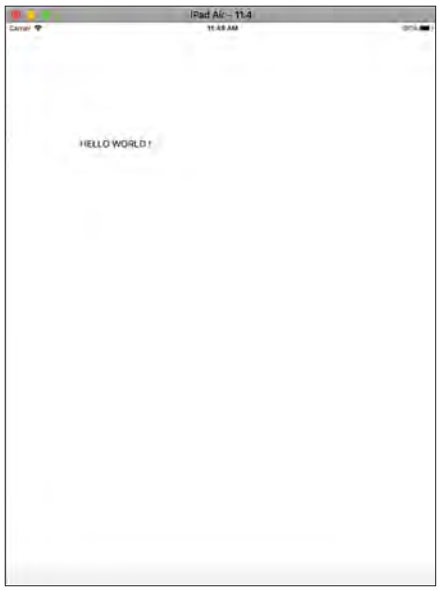
Uygulamanızı simülatörde çalıştırmak için öncelikle simulator cihazınızı seçiniz. Ardından çalıştır (1) düğmesine basarak simülatörün çalışmasını bekleyiniz.



Ekran Görüntüsü 4.18

Simülatör programı ilk defa çalıştırıldığında açılması biraz zaman alabilmektedir.

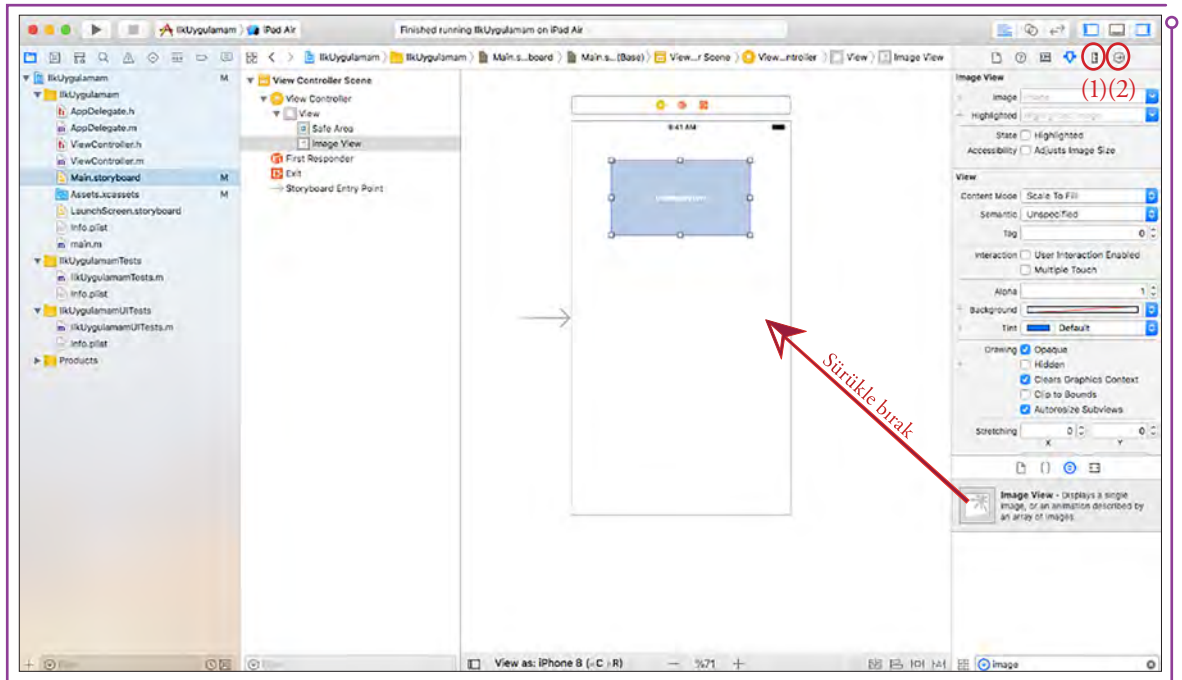
Simülatör açılarak uygulama çalıştığında iPad Air ekranındaki uygulama pencere görüntünüz yandaki gibi olacaktır.



Ekran Görüntüsü 4.19

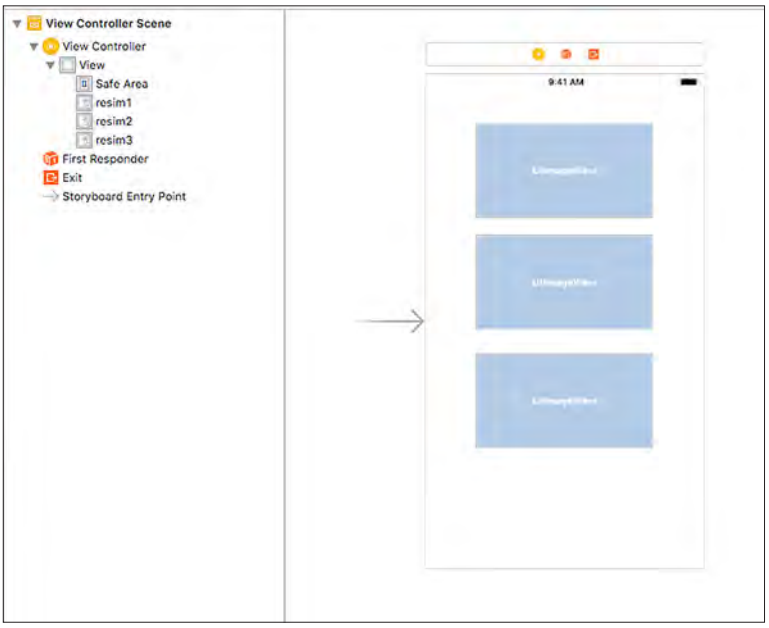
4.2.4. Uygulamaya Resim Ekleme

Sürükle bırak yöntemi ile uygulama penceresine (View'ın üzerine) ImageView eklenir. ImageView nesnesinin boyutlarını değiştirmek için ekran üzerinde fare ile işlem yapılabildiği gibi sağ bölümde cetvel simgesi (1) tıklanarak konum ve boyut bilgileri düzenlenebilir. Resim dosyasının ismi, verilecek numeric tag (sayısal etiket), alfa, arka plan rengi gibi bilgileri düzenlemek için özellikler seçeneğini (2) tıklamanız gerekmektedir.

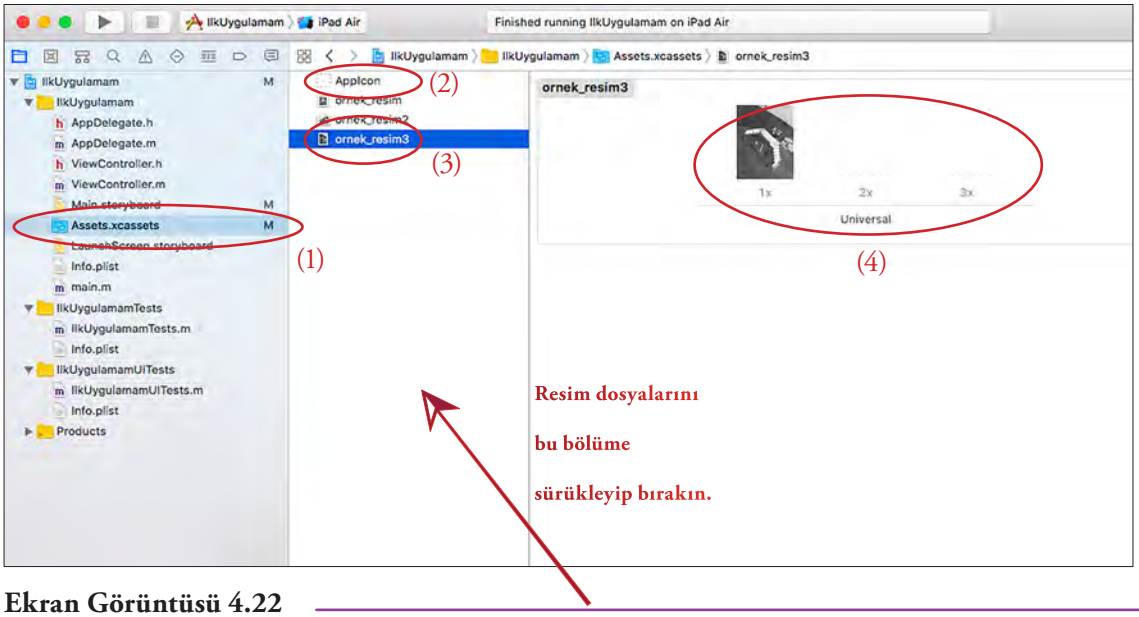


Ekran Görüntüsü 4.20

Sırasıyla ikinci ve üçüncü ImageView'leri ekleyiniz.

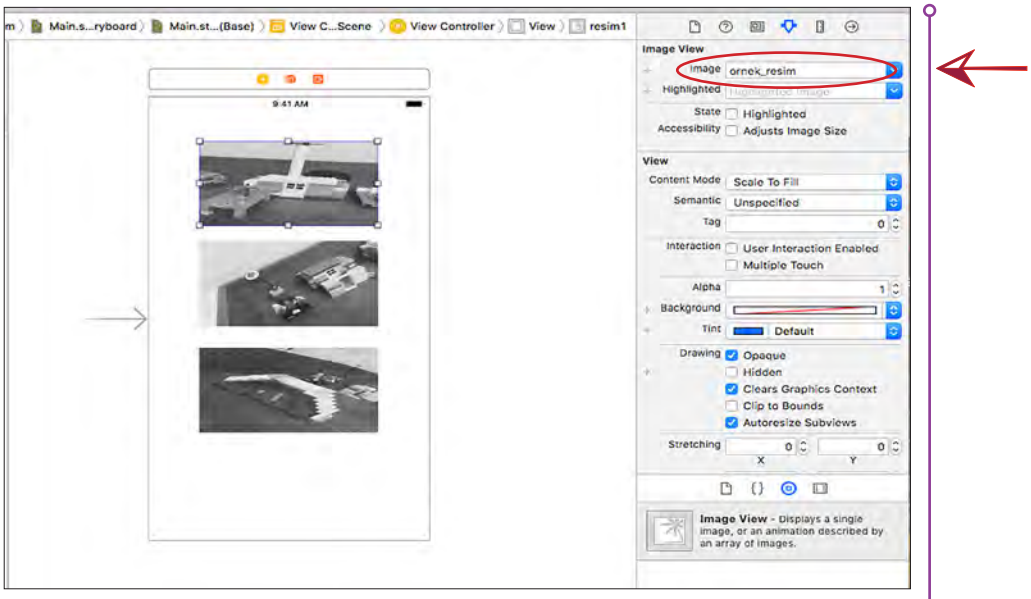


Ekran Görüntüsü 4.21



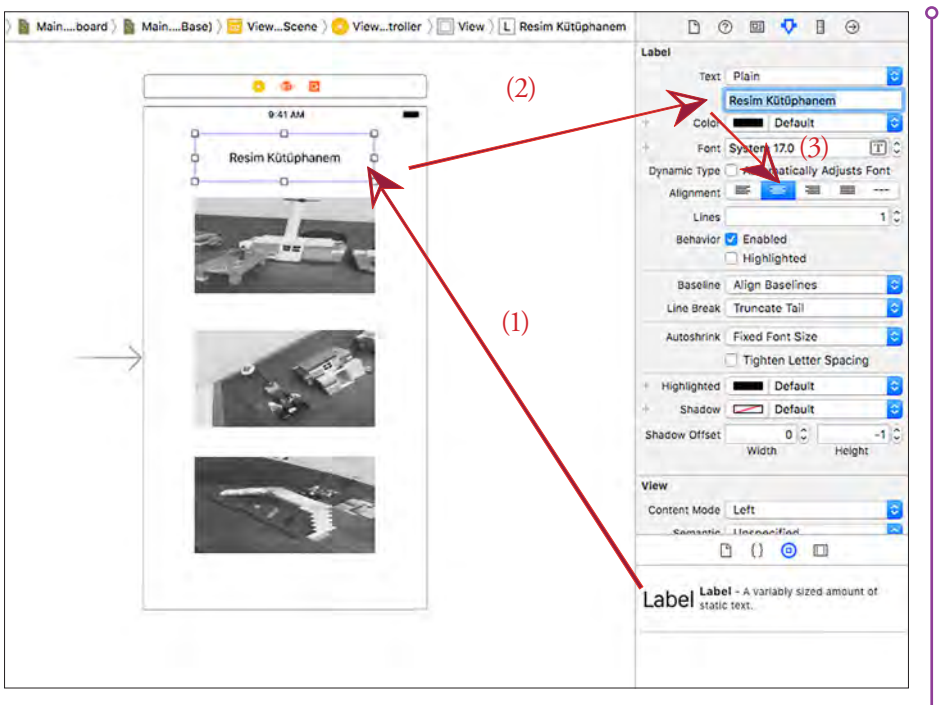
Ekran Görüntüsü 4.22

ImageView içerisine yerleştirilecek resmin dosyasının projeye eklenmesi için sağındaki dosyalar bölmesinden Assets.xcassets (1) tıklanır. AppIcon (2) uygulamanızın cihaz ekranında görünecek başlatma ikonunun görsellerini belirleyeceğiniz bölümdür. Bilgisayarınızda bulunan resim dosyalarını projeye eklemek için orta bölüme sürükleyip bırakınız. İsterseniz resim dosyalarının isimlerini bu bölümden (3) enter tuşuna basarak değiştirebilirsiniz. Proje eklediğiniz resimlerin sağ bölümden görüntülerini izleyebilirsiniz. Kullanılan cihazın ekran çözünürlük özelliklerine göre (Retina gibi) görüntü kalitesinin, uygulamanızın kullanılacağı tüm cihazlarda iyi olmasını sağlamak amacıyla, eklediğiniz resim dosyalarının x2 ve x3 katı boyutlu dosyalarını (4) yükleyebilirsiniz.



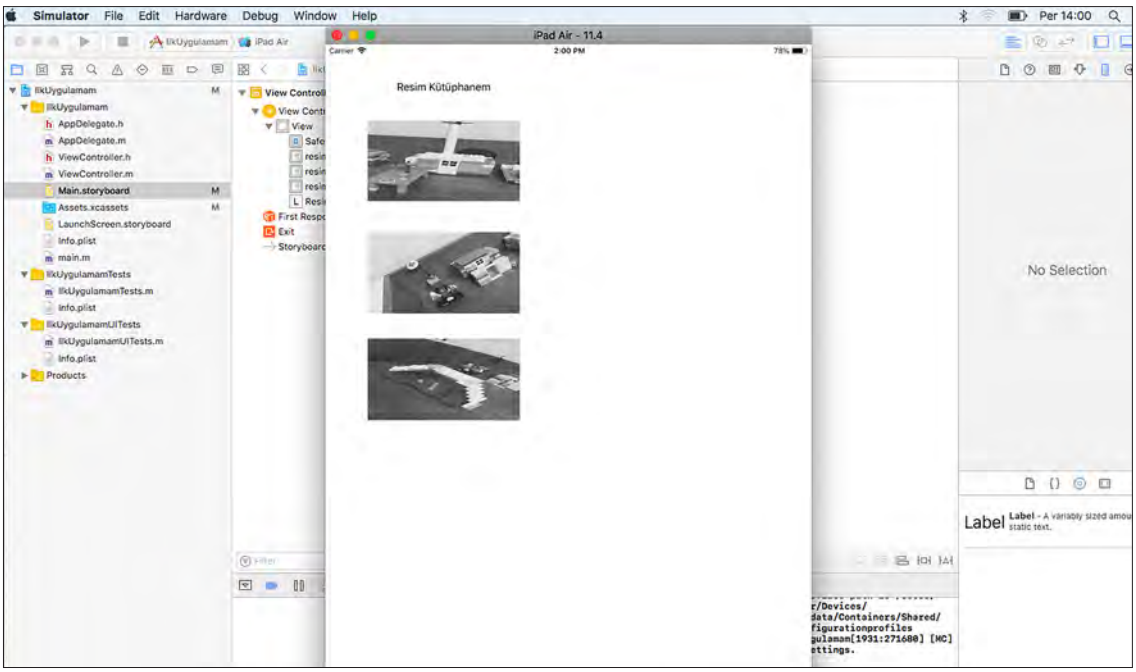
Ekran Görüntüsü 4.23

Storyboard penceresine geri döndüğünüzde içine resim dosyasını eklemek istediğiniz Imageiew'ı seçip sağ taraftaki Image alanına ilgili resim dosyasının adını yazınız veya listeden seçiniz. Sırasıyla diğer ImageView'ler için de aynı işlemi tekrar ediniz.



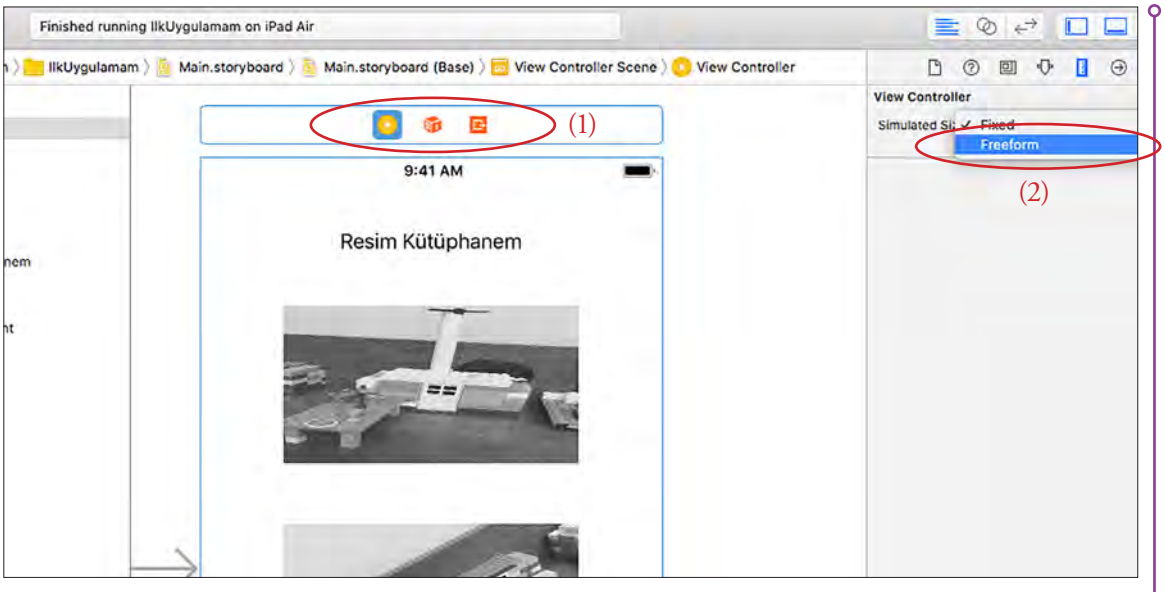
Ekran Görüntüsü 4.24

ImageView'leri fare ile aşağıya doğru sürükleyerek üst kısmına bir Label ekleyiniz. Ardından Label metnini değiştirerek metni ortalayınız.



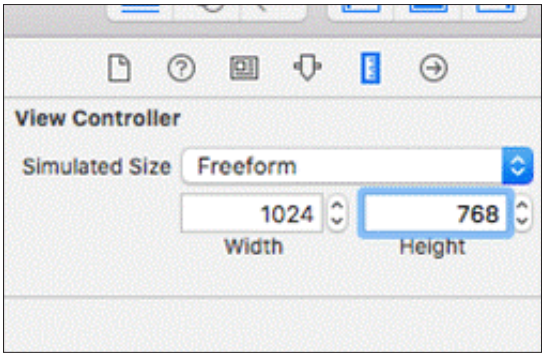
Ekran Görüntüsü 4.25

Simülatördeki ekran görüntüsü ise (Ekran Görüntüsü 4.25) gibi olmaktadır.



Ekran Görüntüsü 4.26

Storyboard ekranındaki ViewController ana penceresinin boyutlarını ayarlayarak iPad simülatör cihaz boyutuna getirebiliriz. Öncelikle ViewController penceresinin üst kısmına (1) tıklayınız. Ardından sağ taraftan Freeform'u(2) seçiniz.

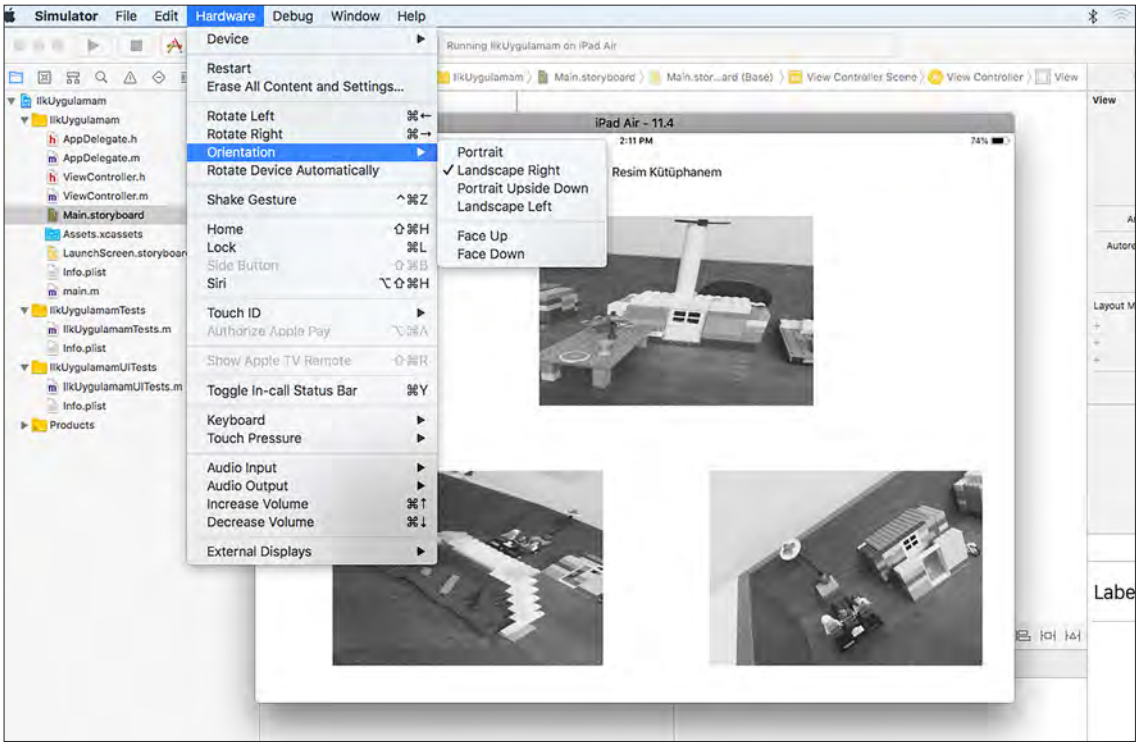


Ekran Görüntüsü 4.27

FreeForm seçilen kutunun altında açılan alandan genişlik ve yükseklik değerlerini giriniz. Burada yatay konum iPad standart boyutları girilmiştir.

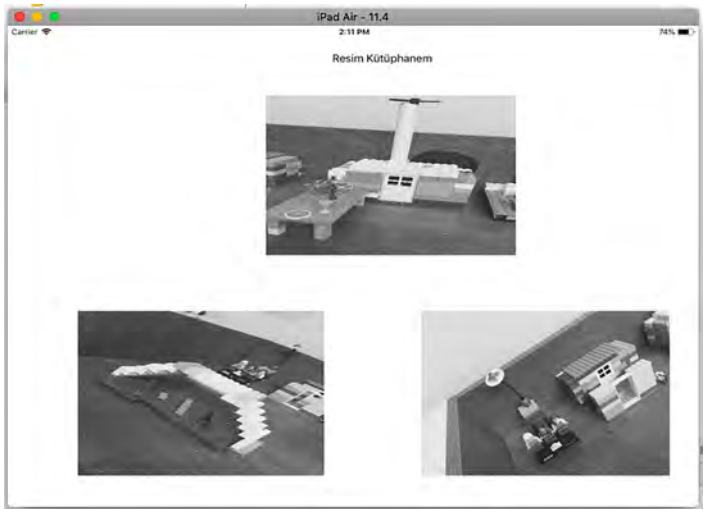
Uygulama iPad Air simülatöründe çalıştırıldığında ekran görüntüsü yandaki gibi olmaktadır.

Ekran Görüntüsü 4.28



Ekran Görüntüsü 4.29

Simülatör cihaz boyutunu yatayda 1024x768 piksel boyutlarına çevirmek için simülatör açık iken Hardware menüsünden Orientation ve LandscapeRight seçeneğini tıklayınız.



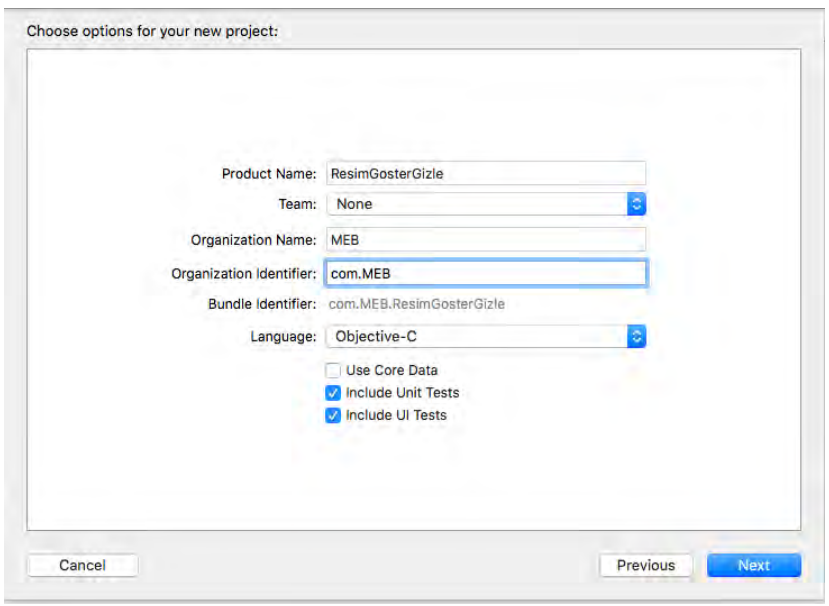
Ekran Görüntüsü 4.30

Yeni simülatör ekran görüntünüz yukarıdaki gibi olacaktır.

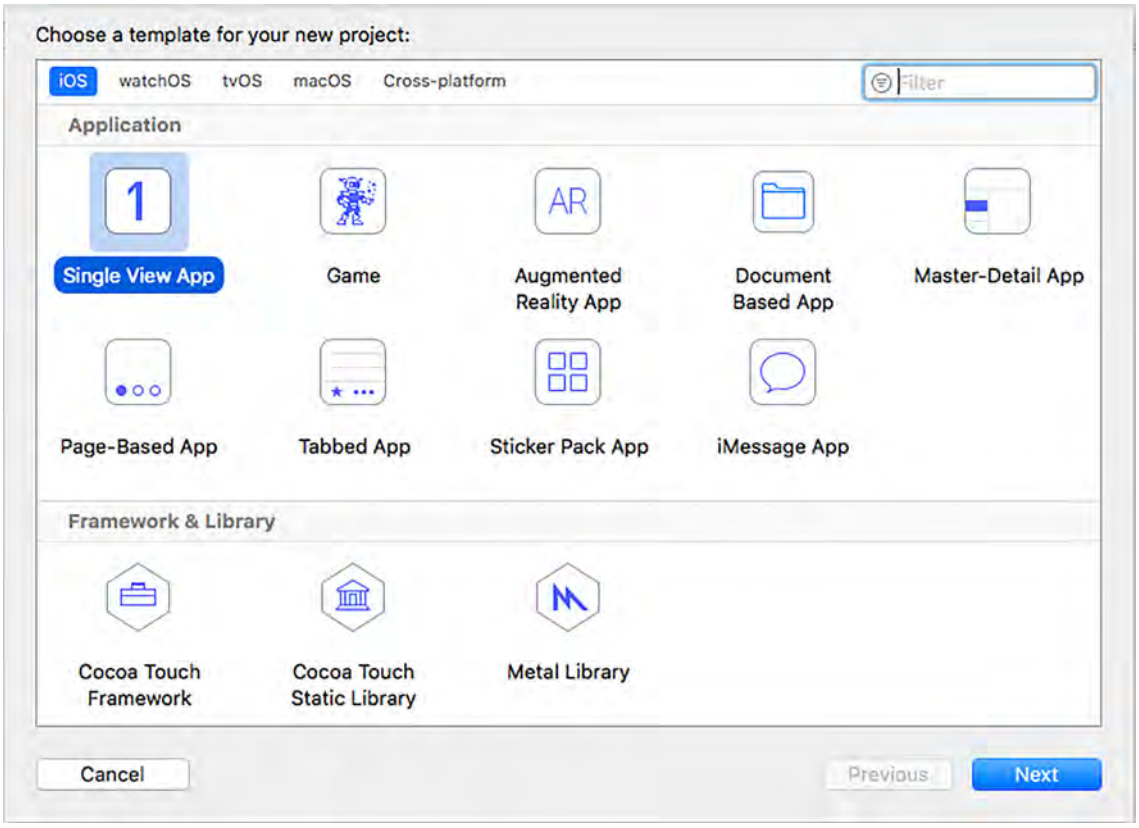
4.2.5. Resim Göster/Gizle Uygulaması

Bu uygulamada ekrana bir ImageView ve Button eklenecektir. Kullanıcı butona tıklayarak ImageView içindeki resmi gösterip gizleyecektir. Resim görünür iken buton başlığı "Gizle", gizli iken de "Görün" olacaktır. Bu uygulama ile oluşturulan nesnelerin kod yazarak title, tag, imageName gibi özellikleri üzerinde değişiklik yapılacaktır.

ResimGosterGizle isimli bir Xcode projesi oluşturunuz. İlk uygulamanın geliştirilmesi sürecindeki başlangıç adımları takip ediniz.

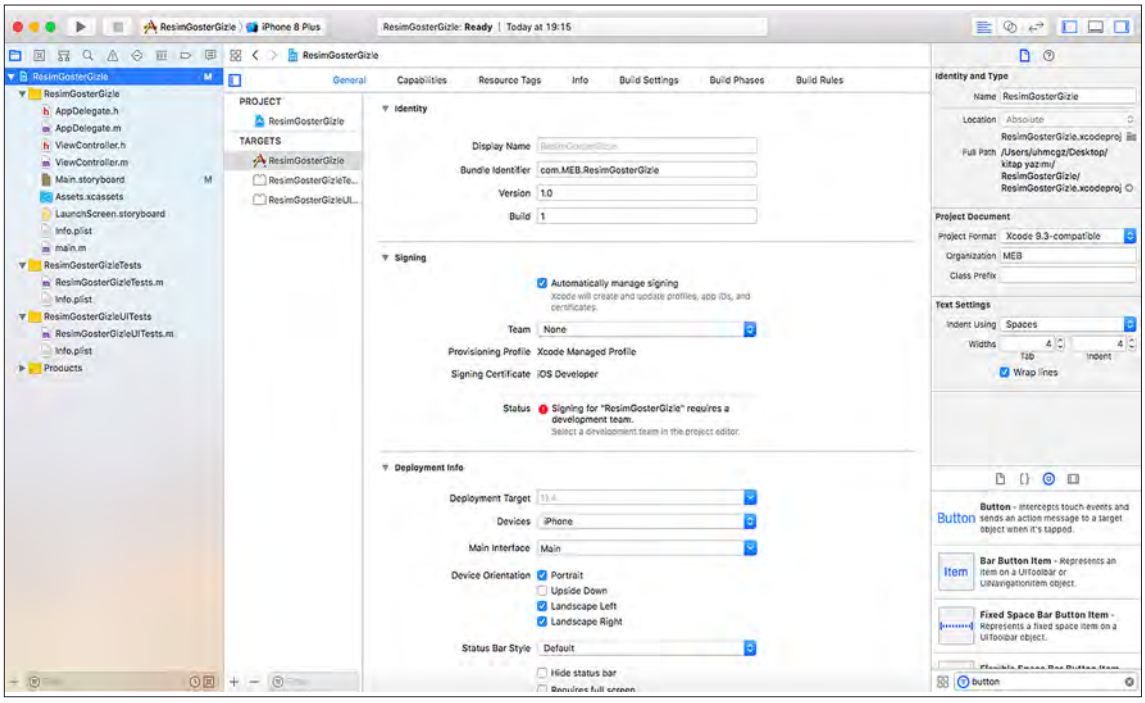


Ekran Görüntüsü 4.31



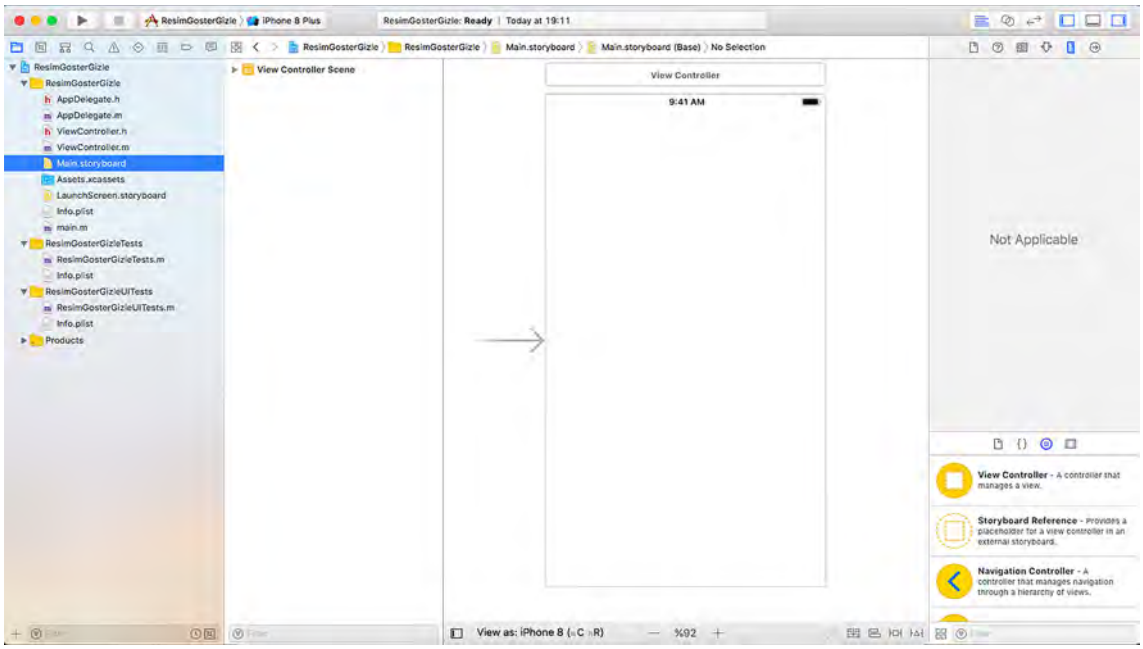
Ekran Görüntüsü 4.32

Bu uygulamada da farklı bir cihazı simülatörde görmek için iPhone seçiniz.



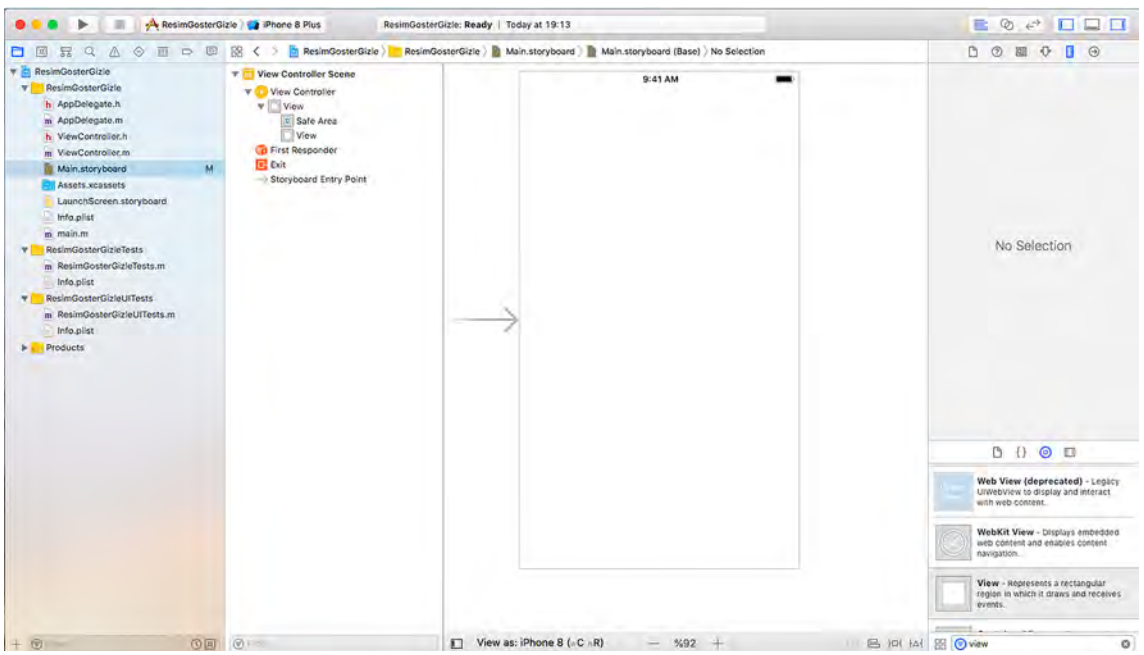
Ekran Görüntüsü 4.33

Main.Storyboard dosyasını açınız.



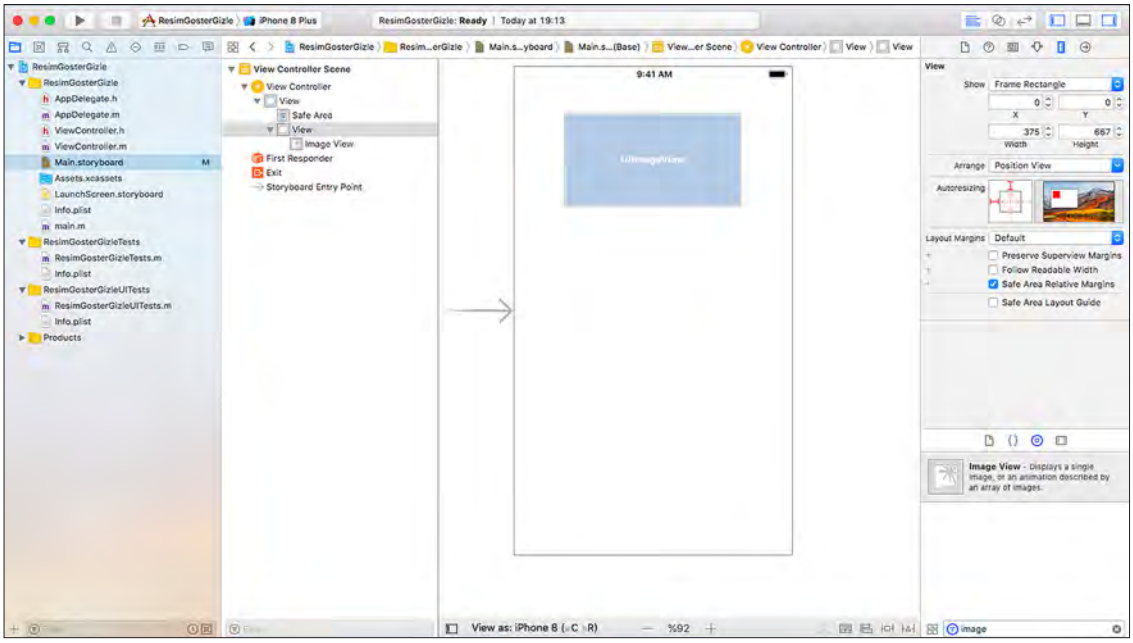
Ekran Görüntüsü 4.34

Sağ alt tarafta bulunan resim kütüphanesinden View nesnesini bulunuz ve ekrandaki ViewController üzerine sürükleyip bırakınız. View'ı tüm çerçeveyi kaplayacak şekilde genişletiniz.



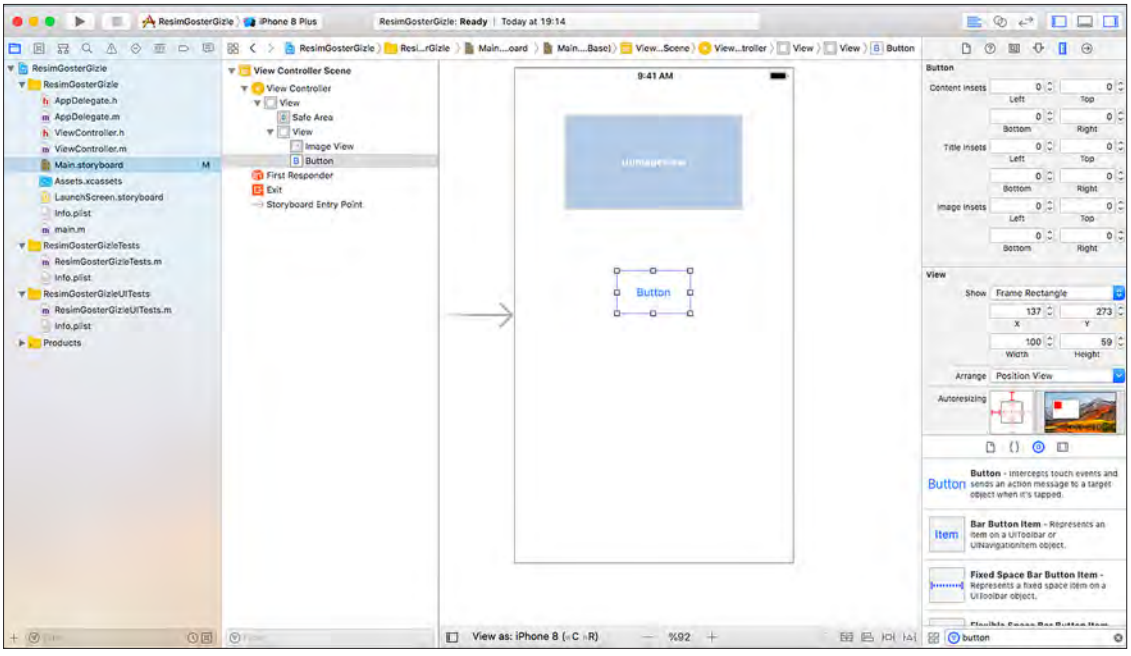
Ekran Görüntüsü 4.35

Ardından aynı sürükle bırak yöntemi ile bir ImageView ekleyiniz. Bu nesneyi de kenar çizgilerinde bulunan kare işaretlerden istenilen boyuta getirebilirsiniz.



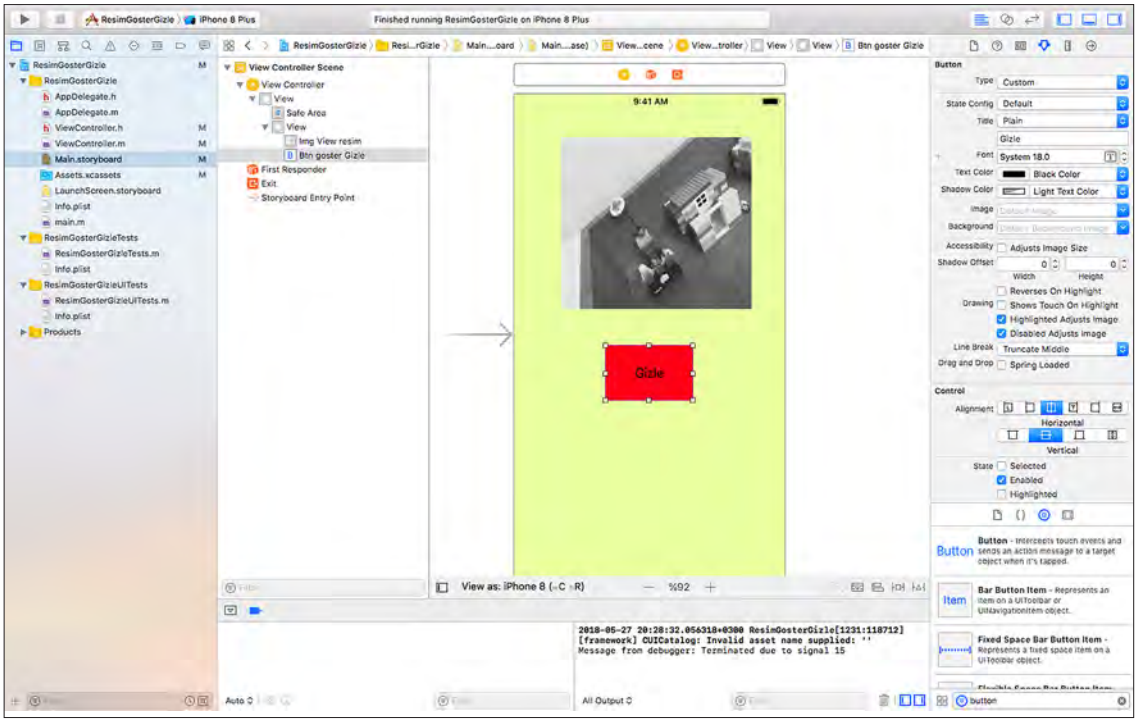
Ekran Görüntüsü 4.36

ImageView'ın altına bir Button sürükleyip bırakınız.



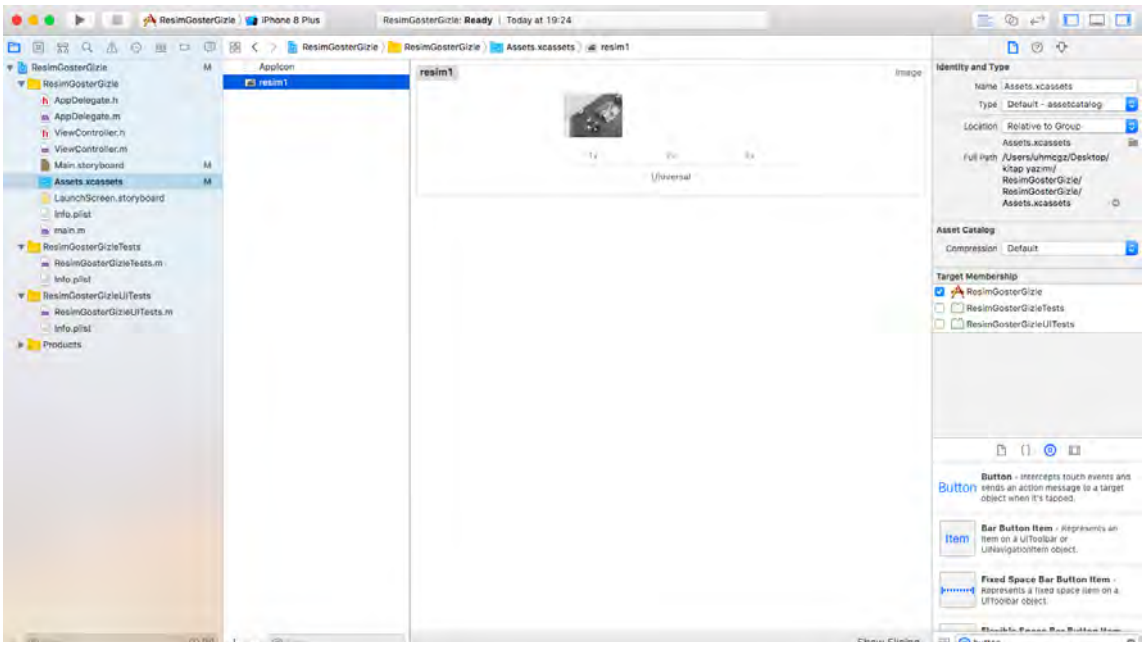
Ekran Görüntüsü 4.37

Buton ekranda seçili iken sağ taraftan bazı Type, Title, TextColor, Background Color özelliklerini (Ekran Görüntüsü 4.37) gibi değiştiriniz. "Type: Custom" seçimi düğmenin (buton) istenildiği gibi özellikleri ile oynanabileceği anlamına gelir. Title buton üzerindeki metnin, TextColor metnin renginin, BackgroundColor ise resim kullanılmayacaksa butonun arkaplan rengini değiştirmeyi sağlamaktadır.



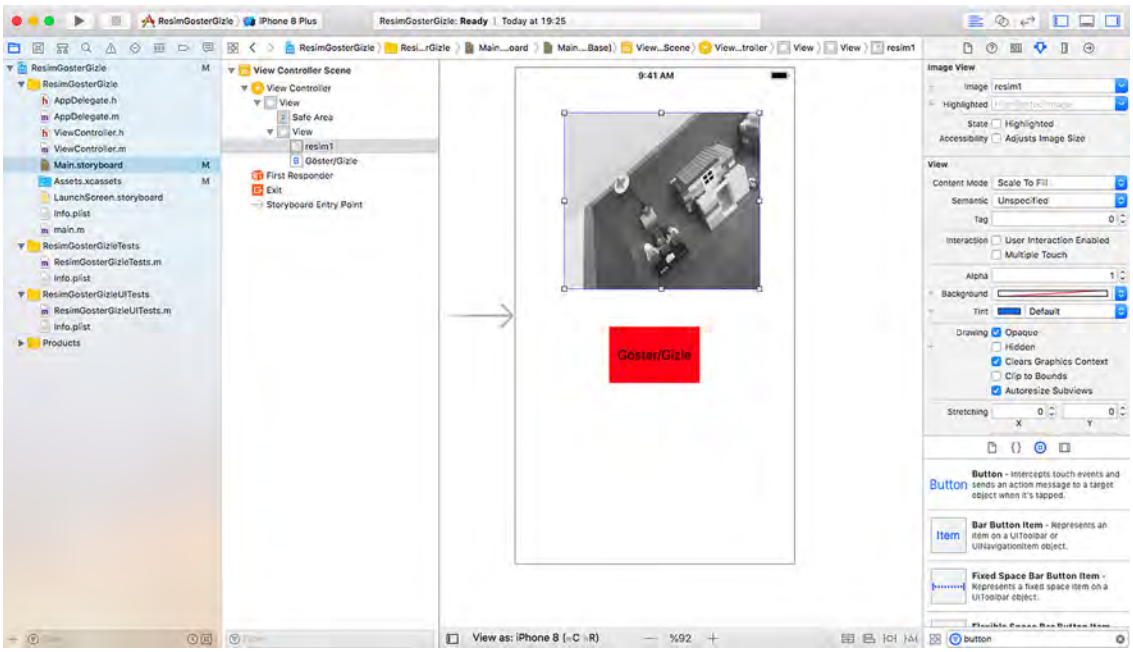
Ekran Görüntüsü 4.38

ImageView'e resim eklemek için öncelikle Assests.xcassets dosyası açılır ve bilgisayara kayıtlı bir png uzantılı resim dosyası sürüklenerek AppIcon yazılı orta alana bırakılır. Üzerinde sağ tıklanarak resim dosyasının adı değiştirilir.



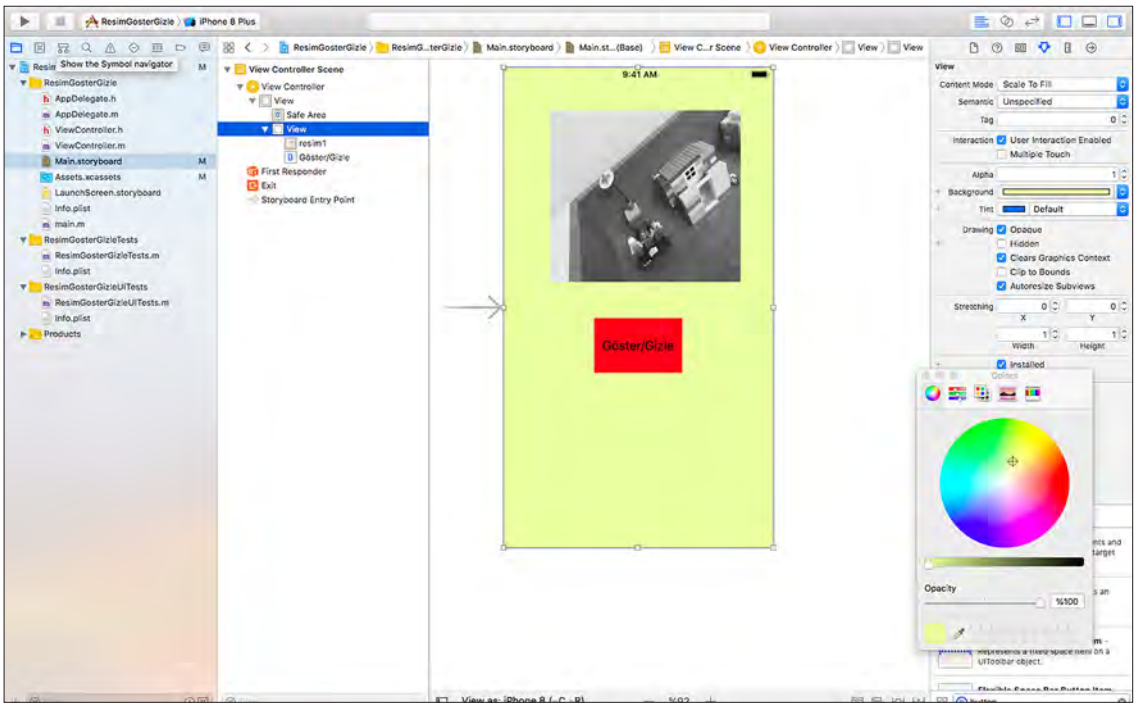
Ekran Görüntüsü 4.39

Storyboard ekranında ImageView tıklanarak sağ taraftan Image alanına Assets dosyasına eklenilen resimin adı yazılır veya dropbox listesinden seçilir.



Ekran Görüntüsü 4.40

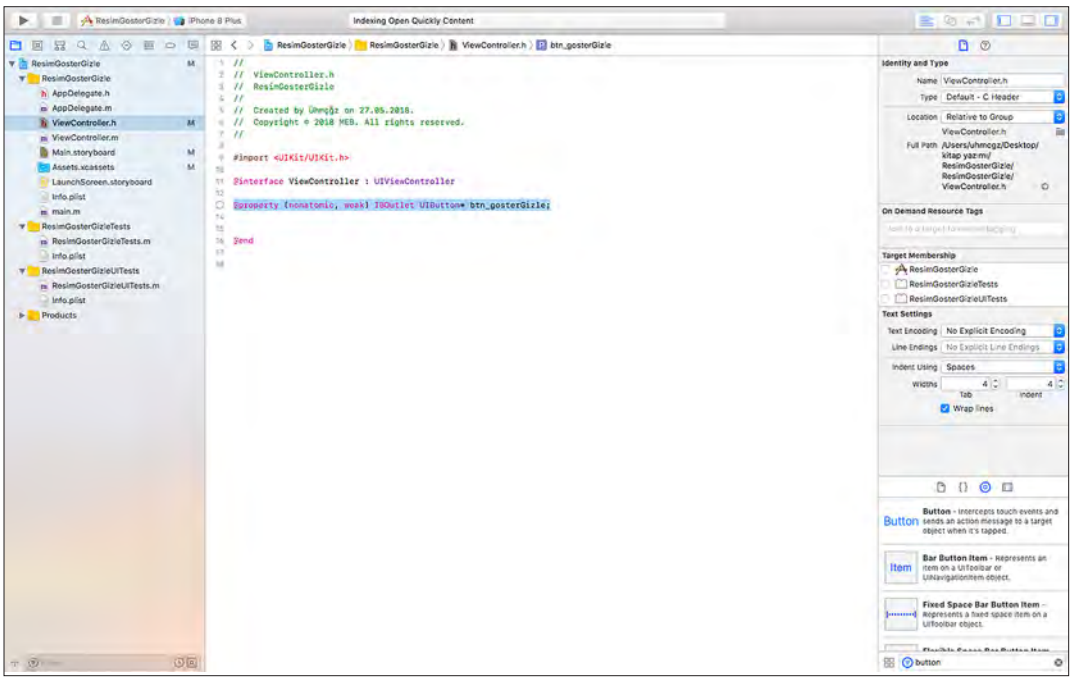
Uygulama arkaplan rengini değiştirmek için sol bölümden View'ı seçerek Background alanından renk paletinden istenilen renk seçilebilir.



Ekran Görüntüsü 4.41

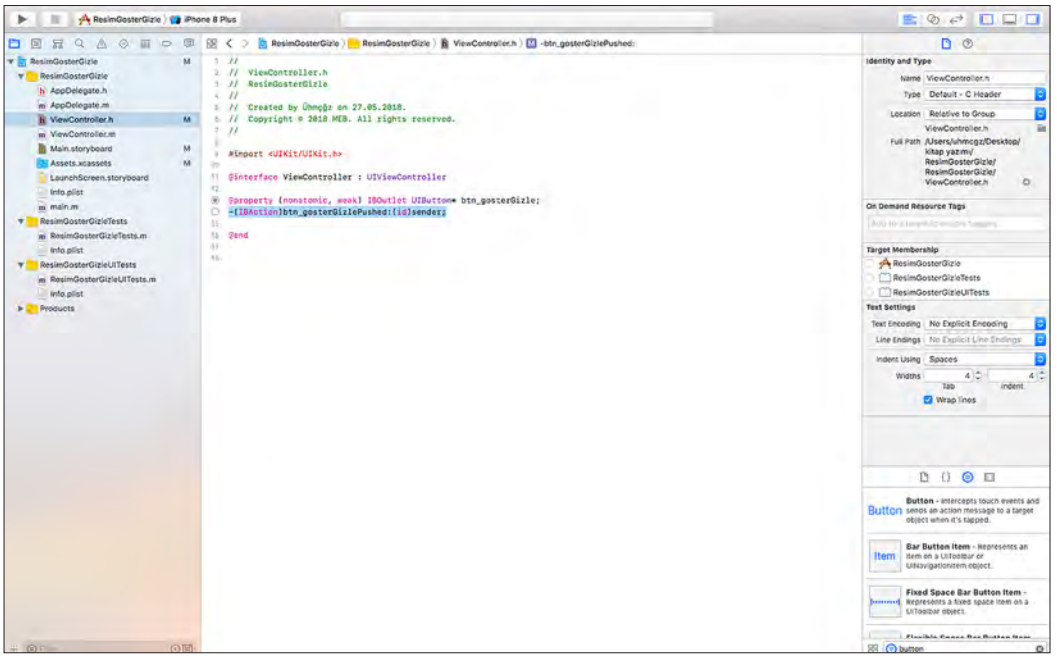
Xcode'un tanıtıldığı giriş bölümünde de belirtildiği gibi nesne tanımlamaları ViewController.h dosyasının içinde yapılmaktadır.

```
@property (nonatomic, weak) IBOutletUIButton* btn_gosterGizle;
```



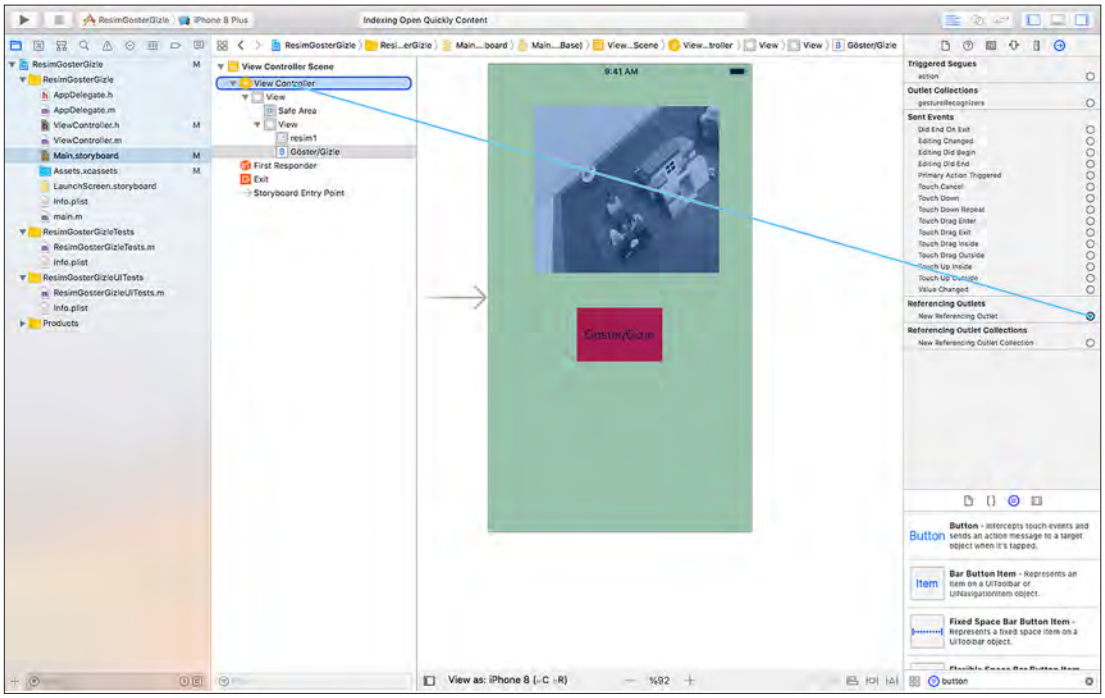
Ekran Görüntüsü 4.42

Button tıklandığında yapılacak işlemlerin yazıldığı fonksiyonun tanımlanması da aynı dosya içinde yapılmaktadır.



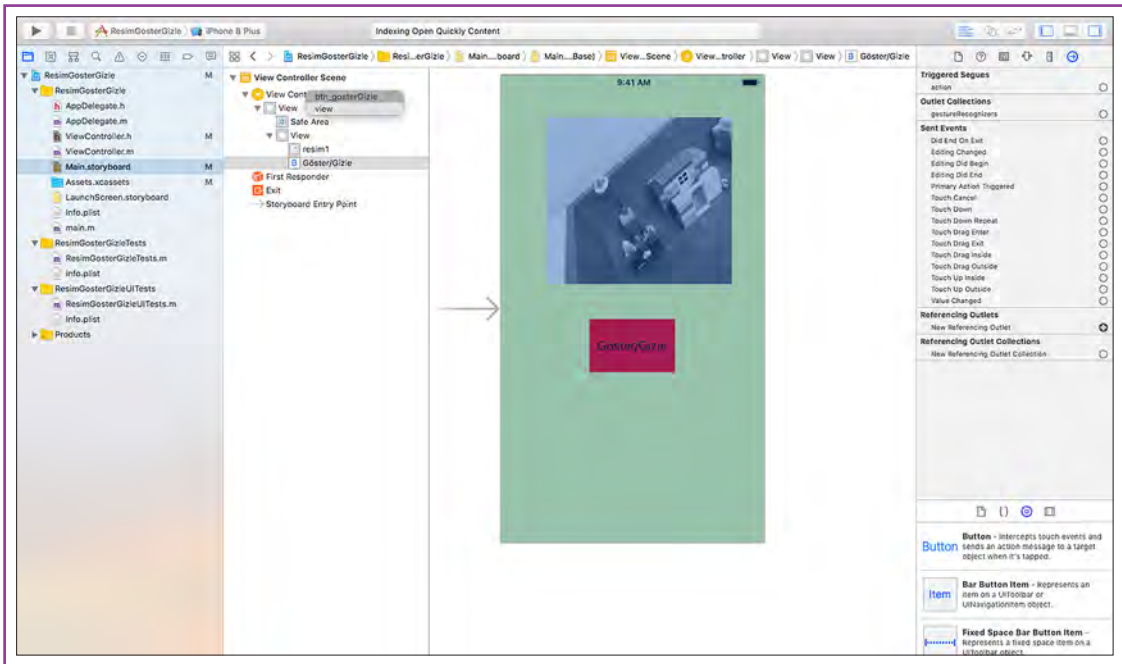
Ekran Görüntüsü 4.43

Yazılan tanımlama kodlarının nesnelerle ilişkilendirilmesi için Storyboard'u açınız. Button seçili iken sağ bölümde References sekmesini açınız ve New Referencing Outlet'in sağ tarafındaki yuvarlak simgeyi tıklayıp bırakmadan sürükleyip sol bölümde bulunan ViewController simgesinin üzerine bırakınız.



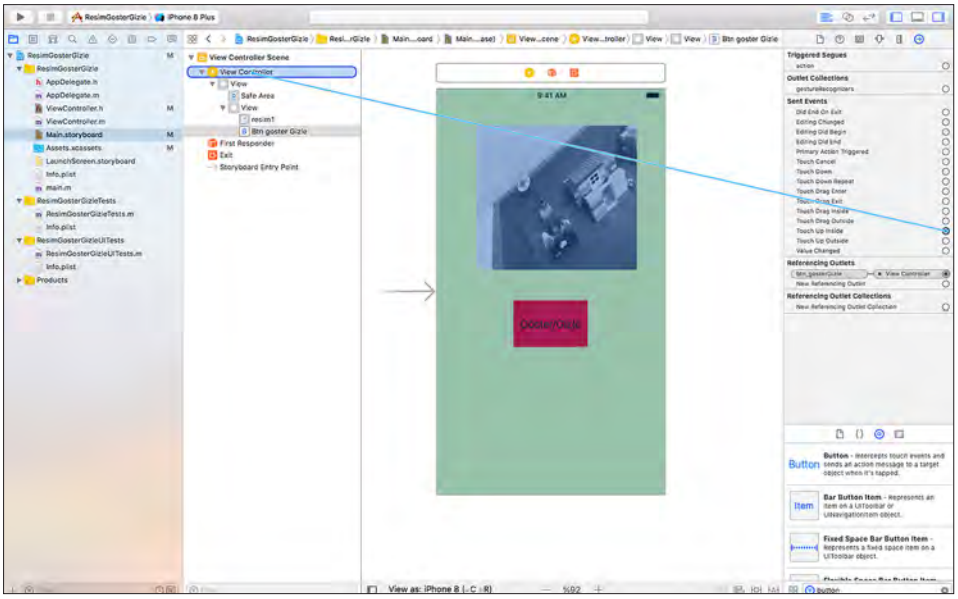
Ekran Görüntüsü 4.44

Karşınıza gelen seçeneklerden nesne tanımlanırken nesneye verilen isim seçilir. Listeden "btn_gosterGizle" yi seçiniz.



Ekran Görüntüsü 4.45

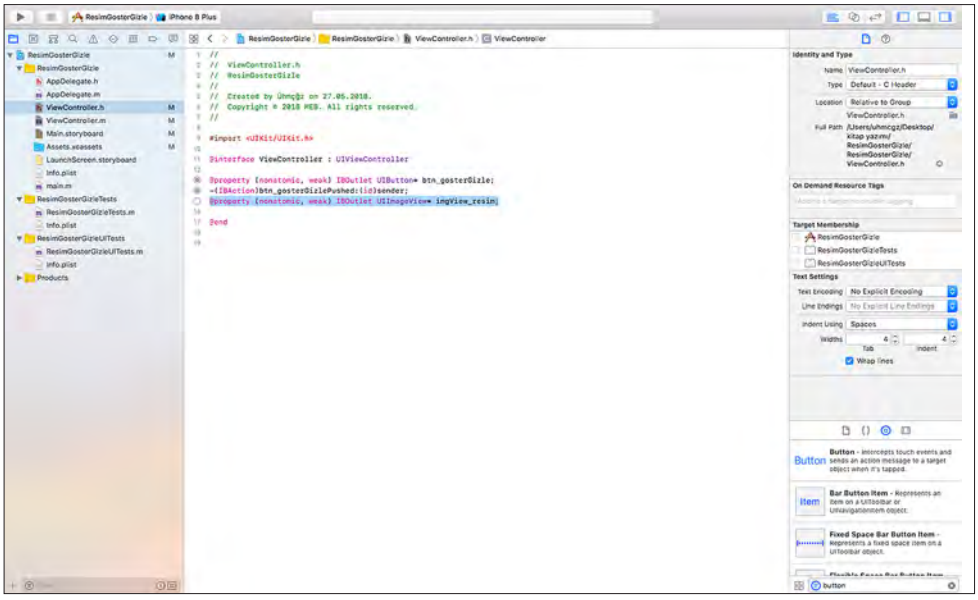
Düğmesine tıklandığında çalışacak fonsiyon ise, Buttonun Referencese sekmesindeki TouchUp Inside ile bir önceki adımdaki benzer yöntemle birbirine bağlanır.



Ekran Görüntüsü 4.46

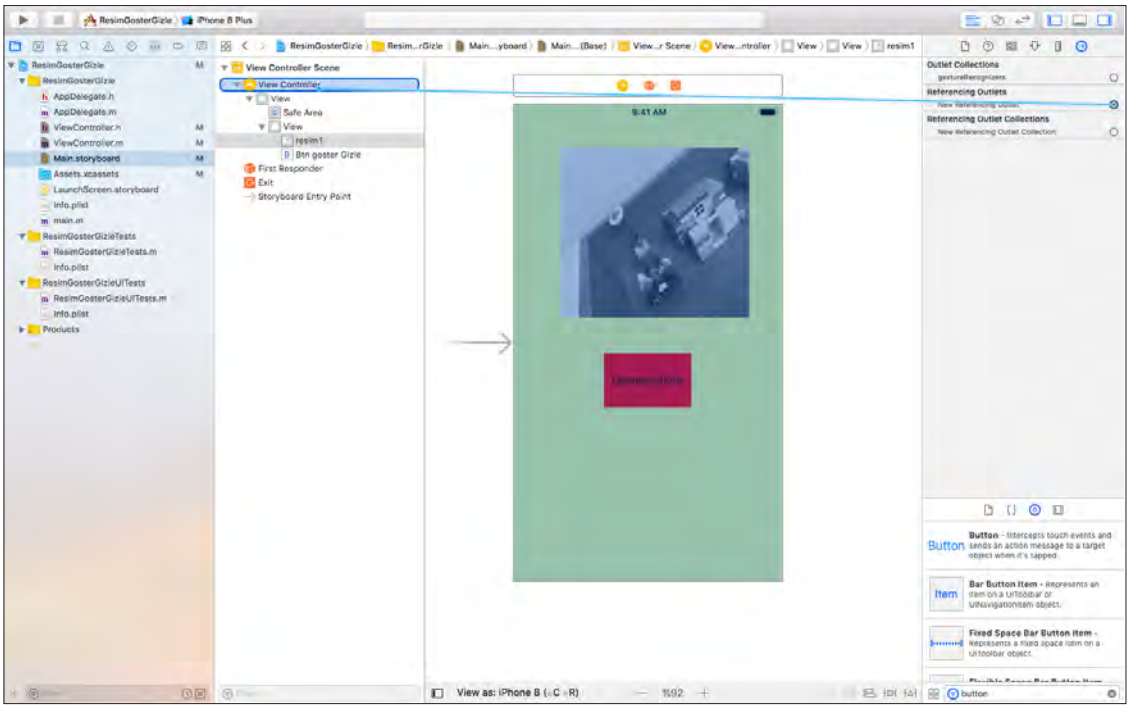
ImageView resminin kullanıcının butona tıklaması ile gösterilip gizlenmesi gerekmektedir. Kod içerisinde ImageView nesnesinin özelliklerinin değiştirilmesi gerektiği için ViewController.h dosyasında tanımlama yapılması gerekmektedir. Aşağıdaki kod satırını dosyaya ekleyiniz.

```
@property (nonatomic, weak) IBOutletUIImageView* imgView_resim;
```

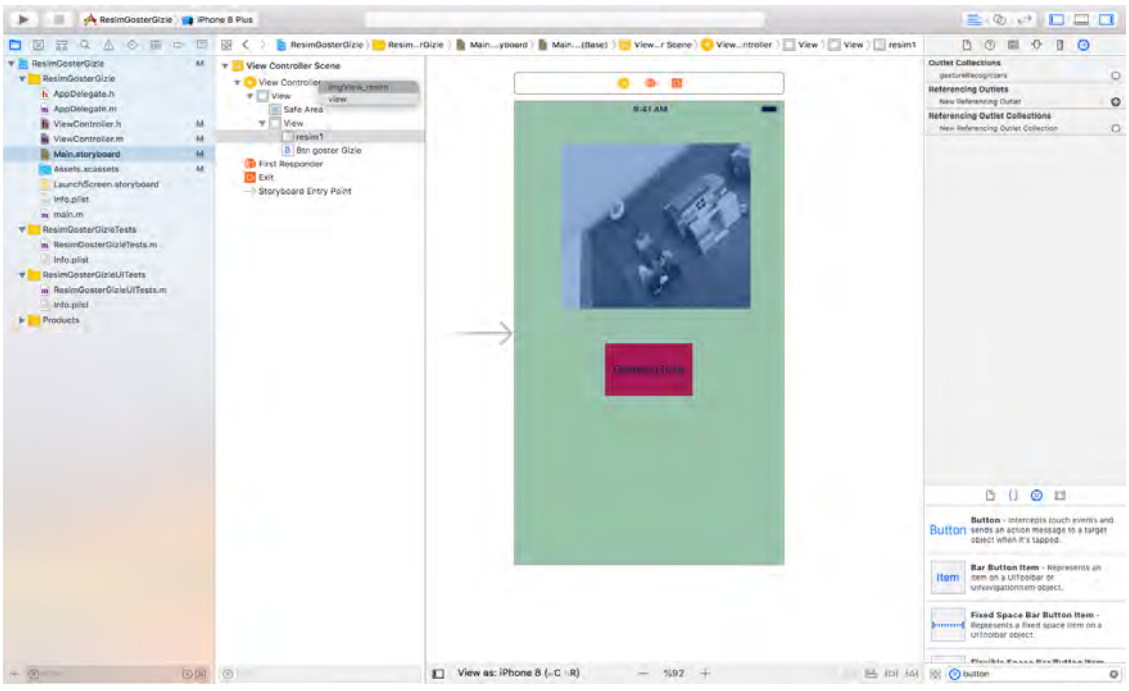


Ekran Görüntüsü 4.47

Ardından Storyboard'da yazılan tanımlama kodu, nesne ile ilişkilendirilir.

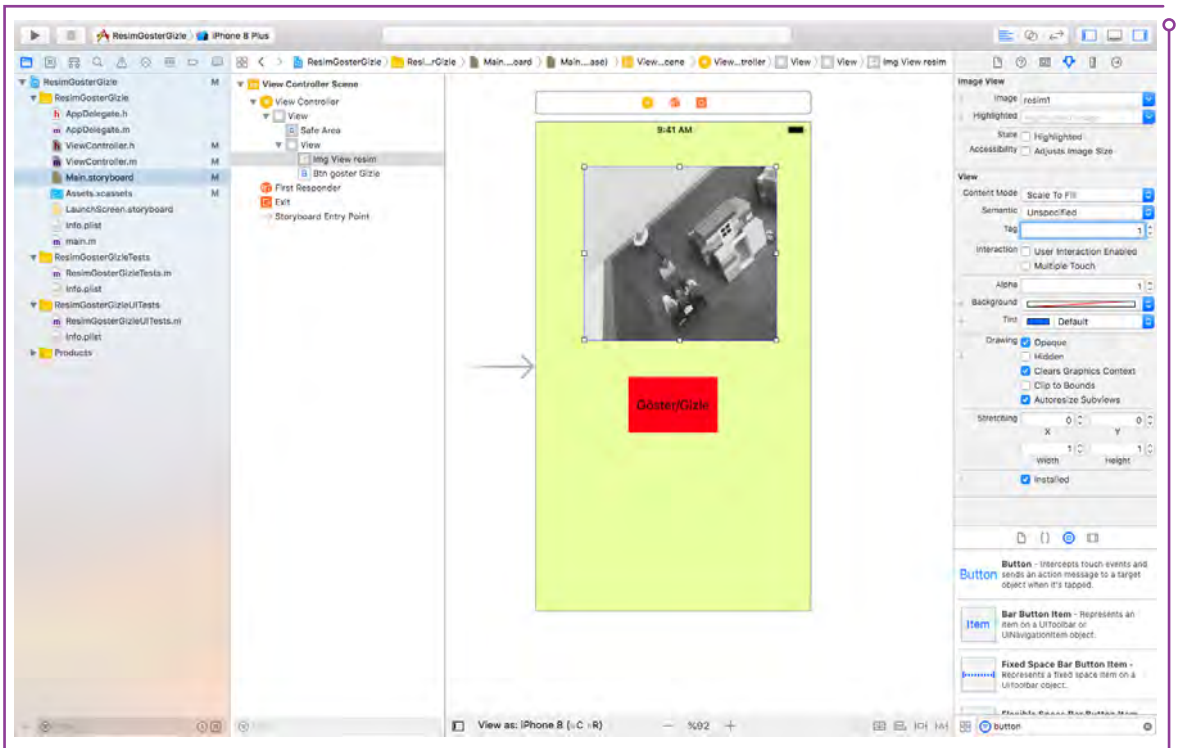


Ekran Görüntüsü 4.48



Ekran Görüntüsü 4.49

Storyboard'da son işlem olarak ImageView nesnesinin tag'i (etiketi) 1 yapılarak düğme tıklandığında resmin ekranda gizlenmesi mi yoksa gösterilmesi mi gerektiğinin kontrolü sağlanacaktır.



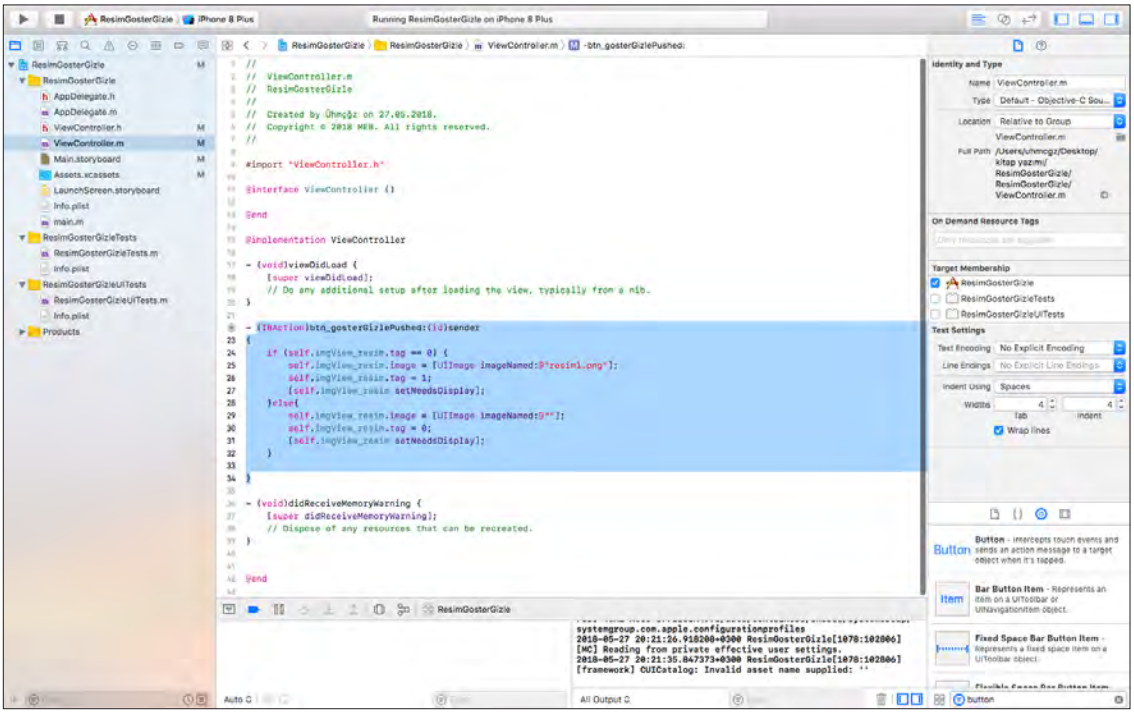
Ekran Görüntüsü 4.50

Bir sonraki adımda, ViewController.m dosyasında Button tıklandığında çalışacak kodlar yazılacaktır. Öncelikle "buton tıklandığında" fonksiyonu yazılır.

```
- (IBAction)btn_gosterGizlePushed:(id)sender
{


}
```

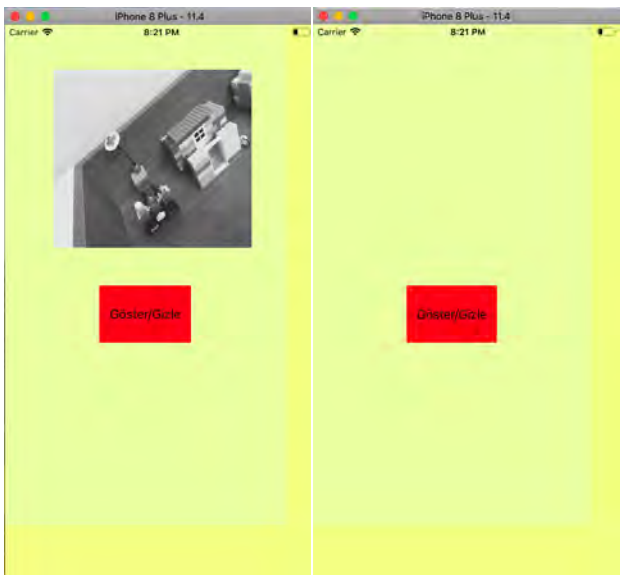
Fonksiyon içerisine *If-Else* koşulu içinde gerekli kodlar yazılır.

```
if (self.imgView_resim.tag == 0) {    // Eğer ImageView'in tag'i 0'a eşit ise
self.imgView_resim.image = [UIImageimageNamed:@"resim1.png"]; // ImageView' e resim1.png isimli resimi yerleş-
tir
self.imgView_resim.tag = 1;  // ImageView'intag'ini 1 yap
[self.imgView_resimsetNeedsDisplay]; // ImageView ile ilgili yapılan tüm değişiklikleri görüntüle
}else{  // Eğer ImageView'intag'i 1'e eşit değilse yani 0 ise
self.imgView_resim.image = [UIImageimageNamed:@""];  // ImageView' e null isimli resmi yerleştir (dolaylı olarak
sil)
self.imgView_resim.tag = 0;       // ImageView'in tag'ini 0 yap
            [self.imgView_resimsetNeedsDisplay];   // ImageView ile ilgili yapılan tüm değişiklikleri görüntüle
}
```



Ekran Görüntüsü 4.51

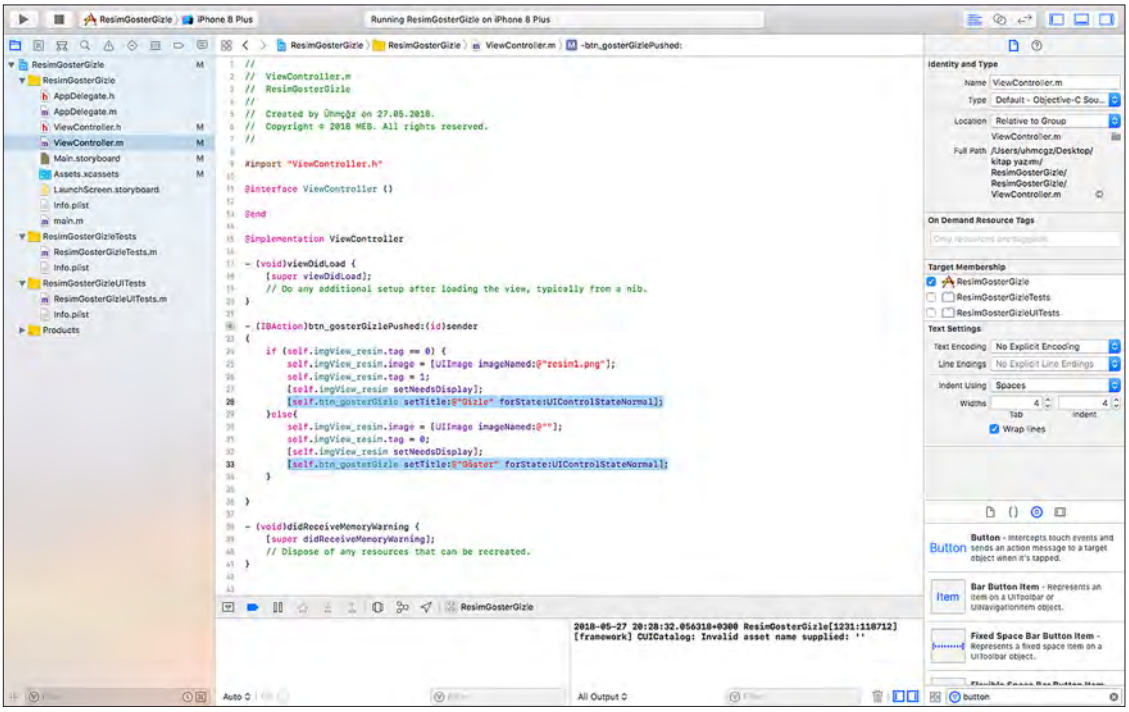
Uygulamayı simülatörde çalıştırdığınızda ve Göster/Gizle düğmesine tıkladığınızda ekran görüntüsü yandaki gibi olacaktır.



Ekran Görüntüsü 4.52

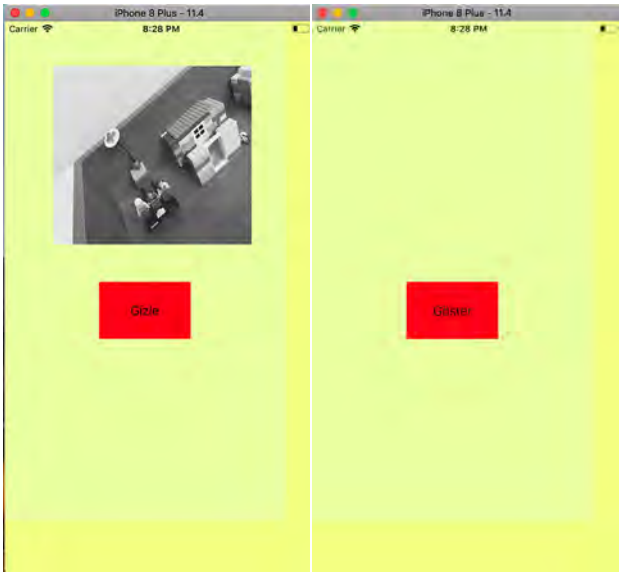
Eğer her tıklamada düğmenin isminin resmin görünürlüğüne göre Göster veya Gizle olarak değişmesini istiyorsanız ViewController.m dosyasına şu kodları da eklemeniz gerekmektedir.

```
[self.btn_gosterGizlesetTitle:@"Gizle"forState:UIControlStateNormal];
[self.btn_gosterGizlesetTitle:@"Göster"forState:UIControlStateNormal];
```



Ekran Görüntüsü 4.53

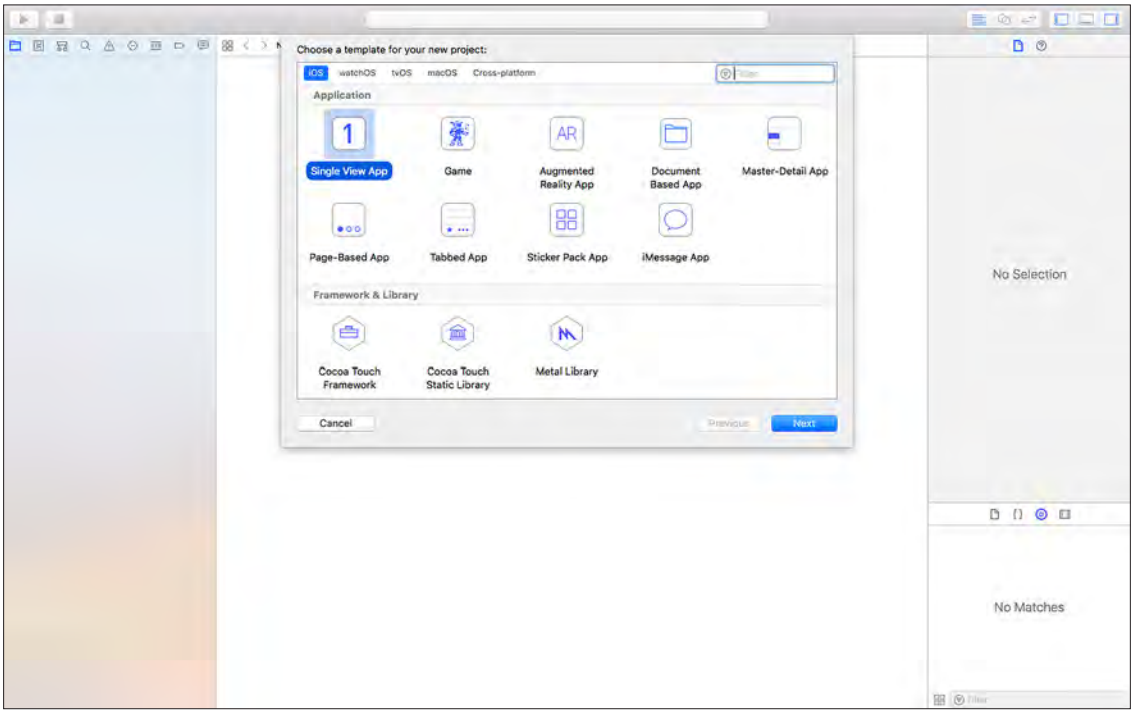
Uygulama çalıştırıldığında ve butona tıklandığında yeni ekran görüntüsü yandaki gibi olacaktır



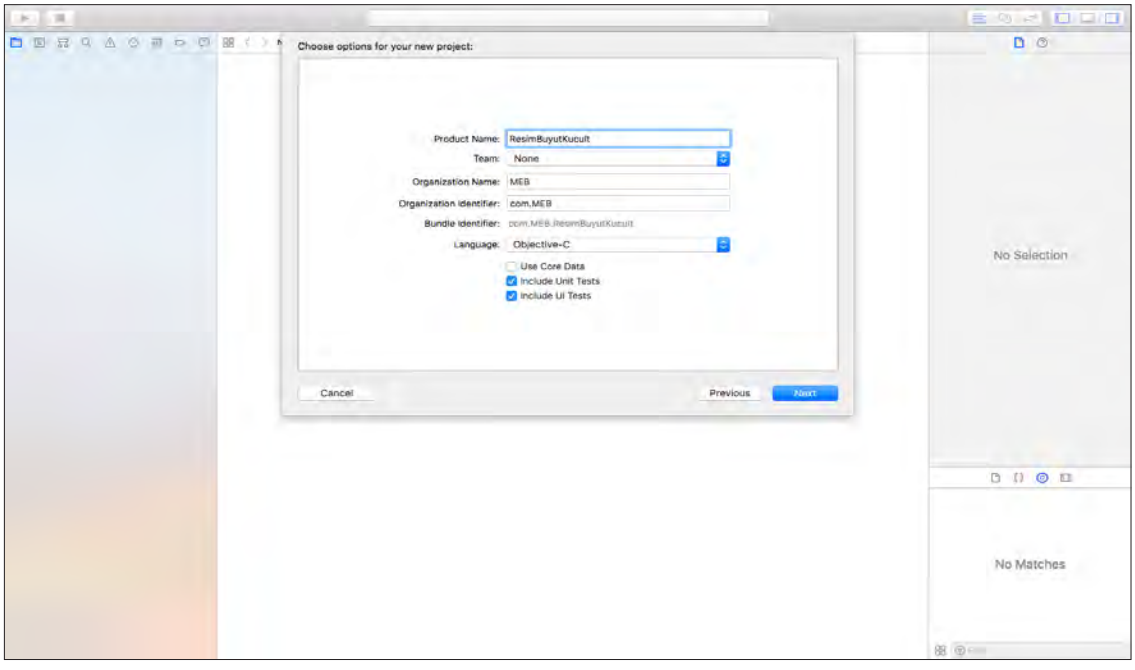
Ekran Görüntüsü 4.54

4.2.6. Resim Büyüt/Küçült Uygulaması

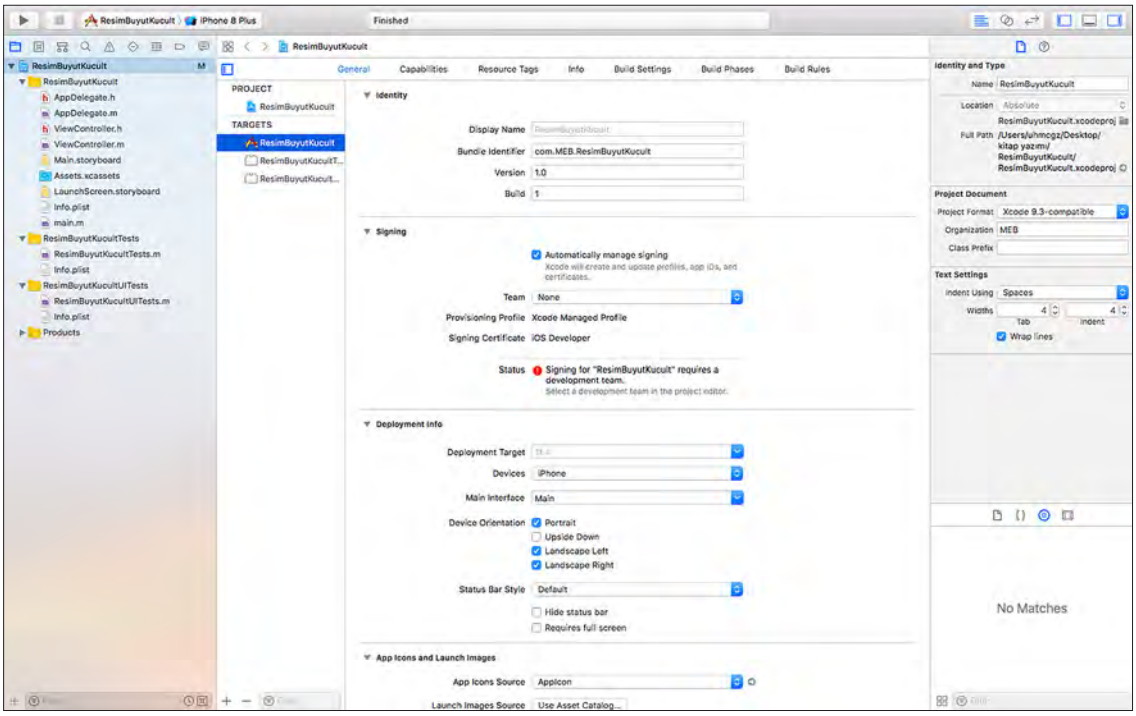
Bu uygulamada ekranda bulunan bir resmin (ImageView) boyutu iki farklı buton ile büyütülüp küçültülecektir. Öncelikle yeni bir proje açarak isim verilir ve cihaz olarak IPhone seçilir.



Ekran Görüntüsü 4.55

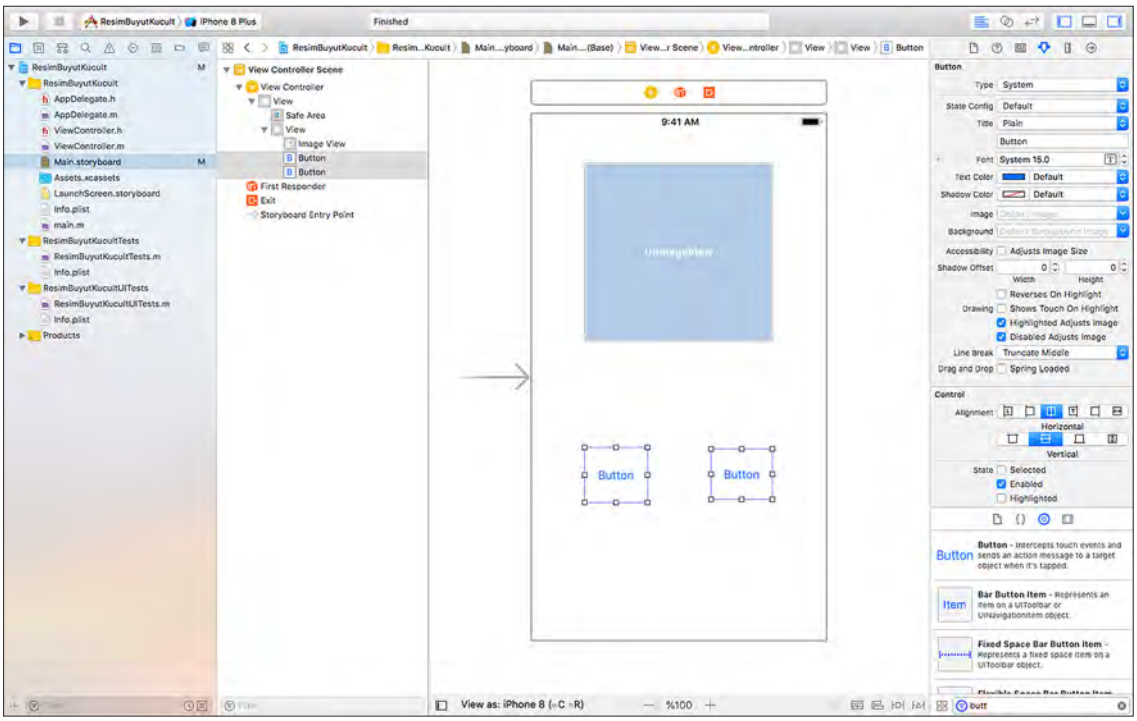


Ekran Görüntüsü 4.56



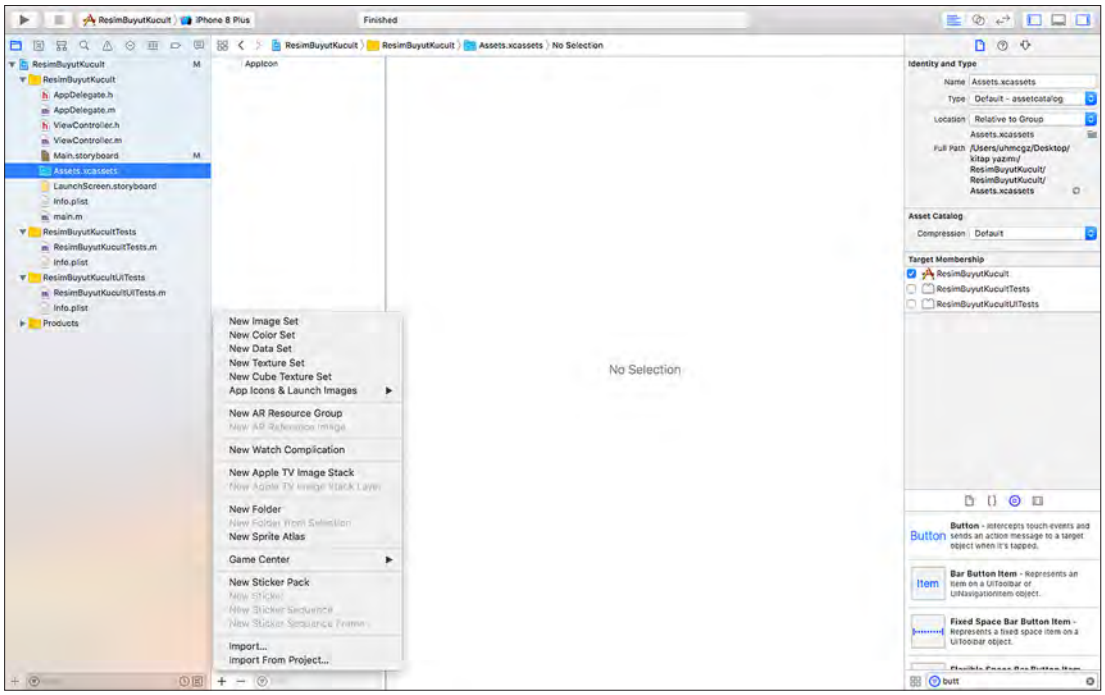
Ekran Görüntüsü 4.57

Main.Storyboard ekranı açılarak öncelikle bir View eklenir ve ekran boyutunda genişletilir. Daha sonra bir ImageView, iki buton ekrana yerleştirilerek uygun boyutlara getirilir.

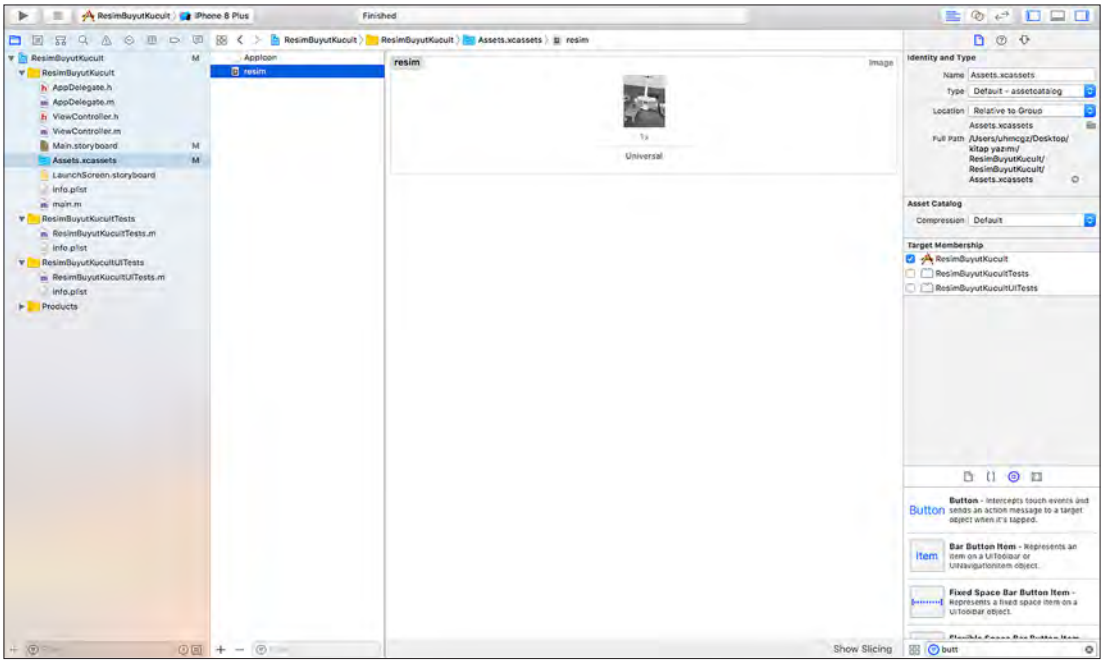


Ekran Görüntüsü 4.58

ImageView'ın içerisine resim eklemek için Assets dosyası açılır ve sürükle bırak yöntemiyle veya alt kısımda bulunan + düğmesine tıklanıp Import seçilerek resim projeye aktarılır.

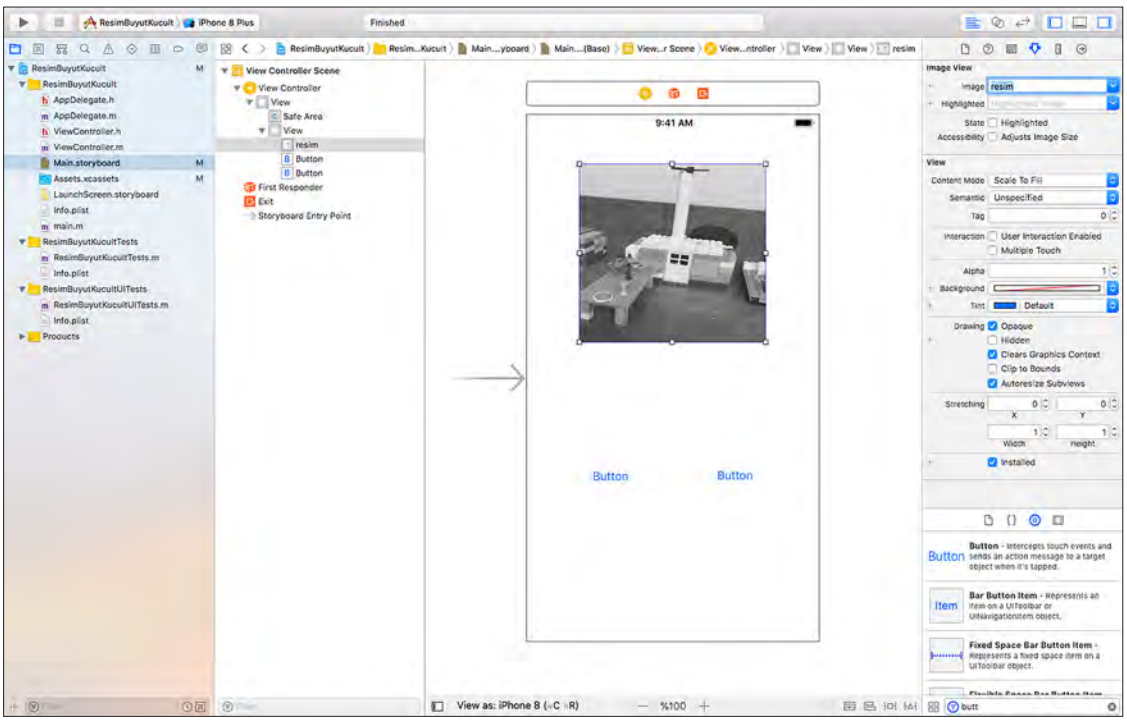


Ekran Görüntüsü 4.59



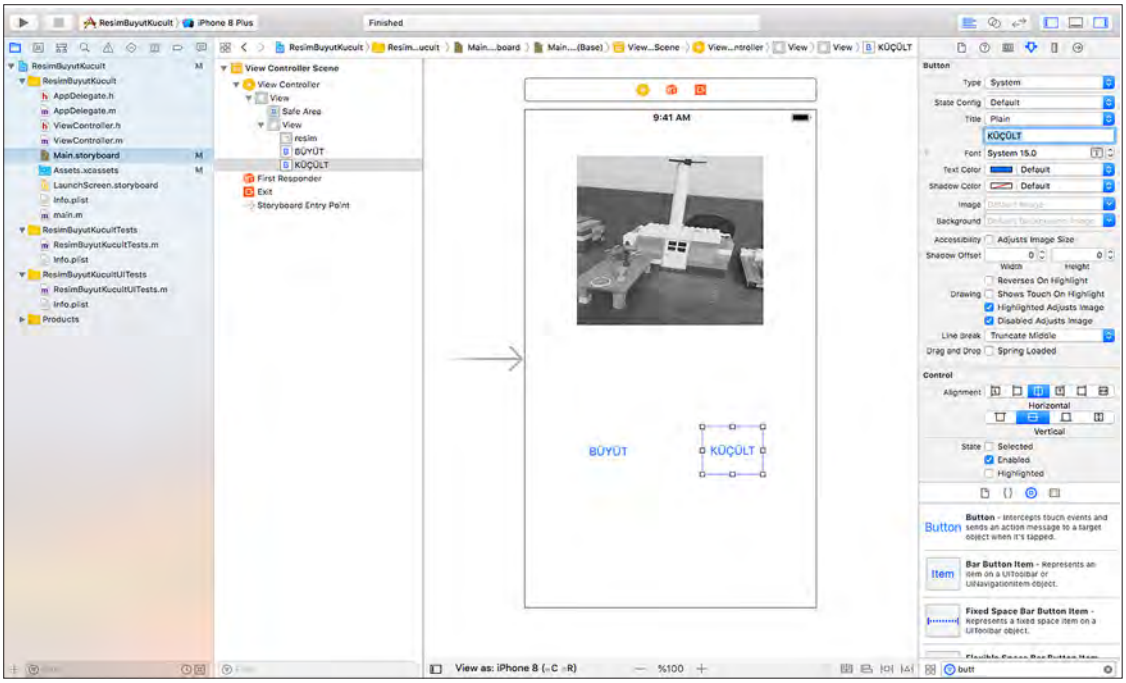
Ekran Görüntüsü 4.60

Storyboard ekranına geri dönülerek Imageview ekranda seçilir ve sağ bölümde "properties" (özellikler) sekmesinde Image alanına resmin adı yazılır veya açılır liste tıklanarak seçilir. Enter tuşuna basılarak değişiklik görüntülenir.



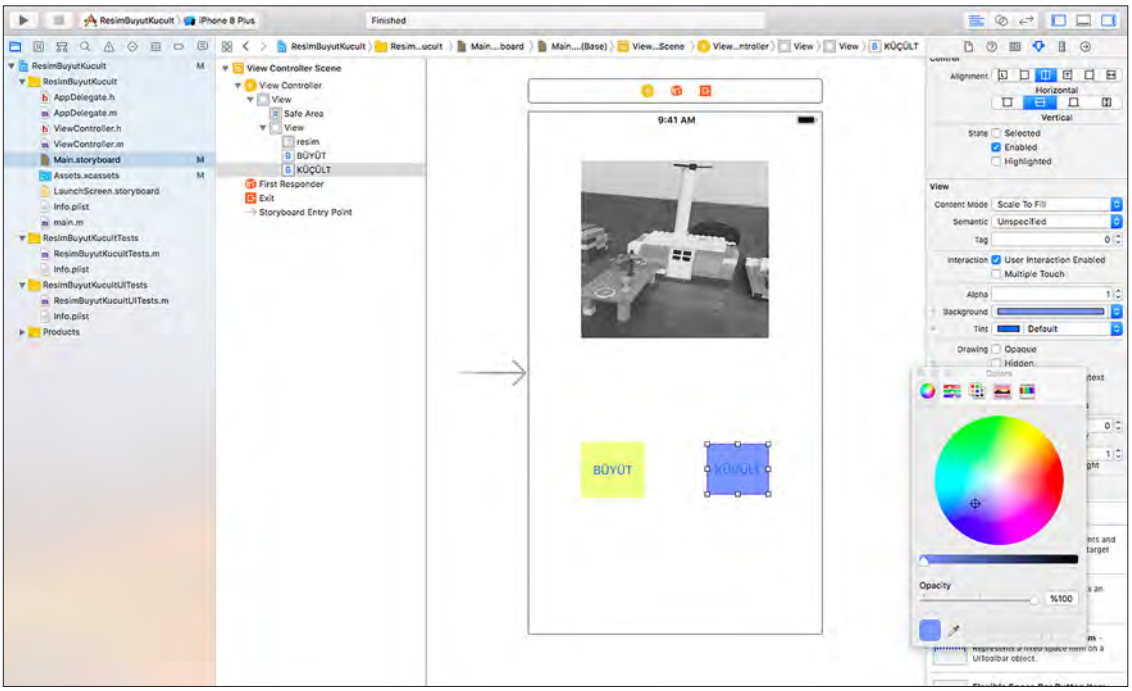
Ekran Görüntüsü 4.61

Düğmeler sırasıyla seçilerek sağ bölmeden isimleri “BÜYÜT” ve “KÜÇÜLT” şeklinde değiştirilir.



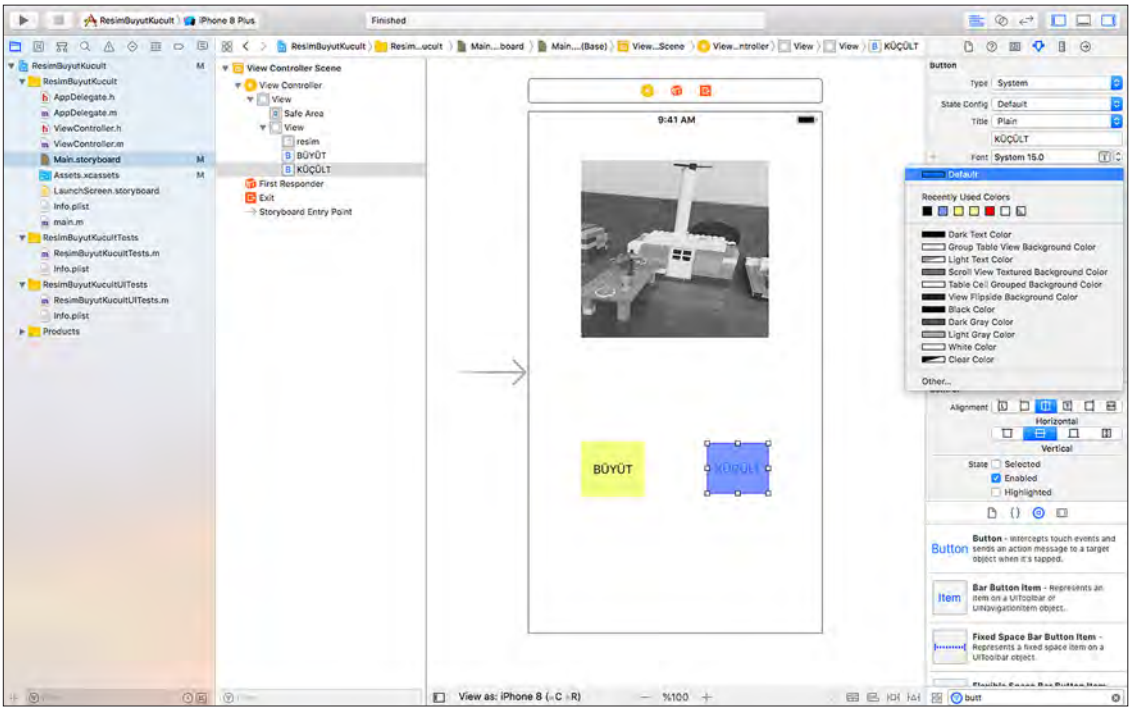
Ekran Görüntüsü 4.62

Düğmelerin belirginleşmesi için arkaplan renklendirmesi yapılabilir. Bunun için de sağ bölümde özellikler sekmesinde Background alanından tercih edilen renk paletinden veya listeden seçilir.



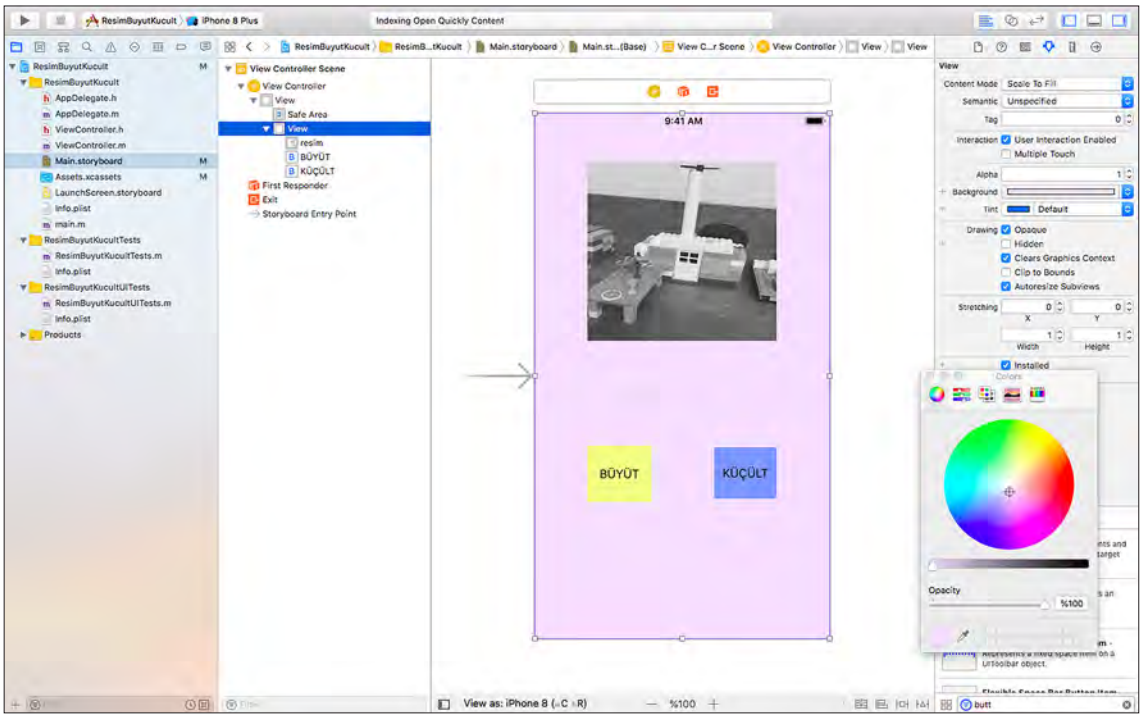
Ekran Görüntüsü 4.63

Düğmelerin metin rengini değiştirmek için TextColor özelliği kullanılır.



Ekran Görüntüsü 4.64

Son olarak StoryBoard ekranında View'a BackGround alanından renk verilerek arkaplan renklendirilir.



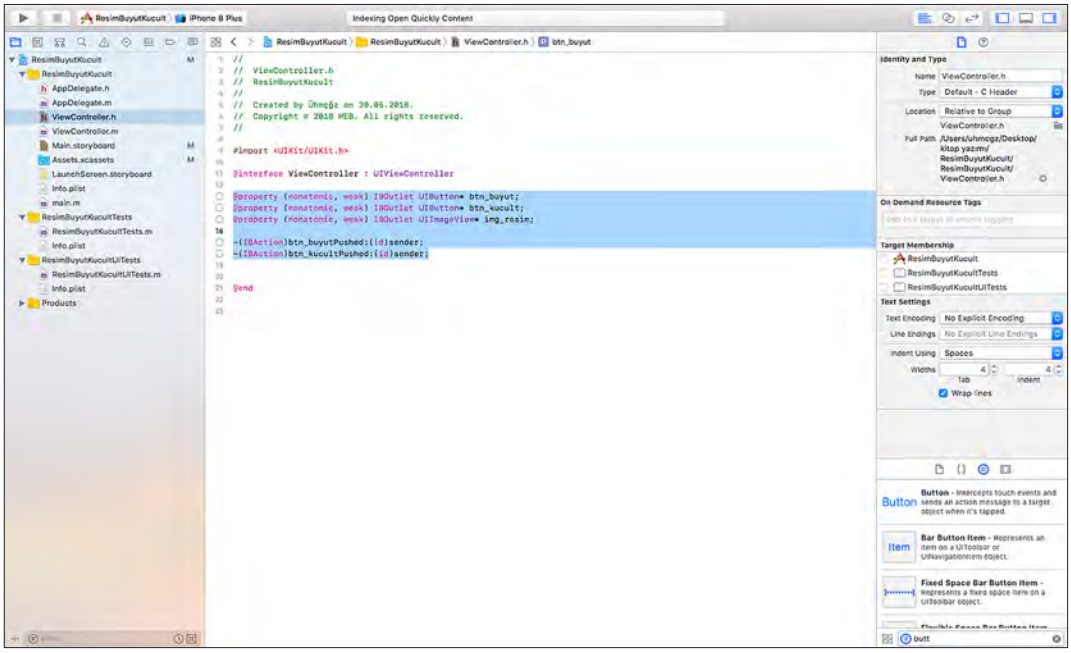
Ekran Görüntüsü 4.65

Main.Storyboard dosyasında oluşturulan nesnelerin (ImageView ve 2 adet Button) programlanabilmesi için isim verilerek proje dosyasına programsal anlamda tanıtılması gerekmektedir. Bunun için tanımlamaların yapıldığı ViewController.h dosyası açılır. Nesneler aşağıdaki kodlar yazılarak projede tanımlanır.

```
@property (nonatomic, weak) IBOutletUIButton* btn_buyut;
@property (nonatomic, weak) IBOutletUIButton* btn_kucult;
@property (nonatomic, weak) IBOutletUIImageView* img_resim;
```

Daha sonra da butonların tıklanma fonksiyonlarının tanımlanması aşağıdaki gibi yapılır.

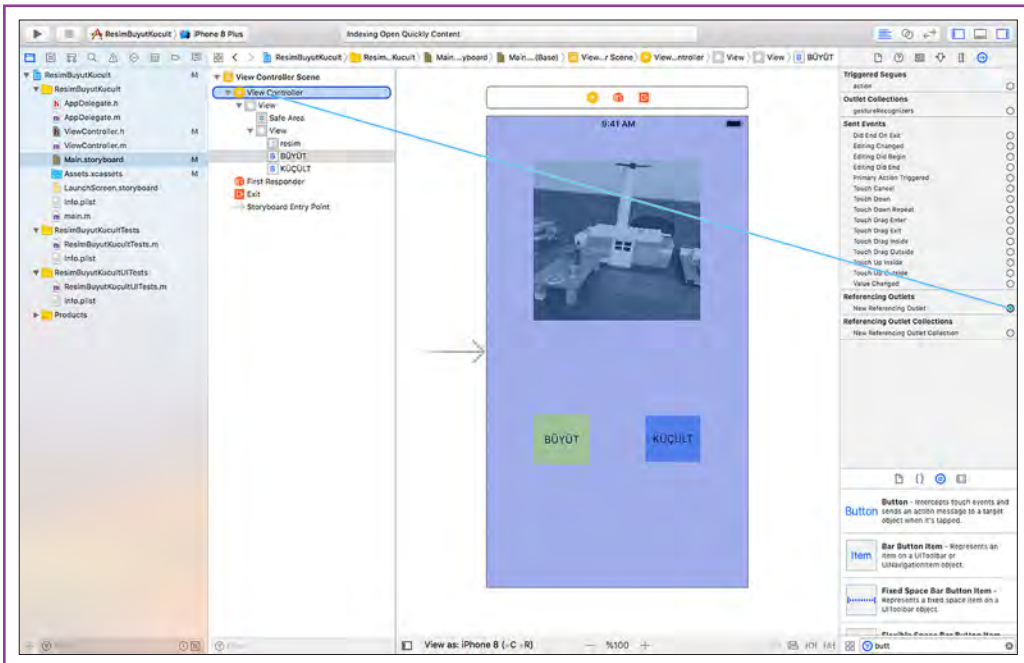
```
-(IBAction)btn_buyutPushed:(id)sender;
-(IBAction)btn_kucultPushed:(id)sender;
```



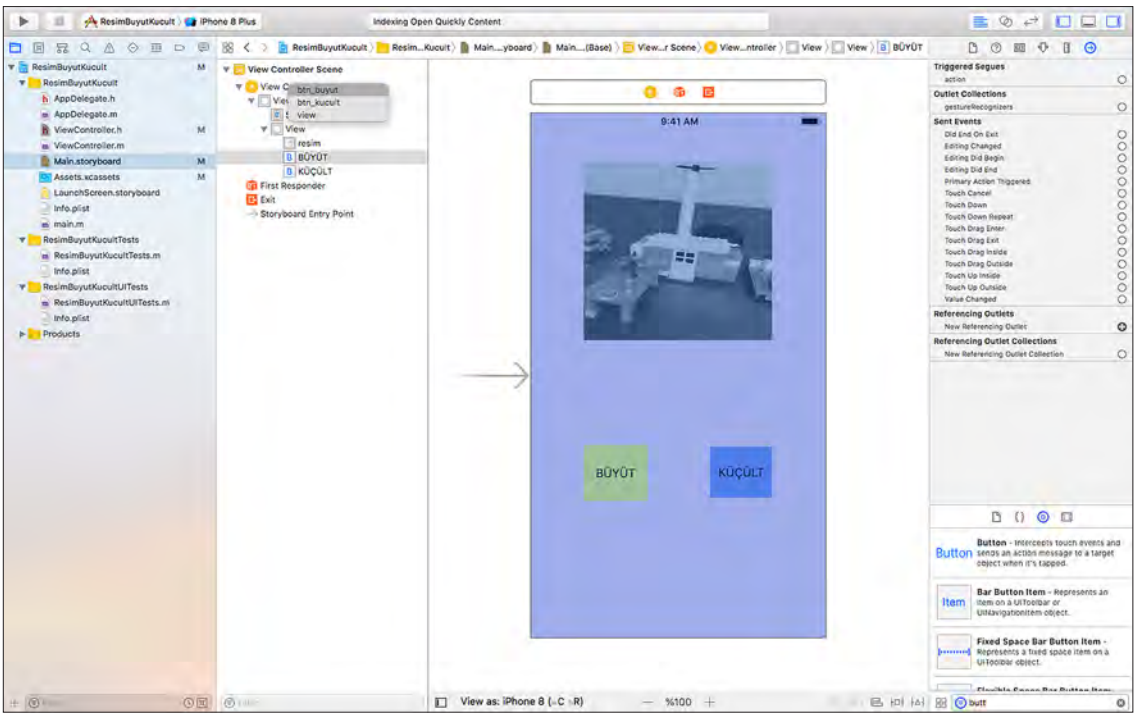
Ekran Görüntüsü 4.66

Programsal olarak yapılan tanımlamalar Storyboard ekranındaki nesneler ile fiziksel olarak ilişkilendirilir. Nesneler sırasıyla tıklanarak bir önceki uygulamada anlatıldığı gibi sağ bölmeden references sekmesi açılır, New Referencing Outlet'ten sol bölmedeki View Controller simgesine farenin sol oku sürüklenerek bırakılır. Ekrana gelen isim listesinden uygun olanı seçilir.

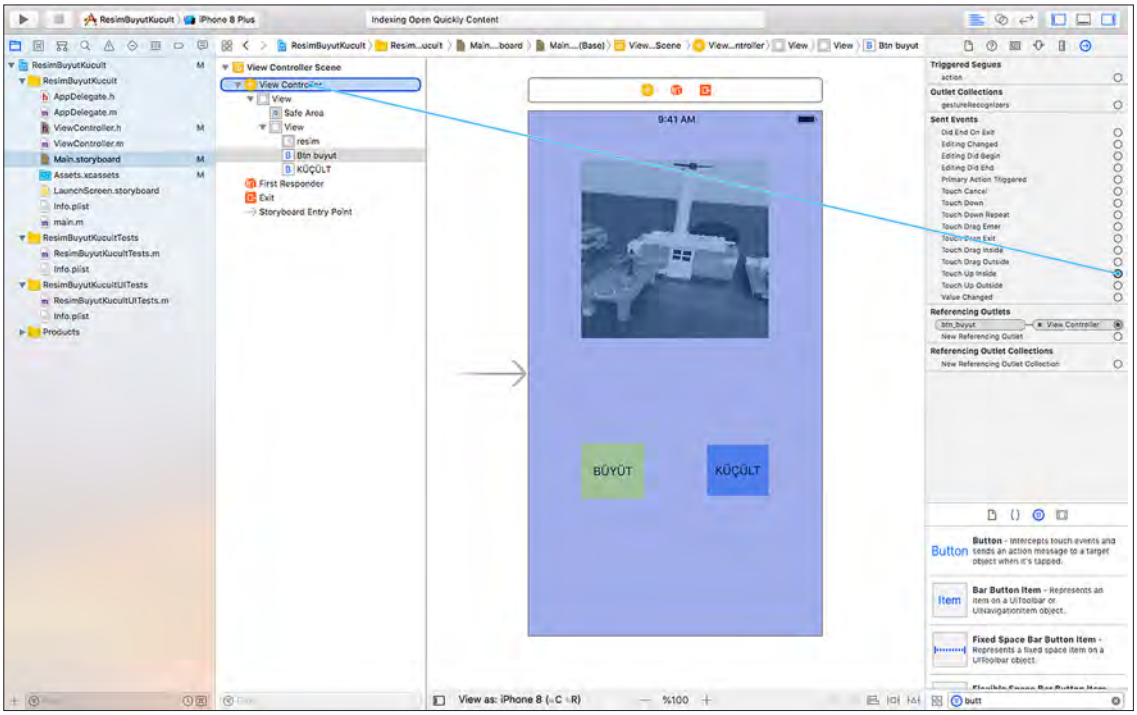
Düğmeler tıklandığında çalışacak fonksiyonun ismi ile de ilişkilendirme gerekmektedir. Bunun için de düğmeler yine sırasıyla tıklanarak ilişkiler sekmesinde Touch Up Inside'dan ViewController simgesine doğru ilişkilendirme yapılır ve ekrana gelen listeden doğru fonksiyon ismi seçilir.



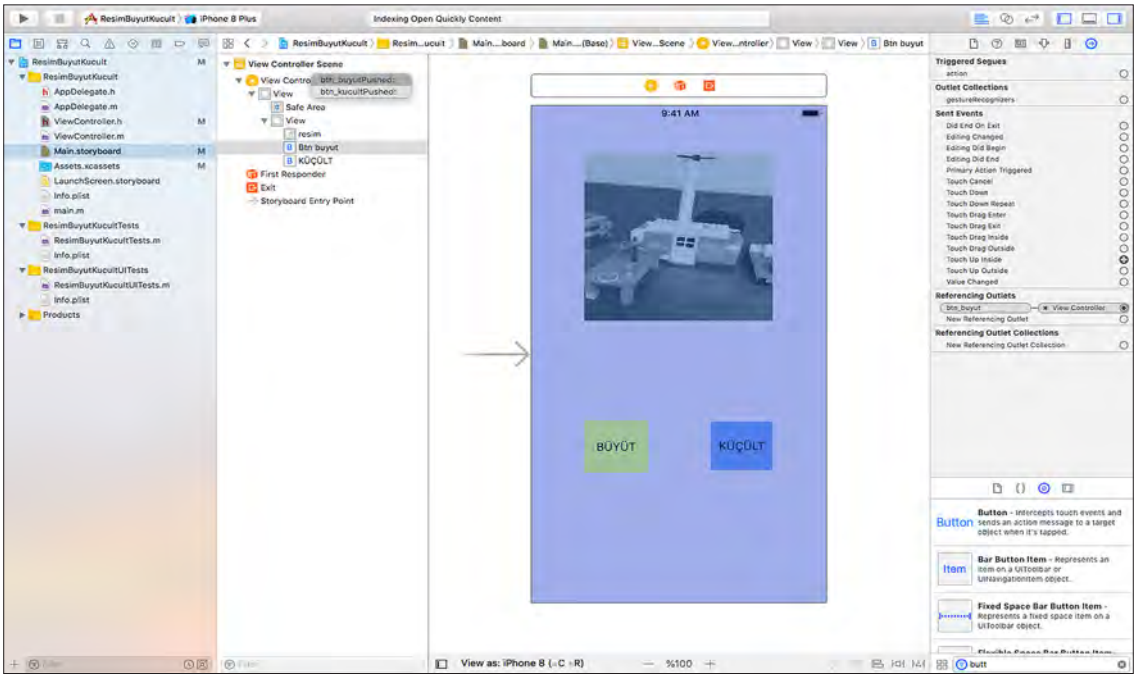
Ekran Görüntüsü 4.67



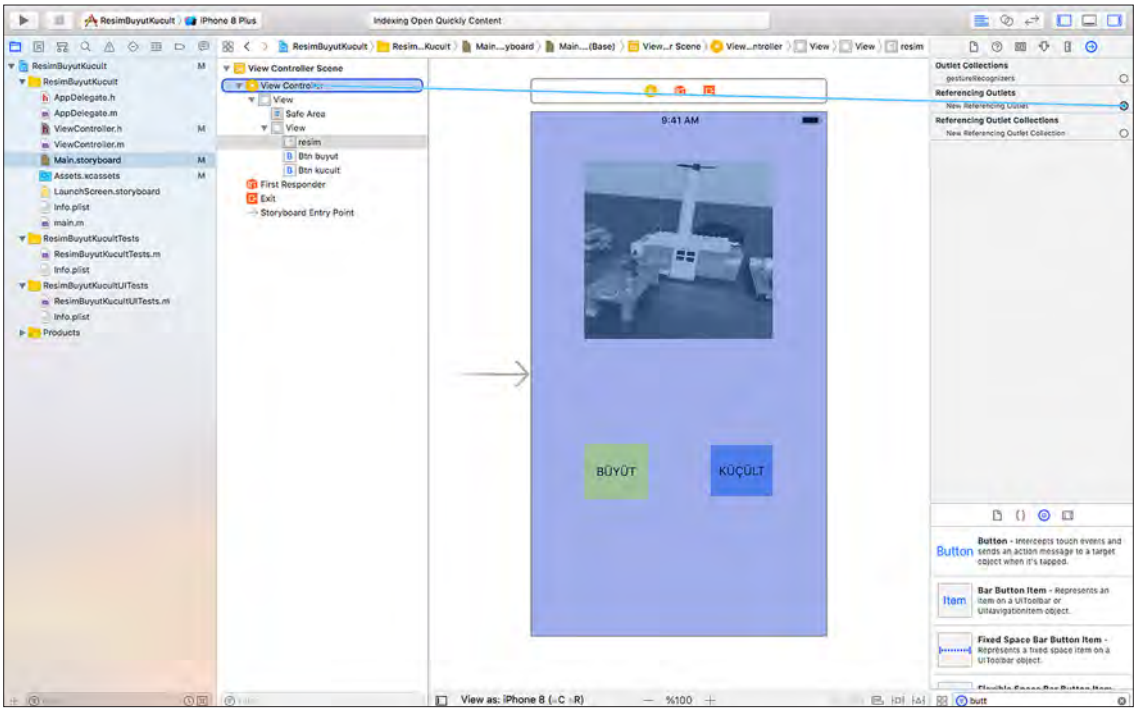
Ekran Görüntüsü 4.68



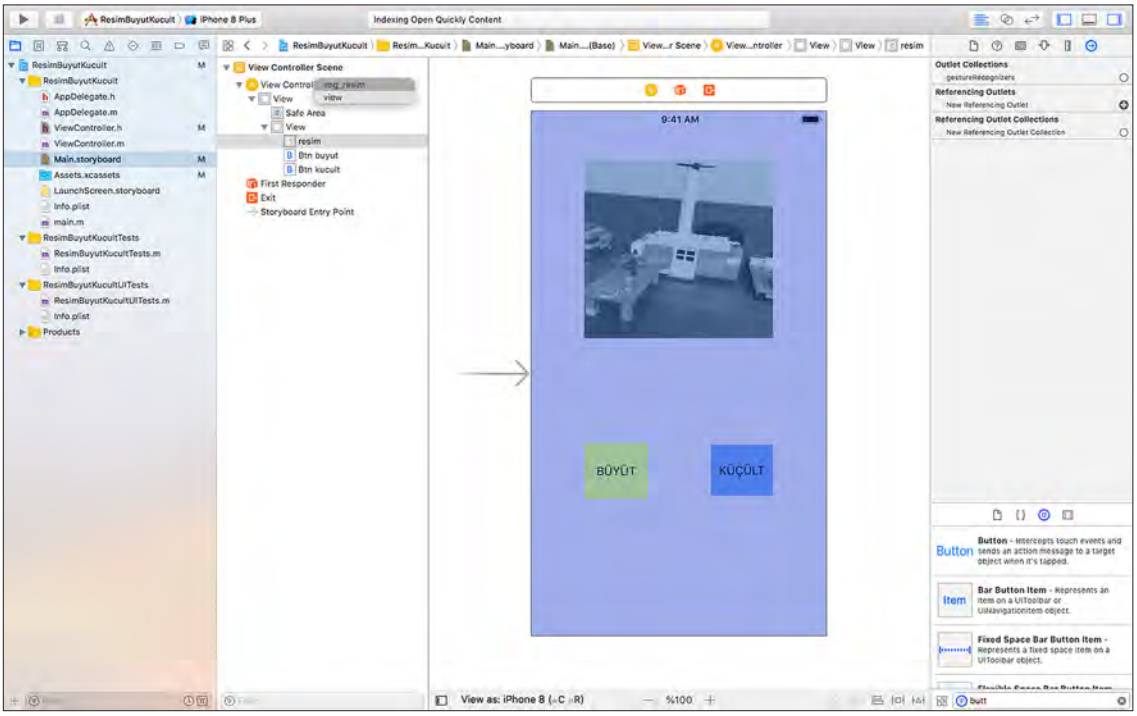
Ekran Görüntüsü 4.69



Ekran Görüntüsü 4.70



Ekran Görüntüsü 4.71



Ekran Görüntüsü 4.72

İlişkilendirmeleri tamamlanan nesneleri programlamak için ViewController.m dosyası açılır. Yazılacak kodda, BÜYÜT düğmesine tıklandığında ImageView'ın yükseklik ve genişliğinin 5'er pixel artması sağlanacaktır. Benzer şekilde KÜÇÜLT düğmesine tıklandığında da ImageView'ın yükseklik ve genişliğinin 5'er pixel azalması sağlanacaktır.

BÜYÜT bdüğmesine tıklandığında çalışacak kodlar:

```
-(IBAction)btn_buyutPushed:(id)sender{
self.img_resim.frame = CGRectMake(self.img_resim.frame.origin.x, self.img_resim.frame.origin.y,  self.img_resim.frame.size.width + 5, self.img_resim.frame.size.width + 5);
}
```

Burada kullanılan CGRectMake fonksiyonu nesnenin x, y koordinatındaki yerini ve "width" ile "height" ise şeklin boyutlarının belirlenmesini sağlamaktadır. Kodun biçimlenmemiş hali aşağıdaki gibidir ve değerler programcı tarafından verilir.

```
CGRectMake(CGFloat x, CGFloat y, CGFloat width, CGFloat height)
```

Ekran Görüntüsü 4.73

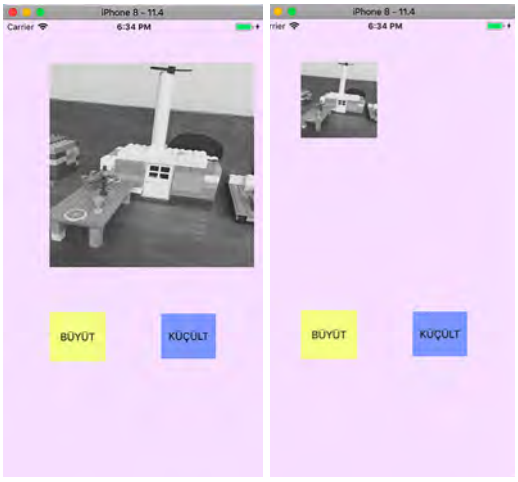
self.img_resim.frame.origin.x: resmin mevcut x konumunu,
self.img_resim.frame.origin.y: resmin mevcut y konumunu,
self.img_resim.frame.size.width: resmin mevcut genişliğini ve
self.img_resim.frame.size.height: resmin mevcut yüksekliğini ifade etmektedir.

KÜÇÜLT butonuna tıklandığında ise çalışacak kodlar:

```
-(IBAction)btn_kucultPushed:(id)sender{
self.img_resim.frame = CGRectMake(self.img_resim.frame.origin.x, self.img_resim.frame.origin.y, self.img_resim.
frame.size.width - 5, self.img_resim.frame.size.width - 5);
}
```

Ekran Görüntüsü 4.74

Uygulama simülatörde çalıştırıldığında BÜYÜT ve KÜÇÜLT düğmesine tıklandıktan sonraki görüntü yandaki gibi olacaktır. İsteğe bağlı olarak kod içerisinden, resmin her tıklamada 5'ten farklı değerlerde büyümesi ve/veya küçülmesi için farklı değerler denenerek değişim gözlenebilir.



Ekran Görüntüsü 4.75

4.2.7. Resim Değiştirme Uygulaması

Bu uygulamada ekranda bulunan bir ImageView içindeki resmi bir buton ile değiştirme işlemi yapılacaktır.

Diğer uygulamalarda olduğu gibi aşağıdaki proje başlangıç adımlarını sırasıyla yapınız:

1. Projenin açılarak isim verilmesi ve kaydedilmesi

2. Uygulamanın çalışacağı cihazın iPhone olarak seçilmesi

3. Main.Storyboard ekran tabanına bir View eklenmesi

4. Eklenen View'in üzerine bir ImageView ve bir Button eklenmesi

5. ImageView içine yerleştirilecek iki adet resmin resim1 ve resim2 isimleri ile Assets dosyasına kaydedilmesi

6. Storyboard ekranında ImageView'in içerisine resim1'in yerleştirilmesi

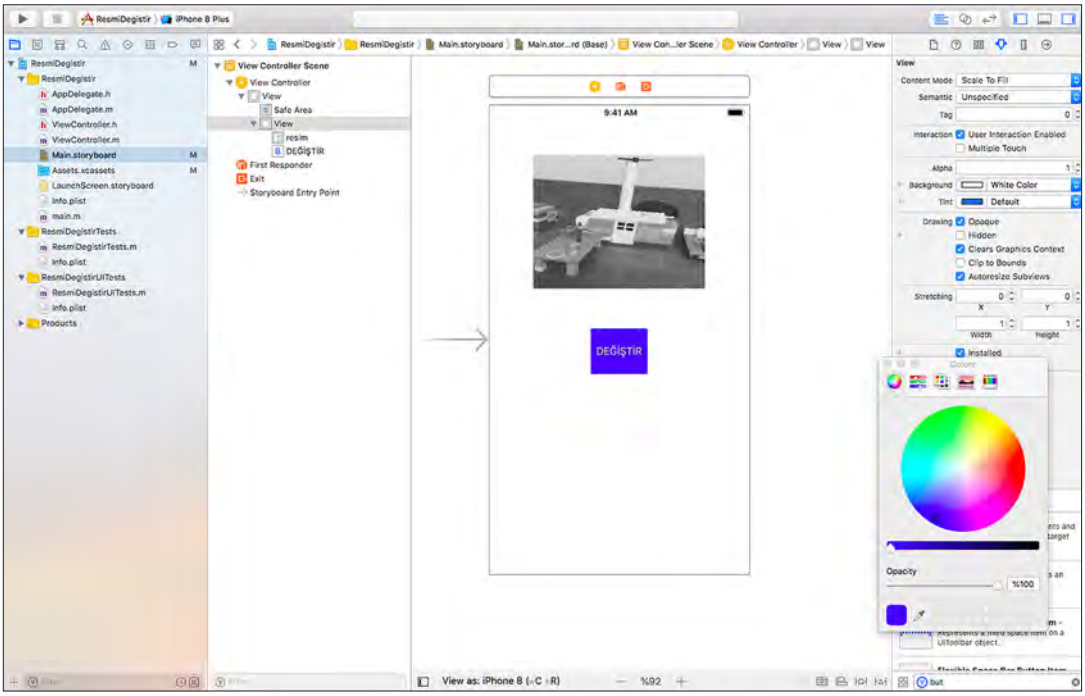
7. Düğme isminin "DEĞİŞTİR" yapılarak renk, boyut gibi özelliklerinin değiştirilmesi

8. Düğme ve ImageView'in ViewController.h dosyasında aşağıdaki kod dizilerinin kullanılarak tanımlanması

```
@property (nonatomic, weak) IBOutlet UIButton* btn_degistir;
@property (nonatomic, weak) IBOutlet UIImageView* img_resim;
-(IBAction)btn_degistirPushed:(id)sender;
```

9. ImageView ve DEĞİŞTİR düğmesinin Storyboard ekranında ilişkilendirmelerinin yapılması.

Adımlar tamamlandıktan sonra oluşacak ekran görüntüsü aşağıdaki gibi olacaktır.

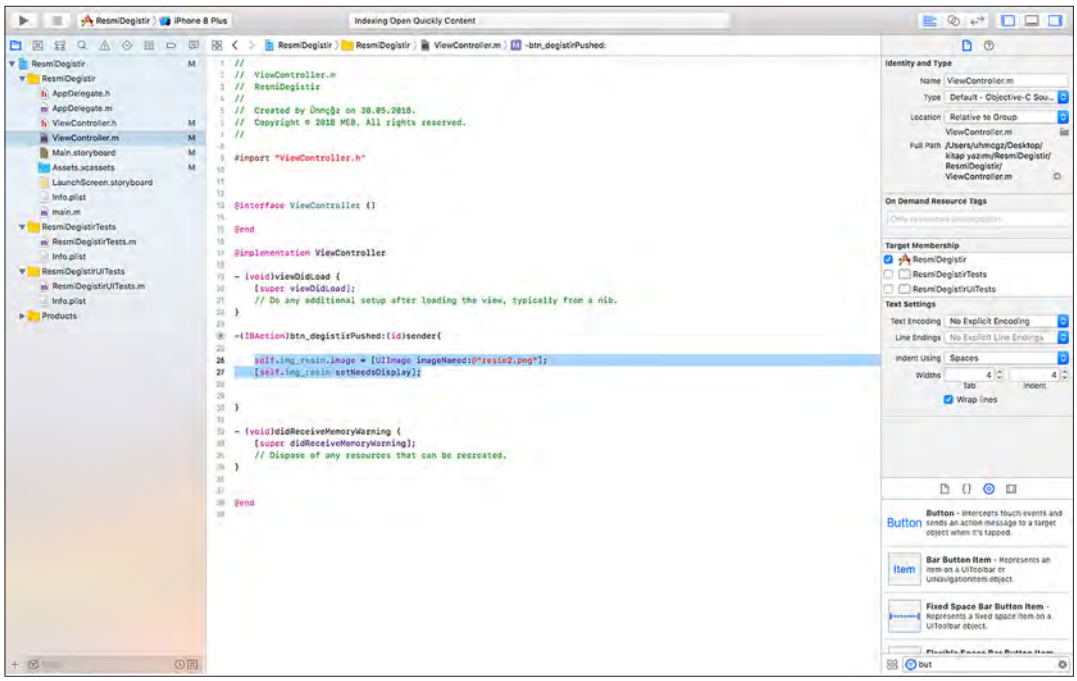


Ekran Görüntüsü 4.76

ViewController.m dosyası açılarak DEĞİŞTİR düğmesine tıklandığında çalışacak kodlar:

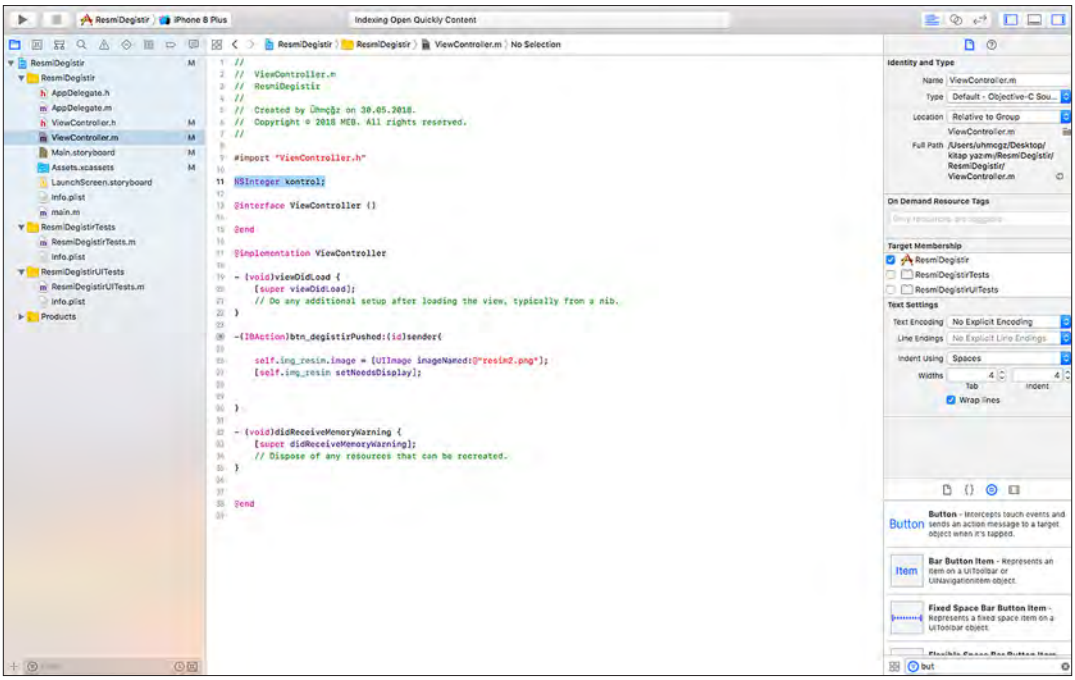
```
self.img_resim.image = [UIImageimageNamed:@"resim2.png"];
[self.img_resimsetNeedsDisplay];
```

şeklinde yazılacaktır. Fakat bu kodlar sadece ilk DEĞİŞTİR düğmesi tıklamasında çalışacak diğer tıklamalarda resim ilk haline dönemeyecektir.



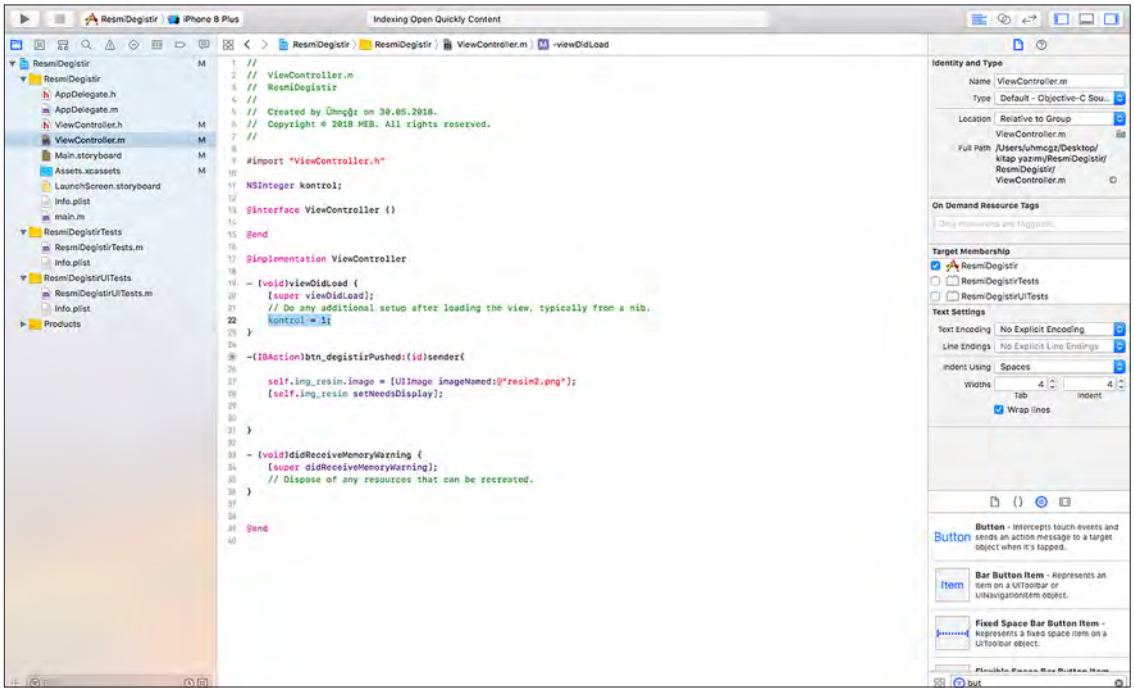
Ekran Görüntüsü 4.77

Hangi resmin ekranda görünür olduğunun kontrol edilerek her düğme tıklamasında diğer resmin gelmesi için bir değişken içerisinde hangi resmin görünür olduğu tutulacaktır. Bunun için öncelikle Integer tipte bir değişken tanımlanmalıdır.



Ekran Görüntüsü4.78

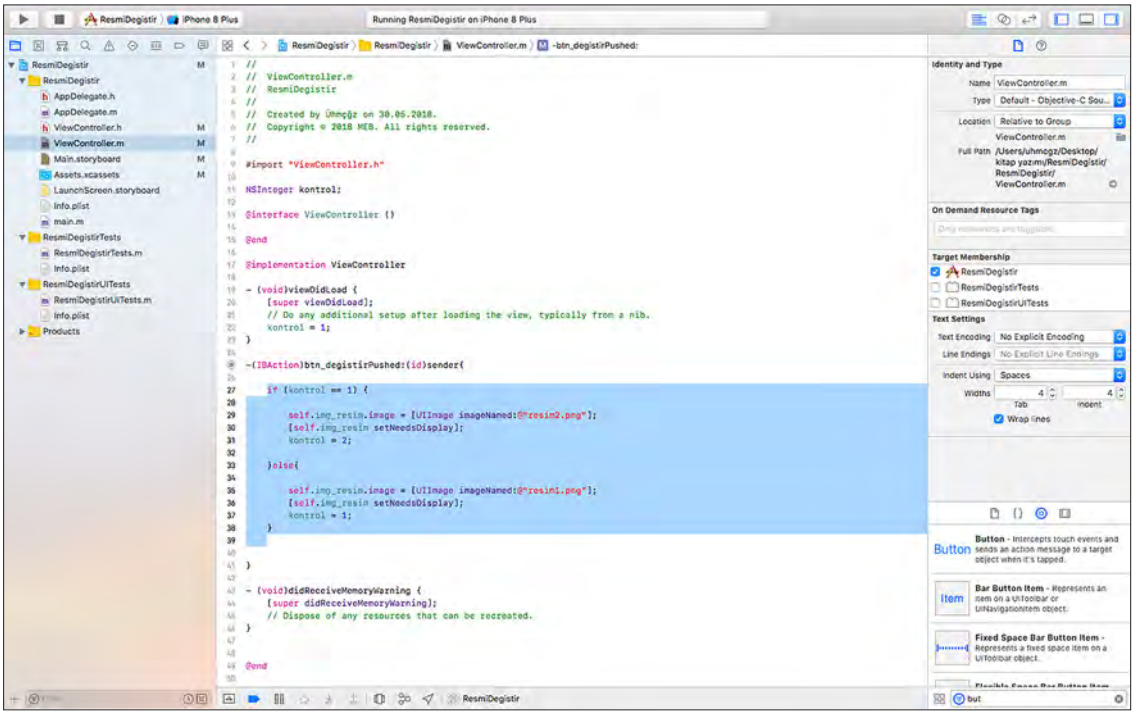
Tanımlanan değişkene uygulama ilk çalıştığında (ViewDidLoad olduğunda) "1" değeri atanır. "1" değeri ekran ilk yüklendiğinde resim1 dosyasının görüntülendiğini ifade etmektedir.



Ekran Görüntüsü 4.79

"kontrol" değişkeninin içindeki sayısal değere göre resim değişikliği yapılması için koşul ifadesi kullanılması gerekmektedir.

```
if (kontrol == 1) {    // eğer kontrol 1'e eşit ise yani resim1 ekranda görüntüleniyorsa
self.img_resim.image = [UIImageimageNamed:@"resim2.png"];        // resim2'yi göster
[self.img_resimsetNeedsDisplay];
kontrol = 2;              //kontrol'ü 2'ye eşitle

}else{                    // eğer değilse yani kontrol 2'ye eşitse

self.img_resim.image = [UIImageimageNamed:@"resim1.png"];        // resim1'i göster
[self.img_resimsetNeedsDisplay];
kontrol = 1;              //kontrol'ü 1'e eşitle
}
```



Ekran Görüntüsü 4.80

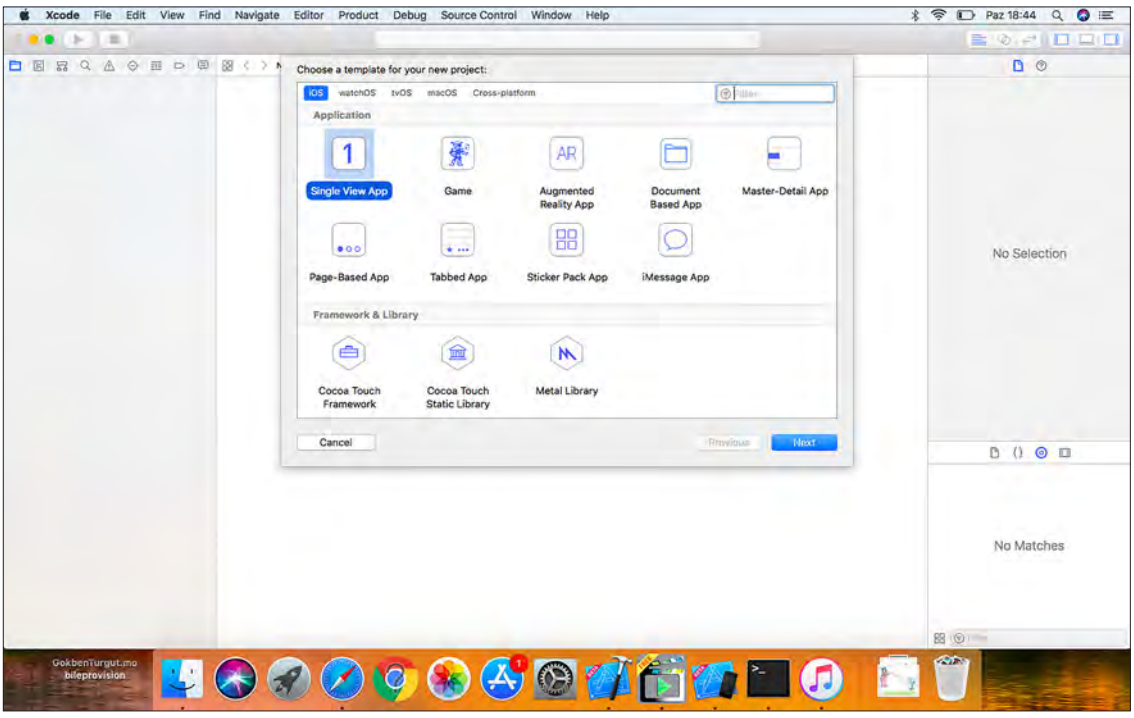
Program çalıştırılarak DEĞİŞTİR düğmesi her tıklandığında ekran görüntüsü yandaki gibi olacaktır.

Ekran Görüntüsü 4.81

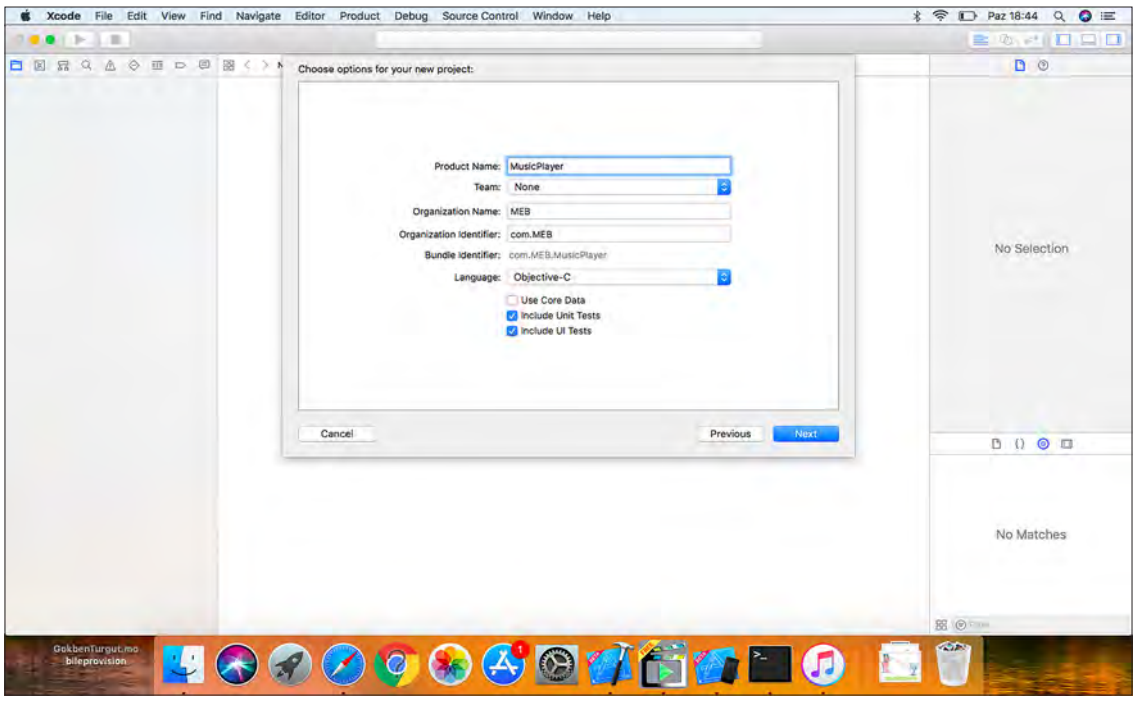
4.2.8. Müzik Çalar Uygulaması

Bu uygulamada harici bir kütüphane projeye eklenerek basit bir müzik oynatıcı yapılacaktır. Oynatıcı üzerinde oynat, durdur, ilerleme çubuğu, tam ekran yap, ses aç/kapa ve tam ekran modunda ileriye, geriye 15 sn sar düğmeleri bulunacaktır. Arayüz tasarımı tamamen kütüphane içindeki class'lardan hazır olarak Storyboard'a yüklenecektir.

Yeni bir Xcode projesi açılarak projeye "MusicPlayer" ismi verilir.

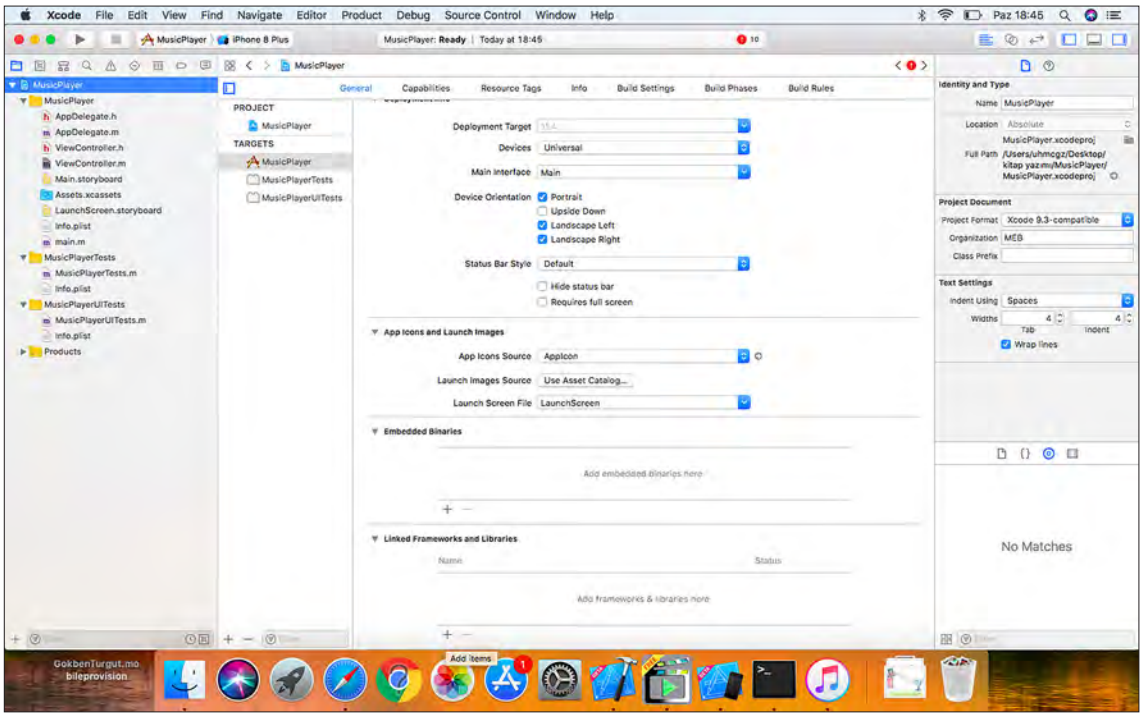


Ekran Görüntüsü 4.82



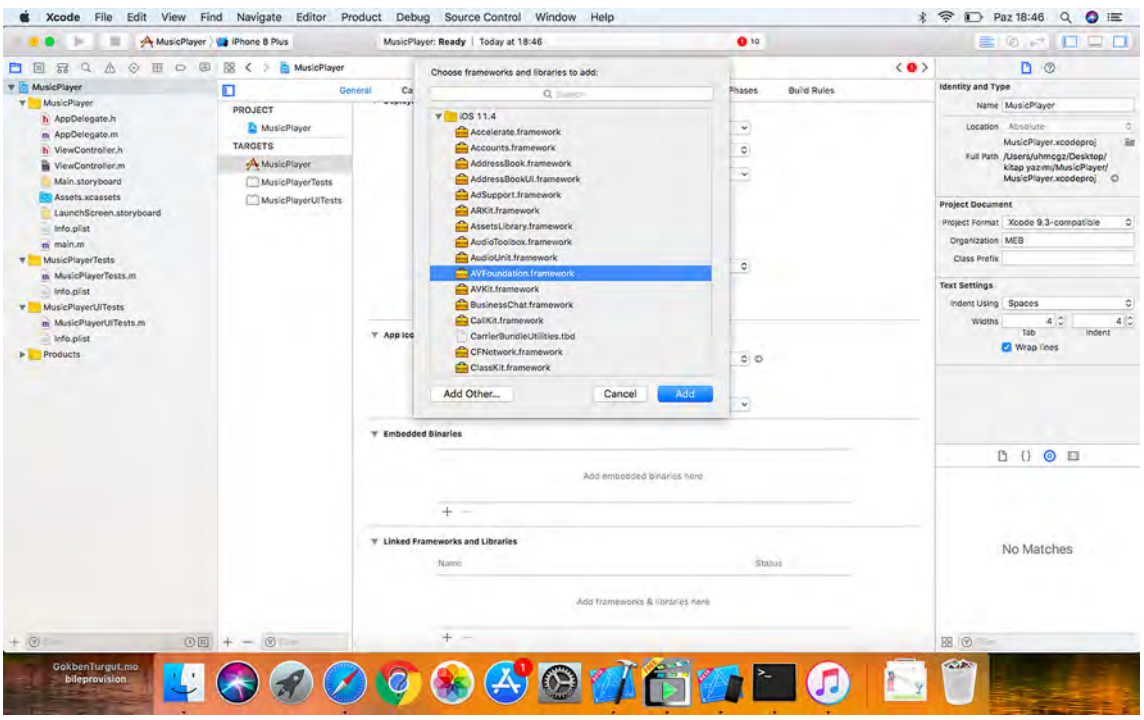
Ekran Görüntüsü 4.83

Proje genel ayarlar penceresinde en alt sırada bulunan "Linked Frameworks and Libraries" bölümünde + (artı) işaretine tıklanarak ses ve videolar üzerinde işlemler yapılmasını sağlayan kütüphaneyi indiriniz.

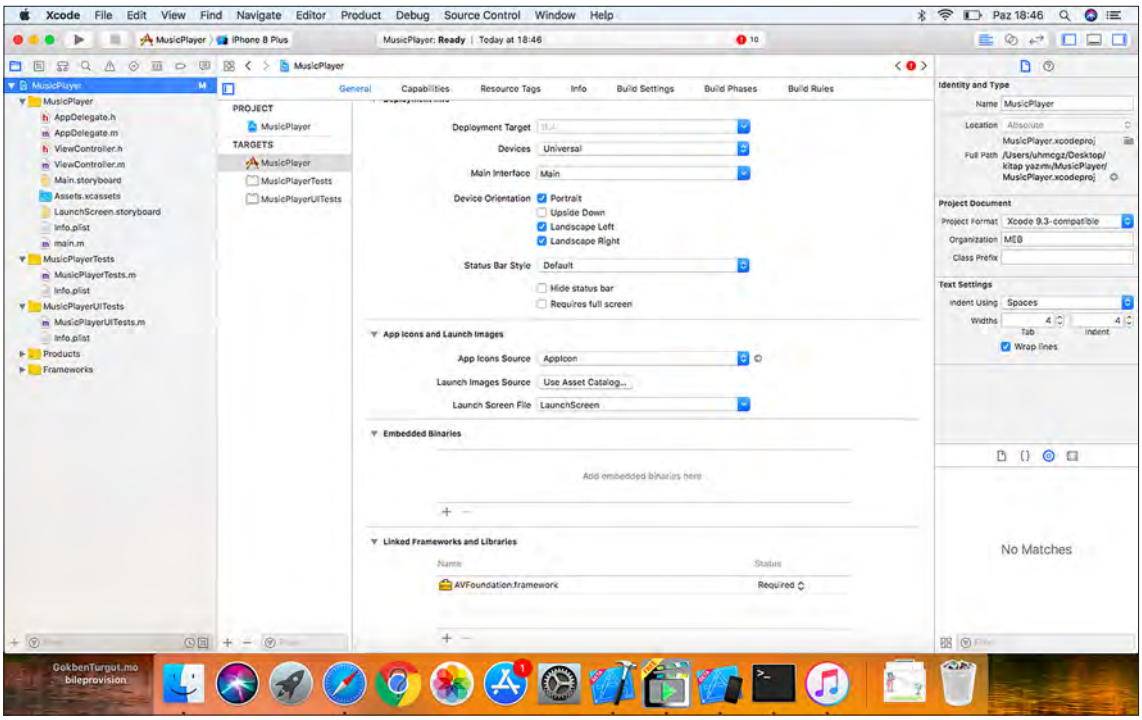


Ekran Görüntüsü 4.84

Ekranda beliren kütüphane listesinden AVFoundation seçilerek kütüphane projeye eklenir.

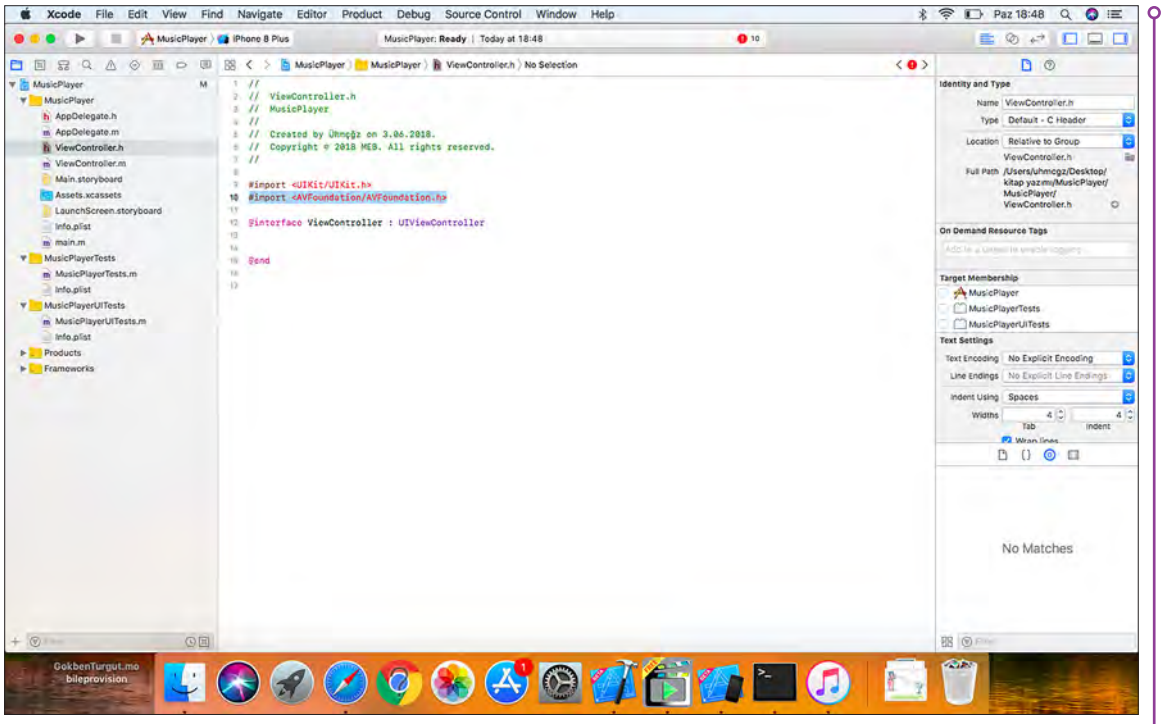


Ekran Görüntüsü 4.85



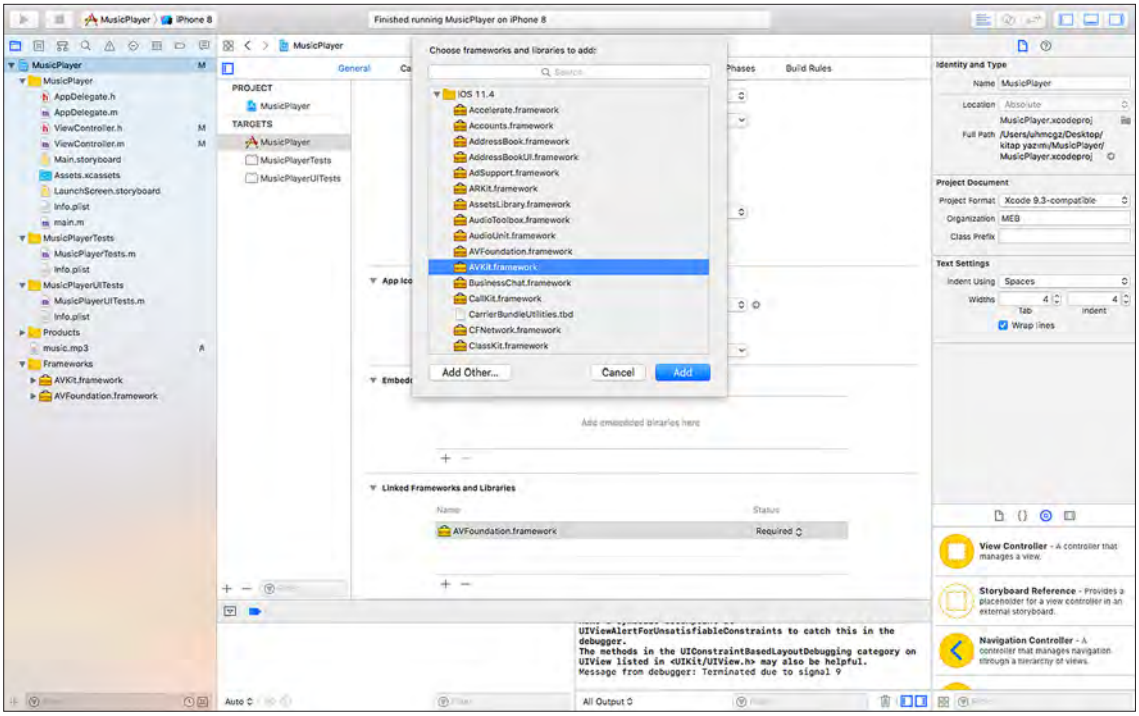
Ekran Görüntüsü 4.86

Kütüphanedeki ilgili class'ın projedeki ekranda yani ViewController'da kullanılabilmesi için AVFoundation.h dosyası ViewController.h dosyasına indirilir.



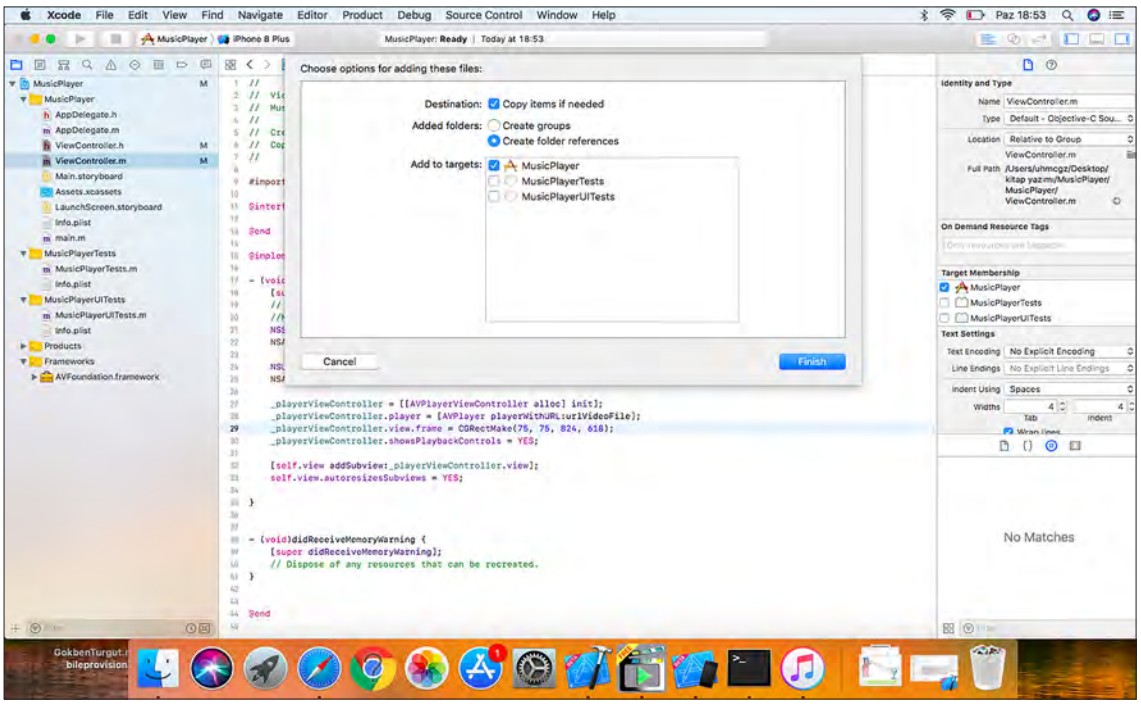
Ekran Görüntüsü 4.87

Benzer şekilde AVKit kütüphanesi de projeye aynı yöntemle eklenerek ViewController.h dosyasına indirilir.



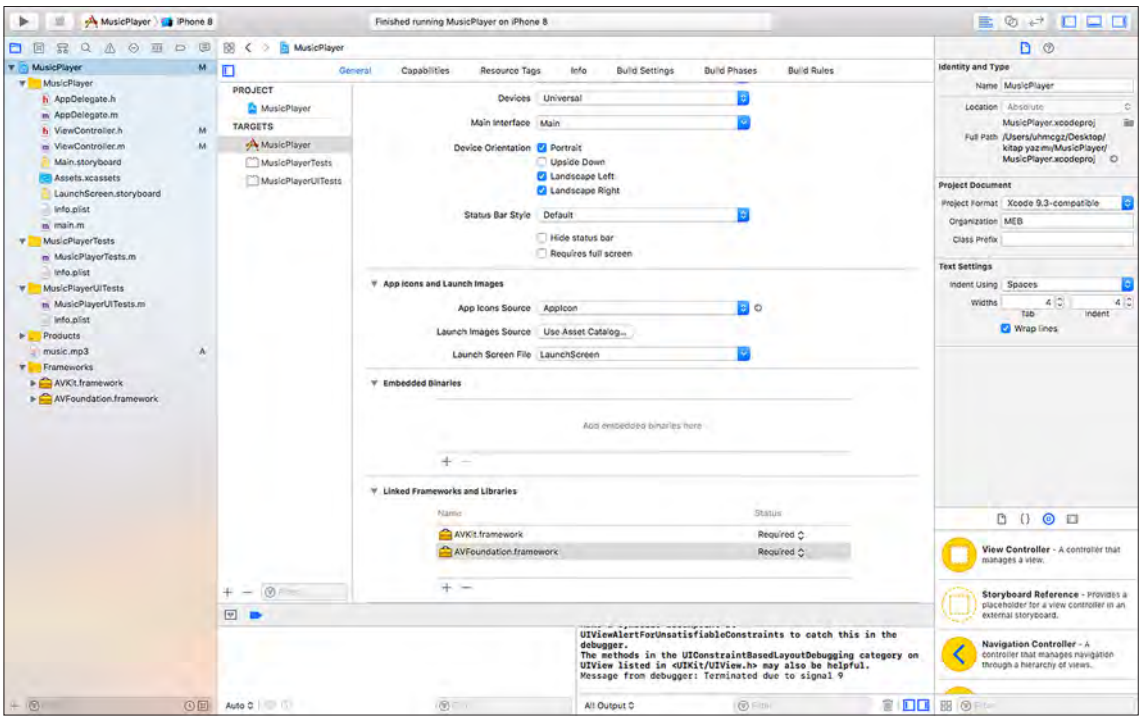
Ekran Görüntüsü 4.88

"music.mp3" isimli bir ses dosyası ise sürükle bırak yöntemi ile ekranın solundaki dosyalar bölümüne eklenir. Karşınıza gelen pencereden "Copy items if needed" alanı işaretlenir.

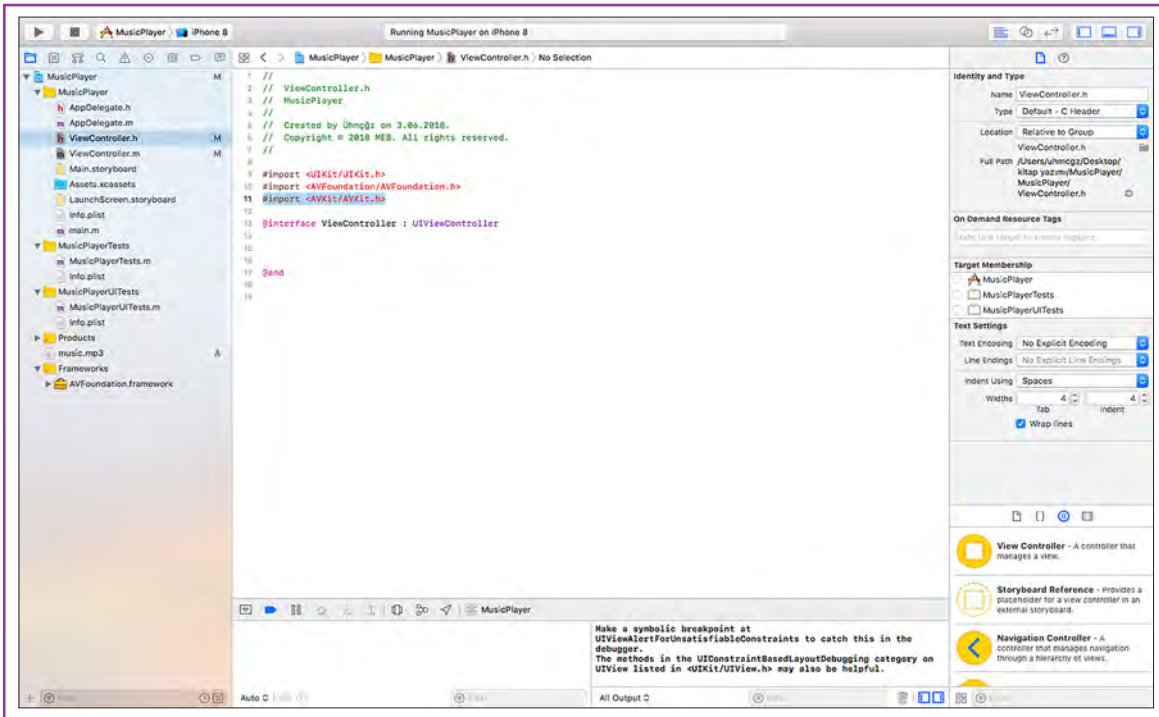


Ekran Görüntüsü 4.89

Aşağıdaki ekran görüntüsünde Products dizini ile Frameworks dizini arasında ses dosyası görülmektedir.

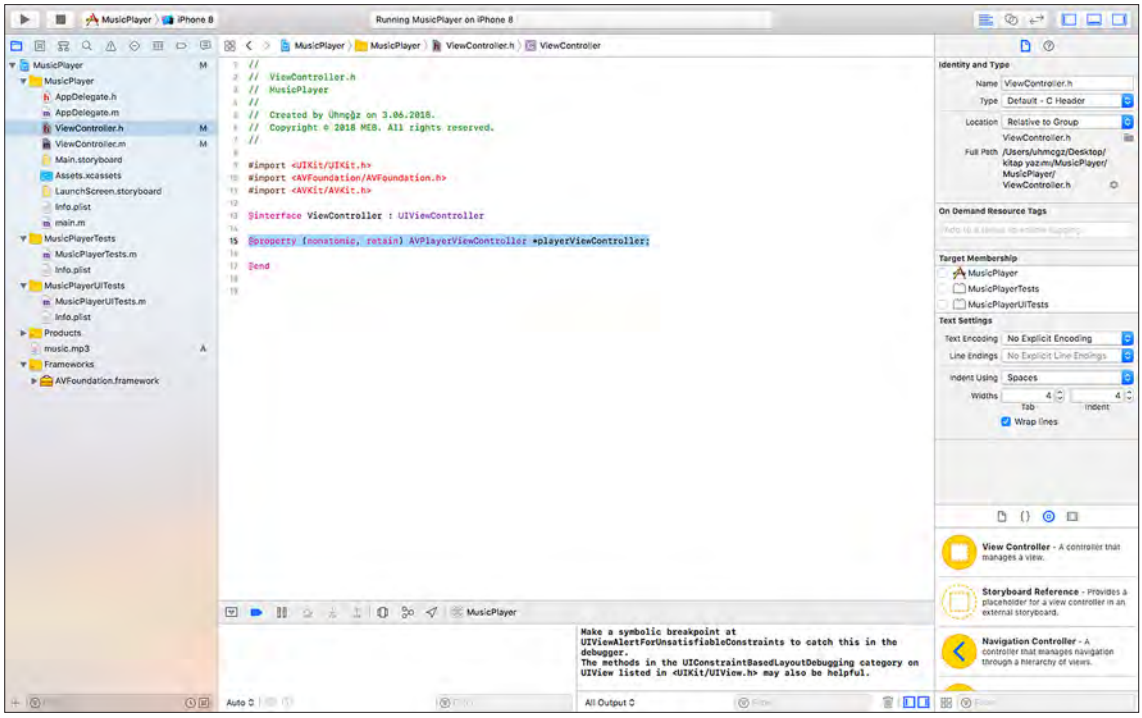


Ekran Görüntüsü 4.90



Ekran Görüntüsü 4.91

İndirilen kütüphanedeki AVPlayerViewController class'ı ViewController.h dosyasında kullanılarak playerViewController değişkeni oluşturulur.



Ekran Görüntüsü 4.92

Ardından ViewController.m dosyasında ekranda görüntülenecek müzik oynatıcısının oluşturulması ve ayarları için programlama yapılır.

Sırasıyla aşağıdaki kodlar yazılır:

```
NSString *stringAudioPath = [[NSBundle mainBundle] pathForResource:@"music" ofType:@"mp3"];
NSAssert(stringAudioPath, @"Expected not nil video file");
```

Yukarıda bir string tipi değişken oluşturularak içine ses dosyasının yolu eklenir. Dosya projenin ana dizininde olduğu için mainBundle ifadesi kullanılır. Dosyanın adı pathForResource içinde, dosyanın tipi ise ofType içinde string tipinde (@"...." ifadesinden ileride bahsedilecektir) yazılır.

```
NSURL *urlAudioFile = [NSURL fileURLWithPath:stringAudioPath];
NSAssert(urlAudioFile, @"Expected not nil video url");
```

URL tipinde bir değişken yaratılarak yol değişkeni URL'e çevrilir.

```
_playerViewController = [[AVPlayerViewController alloc] init];
```

Ekranda müzik oynatıcısı oluşturulur.

```
_playerViewController.player = [AVPlayer playerWithURL:urlAudioFile];
```

Müzik oynatıcısında çalması için parçanın URL'i belirtilir.

```
_playerViewController.view.frame = CGRectMake(20, 120, 350, 460);
```

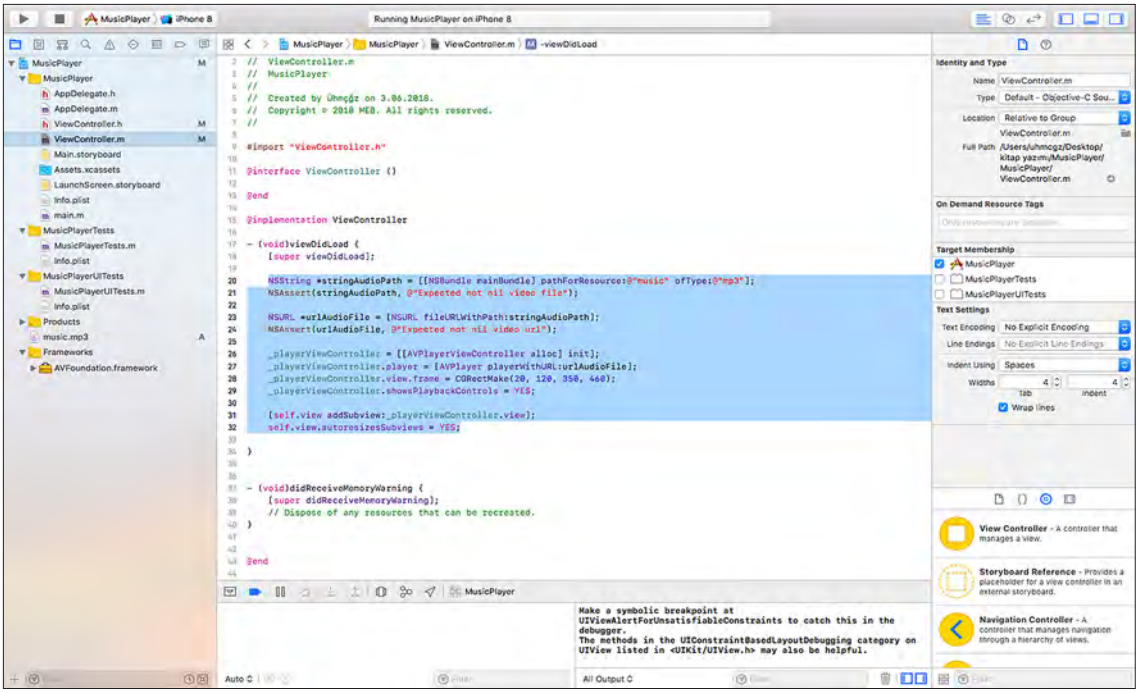
Ekrandaki yeri (x, y olarak) ve boyutları (width, height olarak) pixel cinsinden belirtilir.

```
_playerViewController.showsPlaybackControls = YES;
```

Kontrol düğmeleri aktif hale getirilir.

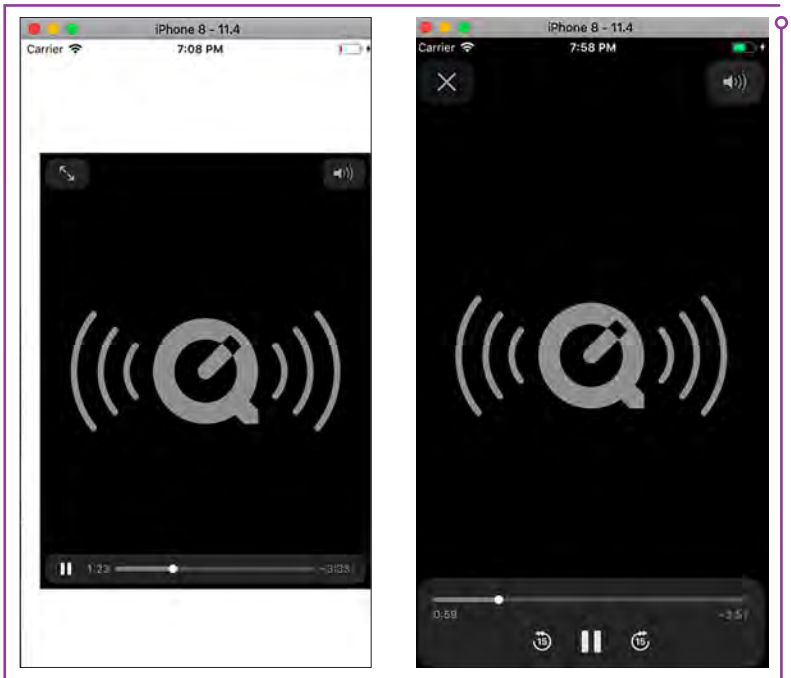
```
[self.view addSubview:_playerViewController.view];
self.view.autoresizesSubviews = YES;
```

addSubview ile müzik oynatıcı View ekranına eklenir.



Ekran Görüntüsü 4.93

Uygulamanın simülatördeki görüntüleri normal ve tam ekran olarak aşağıdaki gibi olacaktır:



Ekran Görüntüsü 4.94

4.2.9. Video Ekleme

Video ekleme işleminin gösterileceği bu uygulamada müzik ekleme uygulamasında yapılan işlemlerin çoğunlukla benzeri işlemler yapılacaktır. AVFoundation kütüphanesi projeye indirilerek içinde bulunan hazır class'lar ile ViewController dosyasına video oynatıcısı ekleme işlemi yapılacaktır.

Öncelikle aşağıdaki başlangıç adımları sırası ile yapılır:

1. Projenin yaratılması

2. Projeye isim verilmesi

3. Projenin bilgisayara kaydedilmesi

4. Genel ayarlar penceresinden "Linked Frameworks and Libraries" bölümünden AVFoundation ve AVKit kütüphanelerinin projeye eklenmesi

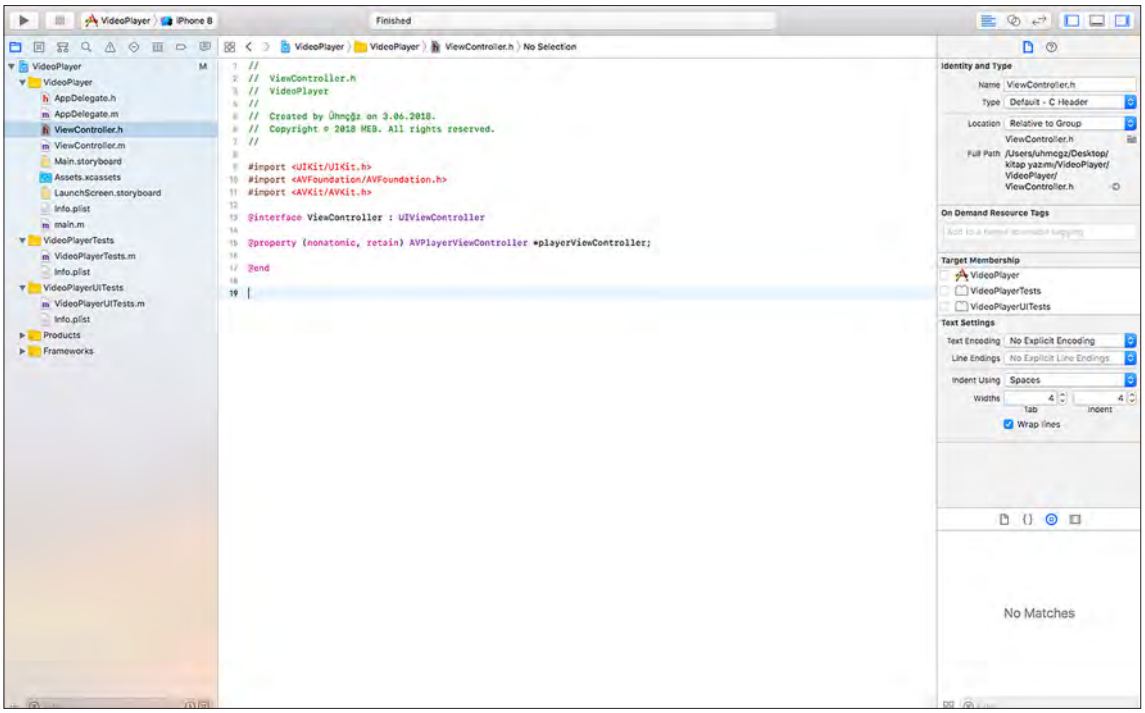
5. ViewController.h dosyasına AVFoundation.h ve AVKit.h dosyalarının indirilmesi

```
#import <AVFoundation/AVFoundation.h>
#import <AVKit/AVKit.h>
```

6. ViewController.h dosyasında AVPlayerViewController class'ından playerViewController isimli bir değişken oluşturulması

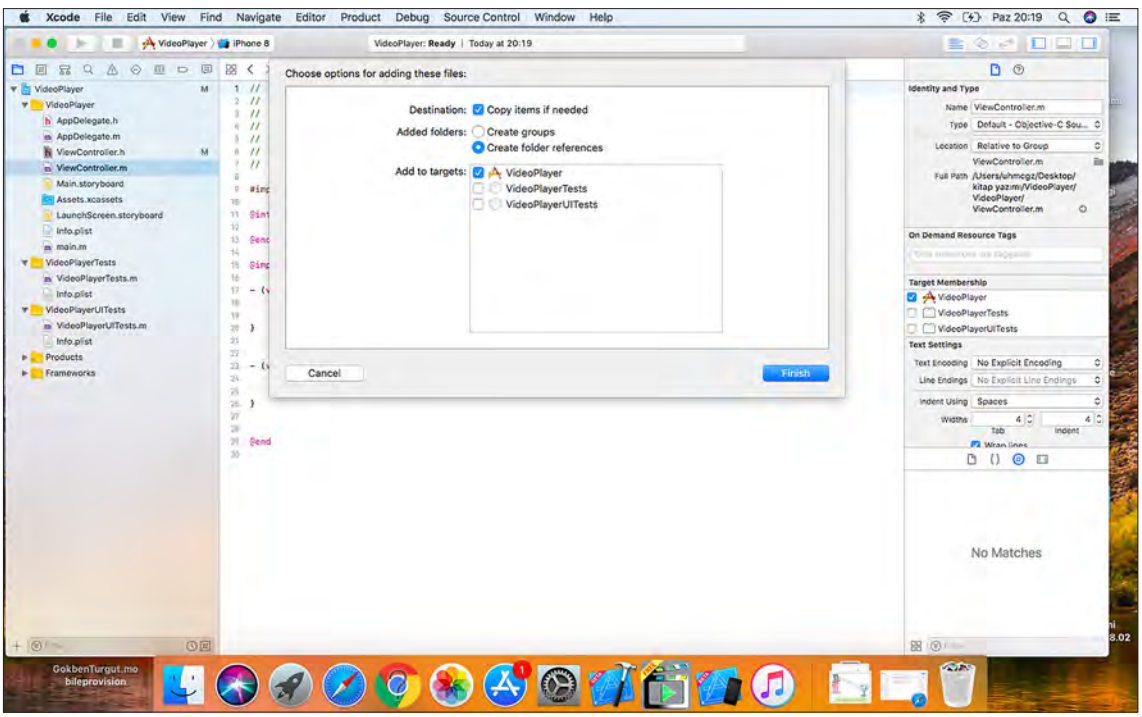
```
@property (nonatomic, retain) AVPlayerViewController *playerViewController;
```

Yukarıdaki adımlar tamamlandığında ViewController.h dosyası aşağıdaki gibi olacaktır:



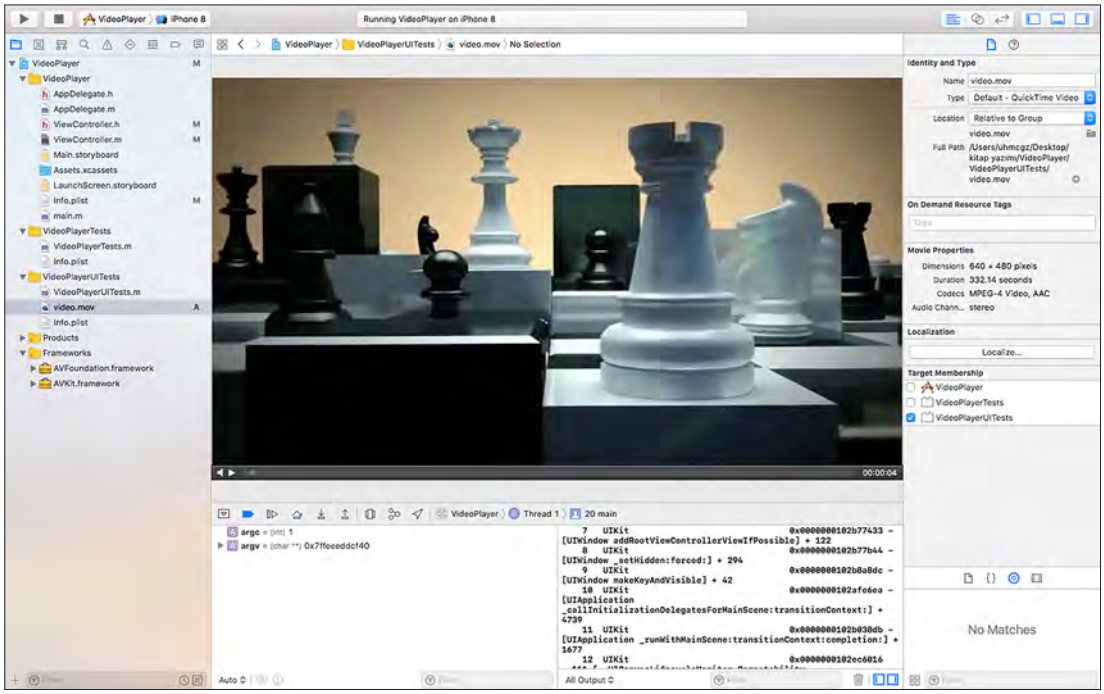
Ekran Görüntüsü 4.95

Projeye bir video dosyasını eklemek için sürükle bırak yöntemi kullanılır. Karşınıza gelen pencereden "Copy items if needed" seçeneği işaretlenir.



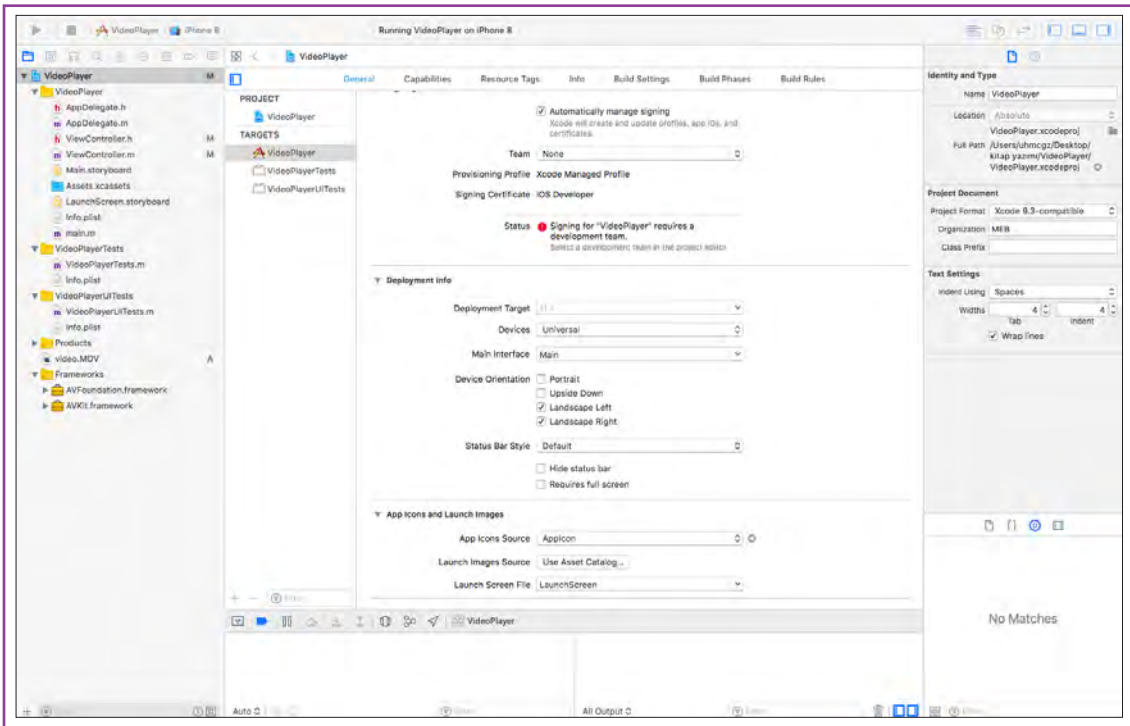
Ekran Görüntüsü 4.96

Video dosyasına tıklandığında projedeki gösterimi aşağıdaki gibi olacaktır.



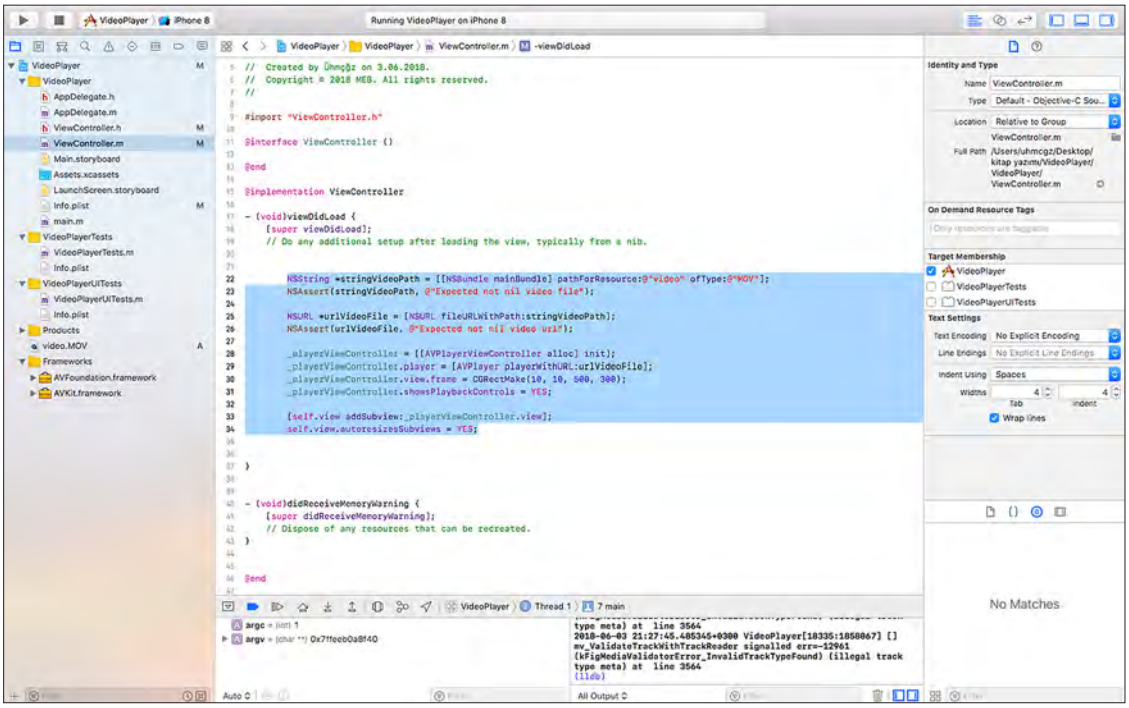
Ekran Görüntüsü 4.97

Bu uygulamada video dosyasının iPhone ekranında sadece yatay görünmesi sağlanacaktır. Bunun için proje genel ayarlar ekranında Deployment Info alanında sadece Landscape Left ve Landscape Right seçeneklerini işaretli bırakınız.



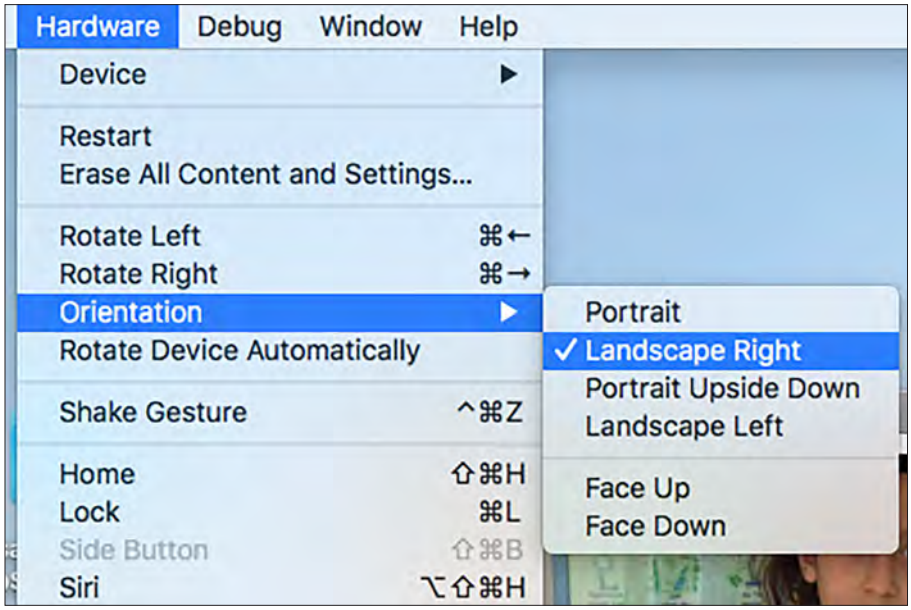
Ekran Görüntüsü 4.98

ViewController.m dosyası açılarak ses ekleme uygulamasında yazılan kodların aynısını yazarak sadece video dosyasının ismi ve uzantısı, videonun ekranda gösterileceği yer (x, y biçiminde) ve boyutları (width, height biçiminde) pixel cinsinden değiştirilir.



Ekran Görüntüsü 4.99

Uygulama çalıştırılır ve simulatör penceresi açıldığında Hardware menüsünden Orientation- Landscape Right seçeneği seçilir.



Ekran Görüntüsü 4.100

iPhone simülatör görüntüsü aşağıdaki gibi olacaktır. Videonun ekran üzerindeki yeri ve boyutları ile ilgili değişiklikler ViewController.m dosyasında;

```
_playerViewController.view.frame = CGRectMake(10, 10, 500, 300);
```

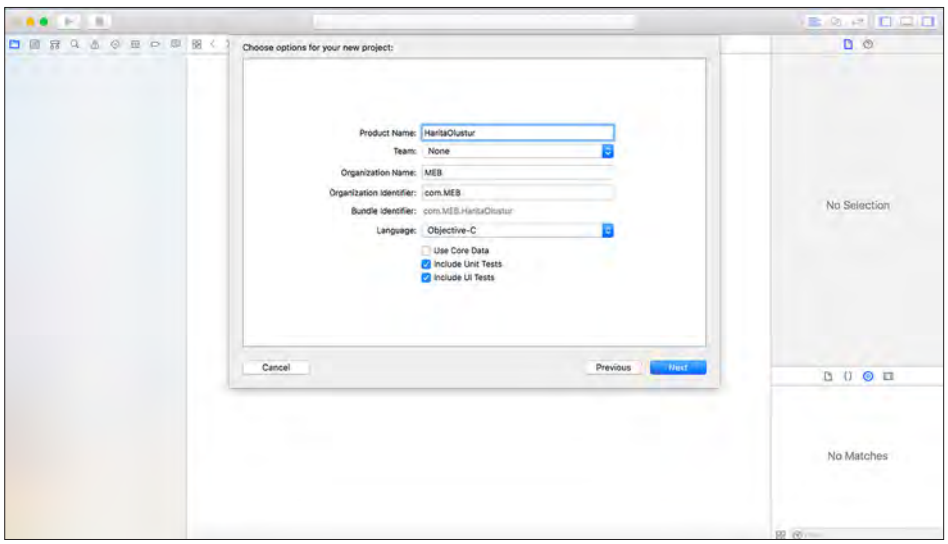
kod satırı üzerinden yapılabilir.

Ekran Görüntüsü 4.101

4.2.10. Harita Uygulaması

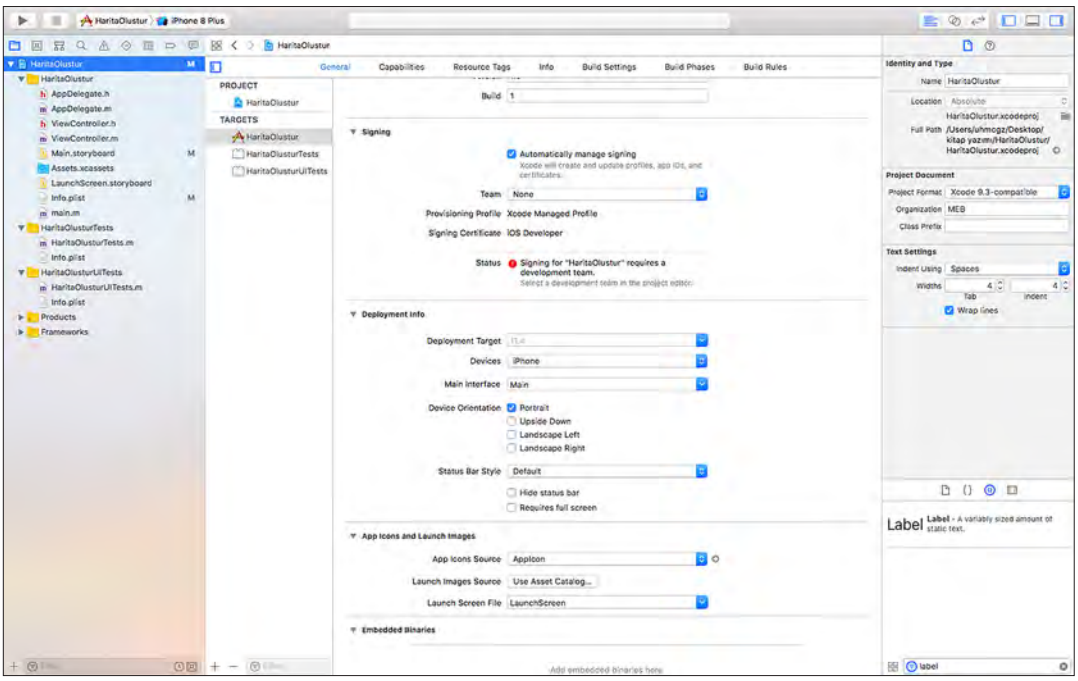
a) Harita Oluşturma

Bu uygulamada kullanıcı konumunu gösteren bir harita ekranda görüntülenecektir. Harita penceresi kodla veya main.storyboard'dan doğrudan bir nesne olarak da eklenebilmektedir. Harita servislerinin kullanımı için bu bölümde geliştirilecek uygulamaların bir kısmı kodla, diğer kısmı da storyboard'dan manuel oluşturulacaktır. "HaritaOlustur" isimli bu uygulamada harita penceresi storyboard'da oluşturulacaktır.



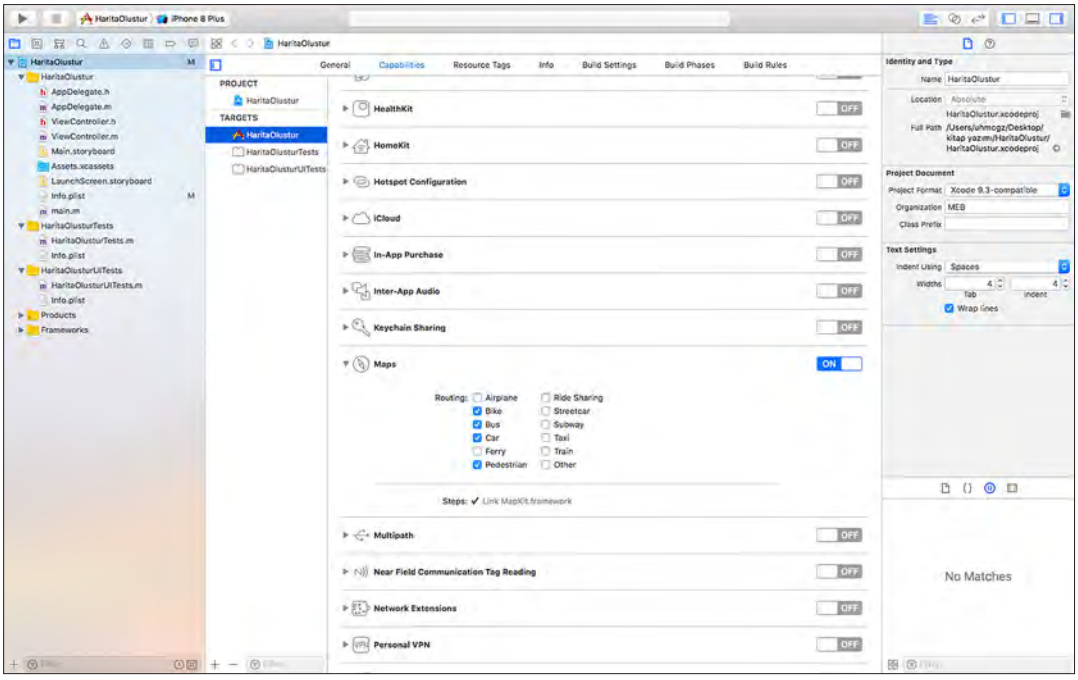
Ekran Görüntüsü 4.102

General ayarlar sekmesinde cihaz yönlendirmesi (device orientation) portre (portrait) olarak seçilir ve cihaz (device) olarak iPhone seçilir.



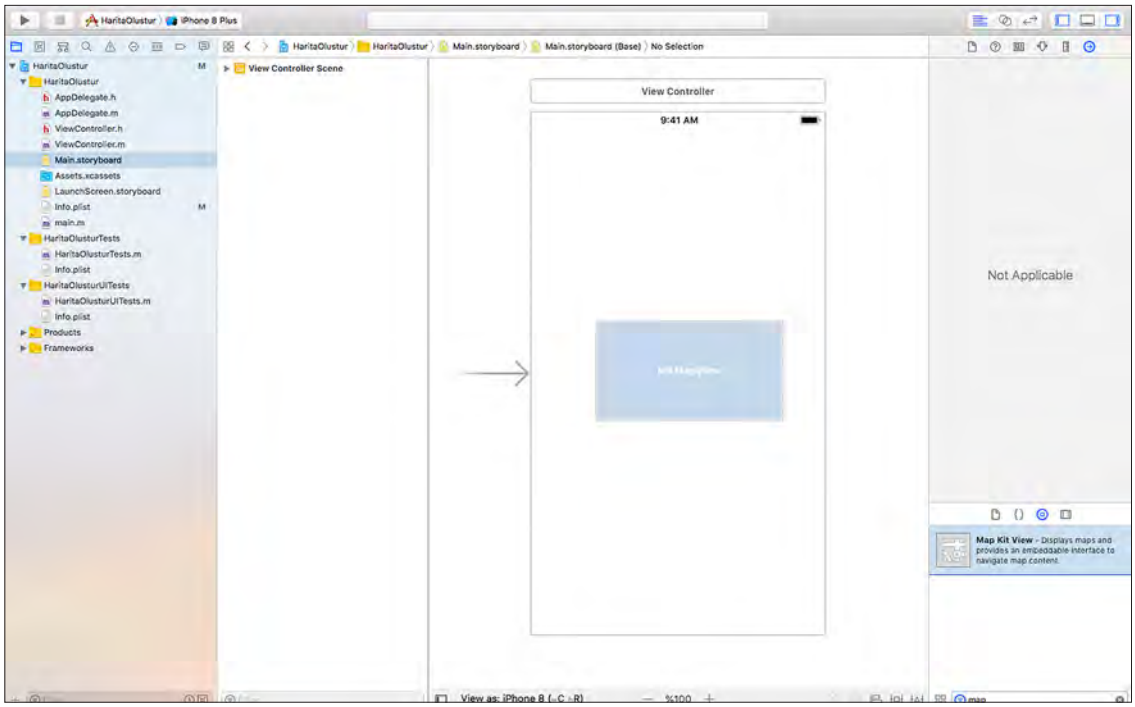
Ekran Görüntüsü 4.103

Capabilities sekmesi açılarak Maps özelliği ON haline getirilirek harita servisleri açılır. Harita servisinin yönlendirmesinin hangi yollarla olacağı seçilir. Bu uygulama için sırasıyla bisiklet, otobüs, araba ve yaya yolları seçilmiştir



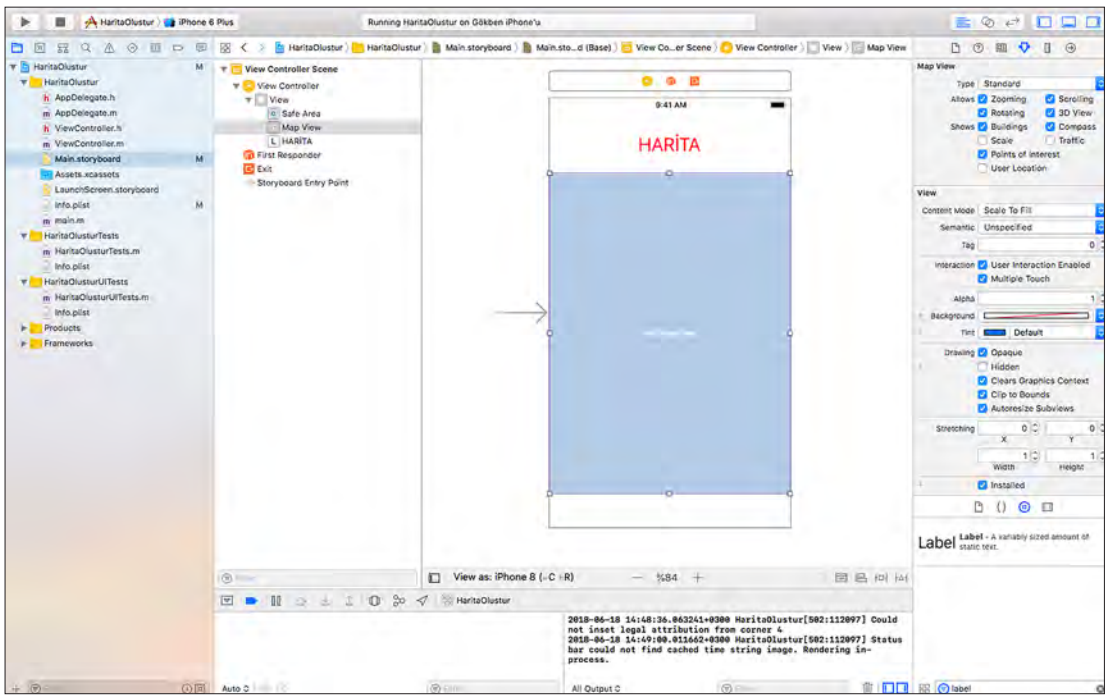
Ekran Görüntüsü 4.104

Main.storyboard açılarak sağ bölmeden Map Kit View nesnesi sürüklenerek View Controller üzerine bırakılır.



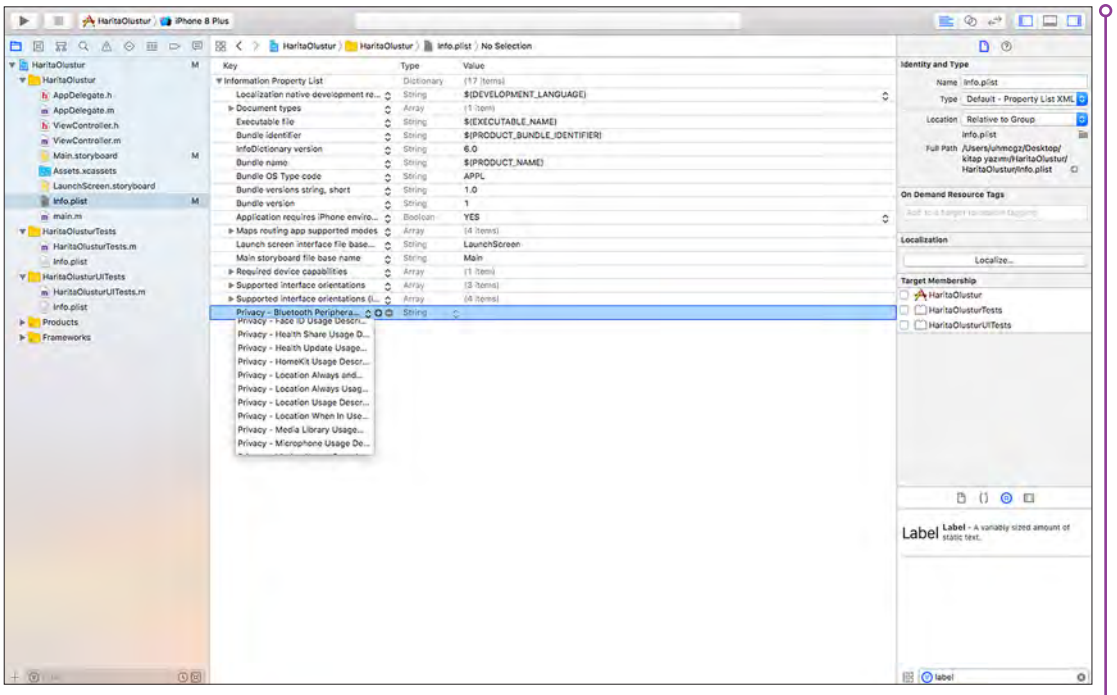
Ekran Görüntüsü 4.105

Harita penceresi uygun boyutlara getirildikten sonra üst bölümüne bir Label eklenerek gerekli biçimsel düzenlemeler yapılır.

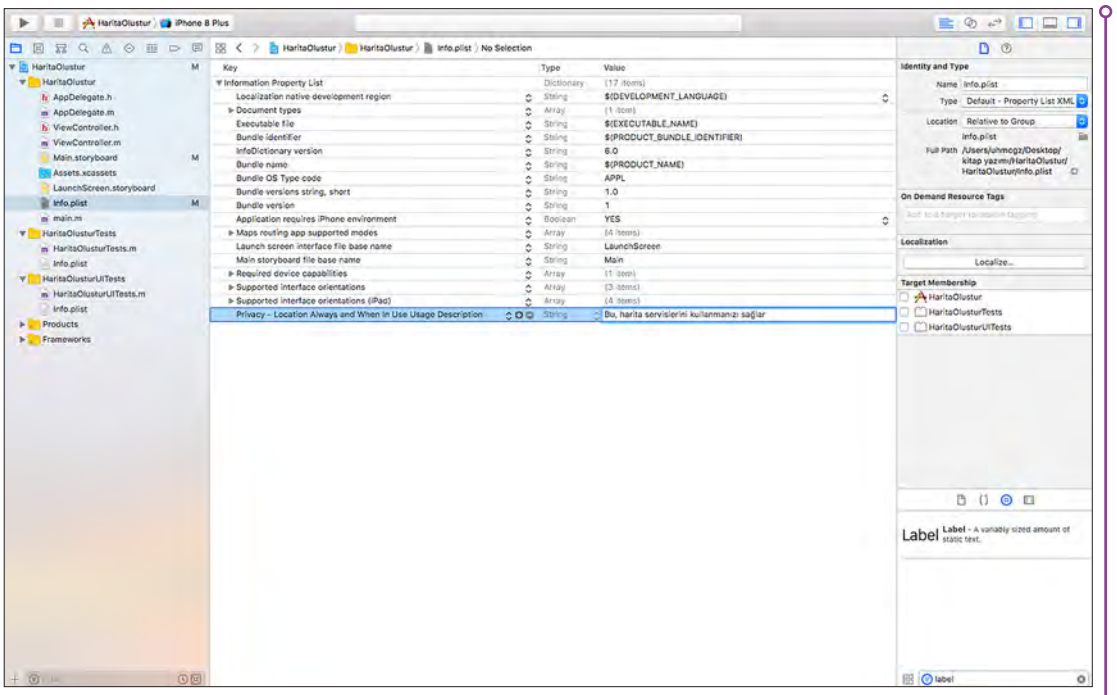


Ekran Görüntüsü 4.106

Kullanıcının konum bilgisi alınacağı için IOS 8'den itibaren kullanıcıdan gizlilik anlaşması (privacy policy) kapsamında izin alınması gerekmektedir. Uygulama dosyaları bölmesinde "info.plist" dosyası açılır. Yeni bir satır eklenerek "Privacy - Location Always and When In Use Usage Description" yazılır. Değeri string olarak kalacaktır. Ardından kullanıcıdan izin alırken ekranda belirecek uyarı penceresinde yazacak metin eklenir.

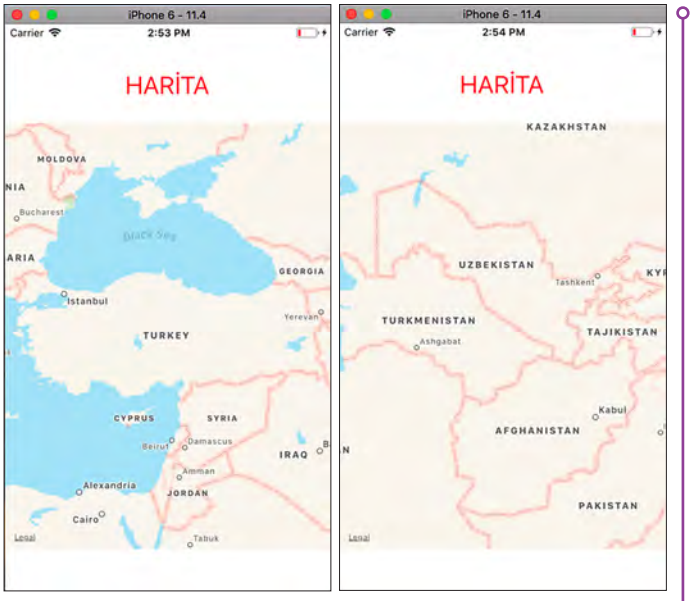


Ekran Görüntüsü 4.107



Ekran Görüntüsü 4.108

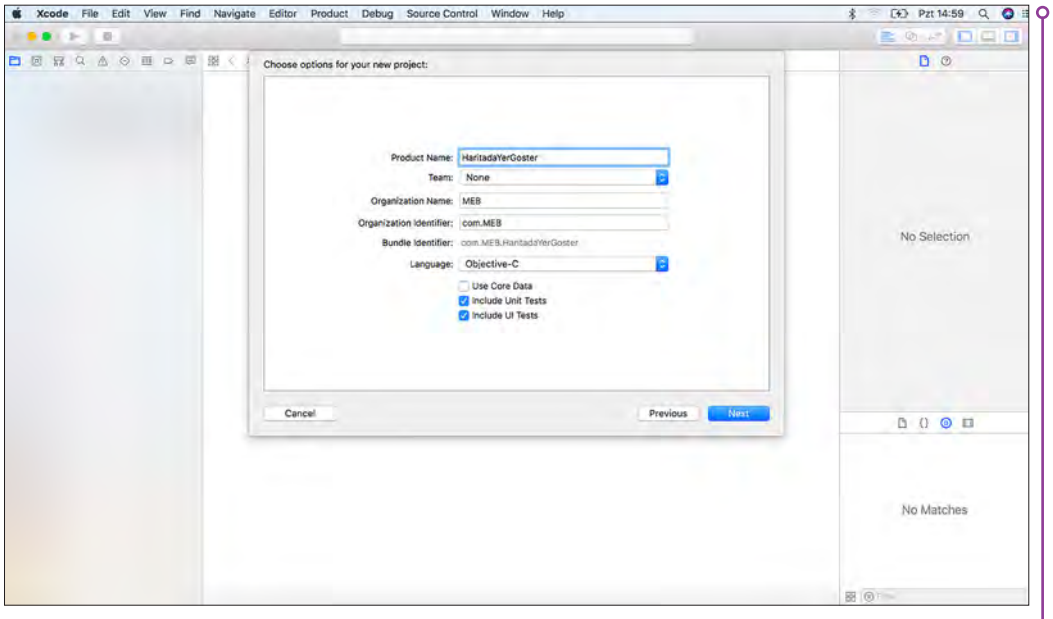
Uygulama simülatörde çalıştırıldığında ekran görüntüsü aşağıdaki gibi olacaktır. Bilgisayar eğer bir internet ağına bağlı olmazsa harita penceresi gelecek fakat harita görünmeyecektir. Ayrıca kullanıcının bulunduğu konum yakınlaştırılmamış şekilde görünmektedir. Daha sonraki uygulamalarda yakınlaştırma (zoom) da kullanılacaktır.



Ekran Görüntüsü 4.109

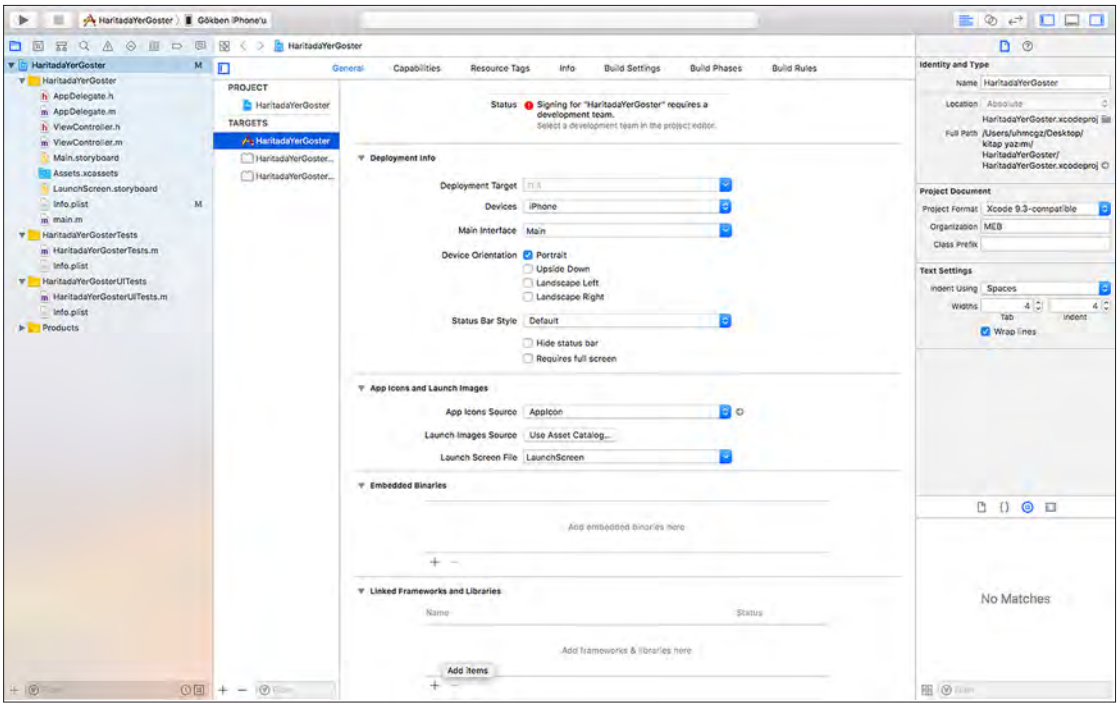
b) Haritada Yer Gösterme

Bu uygulamada, ekrana eklenecek Map View'de (harita penceresinde) küresel koordinat değerleri (enlem ve boylam) verilen özel bir yerin yakınlaştırılmış harita görüntüsü kullanıcıya görüntülenecektir. Bir önceki uygulamada Map View storyboard ekranından eklenirken, bu uygulamada ViewController.m dosyasında kodla oluşturulacaktır.



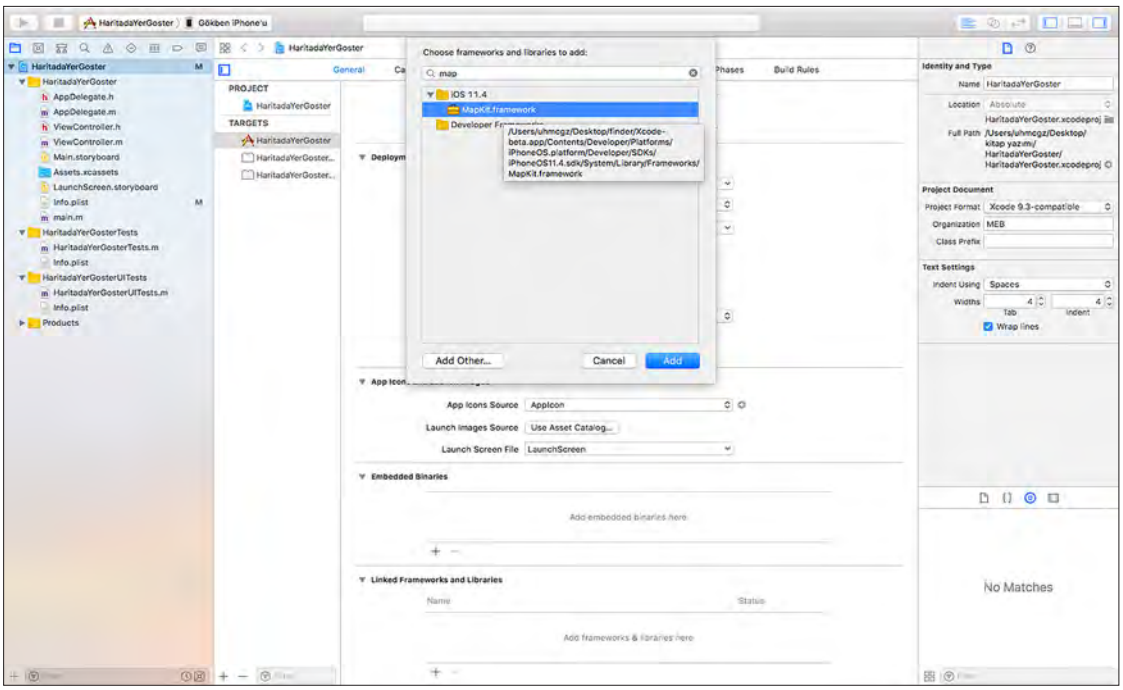
Ekran Görüntüsü 4.110

Genel ayarlardan cihaz yönlendirmesi ve cihaz tipi seçilir.

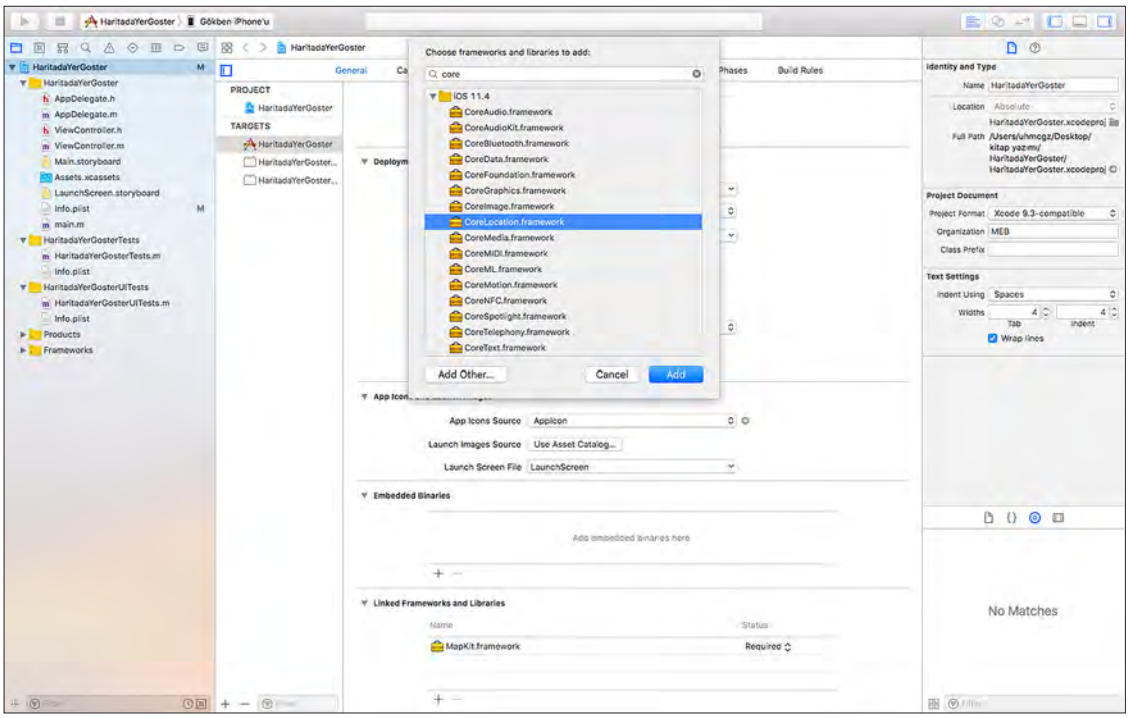


Ekran Görüntüsü 4.111

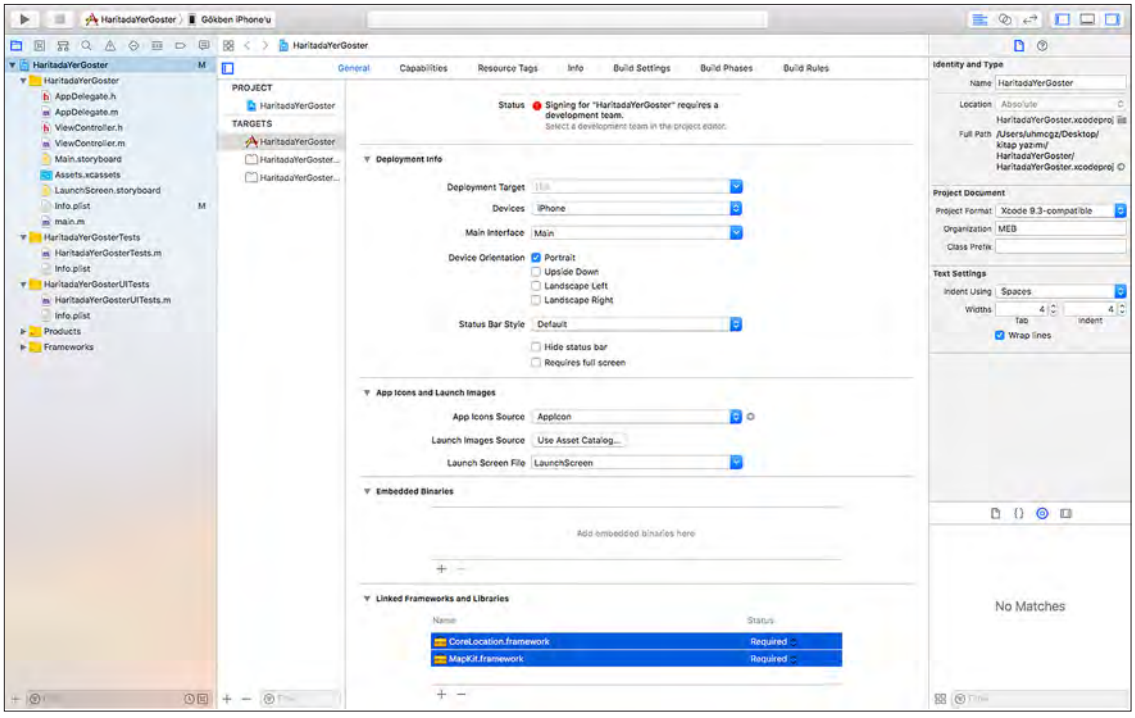
Ardından Linked "Frameworks and Libraries" bölümünden + (artı) işaretine tıklayarak MapKit. Framework ve CoreLocation.Framework kütüphaneleri seçilerek projeye yüklenir.



Ekran Görüntüsü 4.112



Ekran Görüntüsü 4.113



Ekran Görüntüsü 4.114

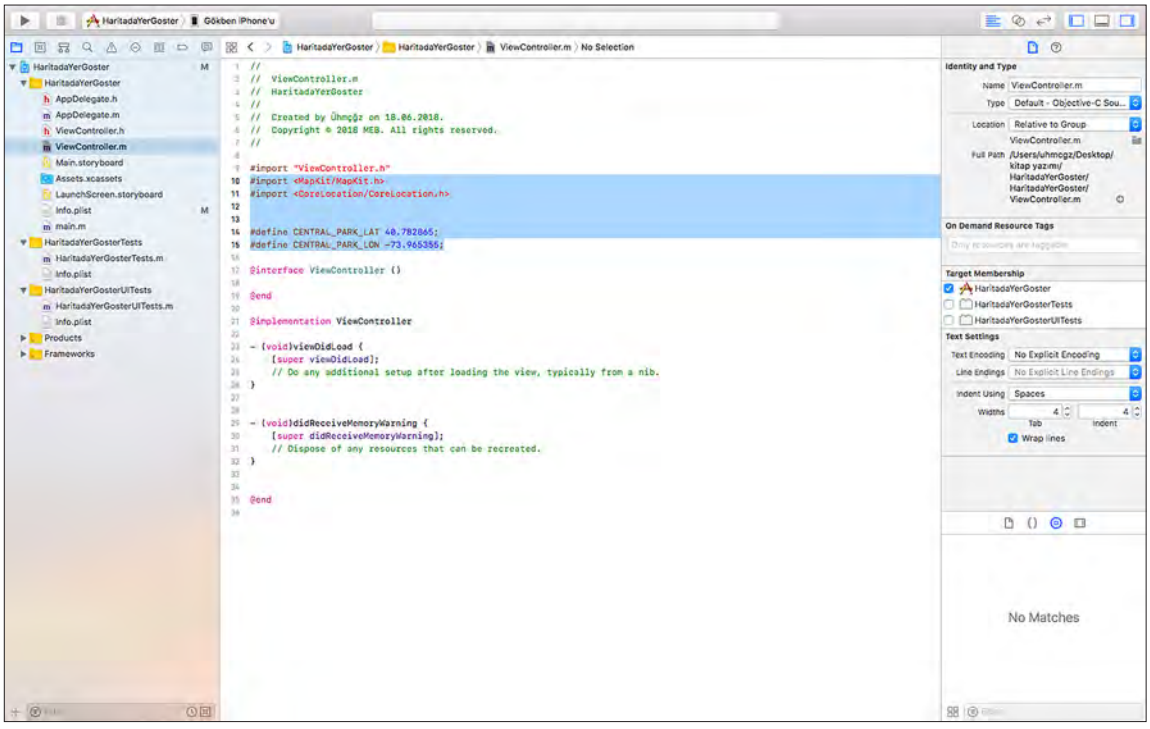
ViewController.m dosyası açılarak indirilen kütüphanelerin header dosyaları

```
#import <MapKit/MapKit.h>
#import <CoreLocation/CoreLocation.h>
```

kodları ile bu dosyaya indirilir. Ardından haritanın konumlanacağı yerin koordinatları

```
#define CENTRAL_PARK_LAT 40.782865;
#define CENTRAL_PARK_LON -73.965355;
```

kodu ile tanımlanır. Bu uygulamada Newyork Central Park adresi kullanıcıya haritada gösterilecektir. Farklı adresler için koordinatlar #define satırında yazılarak tekrarlanabilir.

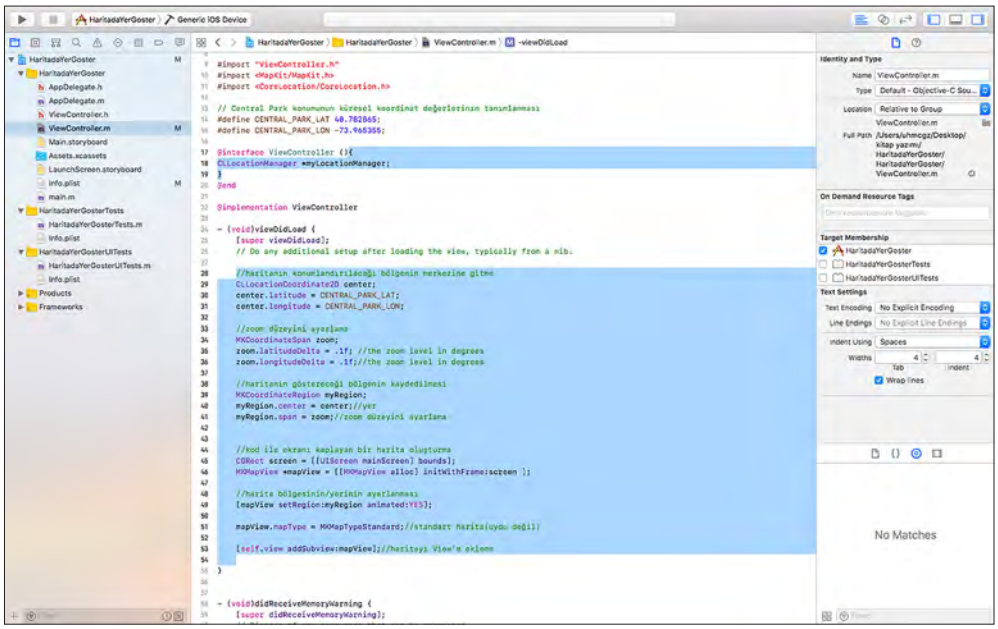


Ekran Görüntüsü 4.115

```
@interface ViewController (){
CLLocationManager *myLocationManager;
}
```

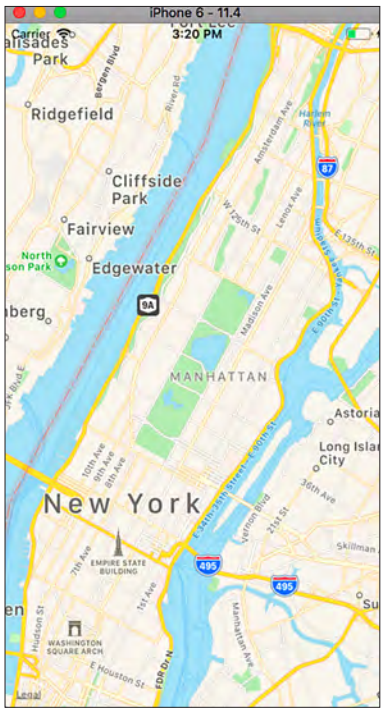
ViewDidLoad fonksiyonunun içine View ekrana yüklendiğinde şunları yap diyen kodlar yazılır.

```
//haritanın konumlandırılacağı bölgenin merkezine gitme
CLLocationCoordinate2D center;
center.latitude = CENTRAL_PARK_LAT;
center.longitude = CENTRAL_PARK_LON;

//zoom düzeyini ayarlama
MKCoordinateSpan zoom;
zoom.latitudeDelta = .1f;
zoom.longitudeDelta = .1f;

//haritanın göstereceği bölgenin kaydedilmesi
MKCoordinateRegion myRegion;
myRegion.center = center; //yer
myRegion.span = zoom; //zoom düzeyini ayarlama


//kod ile ekranı kaplayan bir harita oluşturma
CGRect screen = [[UIScreen mainScreen] bounds];
MKMapView *mapView = [[MKMapView alloc] initWithFrame:screen ];

//harita bölgesinin/yerinin ayarlanması
[mapView setRegion:myRegion animated:YES];
mapView.mapType = MKMapTypeStandard; //standart harita(uydu değil)

[self.view addSubview:mapView]; //haritayı View'e ekleme
```



Ekran Görüntüsü 4.116

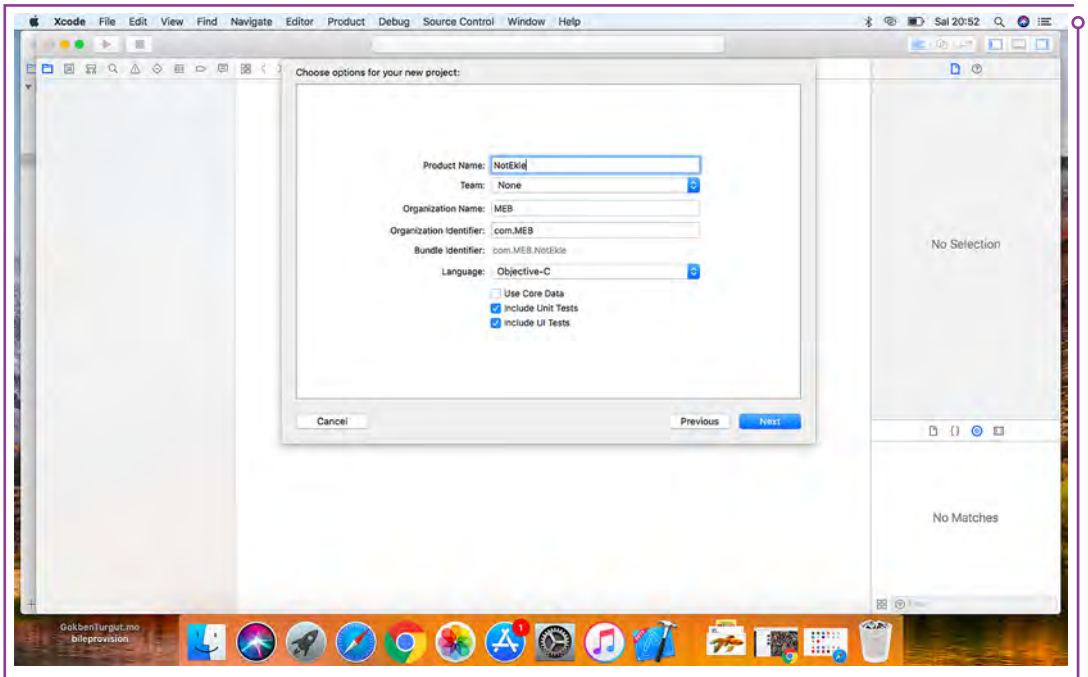
Uygulama simülatörde çalıştırıldığında (bilgisayarınızın internete bağlı olduğundan emin olun) ekran görüntüsü yandaki gibi olacaktır.



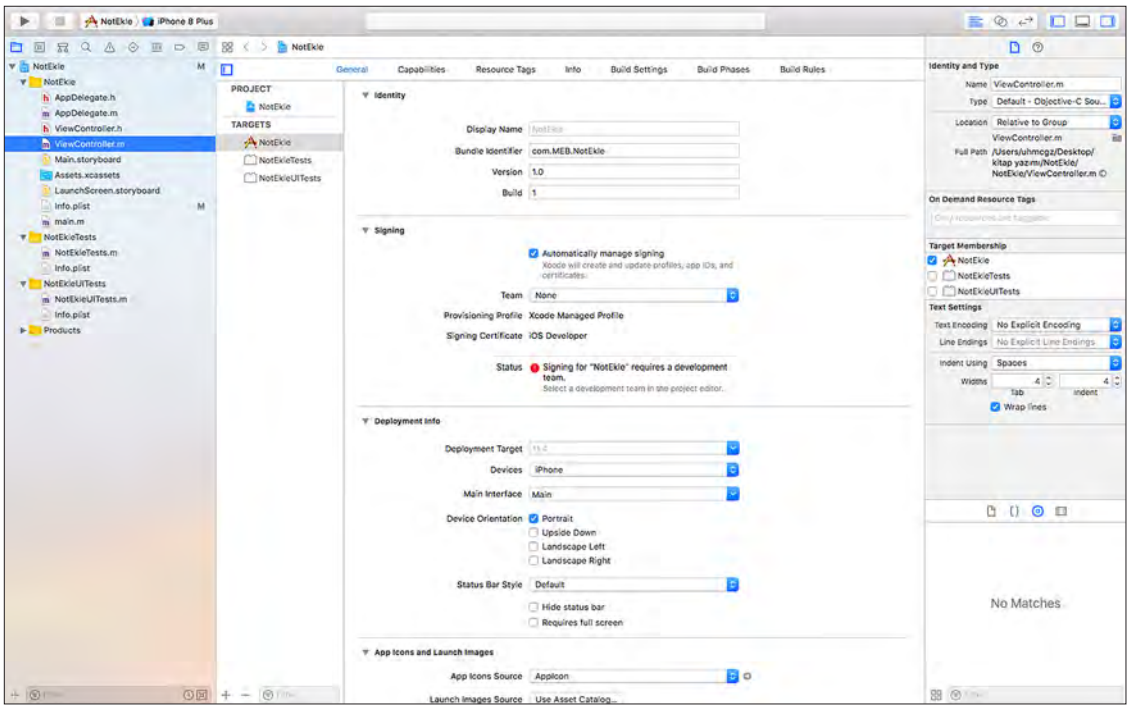
Ekran Görüntüsü 4.117

c) Haritada Toplu İğne Notu (Pin Annotation) Ekleme

Bu uygulamada haritada gösterilen özel bir konumun/adresin üzerine toplu iğne notu eklenerek kullanıcıya yer hakkında bilgi verilecektir. Bu uygulamada da konum olarak Ankara Milli Eğitim Müdürlüğü kullanılacaktır.

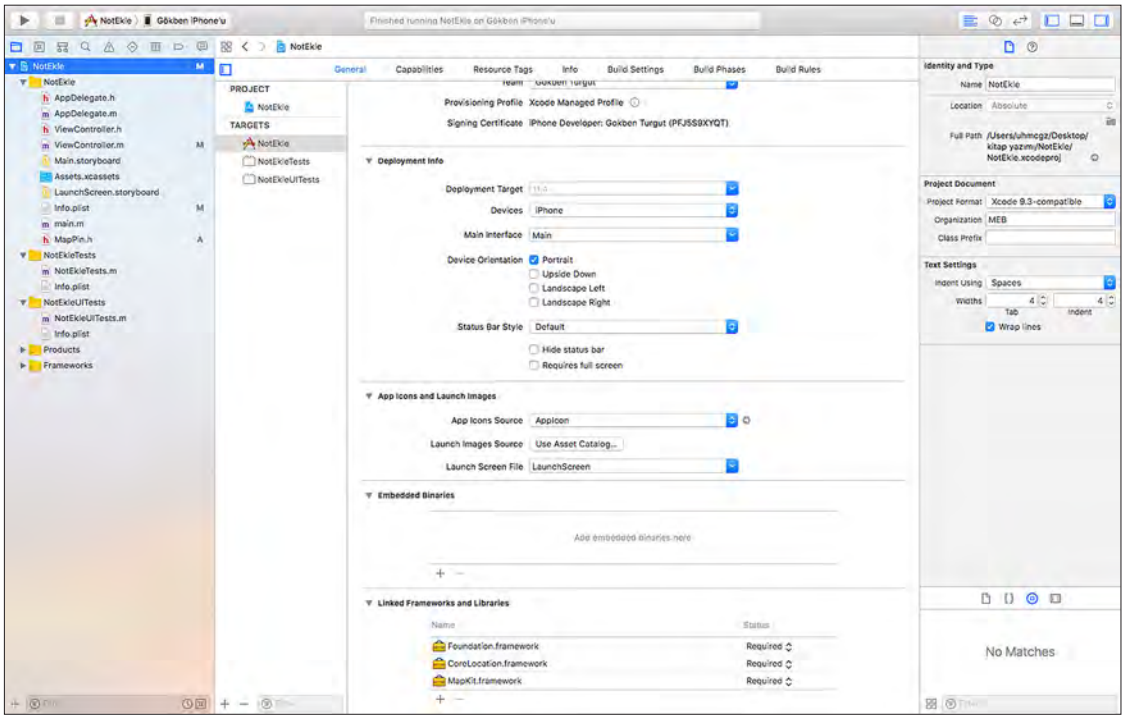


Ekran Görüntüsü 4.118



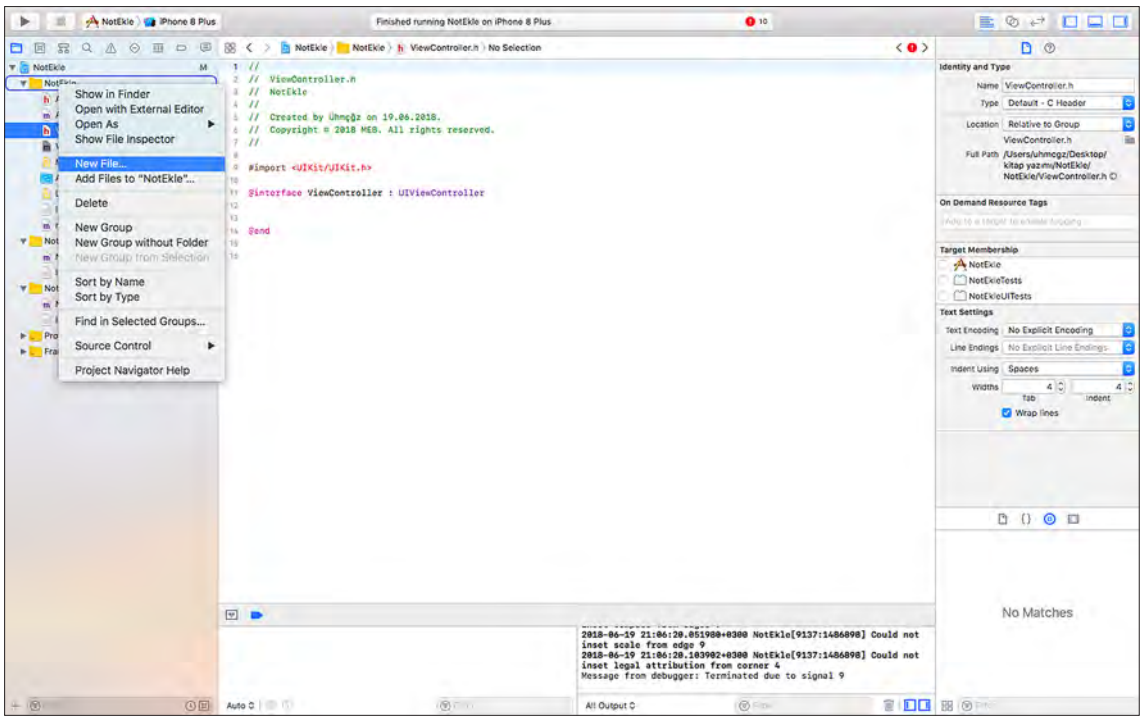
Ekran Görüntüsü 4.119

Foundation, MapKit ve CoreLocation framework'leri indirilir.

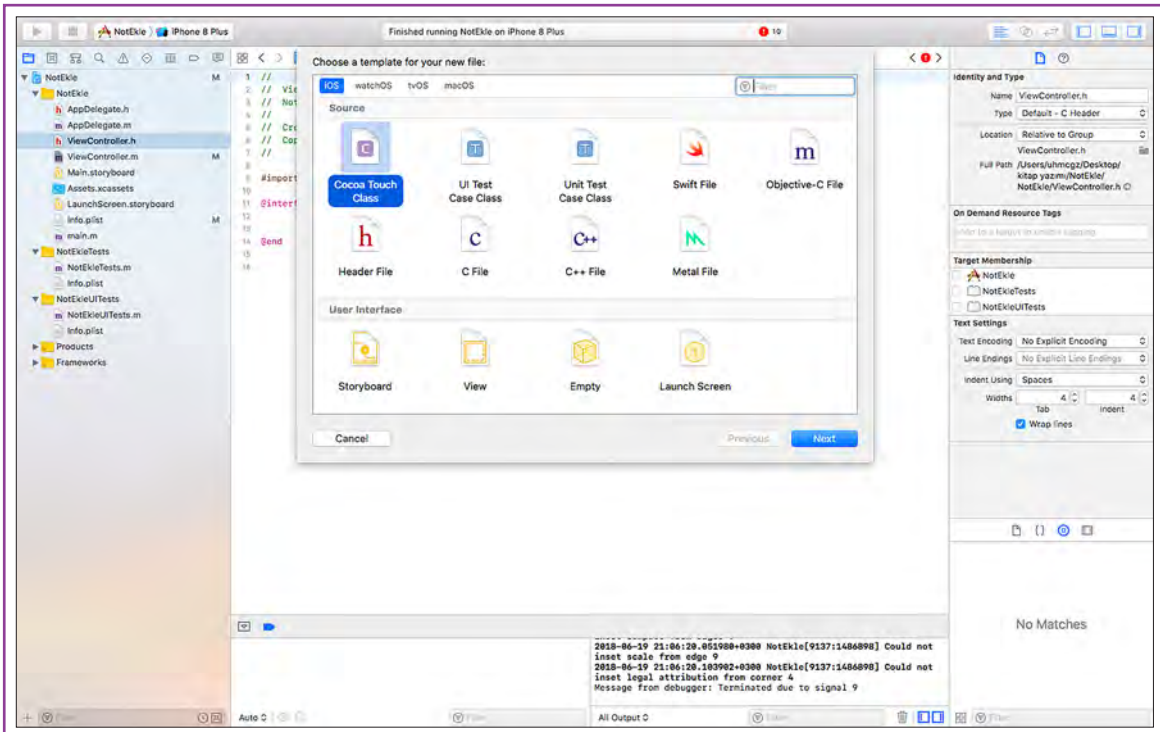


Ekran Görüntüsü 4.120

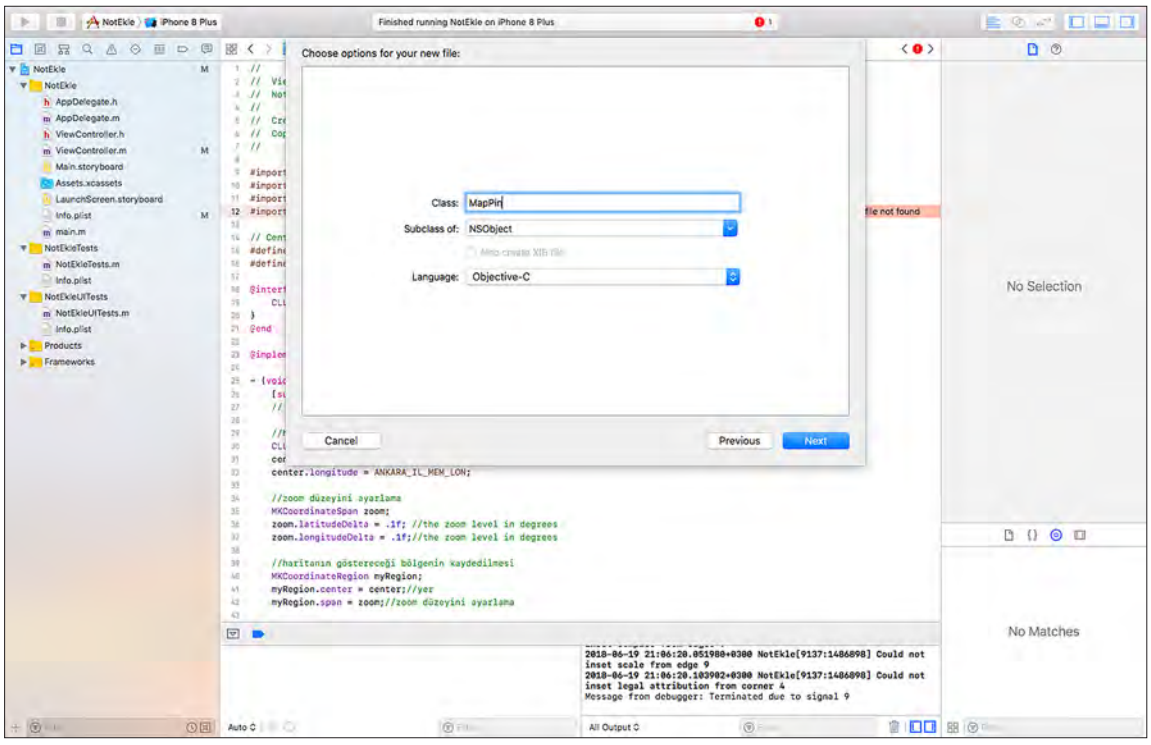
MapPin isimli bir class oluşturulur.



Ekran Görüntüsü 4.121



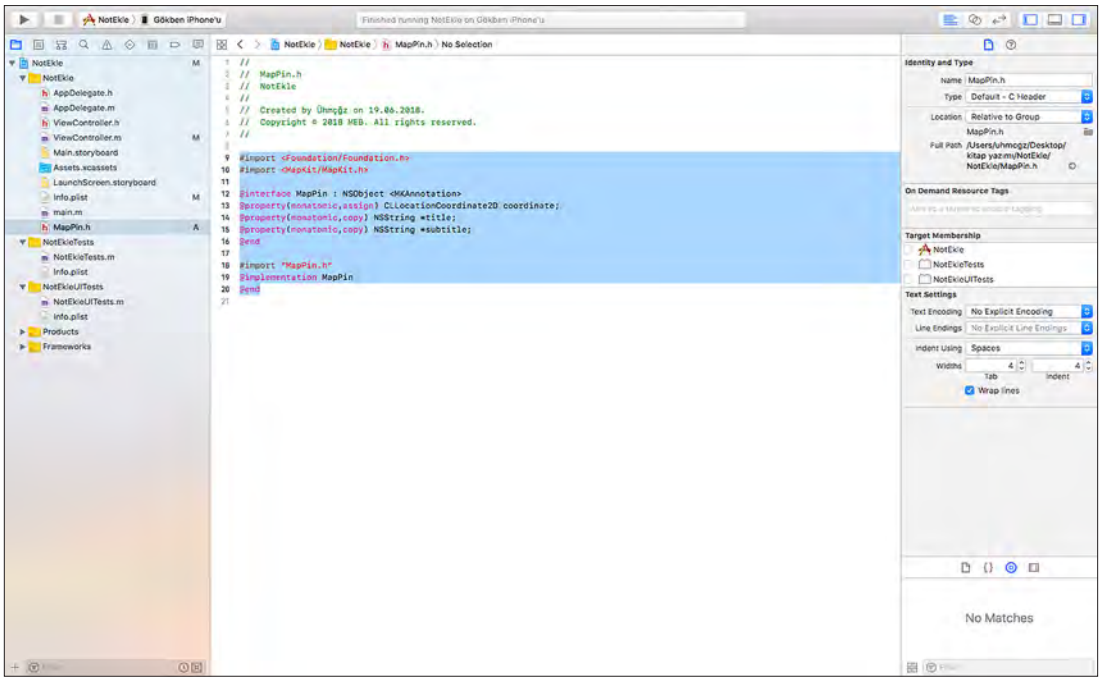
Ekran Görüntüsü 4.122



Ekran Görüntüsü 4.123

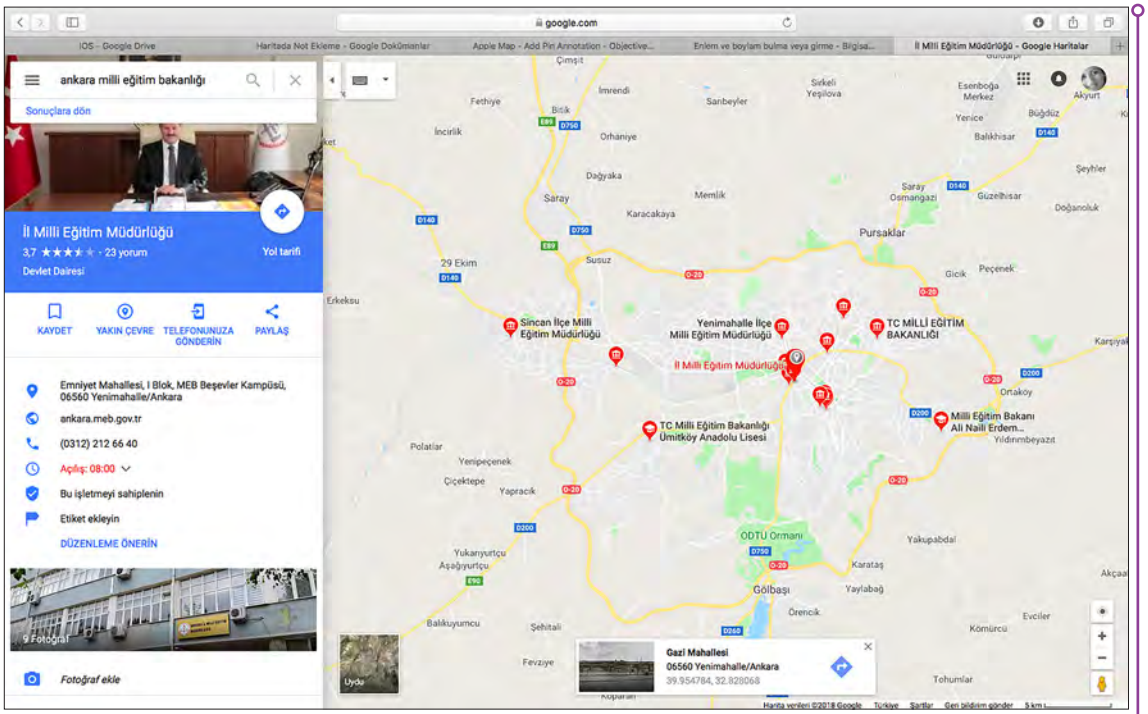
MapPin.m dosyası üzerinde sağ tıklanarak bu dosya silinir. MapPin.h dosyasının içine aşağıdaki kodlar yazılır:

```
#import <Foundation/Foundation.h>
#import <MapKit/MapKit.h>

@interface MapPin : NSObject <MKAnnotation>
@property(nonatomic,assign) CLLocationCoordinate2D coordinate;
@property(nonatomic,copy) NSString *title;
@property(nonatomic,copy) NSString *subtitle;
@end

#import "MapPin.h"
@implementation MapPin
@end
```



Ekran Görüntüsü 4.124

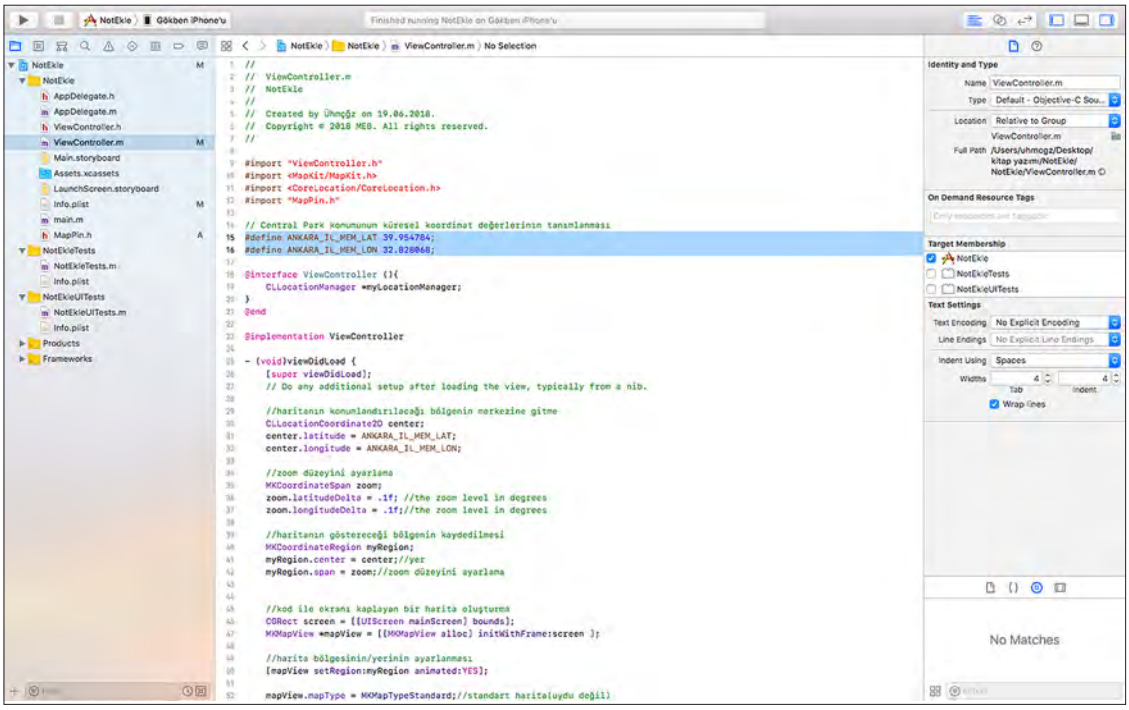
Ankara İl Milli Eğitim Müdürlüğü konumunun yer koordinat (enlem ve boylam) bilgileri Google Haritalar yardımı ile bulunur. Haritada ilgili konumun üzerine tıklandığında açılan bilgilendirme kutucuğundaki enlem ve boylam bilgileri kopyalanır.



Ekran Görüntüsü 4.125

Elde edilen enlem ve boylam bilgileri ViewController.m dosyasında aşağıdaki kod satırına kopyalanır.

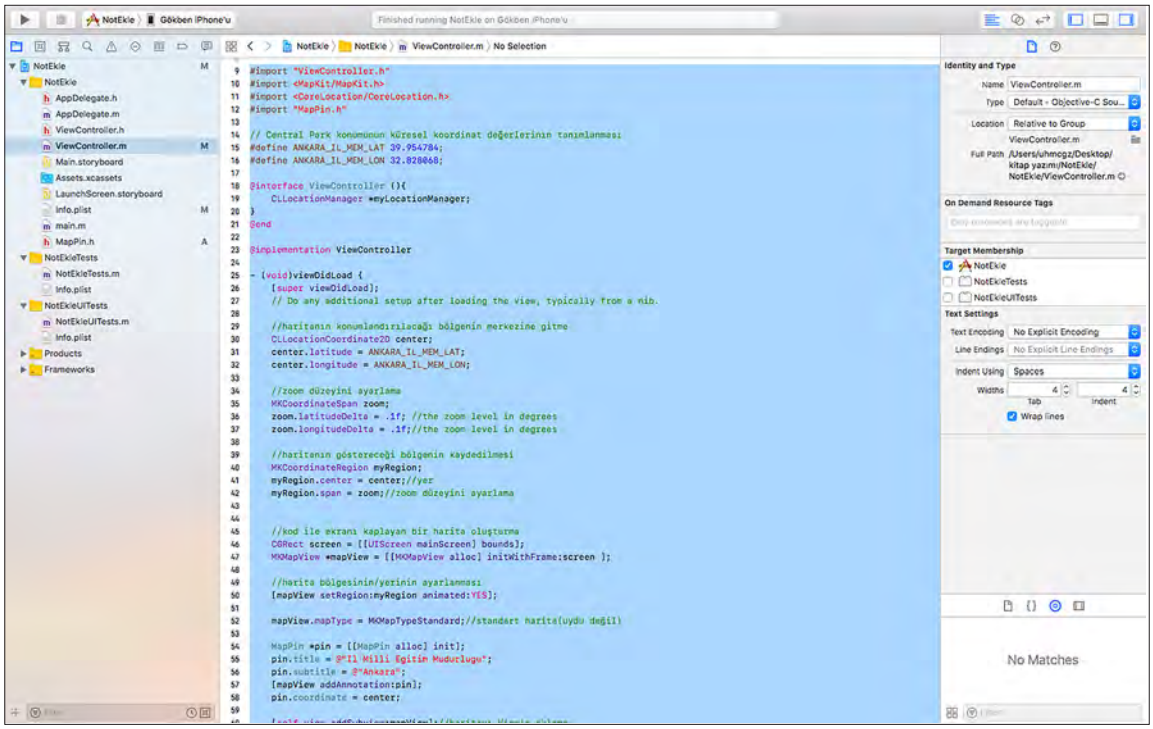
```
#define ANKARA_IL_MEM_LAT 39.954784;
#define ANKARA_IL_MEM_LON 32.828068;
```



Ekran Görüntüsü 4.126

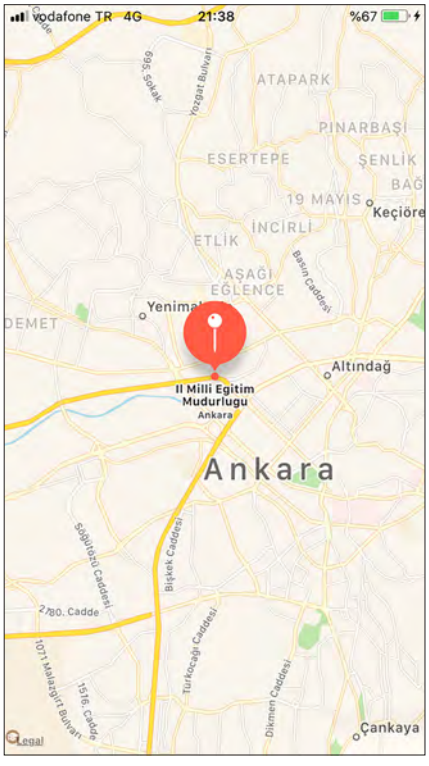
```
#import "ViewController.h"
#import <MapKit/MapKit.h>          // MapKit framework'ünün header dosyasının dosyaya yüklenmesi
#import <CoreLocation/CoreLocation.h>  // CoreLocation framework'ünün header dosyasının dosyaya yüklenmesi
#import "MapPin.h"                 // Oluşturulan MapPin nesnesinin (class'ının) dosyaya yüklenmesi

// Anakara İl Milli Eğitim Müdürlüğü konumunun küresel koordinat değerlerinin tanımlanması
#define ANKARA_IL_MEM_LAT 39.954784;
#define ANKARA_IL_MEM_LON 32.828068;

@interface ViewController (){
CLLocationManager *myLocationManager;
}
@end

@implementation ViewController

- (void)viewDidLoad {
[super viewDidLoad];
// Do any additional setup after loading the view, typically from a nib.

//haritanın konumlandırılacağı bölgenin merkezine gitme
```

```
CLLocationCoordinate2D center;
center.latitude = ANKARA_IL_MEM_LAT;
center.longitude = ANKARA_IL_MEM_LON;

//yakınlaştırma düzeyini ayarlama
MKCoordinateSpan zoom;
zoom.latitudeDelta = .1f; //derece cinsinde zoom düzeyi (enlem)
zoom.longitudeDelta = .1f; //derece cinsinde zoom düzeyi (boylam)

//haritanın göstereceği bölgenin kaydedilmesi
MKCoordinateRegion myRegion;
myRegion.center = center; //yer
myRegion.span = zoom; //zoom düzeyini ayarlama

//kod ile ekranı kaplayan bir harita oluşturma
CGRect screen = [[UIScreen mainScreen] bounds];
        MKMapView *mapView = [[MKMapView alloc] initWithFrame:screen ];

//harita bölgesinin yerinin ayarlanması
[mapView setRegion:myRegion animated:YES];

mapView.mapType = MKMapTypeStandard; //standart harita(uydu değil)

//haritadaki konuma eklenecek not içerisindeki bilgilerin belirlenmesi
MapPin *pin = [[MapPin alloc] init];
        pin.title = @"Il Milli Egitim Mudurlugu";
        pin.subtitle = @"Ankara";
        [mapView addAnnotation:pin];
        pin.coordinate = center;

        [self.view addSubview:mapView]; //haritayı View'e ekleme

}
```



Ekran Görüntüsü 4.127

Uygulama simülatörde çalıştırıldığında ekran görüntüsü yandaki gibi olur.

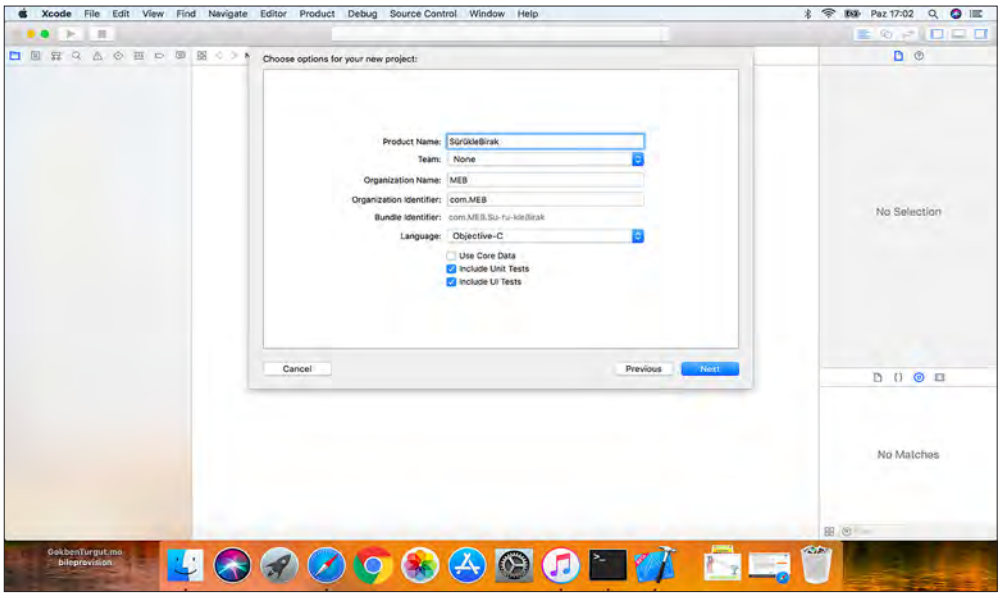


Ekran Görüntüsü 4.128

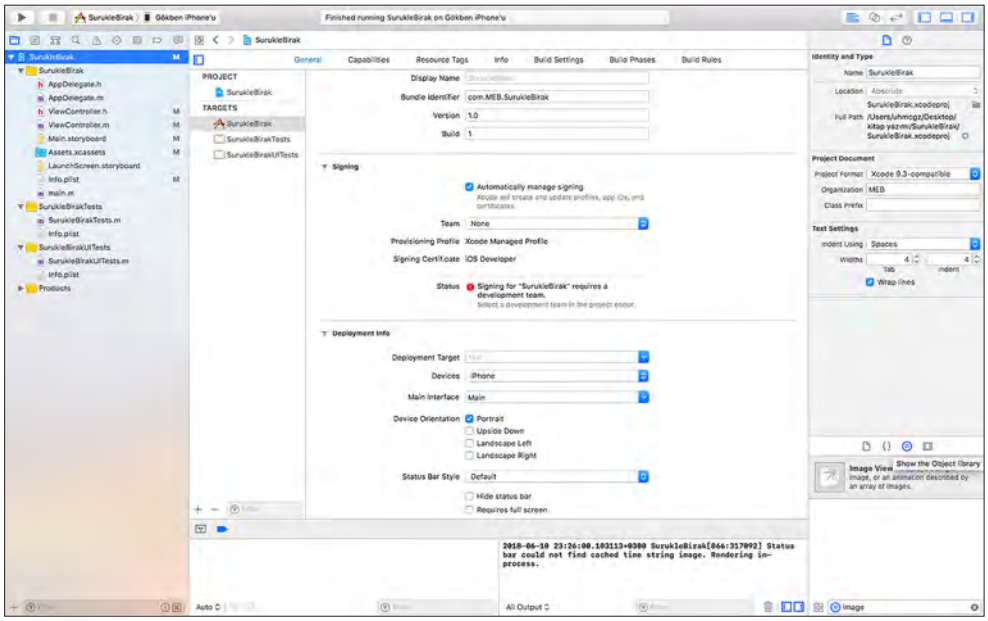
4.2.11. Sürükle Bırak Uygulaması

a) Sürükle-Bırak Uygulaması

Bu uygulamada bir Imageview yani ekrandaki resim sürüklenerek ekran sınırları içerisinde istenilen yere bırakılabilecektir. Öncelikle SurukleBirak isimli bir Xcode projesi açılır. Ekran sadece Portrait yönüne kilitlenir ve device alanından iPhone seçilir.

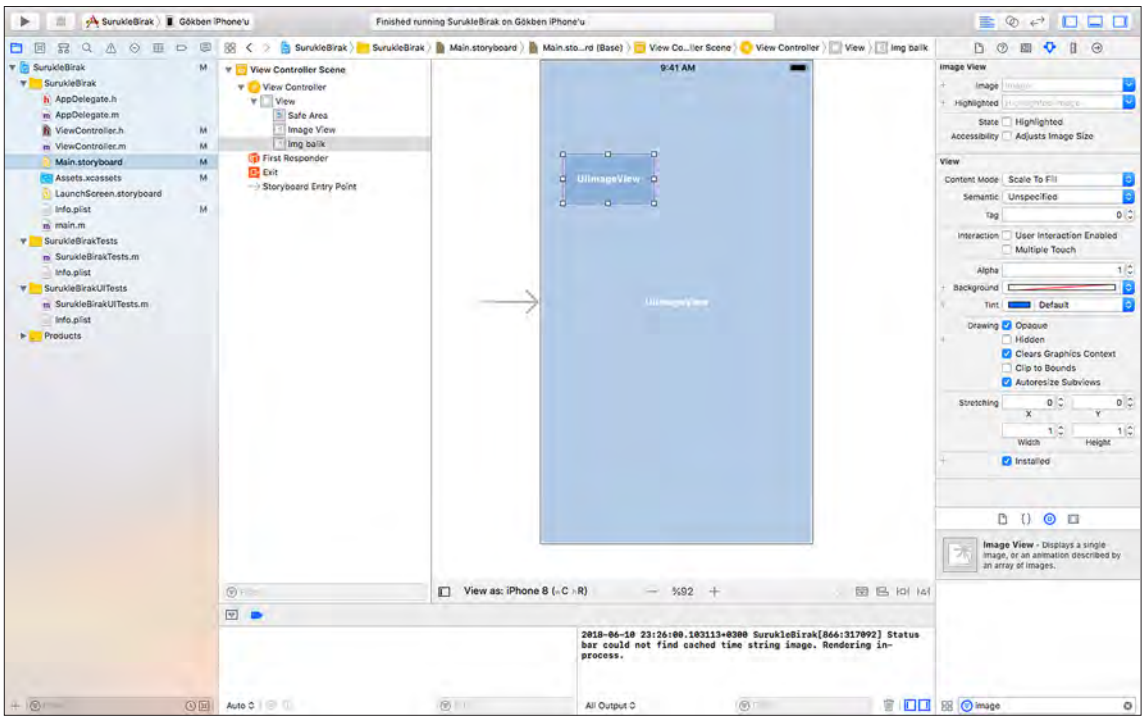


Ekran Görüntüsü 4.129



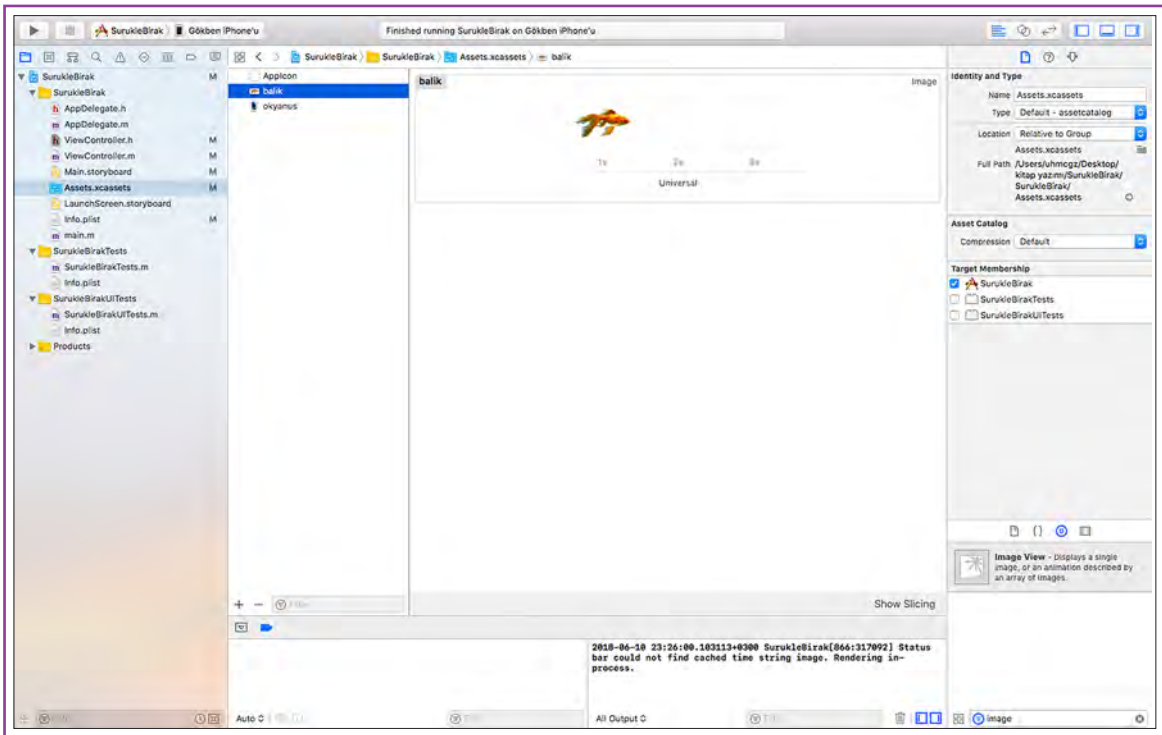
Ekran Görüntüsü 4.130

Main.Storyboard açılarak 2 adet UIImageView biri ekranı tümüyle kaplayacak diğeri ise sürükle bırak yapılacak resim boyutlarında bırakılacaktır. İlk eklenen ImageView ekran arka planı olarak kullanılacağından view hiyerarşisinde sürükle bırak yapılacak ImageView'ın altında olması gerekmektedir.

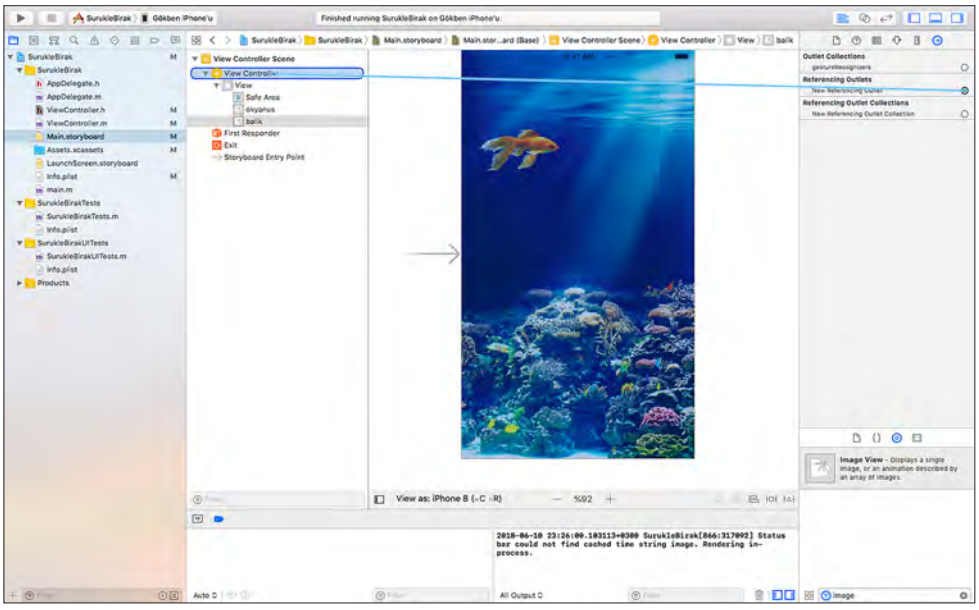


Ekran Görüntüsü 4.131

Assets dosyası açılarak içine balık ve okyanus resimleri kaydedilir. Sonra bu resimler Storyboard'da ImageView'lerin içine yerleştirilir.



Ekran Görüntüsü 4.132

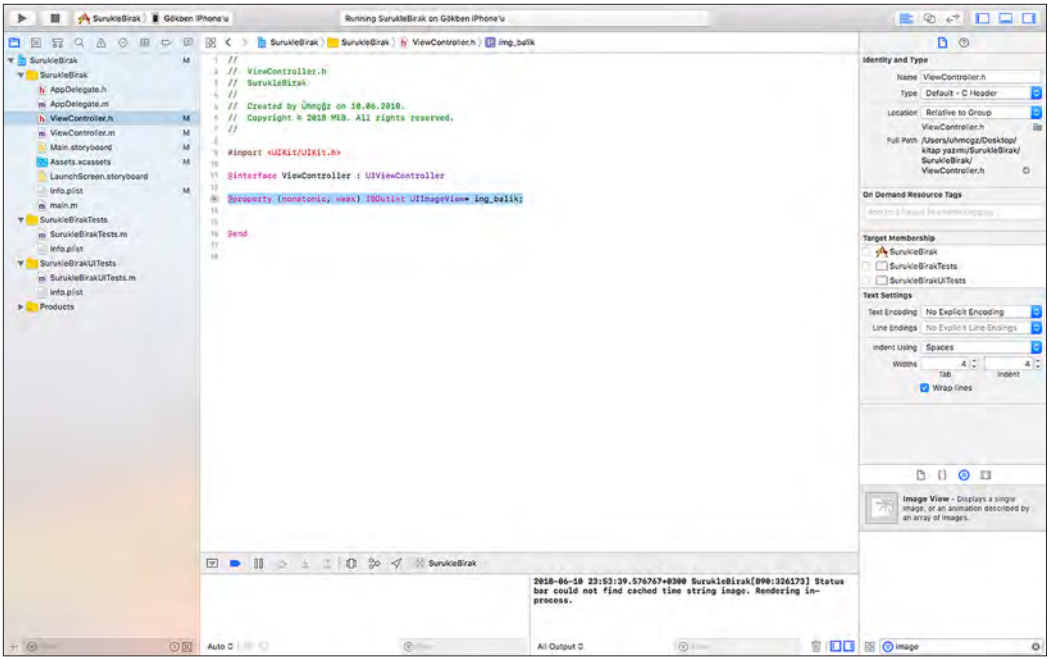


Ekran Görüntüsü 4.133

Balık üzerinde işlem yapılabilmesi için programa tanıtılması gerekmektedir. ViewController.h dosyası açılarak

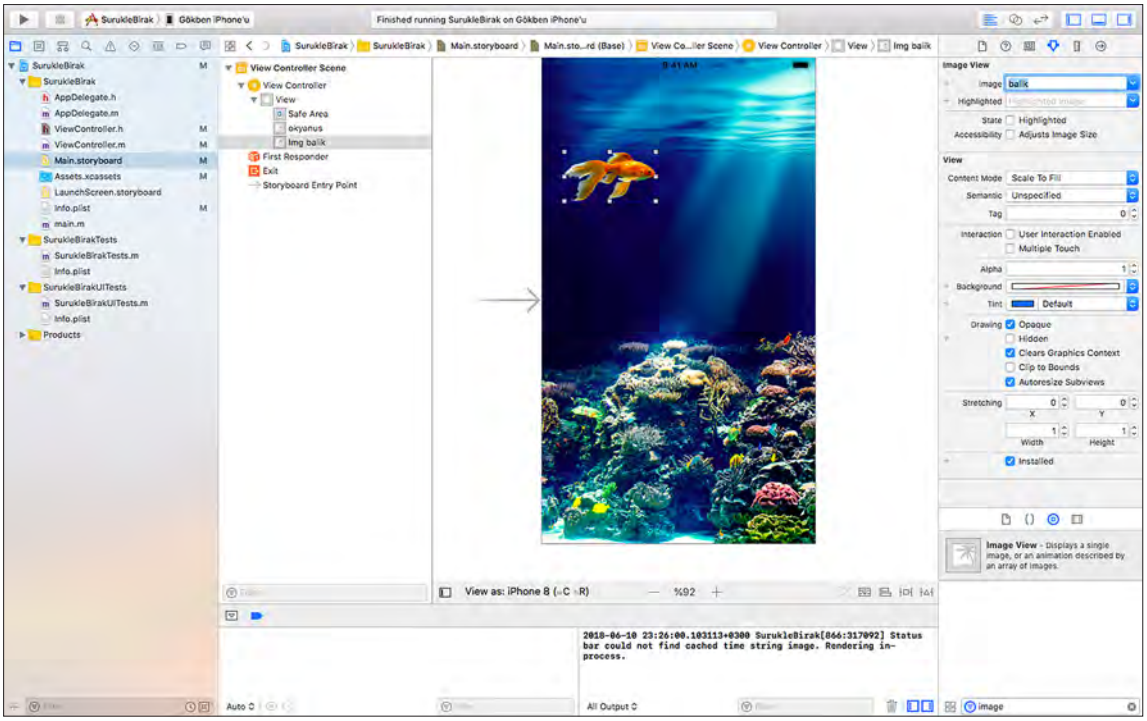
```
@property (nonatomic, weak) IBOutlet UIImageView* img_balik;
```

kodu yazılır.



Ekran Görüntüsü 4.134

Storyboard ekranında sağ bölmede ilişkiler sekmesinde balık ImageView'i ile ViewController.h'de verilen ismi arasında ilişki kurulur.



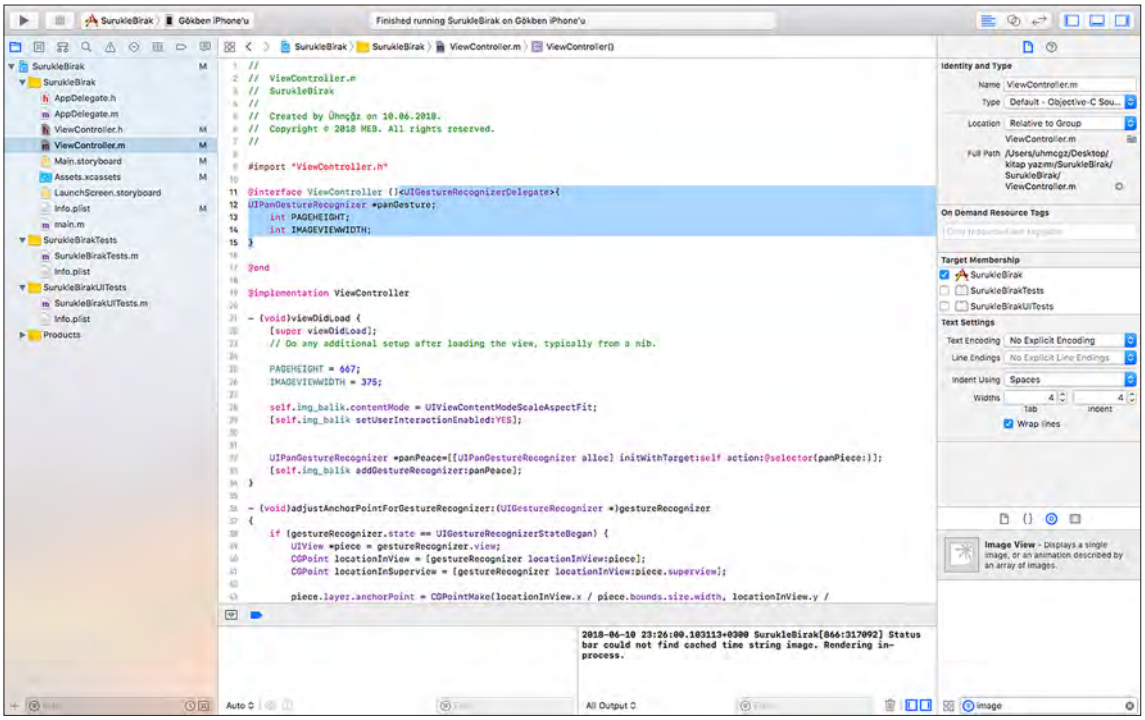
Ekran Görüntüsü 4.135

ViewController.m dosyasında üst kısma <> işaretleri arasında

```
<UIGestureRecognizerDelegate>
```

kodu yazılarak bir nesne üzerindeki touch (dokunma) hareketlerini çağıran "delegate" metodu ilgili dosyada çalışmak üzere çağrılır. Devamında süslü parantez ({....}) içinde bazı değişkenler tanımlanır.

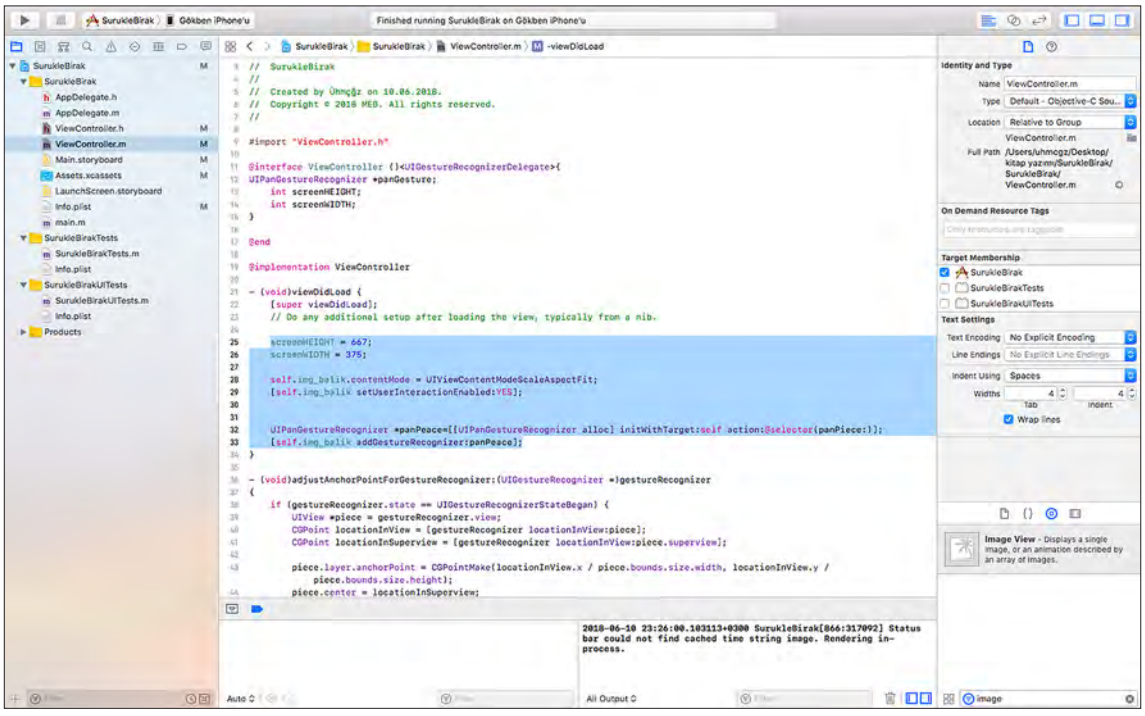
```
{
UIPanGestureRecognizer *panGesture;
int screenHEIGHT;
int screenWIDTH;
}
```



Ekran Görüntüsü 4.136

viewDidLoad (view ekrana yüklendiğinde çalıştırılacak kodlar buraya yazılır) fonksiyonu içerisine aşağıdaki kod yazılır.

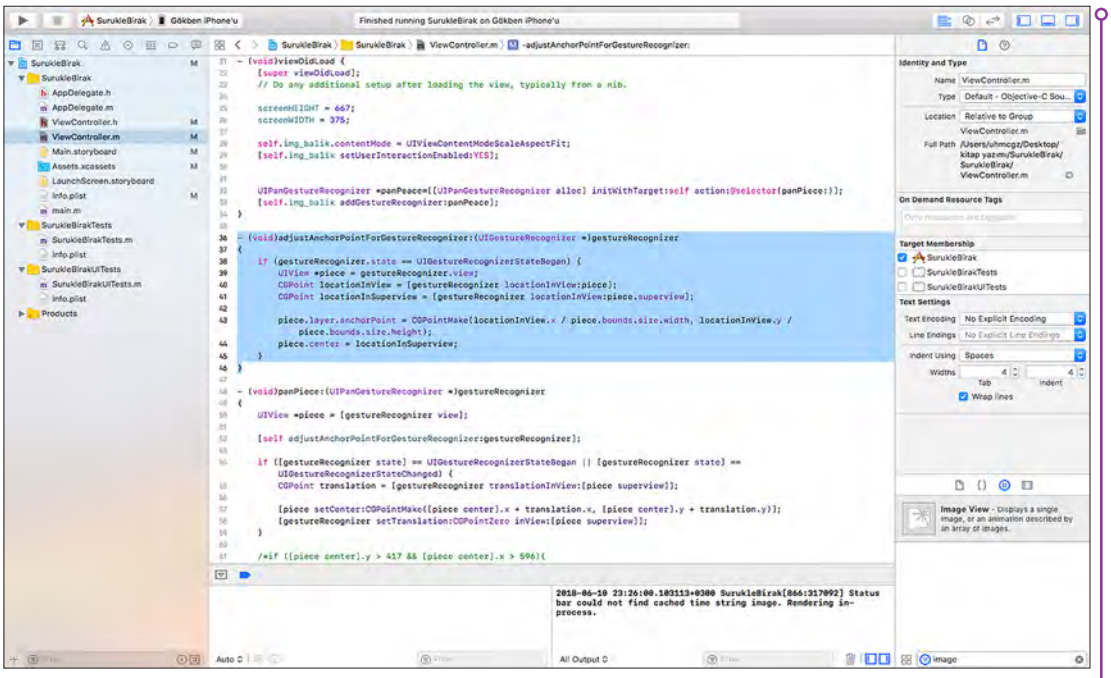
```
screenHEIGHT = 667;      // ekran yüksekliğini int tipindeki değişken içinde tutma
screenWIDTH = 375;     // ekran genişliğini int tipindeki değişken içinde tutma


self.img_balik.contentMode = UIViewContentModeScaleAspectFit;  // Balık üzerinde dokunma özelliği aktif hale getirilir.
[self.img_balik setUserInteractionEnabled:YES];  // Balık üzerinde kullanıcı etkileşimi aktif hale getirilir.
UIPanGestureRecognizer *panPeace=[[UIPanGestureRecognizer alloc] initWithTarget:self action:@selector(panPiece:)];
[self.img_balik addGestureRecognizer:panPeace];
// panPeace isimli UIPanGestureRecognizer sınıfından panPiece fonksiyonu ile tetiklenecek bir değişken yaratılır ve bu değişken balık ImageView'ine tanımlanır.
```



Ekran Görüntüsü 4.137

viewDidLoad'dan çıkılarak aşağıdaki kodlar dosyaya eklenir.

```
- (void)adjustAnchorPointForGestureRecognizer:(UIGestureRecognizer *)gestureRecognizer
{
if (gestureRecognizer.state == UIGestureRecognizerStateBegan) {
UIView *piece = gestureRecognizer.view;
CGPoint locationInView = [gestureRecognizer locationInView:piece];
CGPoint locationInSuperview = [gestureRecognizer locationInView:piece.superview];
piece.layer.anchorPoint = CGPointMake(locationInView.x / piece.bounds.size.width, locationInView.y / piece.bounds.
size.height);
            piece.center = locationInSuperview;
            }
}
```

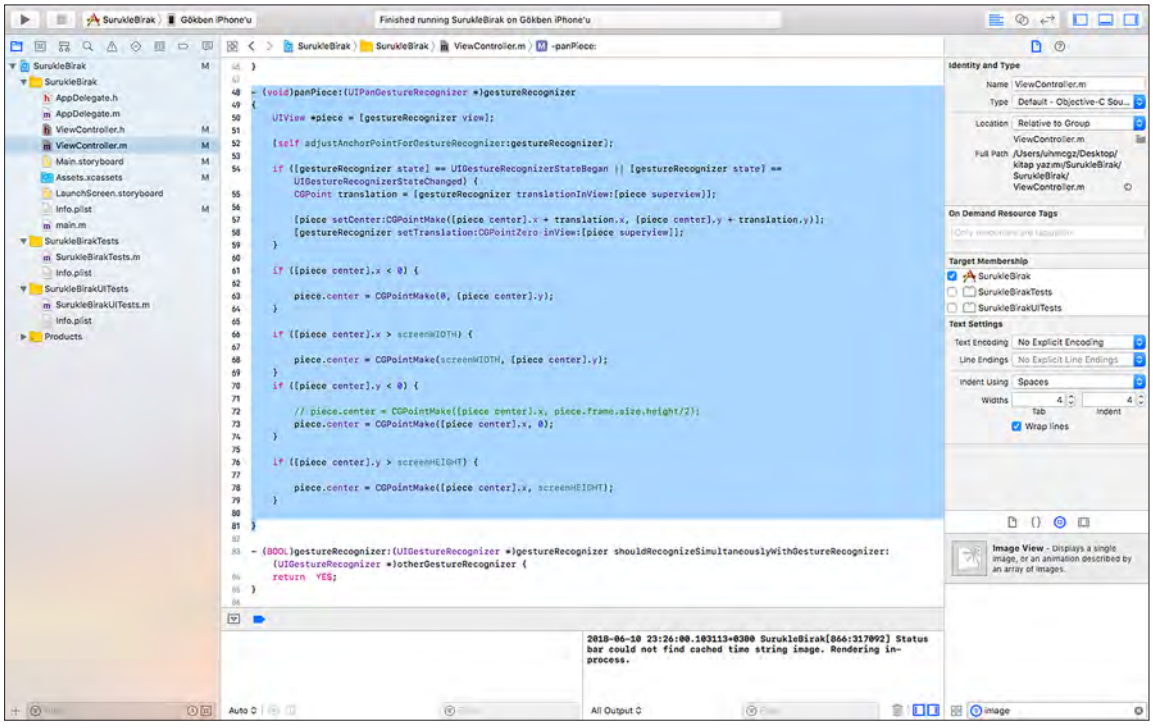


Ekran Görüntüsü 4.138

Ardından panPiece fonksiyonu yazılır:

```
- (void)panPiece:(UIPanGestureRecognizer *)gestureRecognizer
{
UIView *piece = [gestureRecognizer view];

[self adjustAnchorPointForGestureRecognizer:gestureRecognizer];

if ([gestureRecognizer state] == UIGestureRecognizerStateBegan || [gestureRecognizer state] == UIGestureRecogni-
zerStateChanged) {
                    CGPoint translation = [gestureRecognizer translationInView:[piece superview]];
                    [piece setCenter:CGPointMake([piece center].x + translation.x, [piece center].y + translation.y)];
                    [gestureRecognizer setTranslation:CGPointZero inView:[piece superview]];
    }

// Ekranın dışına çıkılmaması için yazılan kodların başlangıcı
if ([piece center].x < 0) {
piece.center = CGPointMake(0, [piece center].y);
}

if ([piece center].x > screenWIDTH) {
piece.center = CGPointMake(screenWIDTH, [piece center].y);
    }

if ([piece center].y < 0) {
```

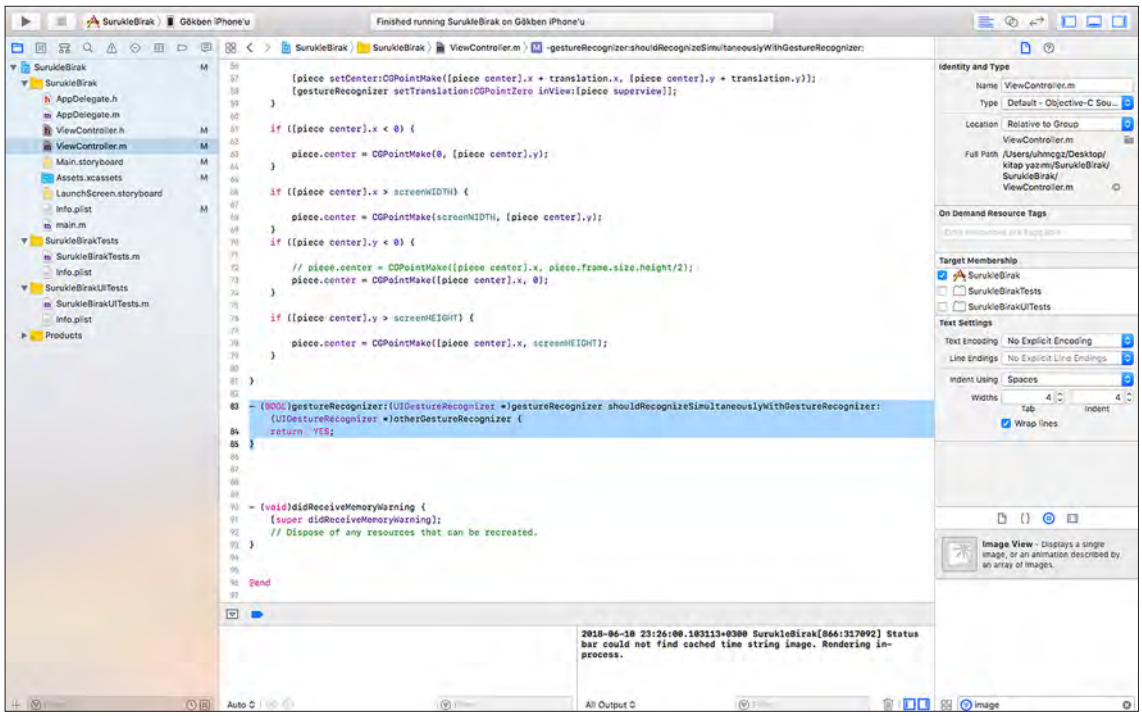
```
piece.center = CGPointMake([piece center].x, 0);
    }

if ([piece center].y > screenHEIGHT) {
piece.center = CGPointMake([piece center].x, screenHEIGHT);
}
  // Ekranın dışına çıkılmaması için yazılan kodların sonu
}
```



Ekran Görüntüsü 4.139

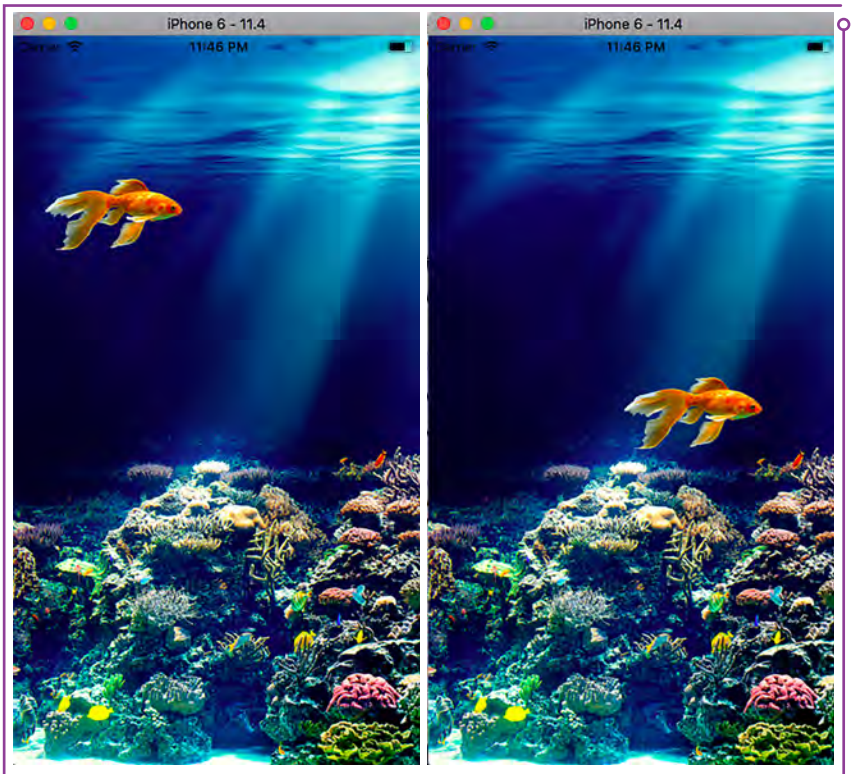
Son olarak aynı dosyaya aşağıdaki kodlar eklenir.

```
-(BOOL)gestureRecognizer:(UIGestureRecognizer *)gestureRecognizer shouldRecognizeSimultaneouslyWithGesture-
Recognizer:(UIGestureRecognizer *)otherGestureRecognizer {
return  YES;
}
```



Ekran Görüntüsü 4.140

Uygulama iPhone 6 simülatöründe çalıştırılarak fare ile balık sürükle bırak yapıldığında ekran görüntüleri aşağıdaki gibi olacaktır.



Ekran Görüntüsü 4.141

B) Resmi İki Parmakla Büyüt/Küçült ve Çevir Uygulaması

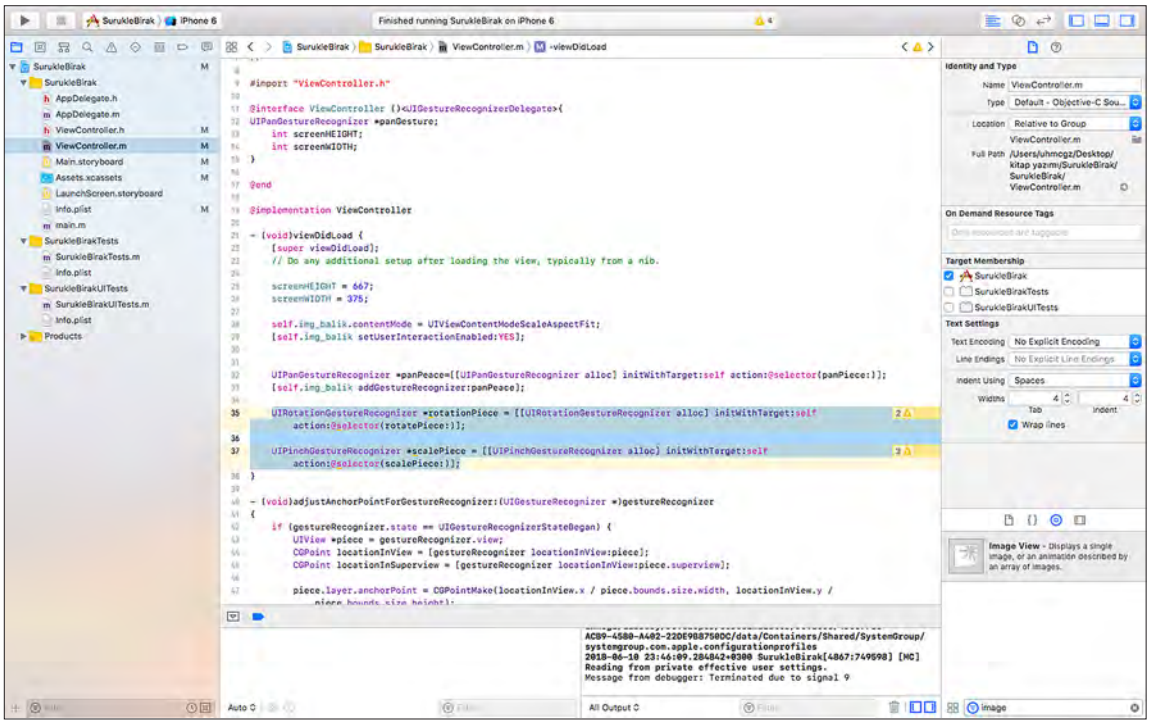
Bu uygulamada kullanıcının ekrandaki bir resmi iki parmak hareketi ile büyütüp küçültmesi ve çevirmesi sağlanacaktır. Bu uygulama sürükle-bırak uygulaması üzerinden devam edecektir.

viewDidLoad içerisine;

viewDidLoad içerisine;

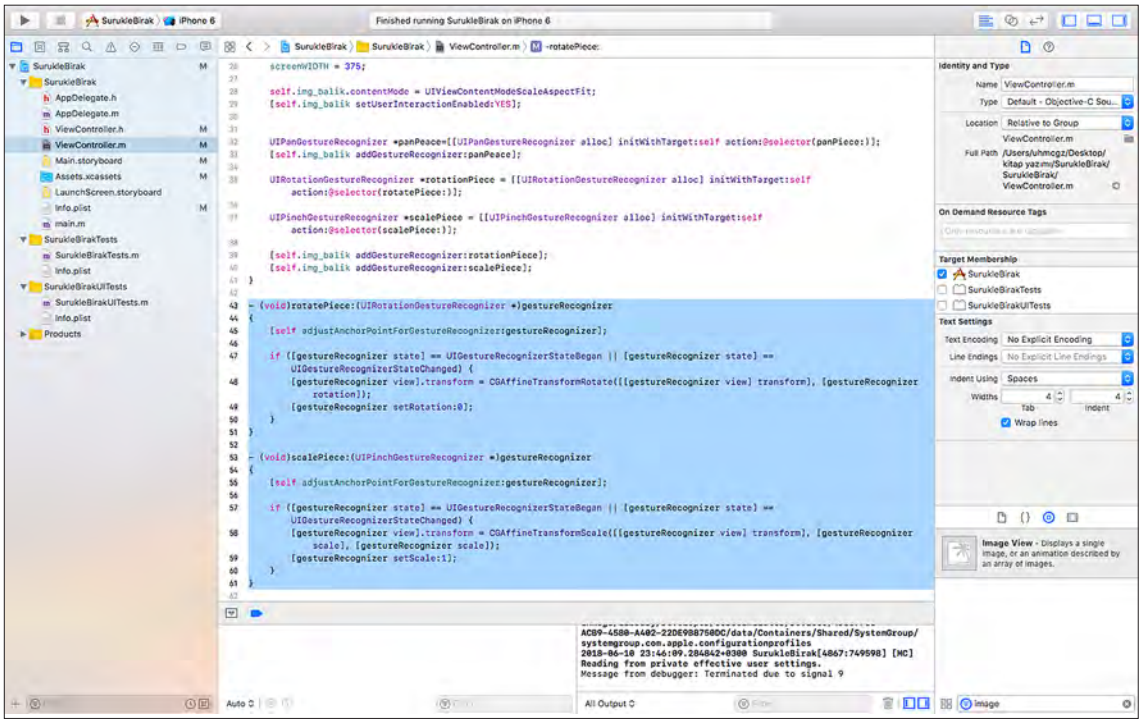
```
// resmi iki parmakla çevirme işlemi için rotatePiece fonksiyonu ile tetiklenecek rotationPiece değişkeni yaratılır.

UIRotationGestureRecognizer *rotationPiece = [[UIRotationGestureRecognizer alloc] initWithTarget:self action:@
selector(rotatePiece:)];

// resmi iki parmakla büyütme/küçültme  işlemi için scalePiece fonksiyonu ile tetiklenecek scalePiece değişkeni yaratılır.

UIPinchGestureRecognizer *scalePiece = [[UIPinchGestureRecognizer alloc] initWithTarget:self action:@selector(s-
calePiece:)];

// Yaratılan değişkenler img_balik için tanımlanır.

[self.img_balik addGestureRecognizer:rotationPiece];
[self.img_balik addGestureRecognizer:scalePiece];
```

kodları yazılır.



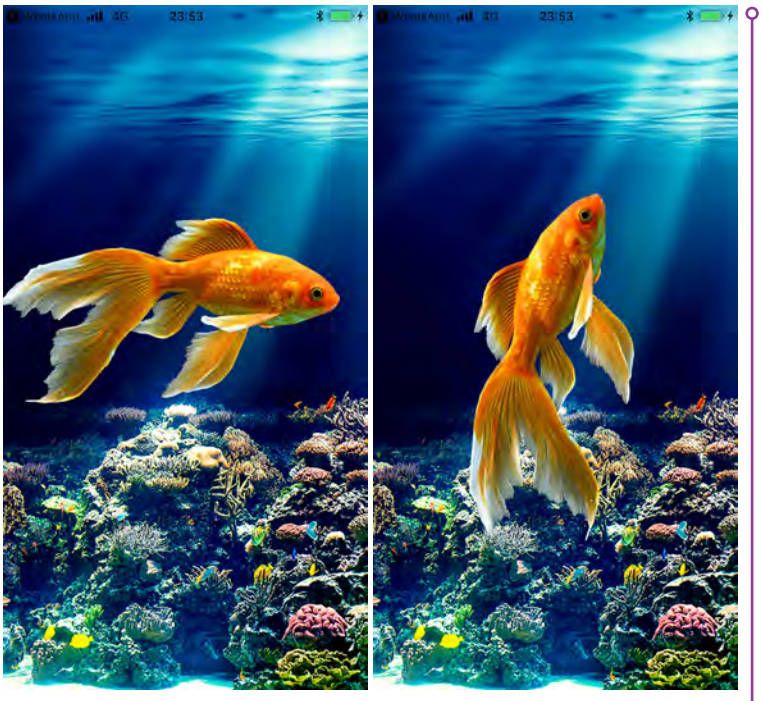
Ekran Görüntüsü 4.142

```
- (void)rotatePiece:(UIRotationGestureRecognizer *)gestureRecognizer
{
[self adjustAnchorPointForGestureRecognizer:gestureRecognizer];

if ([gestureRecognizer state] == UIGestureRecognizerStateBegan || [gestureRecognizer state] == UIGestureRecognizerStateChanged) {
        [gestureRecognizer view].transform = CGAffineTransformRotate([[gestureRecognizer view] transform], [gestureRecognizer rotation]);
        [gestureRecognizer setRotation:0];
    }
}

- (void)scalePiece:(UIPinchGestureRecognizer *)gestureRecognizer
{
[self adjustAnchorPointForGestureRecognizer:gestureRecognizer];

if ([gestureRecognizer state] == UIGestureRecognizerStateBegan || [gestureRecognizer state] == UIGestureRecognizerStateChanged) {
        [gestureRecognizer view].transform = CGAffineTransformScale([[gestureRecognizer view] transform], [gestureRecognizer scale], [gestureRecognizer scale]);
        [gestureRecognizer setScale:1];
    }
}
```



Ekran Görüntüsü 4.143

Uygulama çalıştırılarak balık resmi büyütüldüğünde ve çevrildiğinde ekran görüntüleri aşağıdaki gibi olur.



Ekran Görüntüsü 4.144

Uygulama simülatörde hata vermeden çalışabilmektedir fakat iki parmak hareketlerinin yapılabilmesi için gerçek cihaza ihtiyaç duyulmaktadır. Uygulamanın cihazda çalıştırılabilmesi için "Ios Developer hesabının alınması" ve "Developer hesabına cihaz eklenmesi" gerekmektedir.

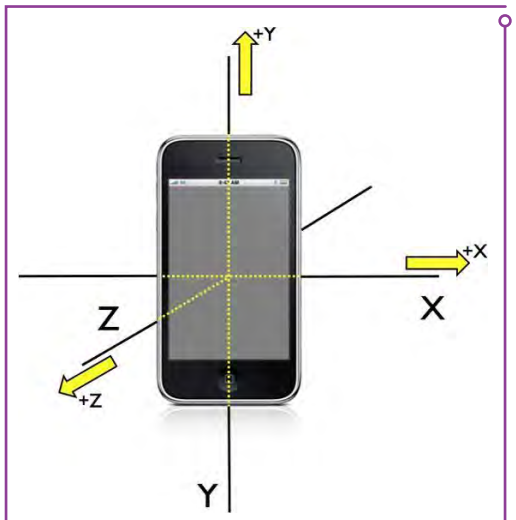
4.2.12. Sensör Uygulamaları

a) Tilt (Eğim) Sensörü

Bu uygulamada telefonun eğim ölçme özelliğini kullanılarak eğimin yönü saptanacaktır. Eğim için kullanılan koordinat düzlemi üzerinde X, Y ve Z sayısal değerlerine göre yönler yorumlanacaktır.

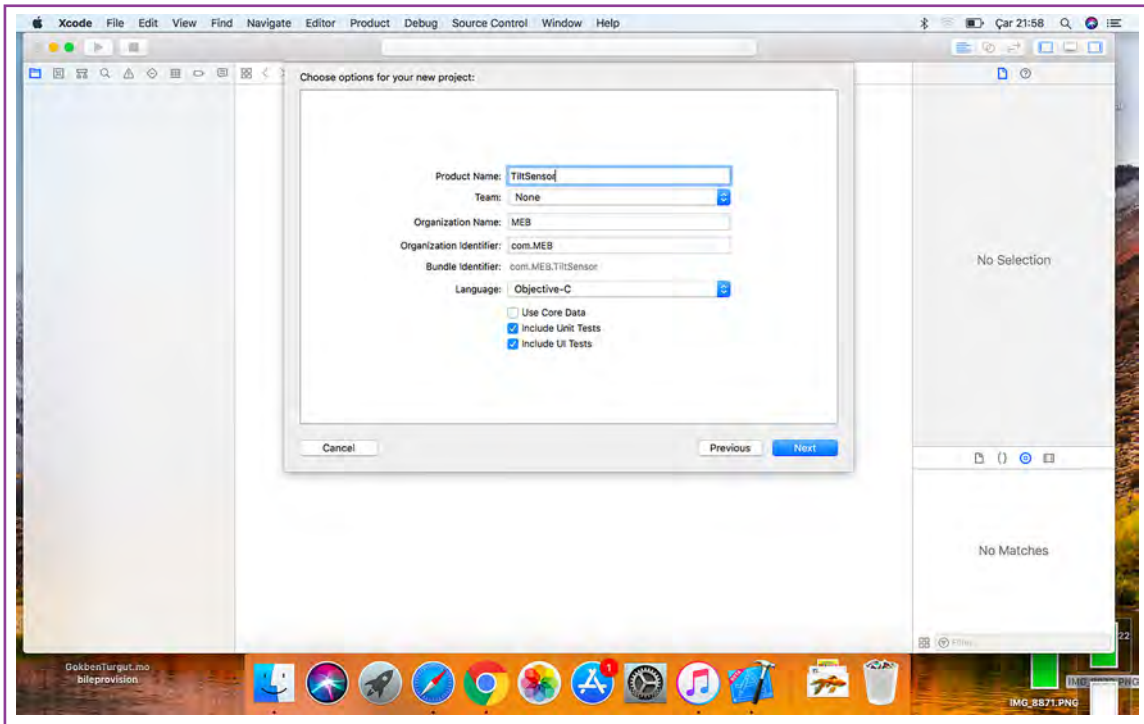
Yandaki şekle göre:

- X değeri 0'dan büyük ise telefonun sağ tarafa, 0'dan küçükse sol tarafa döndürüldüğü yorumlanır.
- Y değeri 0'dan büyük ise telefonun yukarı tarafa, 0'dan küçükse aşağı tarafa döndürüldüğü yorumlanır.
- Z değeri 0'dan büyük ise telefonun ön tarafa, 0'dan küçükse arka tarafa döndürüldüğü yorumlanır.

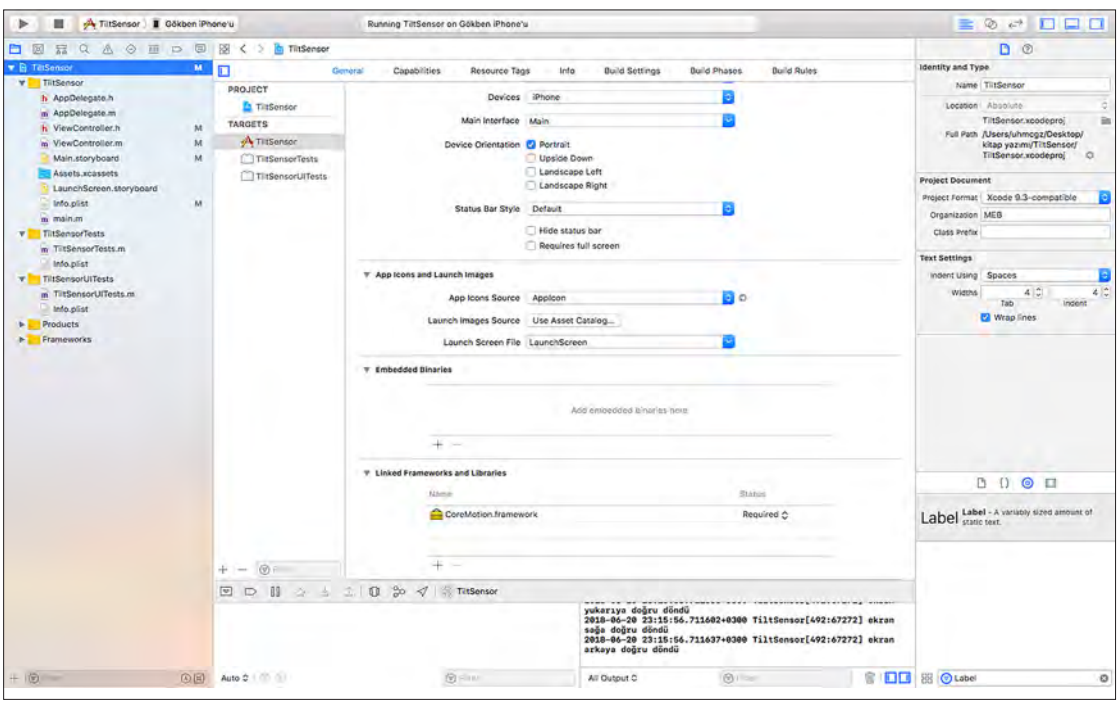


Resim 4.1

TiltSensor isimli bir proje açılarak CoreMotion framework'ü projeye yüklenir. Genel ayarlar kısmında projenin sadece tek bir yöne kilitlenmesi sağlanır. Aksi takdirde telefon döndürülürken ekran da beraberinde döneceği için uygulama istendiği gibi çalışmayabilir. Bu uygulamada Portrait modunda seçilmiştir.



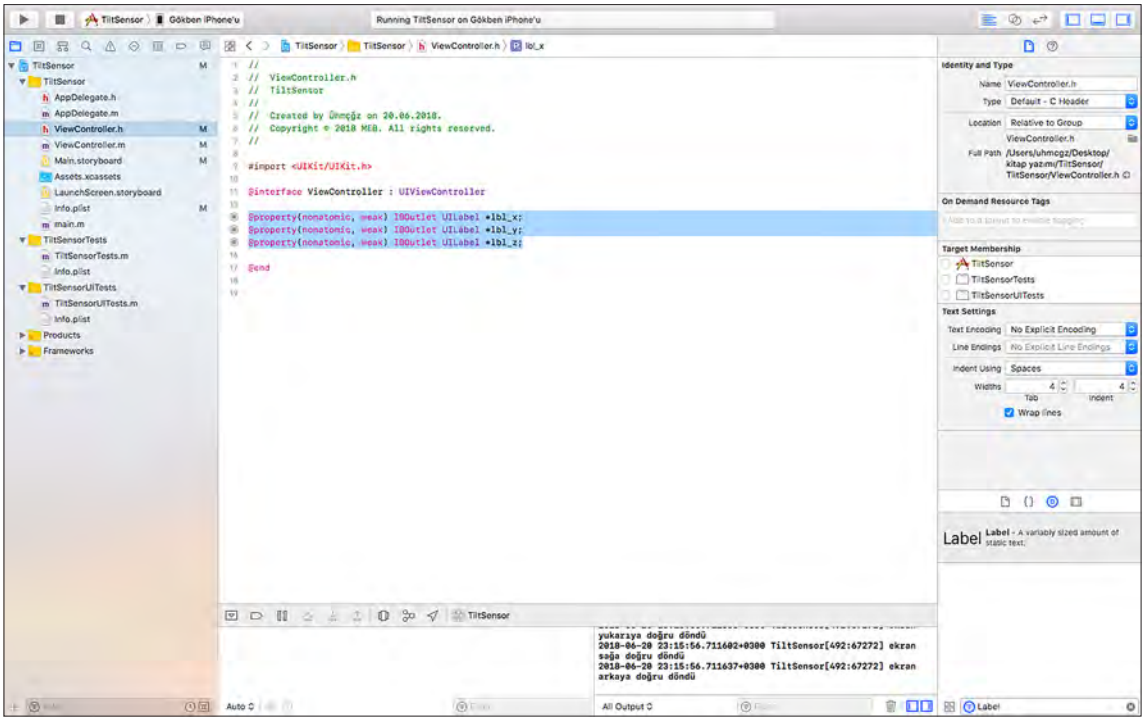
Ekran Görüntüsü 4.145



Ekran Görüntüsü 4.146

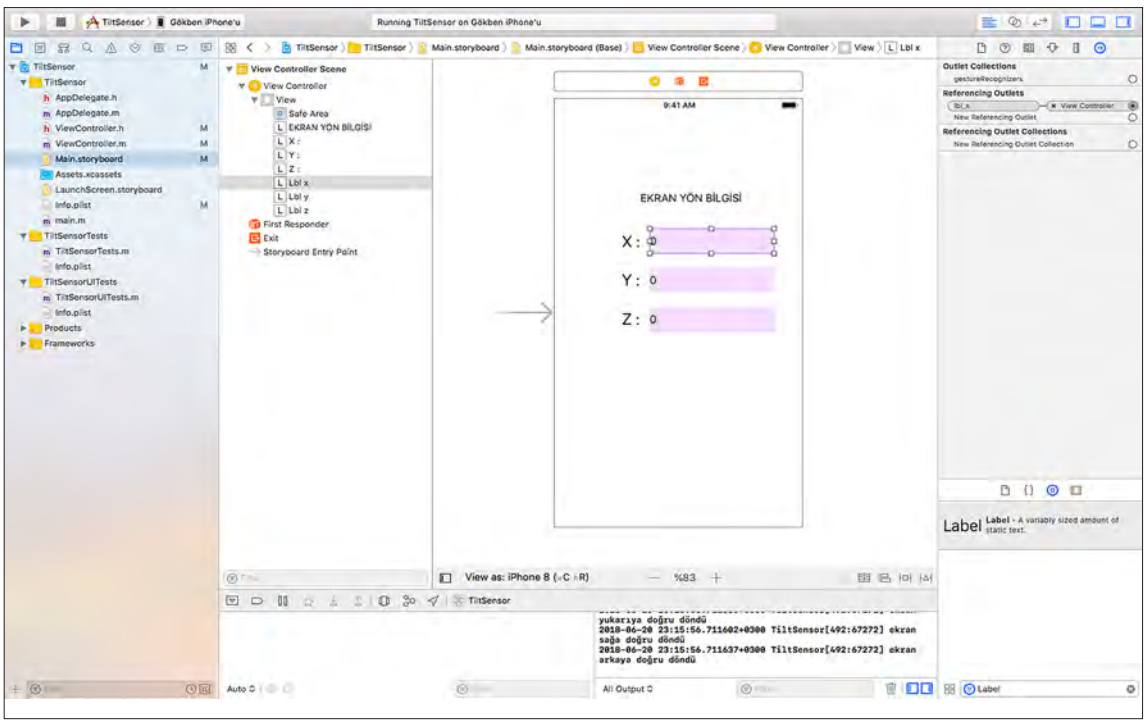
ViewController.h dosyası açılarak ekranda telefonun döndüğü yönleri gösteren Label'lar tanımlanır.

```
@property(nonatomic, weak) IBOutlet UILabel *lbl_x;
@property(nonatomic, weak) IBOutlet UILabel *lbl_y;
@property(nonatomic, weak) IBOutlet UILabel *lbl_z;
```



Ekran Görüntüsü 4.147

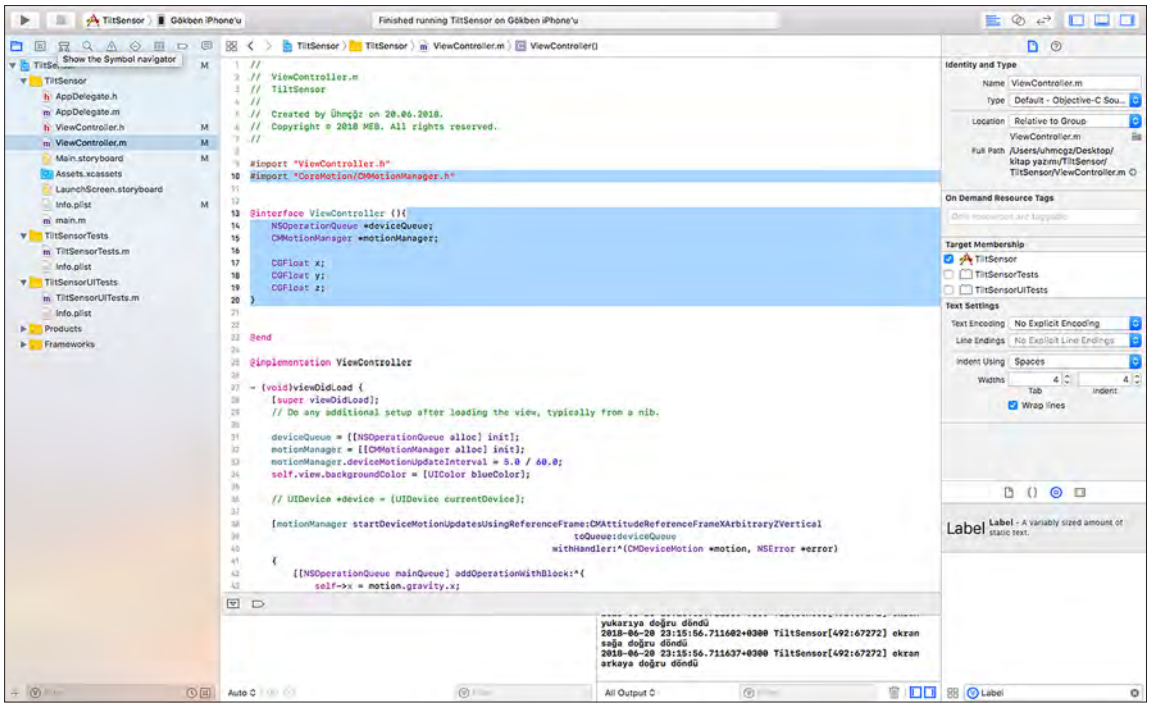
Main.storyboard ekranında Label'lar oluşturularak biçimlendirilir. ViewController.h dosyasında tanımlanan 3 adet label ismi aşağıda gri renkte görünen label'lar ile ilişkilendirilir. Bu işlem için önce label ekranda seçilir, sağ bölmede References sekmesinde Referencing Outlets alanındaki yuvarlak şekil tutup sürüklenerek sol bölümdeki View Controller üzerine bırakılır ve açılan küçük listeden label'a ait isim seçilir. Diğer 2 Label için de aynı işlemler sırasıyla tekrarlanır. Bu 3 Label'in tanımlanmıştır çünkü kodlama ekranında bu Label'ların içindeki metinlerin ve renkler değiştirilecektir. Diğer Label'lar uygulama çalışırken hiç bir değişikliğe uğramayacaktır.



Ekran Görüntüsü 4.148

ViewController.m dosyasına indirilen CoreMotion kütüphanesinden kullanılacak dosya bu sayfaya indirilir.

```
#import "CoreMotion/CMMotionManager.h"
Program içerisinde kullanılacak değişkenler dosyanın ilgili kısmında tanımlanır.

@interface ViewController (){
NSOperationQueue *deviceQueue;
CMMotionManager *motionManager;

CGFloat x;
CGFloat y;
CGFloat z;
}

@end
```

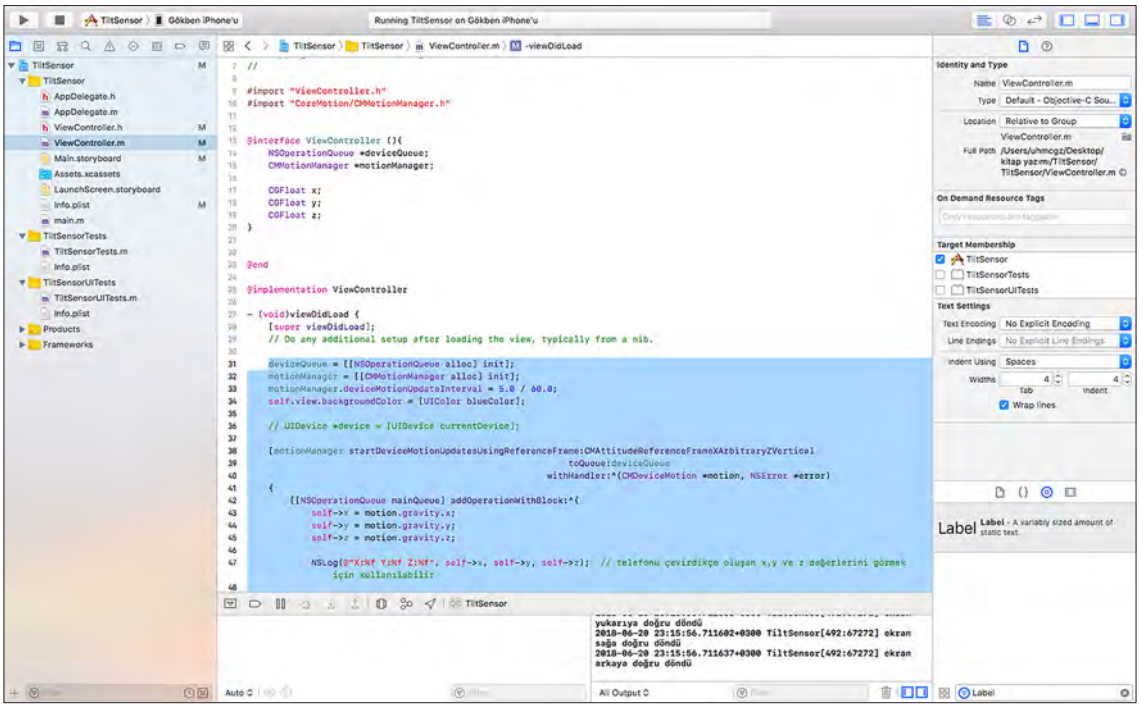


Ekran Görüntüsü 4.149

View ekrana yüklendiğinde çalışacak kodlar yazılır.

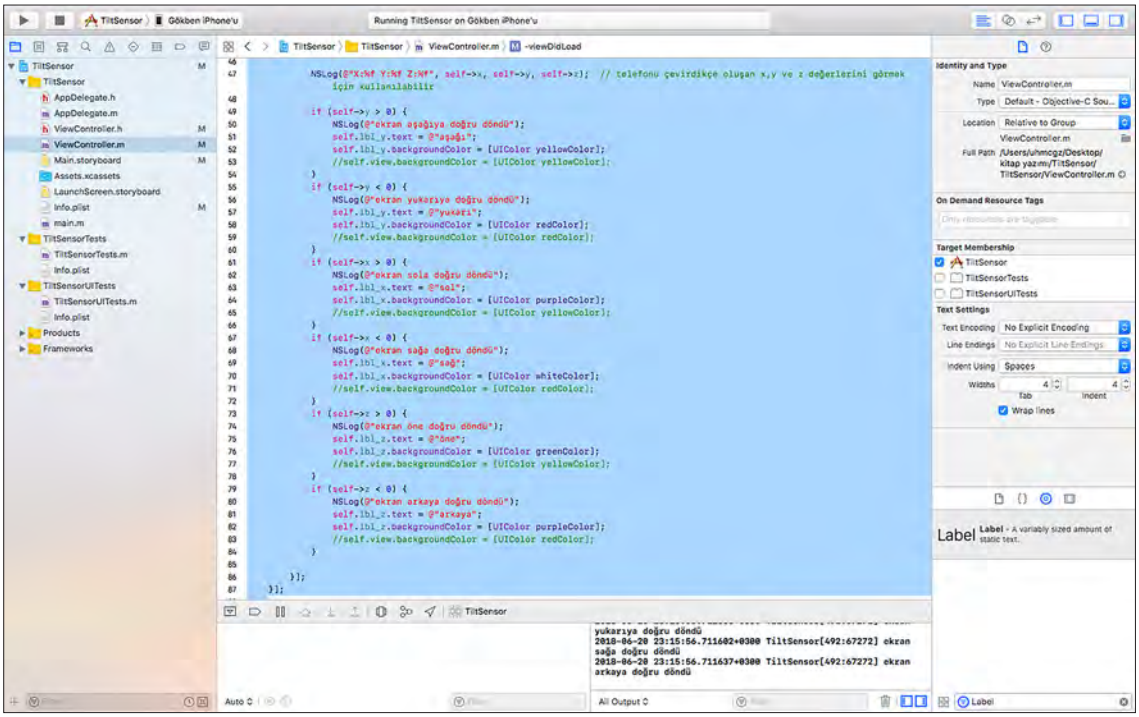
```
- (void)viewDidLoad {
[super viewDidLoad];

deviceQueue = [[NSOperationQueue alloc] init];
motionManager = [[CMMotionManager alloc] init];
motionManager.deviceMotionUpdateInterval = 5.0 / 60.0;
self.view.backgroundColor = [UIColor blueColor];


[motionManager startDeviceMotionUpdatesUsingReferenceFrame:CMAttitudeReferenceFrameXArbitraryZVertical
                                                   toQueue:deviceQueue
                                                 withHandler:^(CMDeviceMotion *motion, NSError *error)
  {
          [[NSOperationQueue mainQueue] addOperationWithBlock:^{
          self->x = motion.gravity.x;
          self->y = motion.gravity.y;
          self->z = motion.gravity.z;

      NSLog(@"X:%f Y:%f Z:%f", self->x, self->y, self->z);  // telefonu çevirdikçe oluşan x,y ve z değerlerini kod penceresinden görmek için kullanılabilir
```

```
            if (self->y > 0) {
            NSLog(@"ekran aşağıya doğru döndü");
            self.lbl_y.text = @"aşağı";
            self.lbl_y.backgroundColor = [UIColor yellowColor];
            }

if (self->y < 0) {
            NSLog(@"ekran yukarıya doğru döndü");
            self.lbl_y.text = @"yukarı";
            self.lbl_y.backgroundColor = [UIColor redColor];
            }

            if (self->x > 0) {
            NSLog(@"ekran sola doğru döndü");
             self.lbl_x.text = @"sol";
            self.lbl_x.backgroundColor = [UIColor purpleColor];
            }

            if (self->x < 0) {
            NSLog(@"ekran sağa doğru döndü");
            self.lbl_x.text = @"sağ";
            self.lbl_x.backgroundColor = [UIColor whiteColor];
            }

            if (self->z > 0) {
            NSLog(@"ekran öne doğru döndü");
            self.lbl_z.text = @"öne";
            self.lbl_z.backgroundColor = [UIColor greenColor];
            }

            if (self->z < 0) {
            NSLog(@"ekran arkaya doğru döndü");
            self.lbl_z.text = @"arkaya";
            self.lbl_z.backgroundColor = [UIColor purpleColor];
            }

      }];
   }];

   }
```



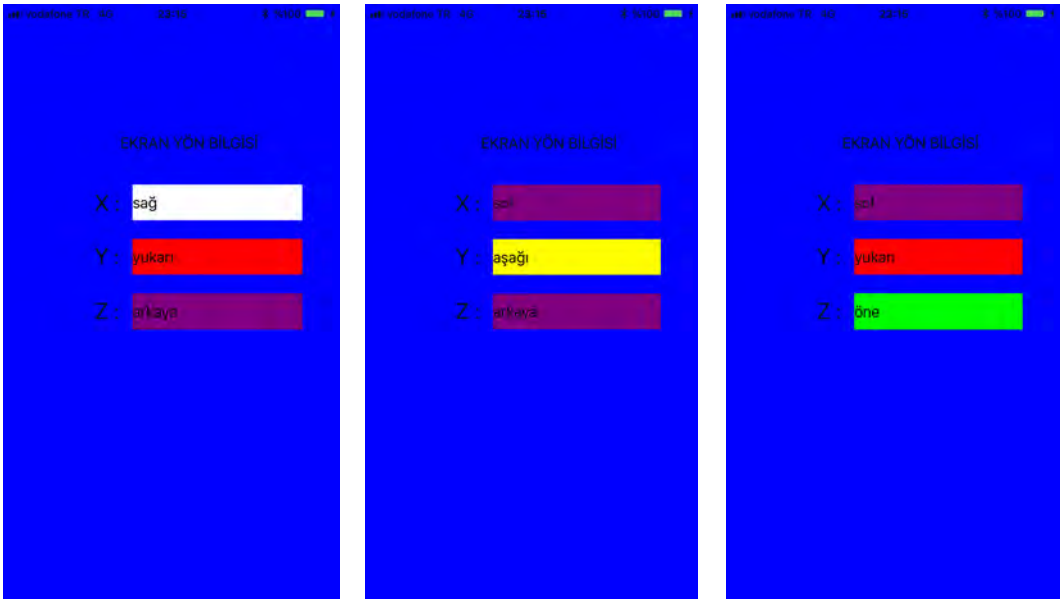
Ekran Görüntüsü 4.150

Elde edilen X,Y, Z değerlerine göre ekranda bulunan Label'ların rengi ve yön bilgisi metin şeklinde değiştirilir.



Ekran Görüntüsü 4.151

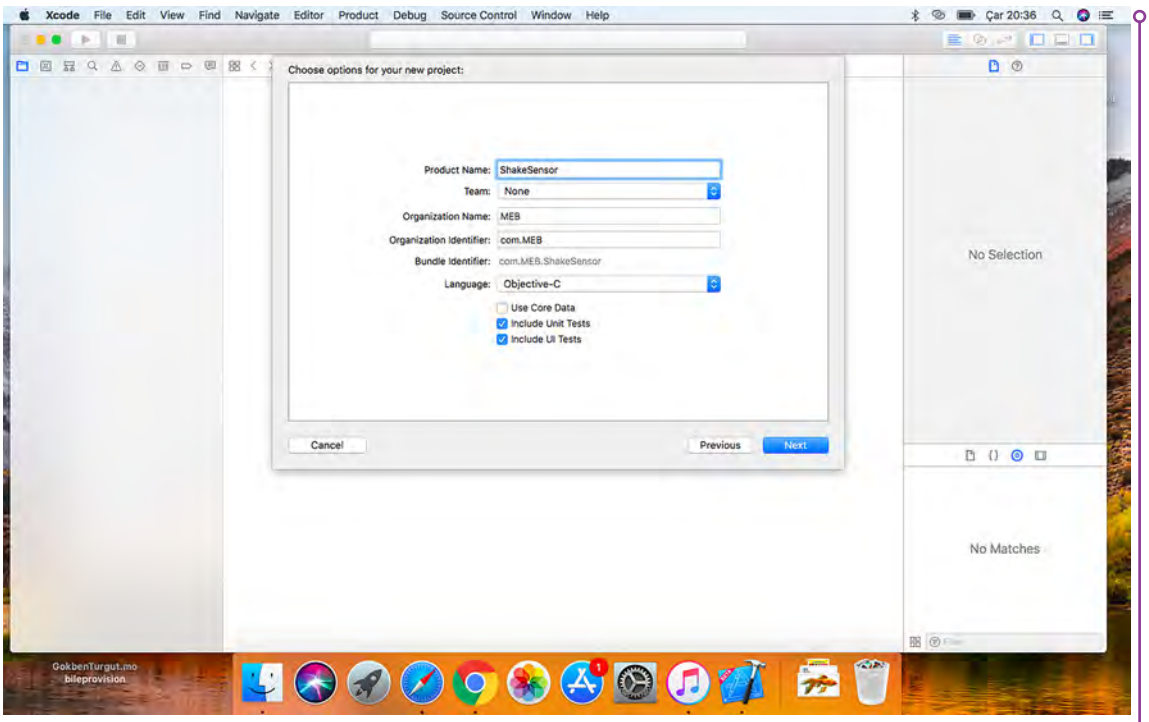
Uygulama iPhone cihazında çalıştırılarak cihaz sağa, sola, öne, arkaya, aşağı, yukarı yönlerinde çevrildiğinde ekran görüntülerinden birkaçı aşağıdaki gibi olacaktır.



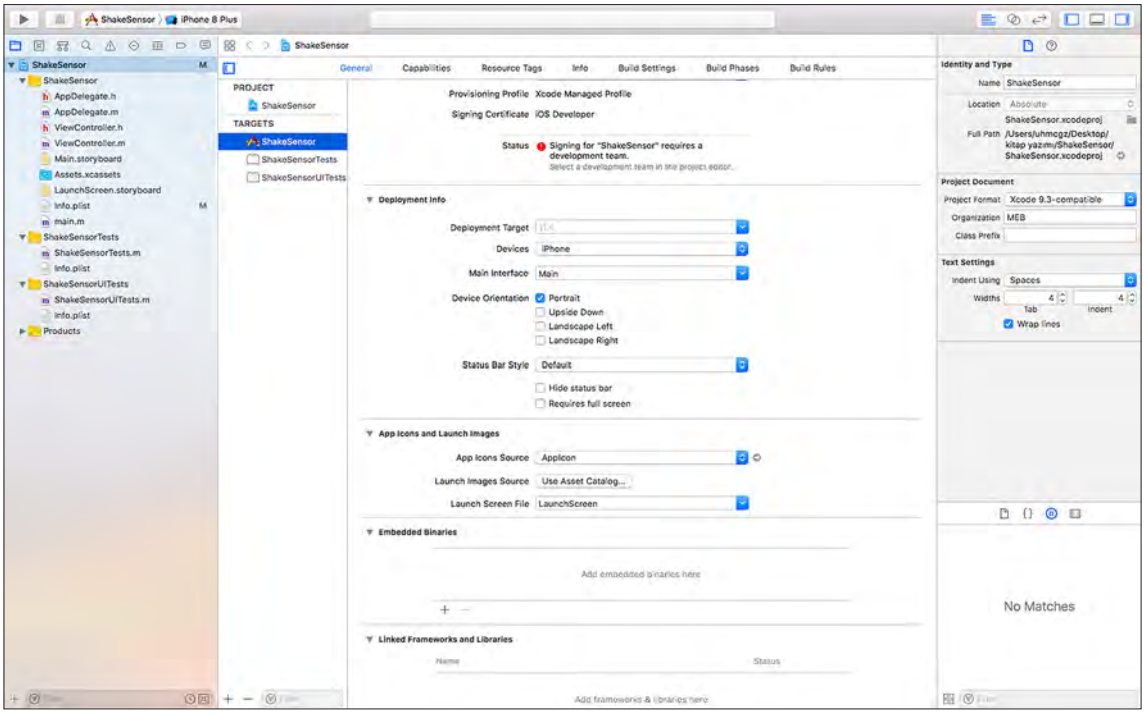
Ekran Görüntüsü 4.152

b) Shake (Sallama) Sensörü

Bu uygulamada kullanıcı telefonu salladığında ekran yeşil renge dönecek ve ekranda "Renk Değiştir" butonu belirecektir. Kullanıcı butona tıkladığında ekran tekrar beyaz renge dönecek ve sallama hareketi tekrar yapılırsa ekran tekrar yeşil rengi alacaktır.



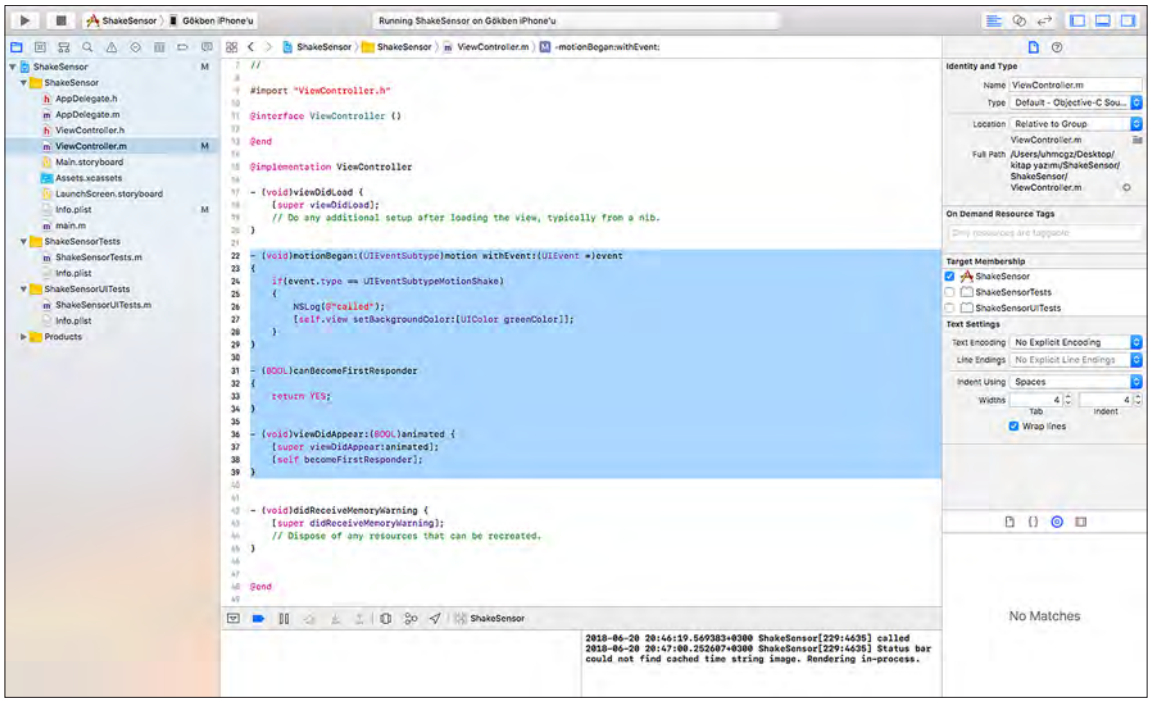
Ekran Görüntüsü 4.153



Ekran Görüntüsü 4.154

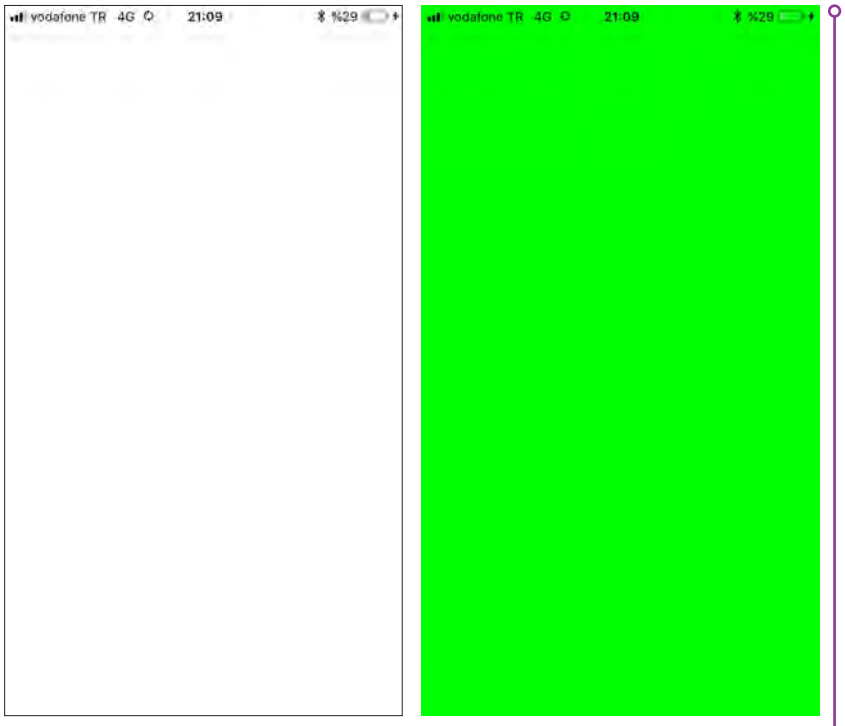
ViewController.m dosyasında aşağıdaki kodlar yazılır:

```
- (void)motionBegan:(UIEventSubtype)motion withEvent:(UIEvent *)event
{
if(event.type == UIEventSubtypeMotionShake){
NSLog(@"sallandı");
[self.view setBackgroundColor:[UIColor greenColor]];     // Ekran rengini yeşile çevirme
}
}

- (BOOL)canBecomeFirstResponder
{
return YES;
}

- (void)viewDidAppear:(BOOL)animated {
[super viewDidAppear:animated];
[self becomeFirstResponder];
}
```



Ekran Görüntüsü 4.155

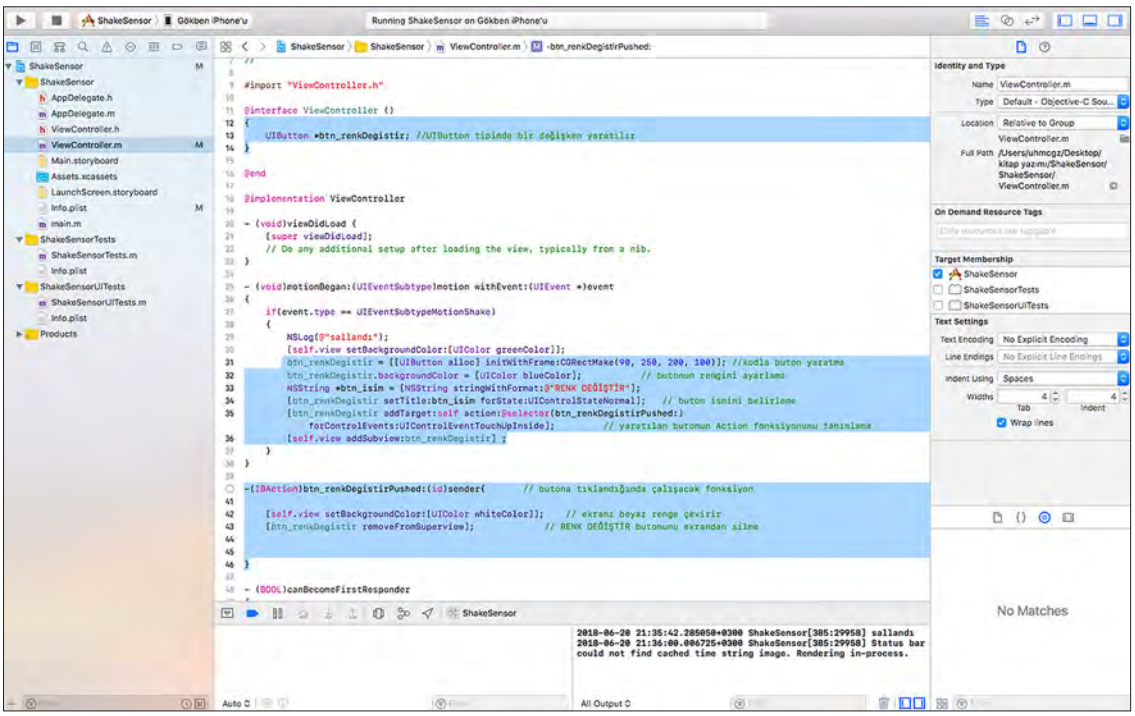
Uygulama iPhone cihazında çalıştırıldığında (Ekran Görüntüsü 4.156 - Soldaki Resim) ve telefon sallandığında (Ekran Görüntüsü 4.156 - Sağdaki Resim) ekran görüntüsü aşağıdaki gibi olacaktır.



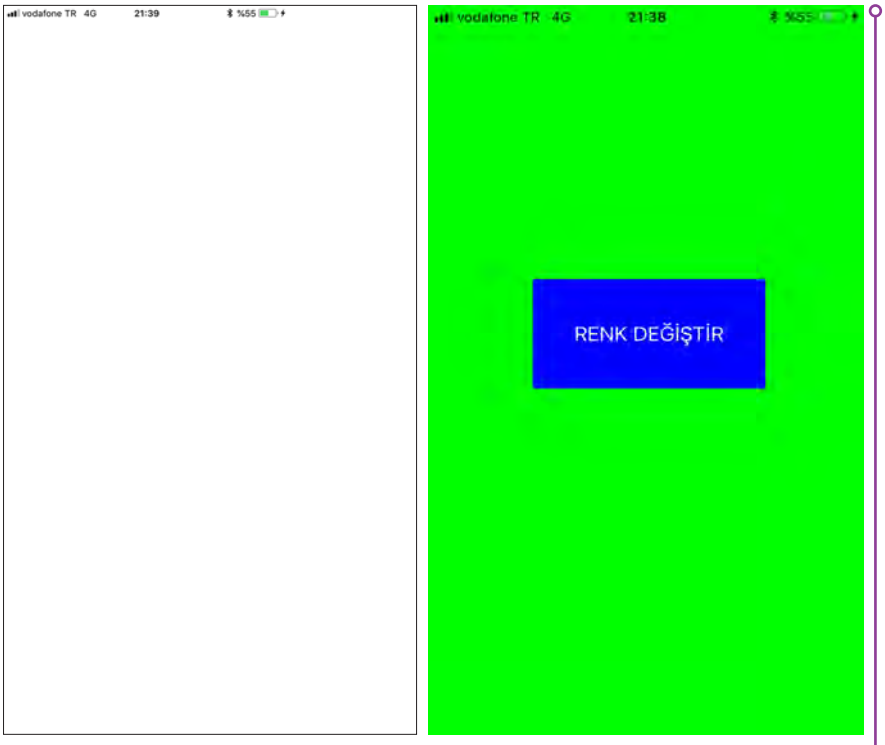
Ekran Görüntüsü 4.156

Ekranı tekrar eski haline döndürmek için aşağıdaki kodlar eklenir:

```
@interface ViewController ()
{
UIButton *btn_renkDegistir; //UIButton tipinde bir değişken yaratılır
}

- (void)motionBegan:(UIEventSubtype)motion withEvent:(UIEvent *)event
{
if(event.type == UIEventSubtypeMotionShake){
NSLog(@"sallandı");
[self.view setBackgroundColor:[UIColor greenColor]];
btn_renkDegistir = [[UIButton alloc] initWithFrame:CGRectMake(90, 250, 200, 100)]; //kodla buton yaratma
btn_renkDegistir.backgroundColor = [UIColor blueColor];          // butonun rengini ayarlama
NSString *btn_isim = [NSString stringWithFormat:@"RENK DEĞİŞTİR"];
[btn_renkDegistir setTitle:btn_isim forState:UIControlStateNormal];   // buton ismini belirleme
[btn_renkDegistir addTarget:self action:@selector(btn_renkDegistirPushed:) forControlEvents:UIControlEventTouc-
hUpInside];        // yaratılan butonun Action fonksiyonunu tanımlama
                        [self.view addSubview:btn_renkDegistir] ;
            }
}

-(IBAction)btn_renkDegistirPushed:(id)sender{       // butona tıklandığında çalışacak fonksiyon

[self.view setBackgroundColor:[UIColor whiteColor]];    // ekranı beyaz renge çevirir
[btn_renkDegistir removeFromSuperview];           // RENK DEĞİŞTİR butonunu ekrandan silme

}
```



Ekran Görüntüsü 4.157



Ekran Görüntüsü 4.158

Uygulama iPhone cihazında çalıştırıldığında ekran görüntüsü yanaki gibi olacaktır.

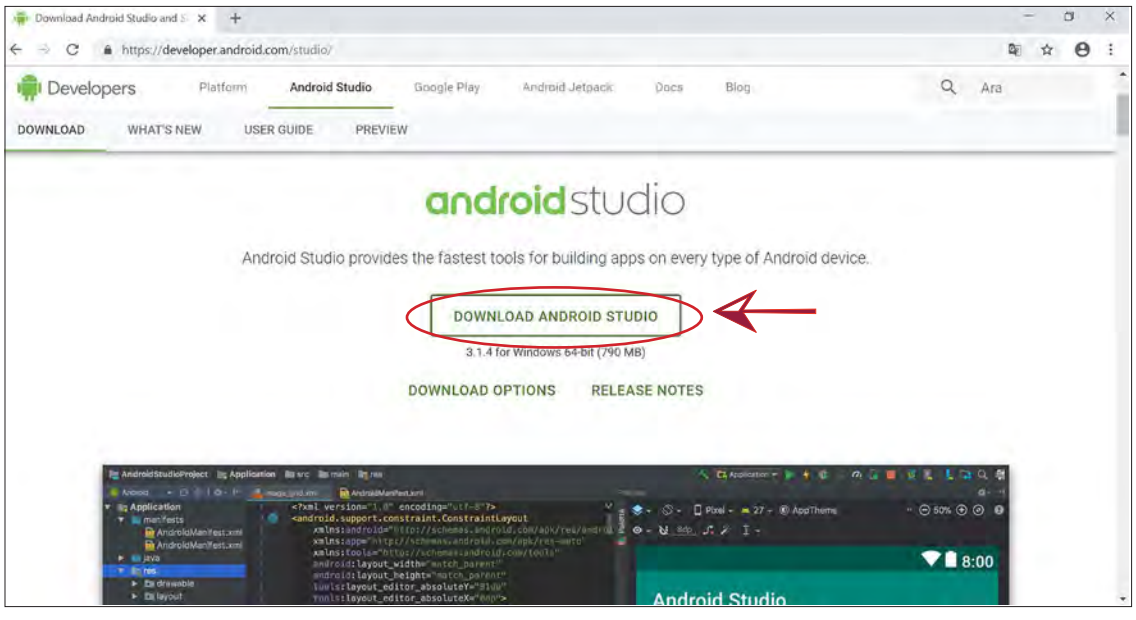
4.3. Android Studio

Android Studio masaüstü uygulaması Windows, MacOs ve Linux işletim sistemlerinde kullanılabilmektedir. Kitabımızda Windows işletim sisteminde Android Studio kurulum ve kullanımı anlatılacaktır.

4.3.1. Android Studio Programının Kurulumu

Android Studio'un kendi resmi web sayfasından indirilebilmektedir. www.developer.android.com adresinden ücretsiz olarak bilgisayarlara kurulabilmektedir. Android Studio programının bilgisayarınıza yüklenme adımları aşağıdaki gibidir:

(1) Öncelikle internet tarayıcınızın adres kısmına http://developer.android.com/studio/ adresini yazarak Android'in geliştiriciler için çeşitli hizmet ve bilgiler verdiği resmi sitesine bağlanın.

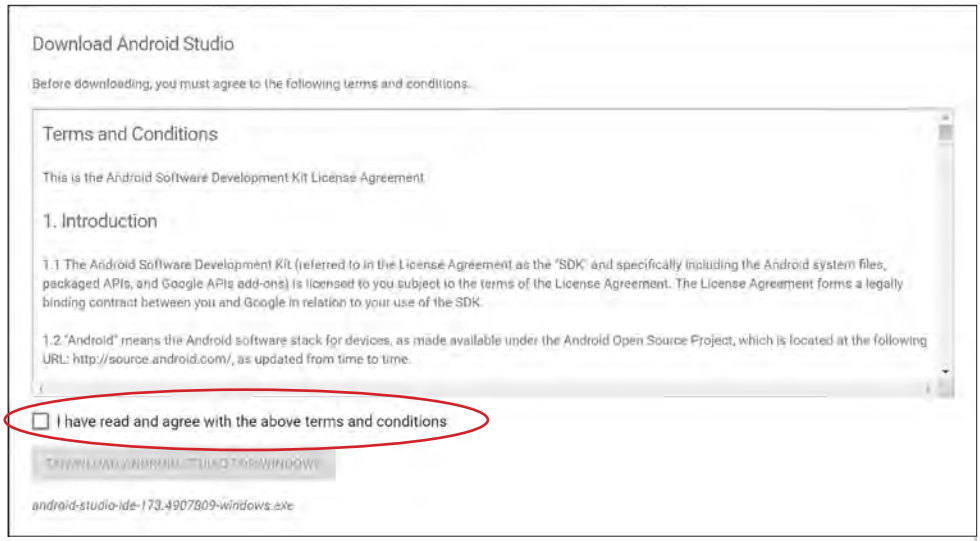


Ekran Görüntüsü 4.159

(2) İndirme işlemi tamamlandıktan sonra kurulum dosyası İndirilenler klasöründen çift tıklanarak, kurulum başlatılır.

(3) Daha sonraki ekranda yazılım sözleşmesinin "I Agree" düğmesine basarak onaylanması gerekmektedir.

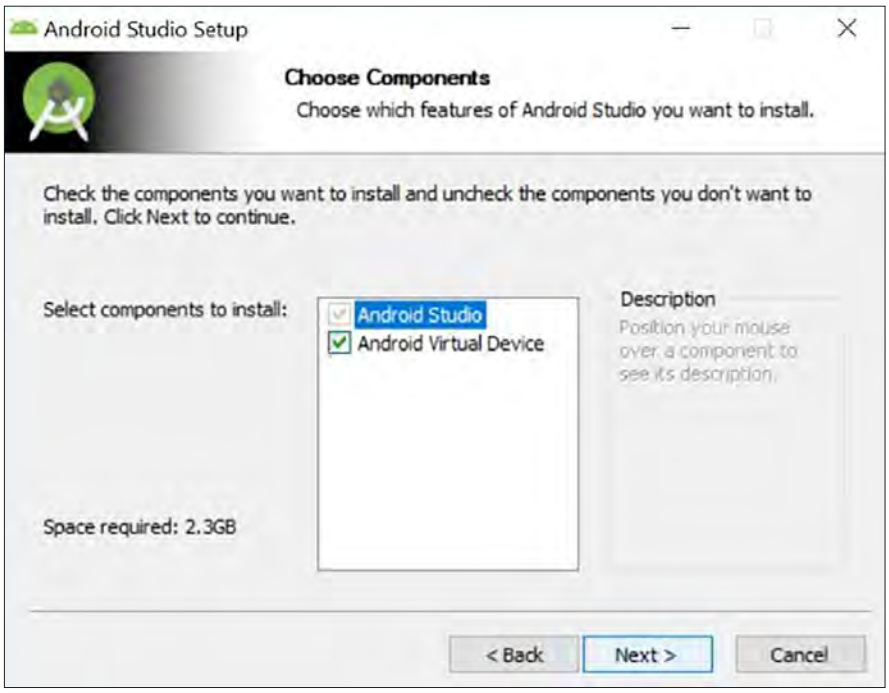
(4) Ardından "Next" düğmesine basarak kurulumu başlatınız.



Ekran Görüntüsü 4.160

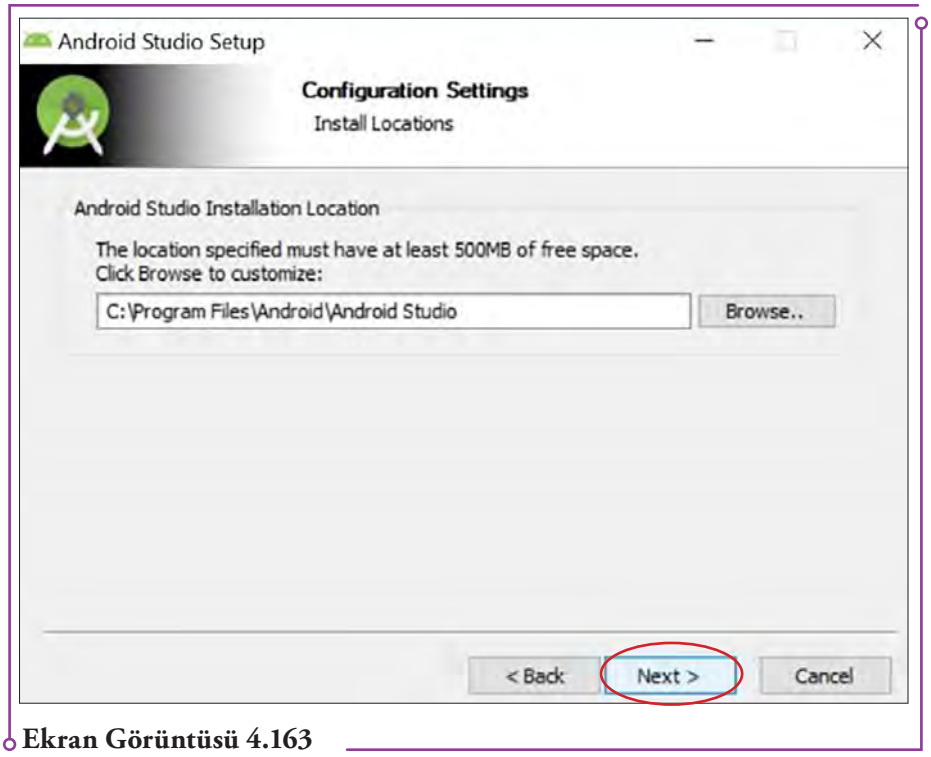
Ekran Görüntüsü 4.161

(5) Aşağıdaki ekranda, seçeneklerin ikisinin de seçili olması gerekmektedir. Android Virtual Device seçeneği sanal cihazın (simülatör) bilgisayarımıza kurulmasını sağlamaktadır.



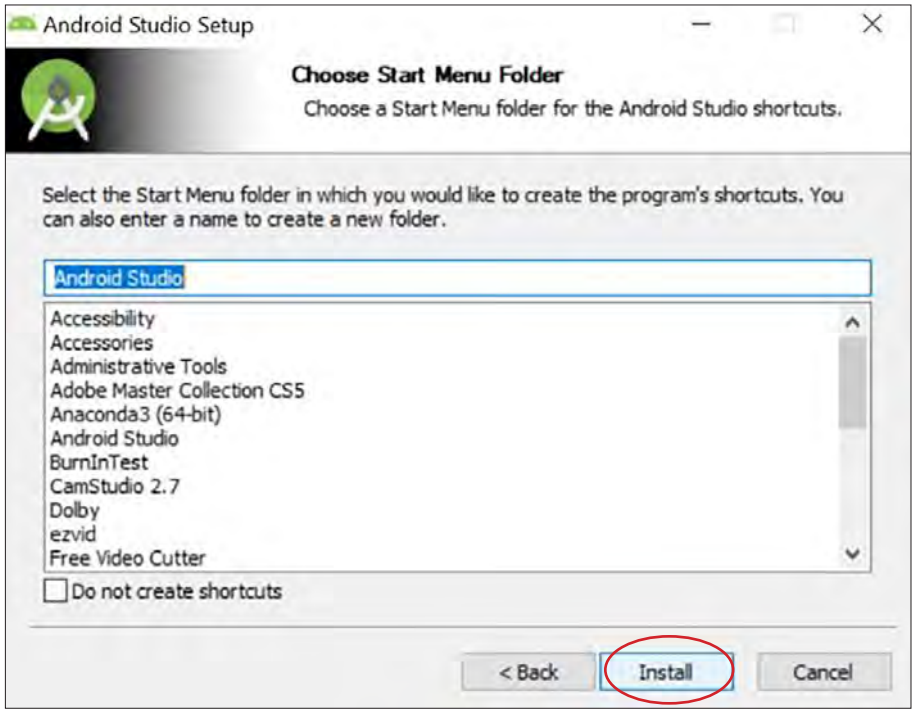
Ekran Görüntüsü 4.162

(6) Programın kurulması için en az 500 mb alana ihtiyaç vardır. Programın kurulacağı alanı belirleyip "Next" düğmesine basınız.

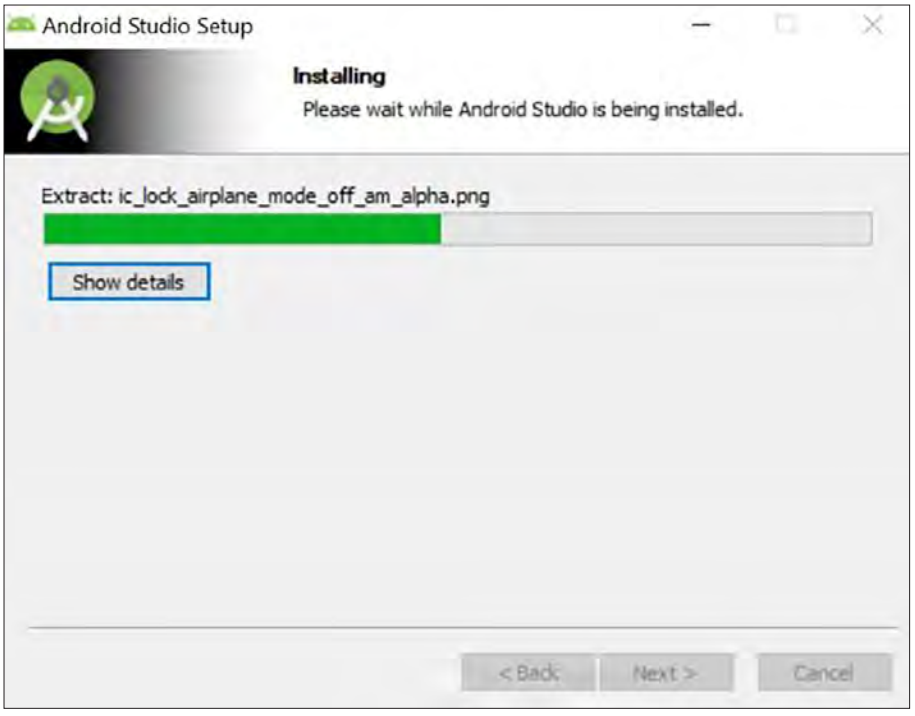


Ekran Görüntüsü 4.163

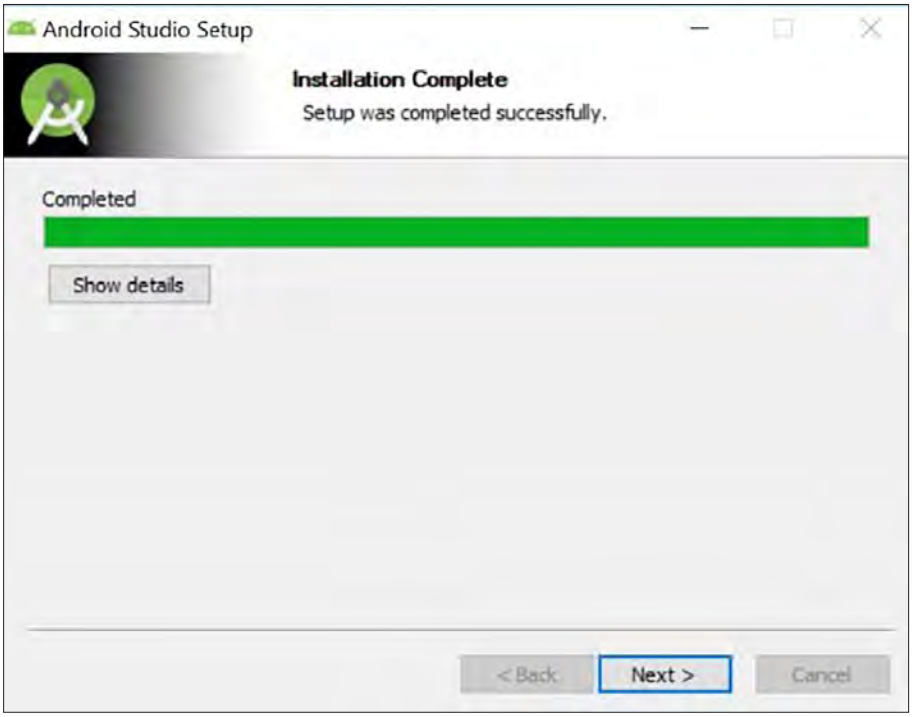
(7) “Install” düğmesine basarak kuruluma devam ediniz.



Ekran Görüntüsü 4.164

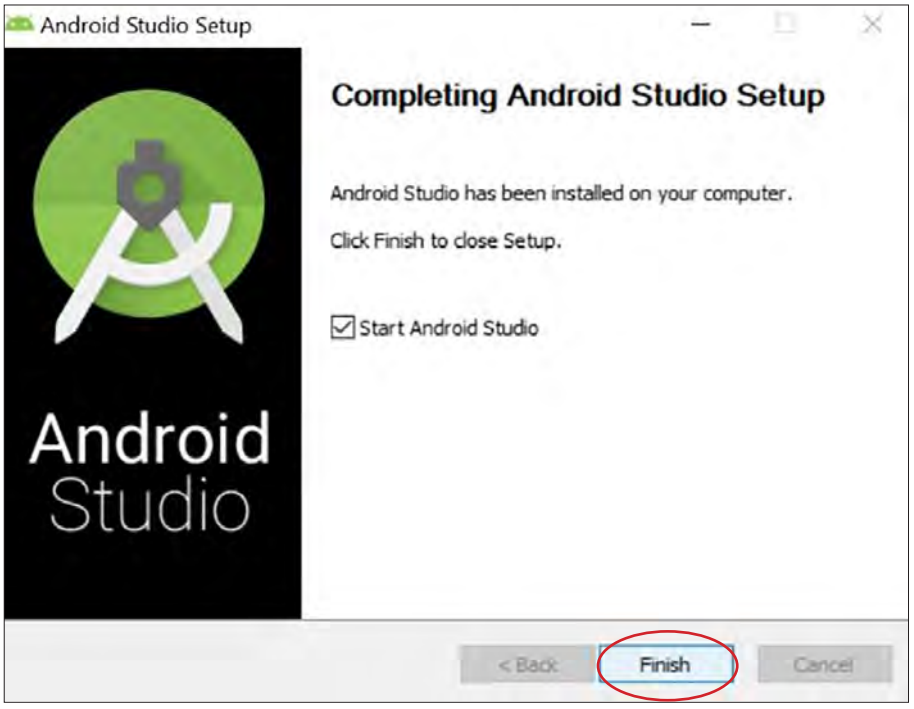


Ekran Görüntüsü 4.165



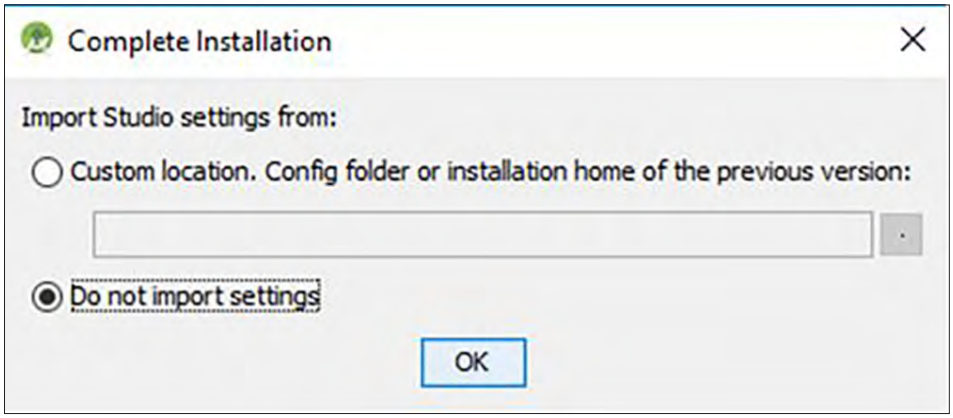
Ekran Görüntüsü 4.166

(8) Finish düğmesine basarak kurulumu sonlandırınız.

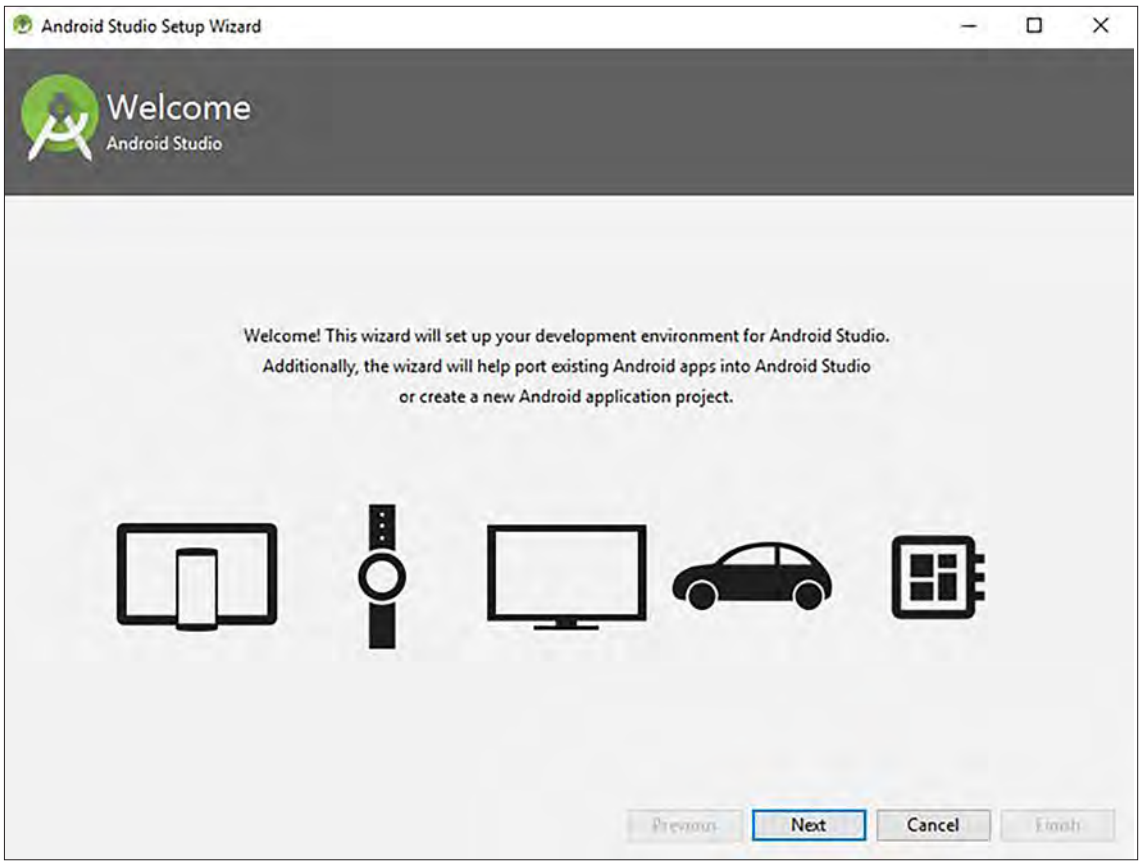


Ekran Görüntüsü 4.167

(9) Program kurulduktan sonra çalıştırdığınızda, karşınıza çıkan ekran aşağıdaki gibi olacaktır. Burada daha önce kurulmuş olan Android Studio varsa ilk seçenek, ilk kez kurulduysa ikinci seçenek seçilmelidir.

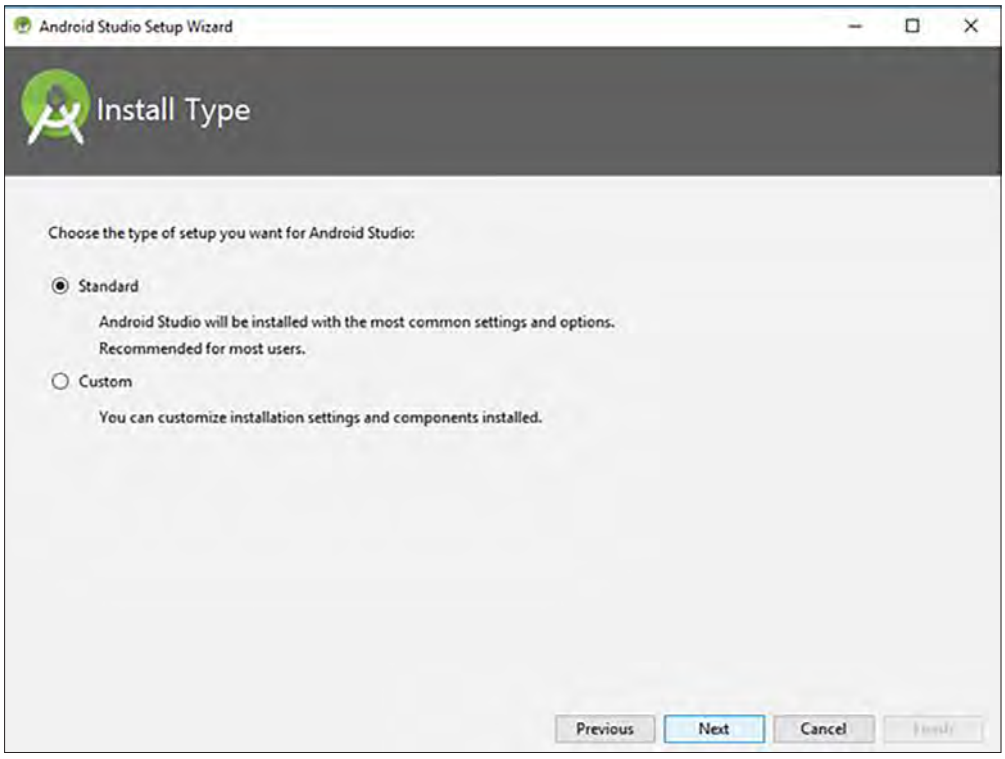


Ekran Görüntüsü 4.168



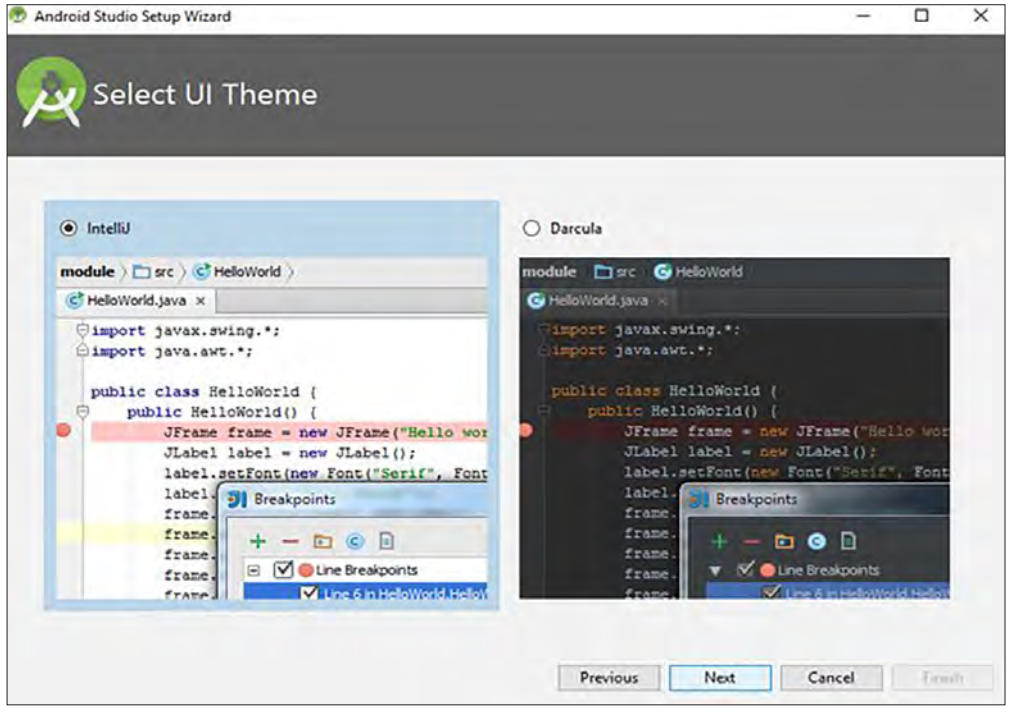
Ekran Görüntüsü 4.169

(10) Karşınıza çıkan aşağıdaki ekranda InstallType olarak Standard seçilip "Next" düğmesine basılmalıdır.

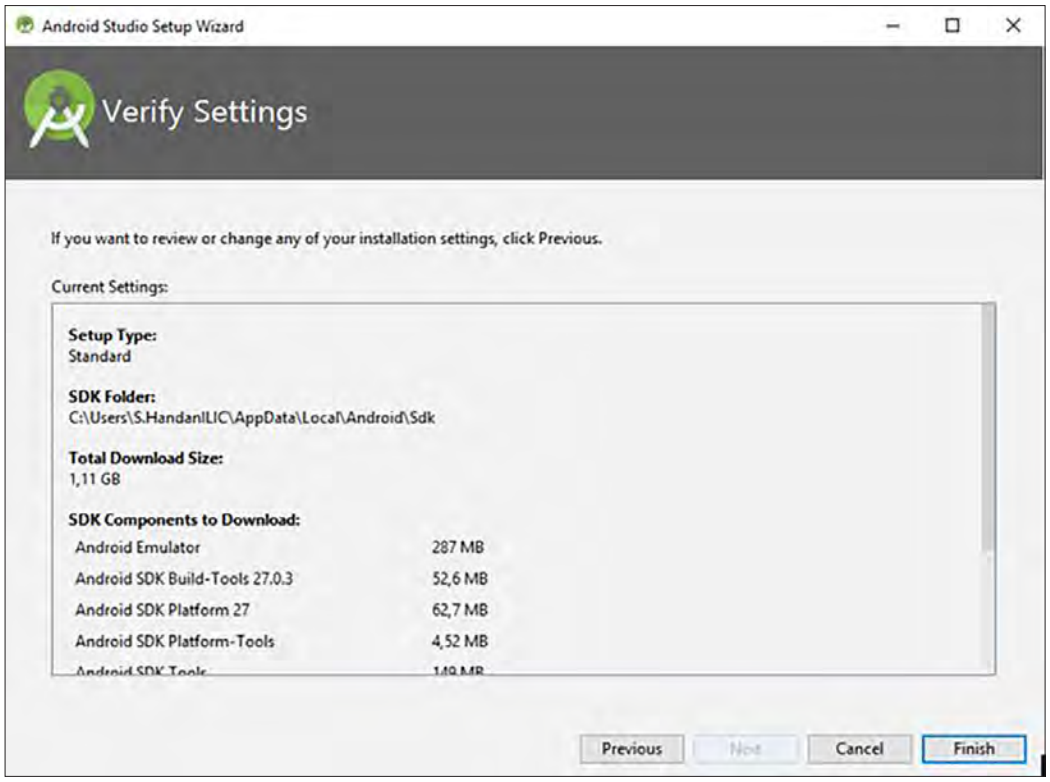


Ekran Görüntüsü 4.170

(11) Daha sonra kullanacağınız Arayüzü belirleyip devam ediniz.

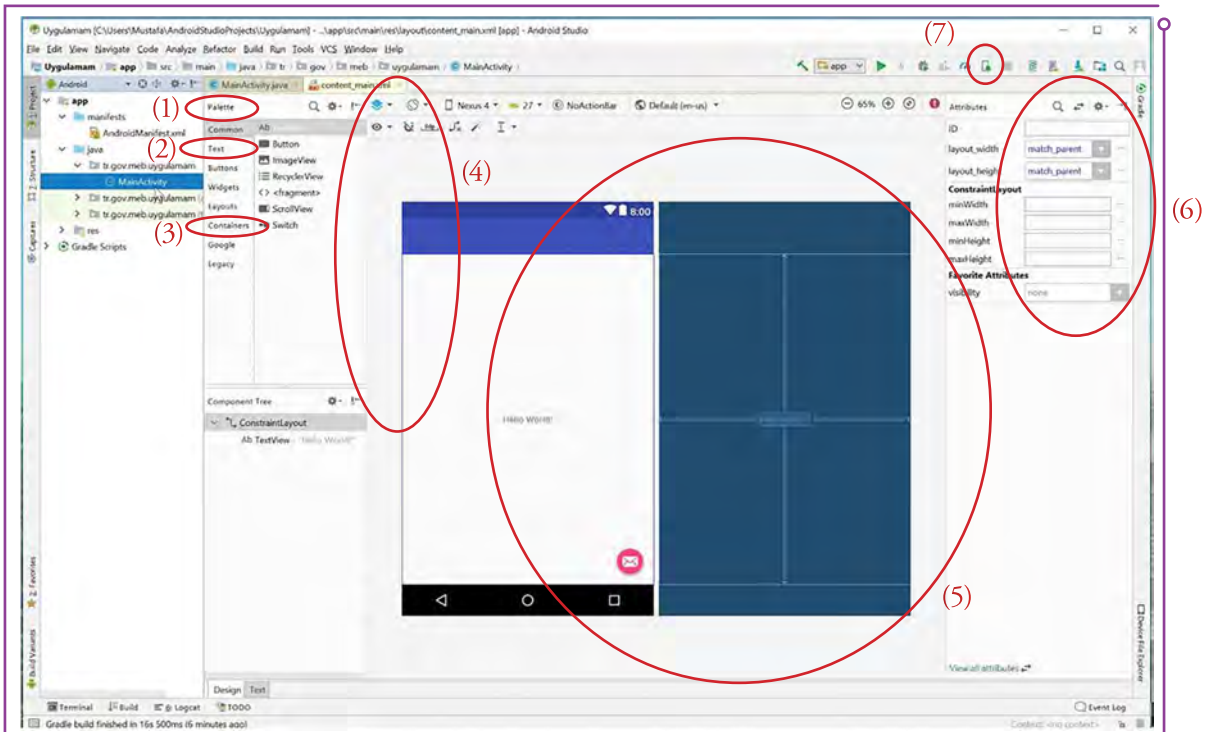


Ekran Görüntüsü 4.171



Ekran Görüntüsü 4.172

4.3.2. Android Studio Uygulama Ekranı



Ekran Görüntüsü 4.173

(1) Manifests: Android projesinin önemli parçalarından biridir ve XML uzantılıdır. Uygulamanın temel bilgileri bu dosyada yer alır. Ayrıca uygulama için gerekli olan izin istekleri bu dosya içinde oluşturulur.

(2) Java: Java kaynak kodları bu bölümde yer alır.

(3) Res: Kaynak kodların bulunduğu bölümdür. Resim dosyaları, ekran tasarımları, metin dosyaları vb. bu bölümde bulunur.

(4) Uygulama ara yüzüne veya projeye çeşitli objectler (nesneler) eklemek için kullanılan alandır.

(5) Yukarıdaki alanda seçilen dosya içeriğinin görüntülendiği alandır.

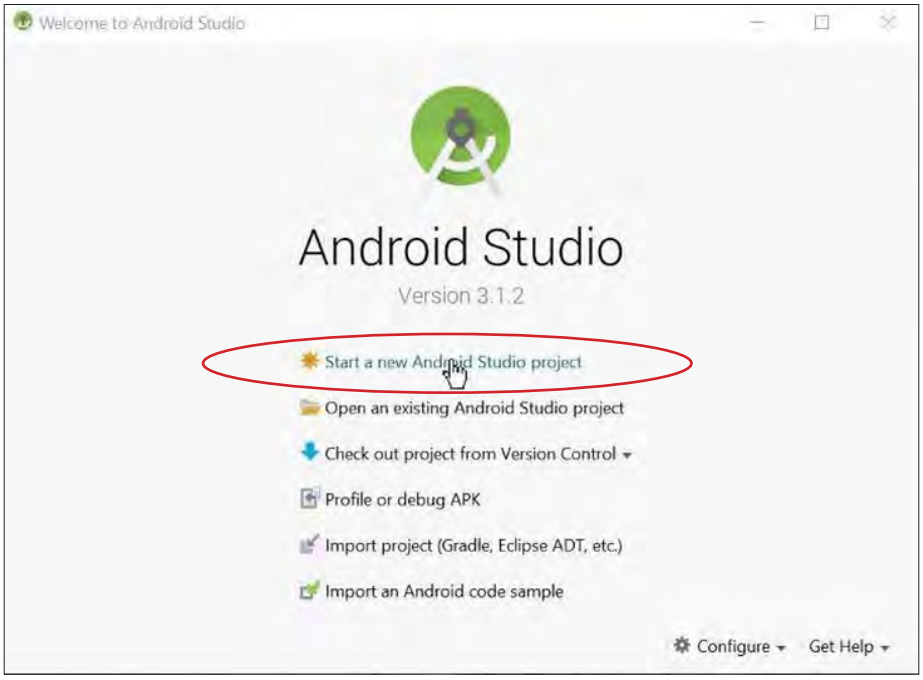
(6) Uygulama dosyalarının ve içindeki öğelerin özelliklerinin görüntülendiği ve ayarlandığı alandır.

(7) Uygulamanın emülatör veya cihaz üzerinde çalıştırılması için Çalıştır düğmesinin bulunduğu bölümdür.

4.3.3. Hello World Uygulaması ve Emülatör Çalıştırma

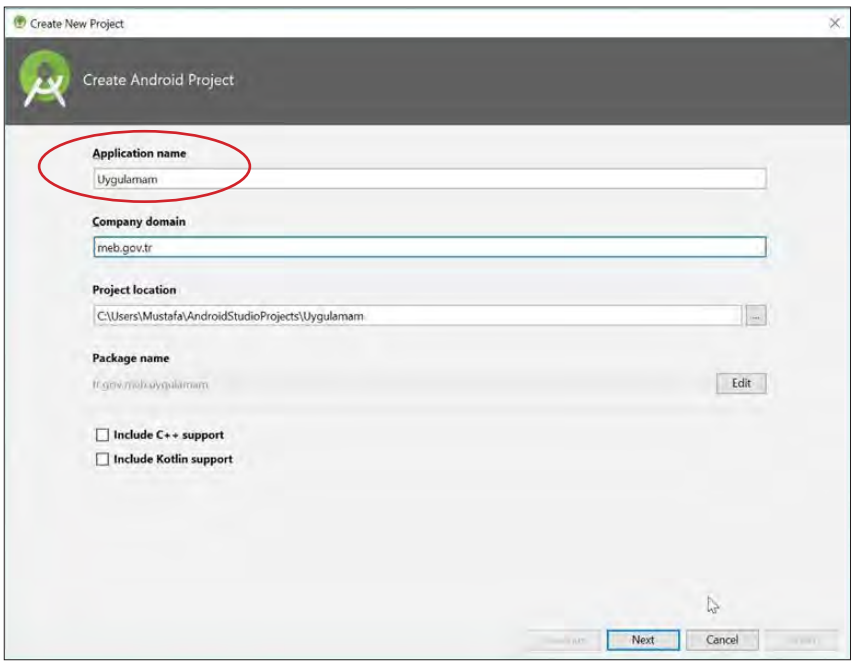
Android Studio'daki ilk uygulamada emülatör çalıştırma ve kısa text yazımı gösterilecektir.

İlk olarak Android Studio'yu başlatınız. "Start a new Android Studio project" diyerek yeni bir proje oluşturunuz.



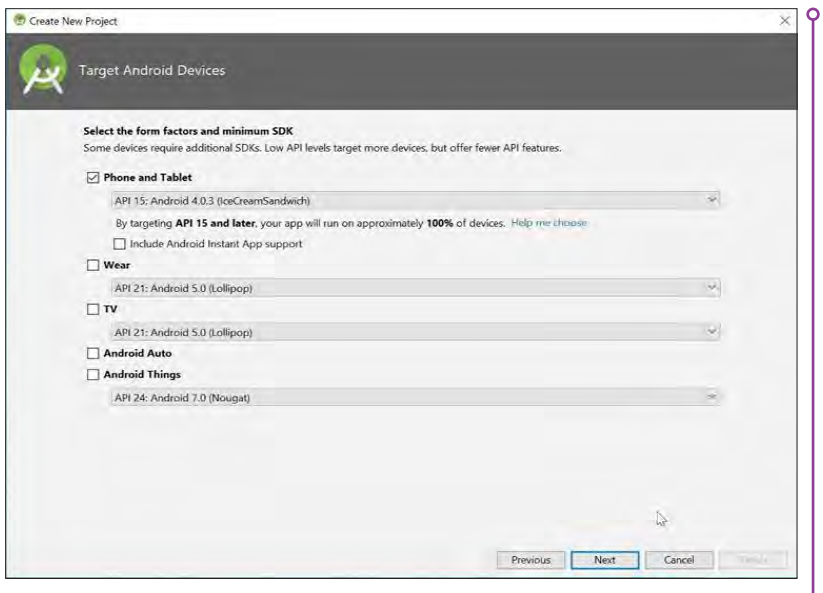
Ekran Görüntüsü 4.174

Ardından projenize Application name bölümünden isim veriniz. Burada dikkat edilmesi gereken nokta proje adındaki ilk harf büyük olmalı ve rakam ya da işaret ile başlamamalıdır. Company Domain kısmını, uygulamanızı ürettiğiniz şirket olarak düşünebilirsiniz. Company Domain girdikten sonra Package name otomatik olarak gelmektedir. Next düğmesine basıp devam ediniz.



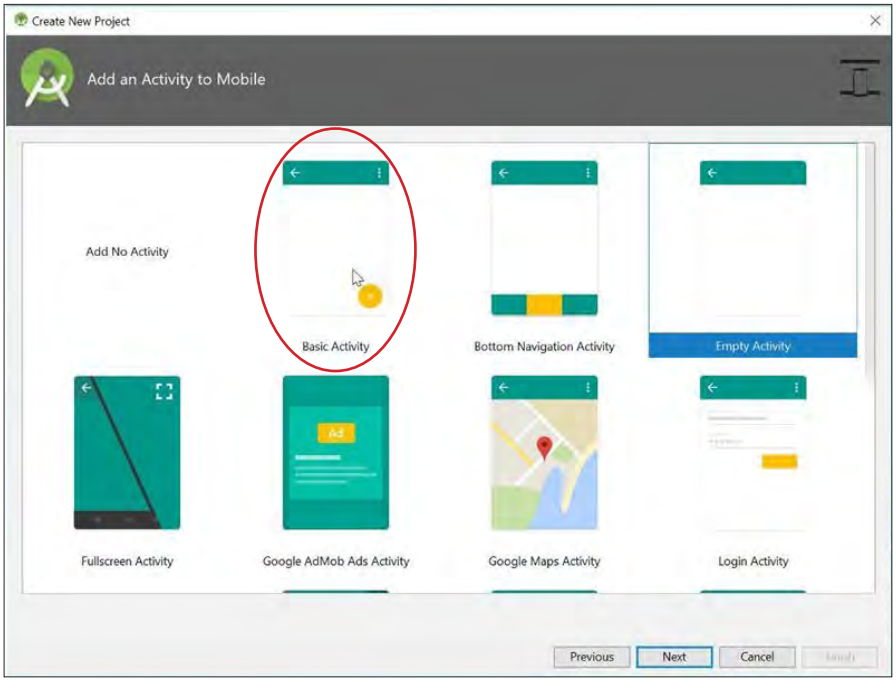
Ekran Görüntüsü 4.175

Target Android Devices bölümünde uygulamanızın görüntülenmesini istediğiniz cihazı seçiniz. Ayrıca uygulamanın çalışmasını istediğiniz en düşük Android sürümünü de bu ekranda belirlemeniz gerekmektedir. Eğer uygulamanızın akıllı saat, televizyon ya da gözlükte kullanılmasını istiyorsanız, bu ayarlamaları da yapınız.



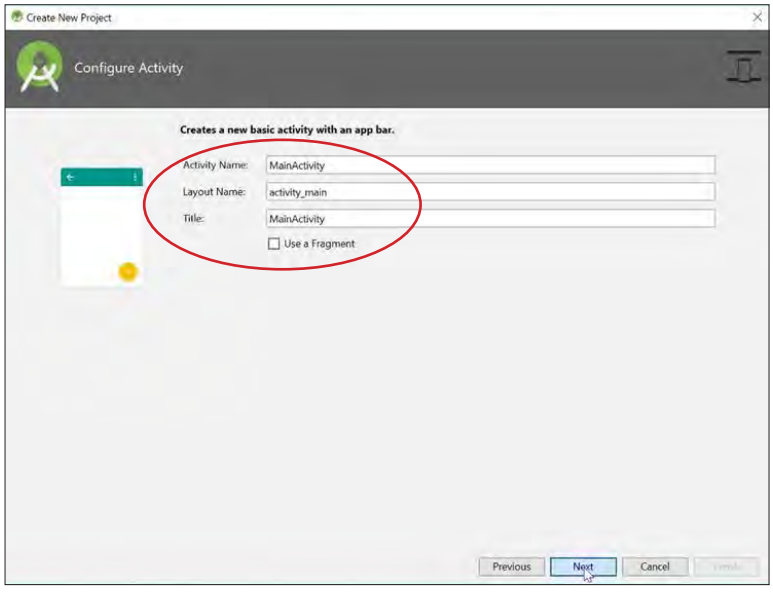
Ekran Görüntüsü 4.176

Şimdi uygulamanın görüntüleneceği tasarım ekranını (Activity) seçiniz. Yapacağınız uygulama türüne göre tasarım ekranınız değişebilmektedir. Eğer bir harita uygulaması yapacaksanız Google Maps Activity, kullanıcı girişli bir uygulama yapacaksanız Login Activity vb. gibi uygulama türünüze göre activity ekranı seçebilirsiniz. Bu uygulamada Basic Activity seçip Next düğmesine basınız.



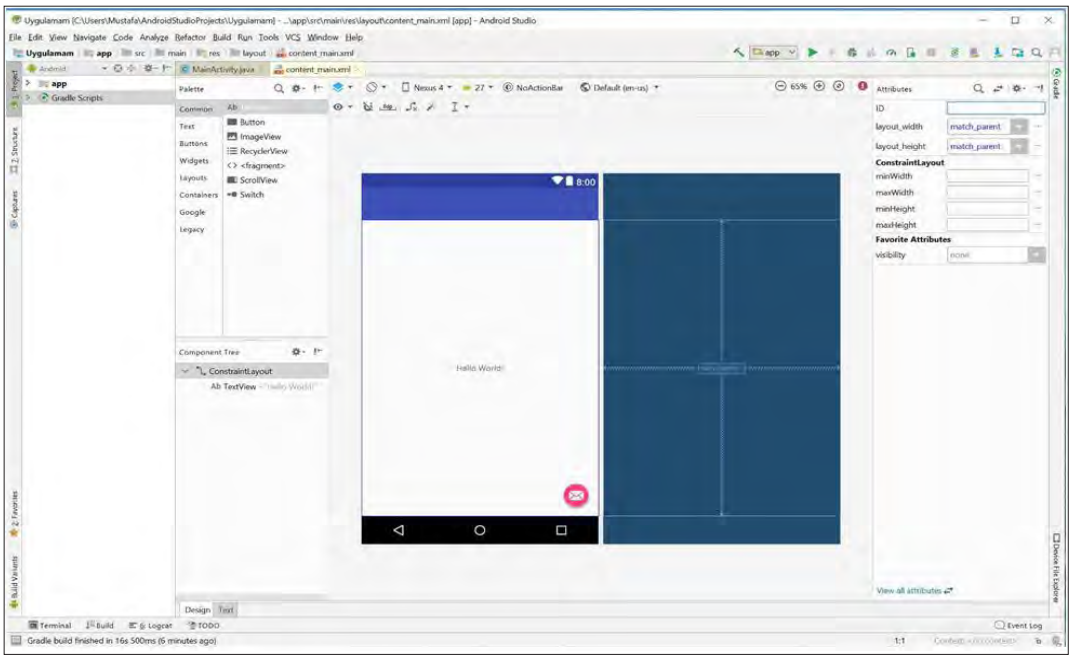
Ekran Görüntüsü 4.177

Configure Activity bölümü, Tasarım ekranının ayarlarının yapıldığı bölümdür. Tasarım ekranının ismini ve başlığını veriniz.



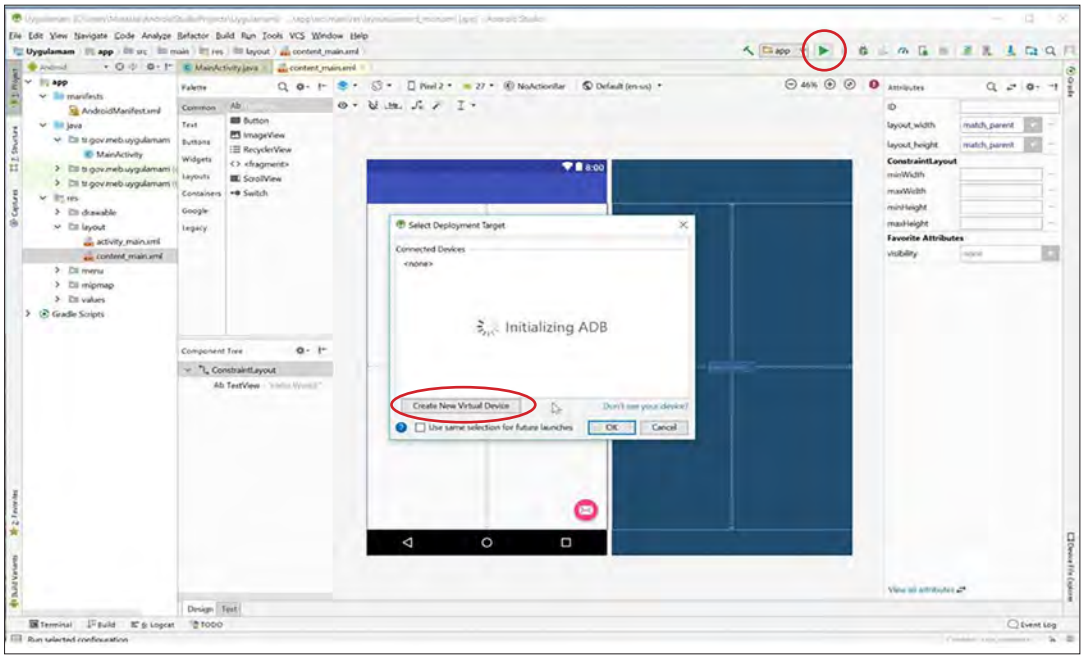
Ekran Görüntüsü 4.178

Projenizi oluşturduktan sonraki görüntü aşağıdaki gibidir. Uygulamanıza Palette bölümünden bir TextView sürükleyerek ekleyip içerisine yazınızı yazınız. TextView içindeki metni sağ tarafta yer alan bölümden değiştirebilirsiniz. Ayrıca metin ile ilgili özellikleri değiştirmek için textAppearence kısmının açılması gerekmektedir. Burada yer alan fontFamily ile yazı tipini, textSize ile yazı boyutunu, textColor ile yazı rengini ve textStyle ile kalın, italik gibi özellikleri değiştirebilirsiniz.



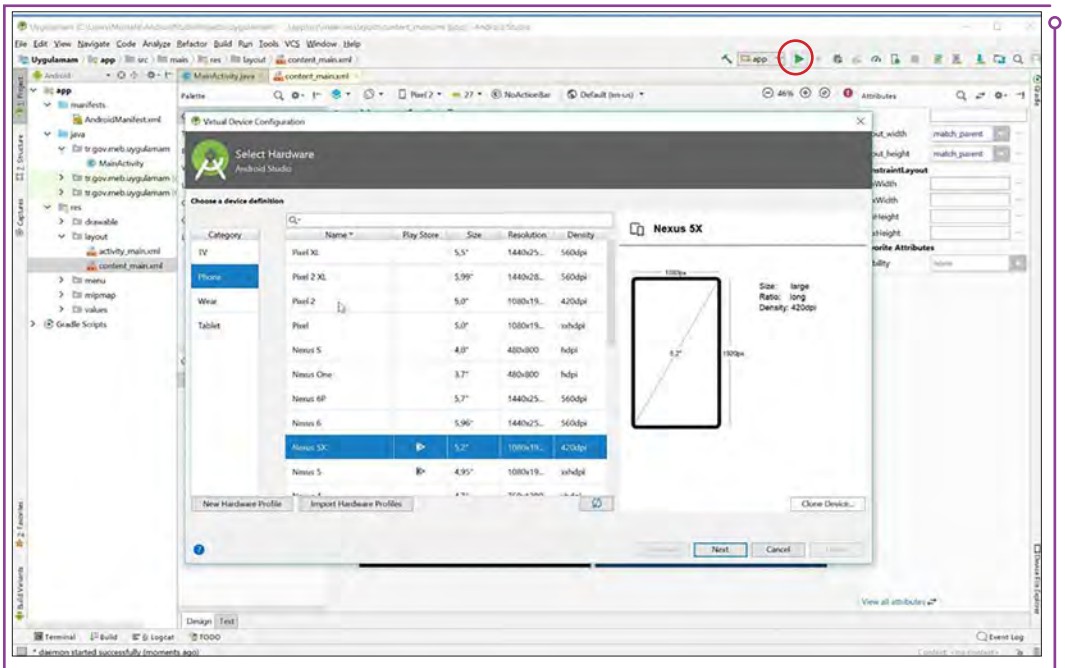
Ekran Görüntüsü 4.179

Run düğmesine basarak Merhaba Dünya uygulamanızı çalıştırınız.



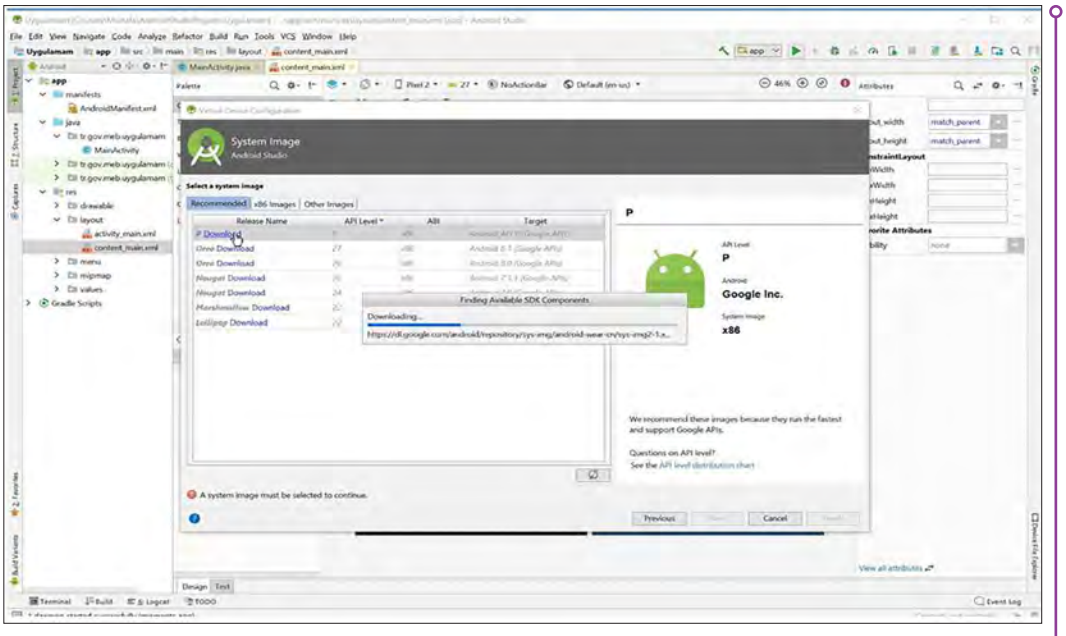
Ekran Görüntüsü 4.180

Uygulamayı çalıştırdıktan sonra, Create New Virtual Device düğmesine basarak uygulamanızın görüntüleneceği sanal cihaz (emülatör) için ekran ayarları, boyutu gibi özellikleri seçiniz.



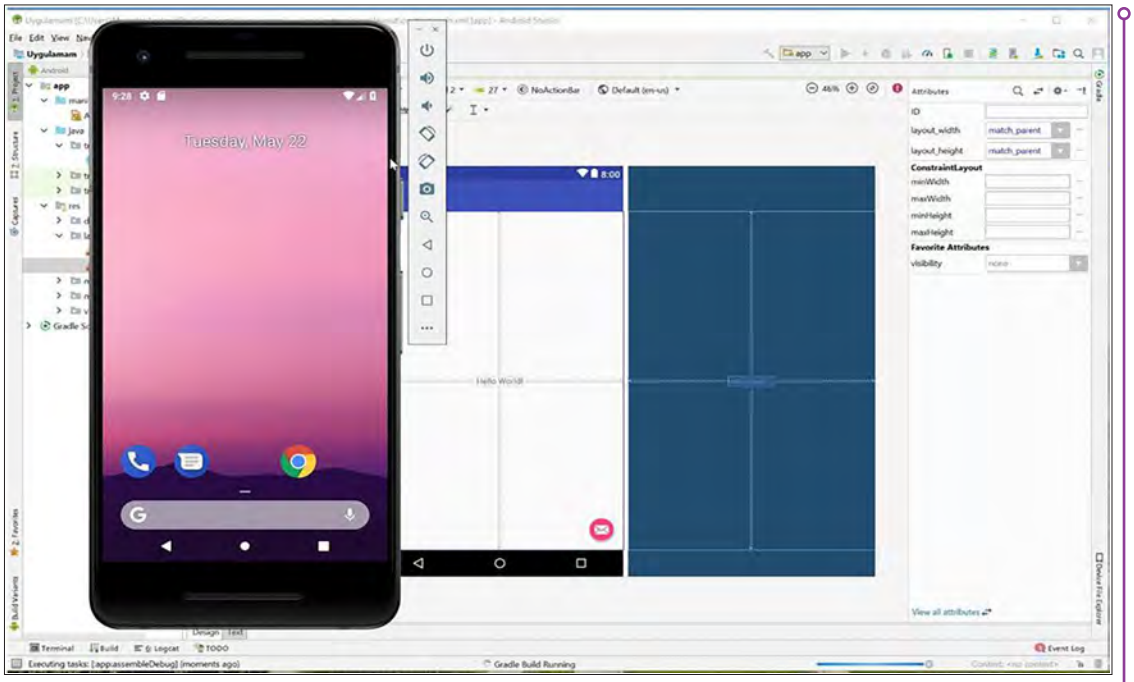
Ekran Görüntüsü 4.181

Seçtiğiniz sanal cihazın bilgisayarınıza kurulması için internete gerek vardır. Download'a tıklayıp bu cihazın internetten indirilip bilgisayara yüklenmesini sağlayınız.



Ekran Görüntüsü 4.182

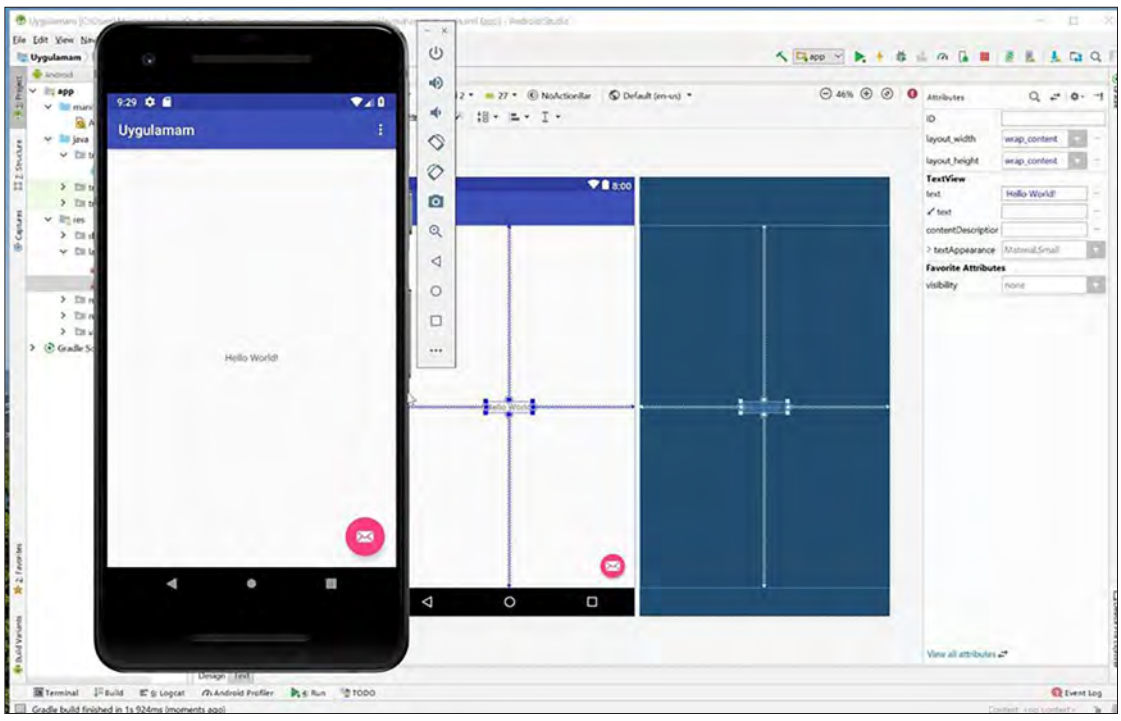
Yükleme tamamlandıktan sonra Next düğmesine basarak uygulamanızı çalıştırınız. Sanal cihazınızı ilk kez çalıştırdığınızda uygulamanın ekrana gelmesi biraz zaman alabilir.



Ekran Görüntüsü 4.183

Sanal zihazınız yüklendiğinde yukarıdaki gibi bir görüntüsü olacaktır.

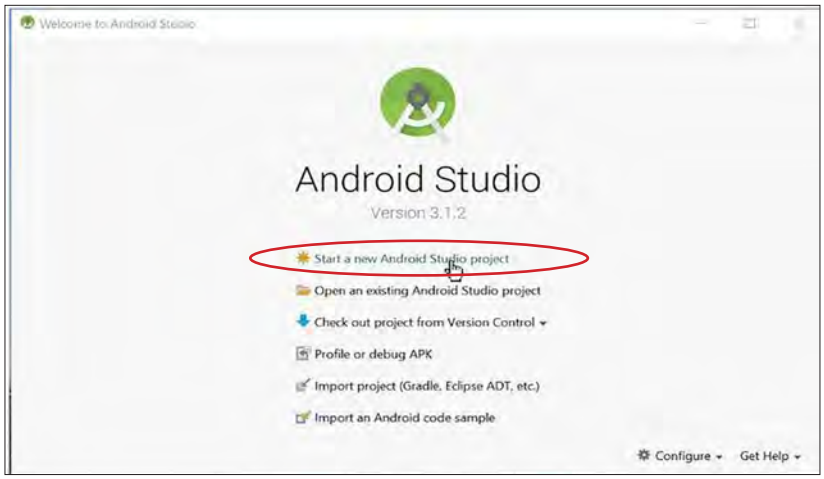
İlk uygulamanızın görüntüsü aşağıdaki gibi olacaktır.



Ekran Görüntüsü 4.184

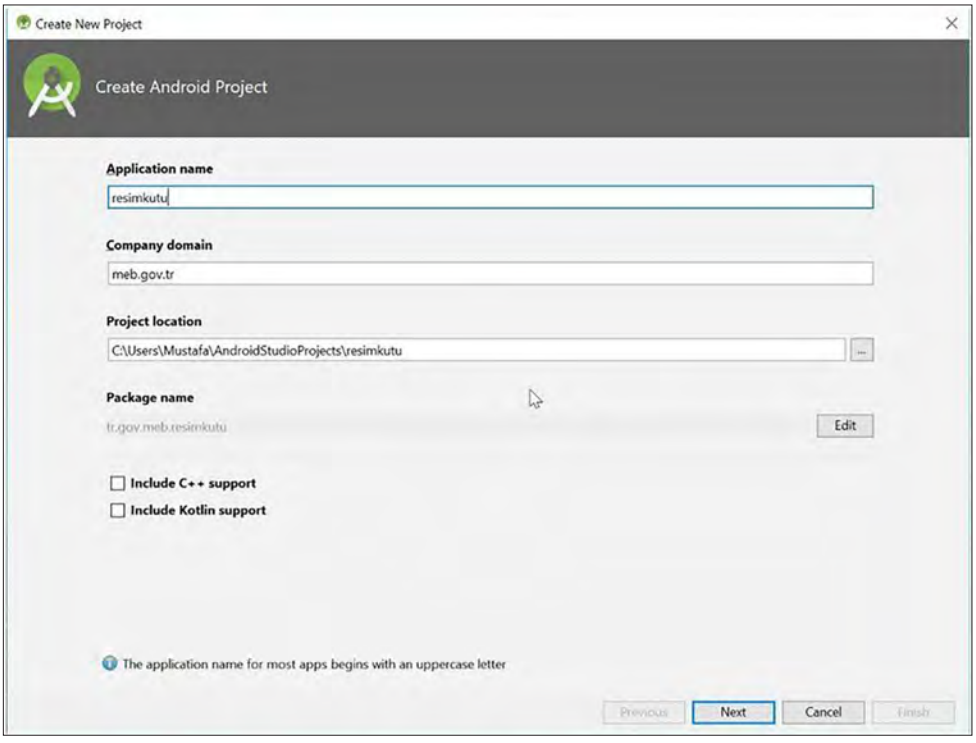
4.3.4. Uygulamaya Resim Ekleme

Sıradaki uygulamada, Android Studio'da uygulamaya resim eklenerek resim kütüphanesi yapılacaktır. "Start a new Android Studio Project" seçtikten sonra projenizi isimlendiriniz.



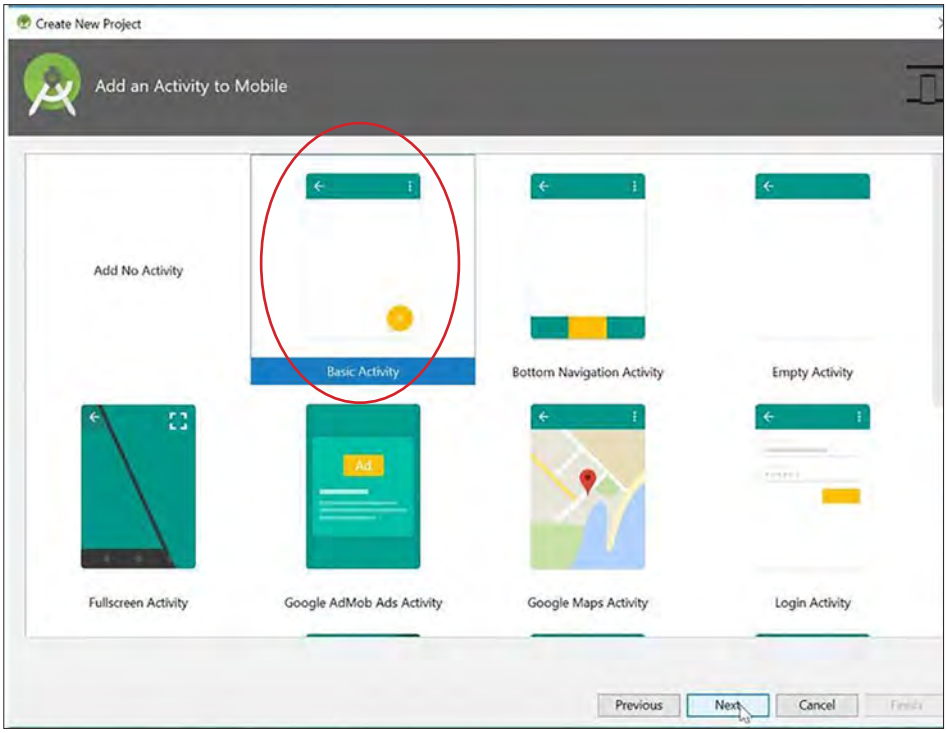
Ekran Görüntüsü 4.185

Bu projeye "resimkutuphanem" ismini veriniz. Türkçe karakter kullanmamaya dikkat etmeniz gerekmektedir.



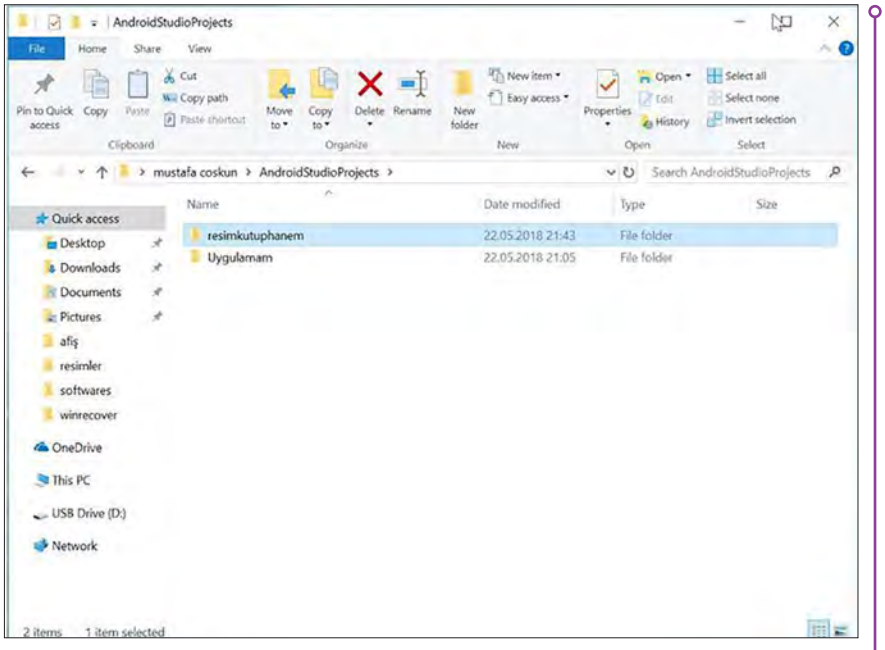
Ekran Görüntüsü 4.186

Tasarım ekranı olarak Basic Activity seçiniz.



Ekran Görüntüsü 4.187

Android Studio programında yapılan tüm projelerin kayıt edildiği dizin C:/Kullanıcılar/Kullanıcı/AndroidStudioProjects/ProjeAdı'dır. resimkutuphanem isimli proje, bu dizin altında oluşturulacaktır. Projeye eklenecek resimlerde bu dizine tekrar geri dönüş yapılacaktır.

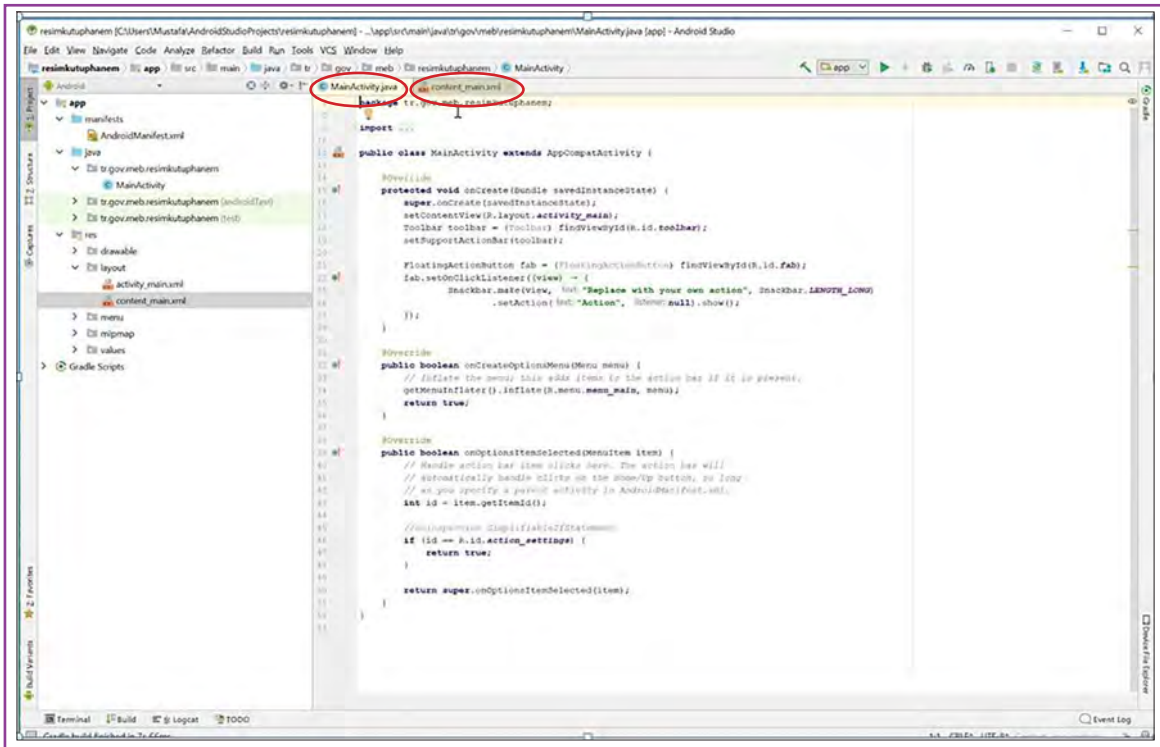


Ekran Görüntüsü 4.188

Kodlama işlemleri, Java'nın altındaki "MainActivity.java'da" ve Res/layout'un altındaki "content_main.XML'de" yapılmaktadır. MainActivity'de kodlar yazılmaktadır. MainActivity.java dosyası projede oluşturulan ana aktivitenin .java uzantılı kaynak dosyasıdır. Projenizde yapacağınız birçok durum için kodlarınızı bu dosya içerisine yazmalısınız. content_main.XML dosyası ise aktivitenizde yer alan bileşenlerin (TextView, Button, ImageView vb.) tanımlanmasının yapıldığı yerdir.

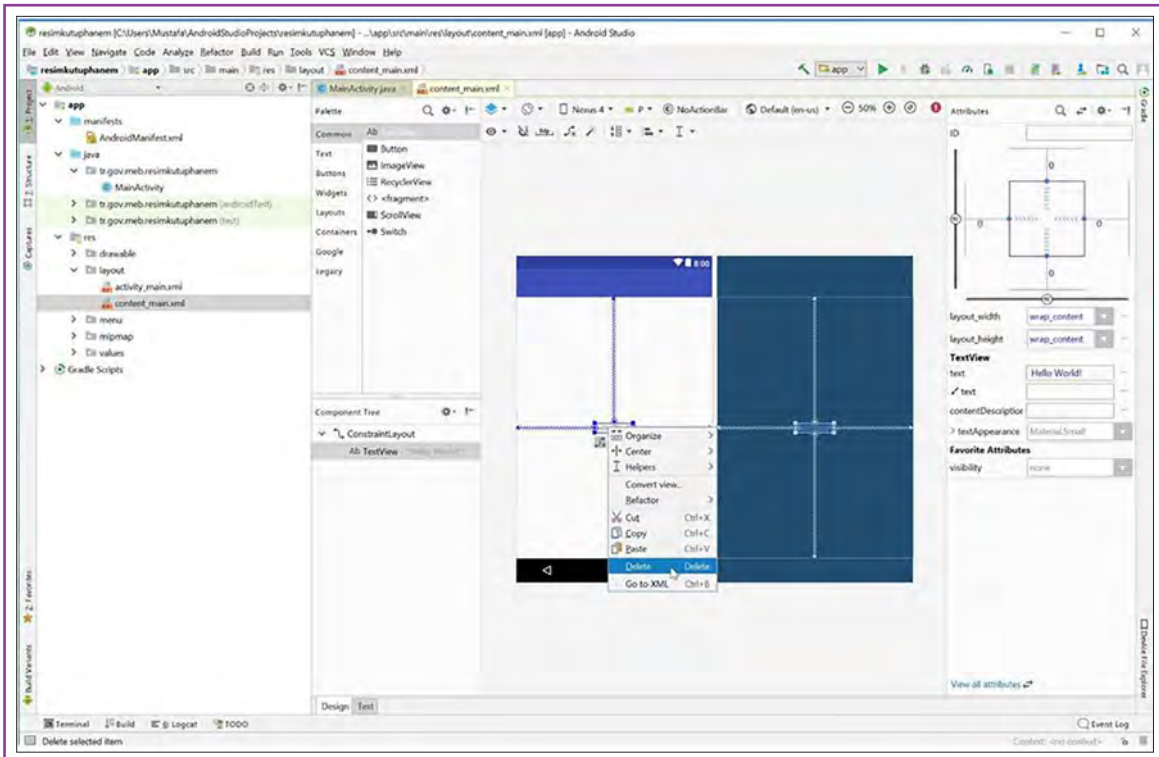
XML'ler design (tasarım) ekranındaki nesnelerin bilgilerini tutan veri dosyalarıdır. Uygulamanızda kullandığınız her nesne bu XML dosyasında properties (özellikler) penceresindeki belirlediğiniz özellikleri ile birlikte yer tutar. Örneğin bir Imageview nesnesi koyarsanız aşağıdaki gibi; id'si, width (genişlik), height (yükseklik) bilgisi XML dosyasına konur. Sizler isterseniz design ekranında isterseniz de XML dosyasında ekranınızdaki nesneleri ve bilgilerini değiştirebilirsiniz. Aşağıdaki kodlar tüm XML dosyalarında ortak olarak gelmektedir.

```
<android.support.constraint.ConstraintLayout XMLns:android="http://schemas.android.com/apk/res/android"
   XMLns:app="http://schemas.android.com/apk/res-auto"
   XMLns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   app:layout_behavior="@string/appbar_scrolling_view_behavior"
   tools:context=".MainActivity"
   tools:showIn="@layout/activity_main">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello World!"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```



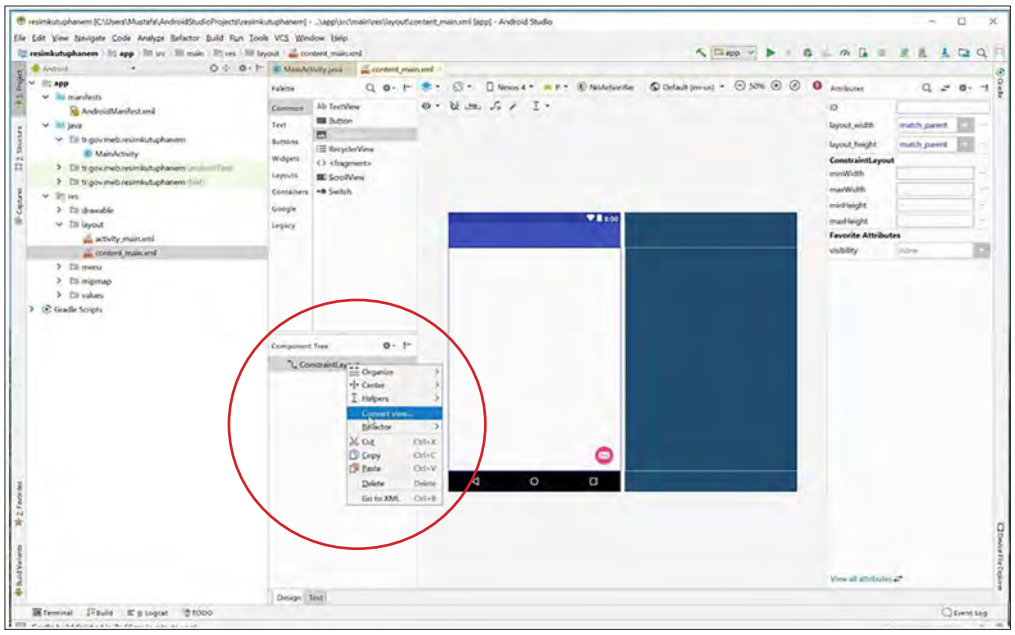
Ekran Görüntüsü 4.189

Varsayılan olarak gelen helloworld uygulamasını sağ tıklayıp delete diyerek kaldırınız.



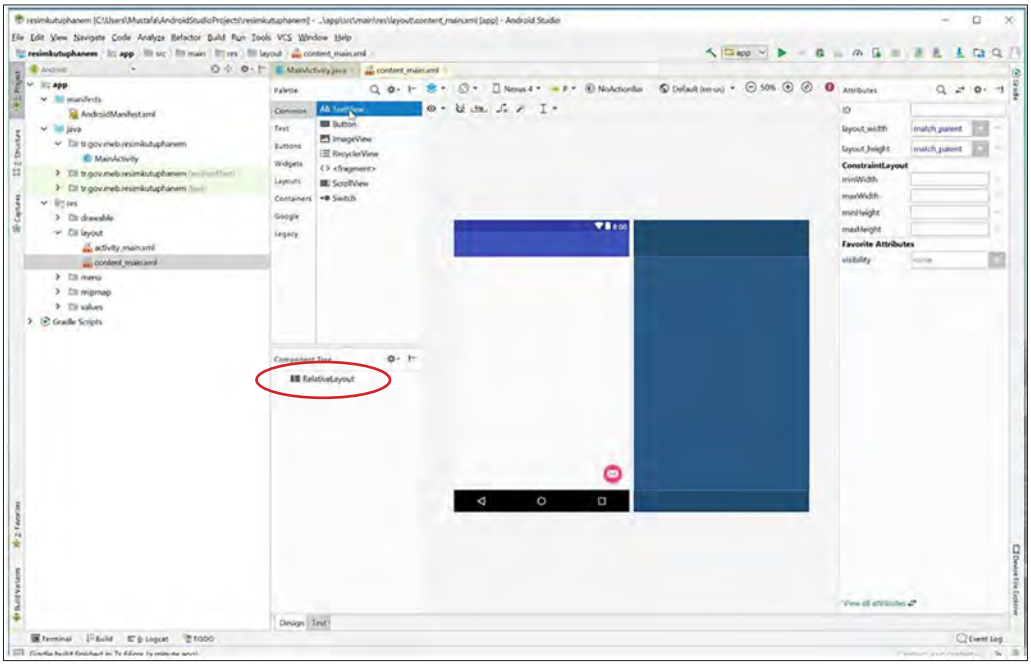
Ekran Görüntüsü 4.190

Bu uygulamada imageview ve TextView'lerle çalışılacaktır. Nesnelerin tasarım ekranında düzenlenmesi Layoutlar ile yapılmaktadır. Projenin türüne göre farklı Layoutlarda çalışılabilir. Bu uygulamada RelativeLayout kullanılacaktır. RelativeLayout; tasarım ekranına koyduğunuz nesneleri aynı koordinatta bırakan bir layouttur. Bu uygulamada, ConstraintLayout'u RelativeLayout'a çevirmeniz gerekmektedir. ConstraintLayout'a sağ tıklayıp Convert view'e tıklayınız. Çıkan layout seçeneklerinden RelativeLayout'u seçiniz.



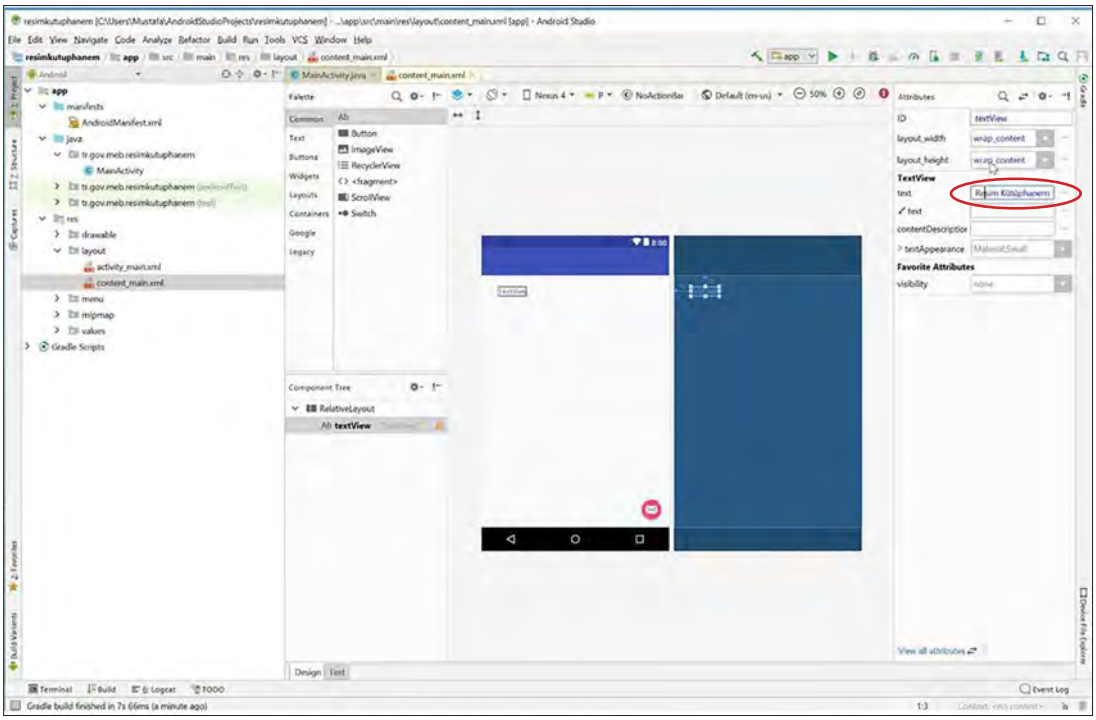
Ekran Görüntüsü 4.191

Şimdi proje görüntüsü Relative yani 'Göreli' hale dönüşmüştür.



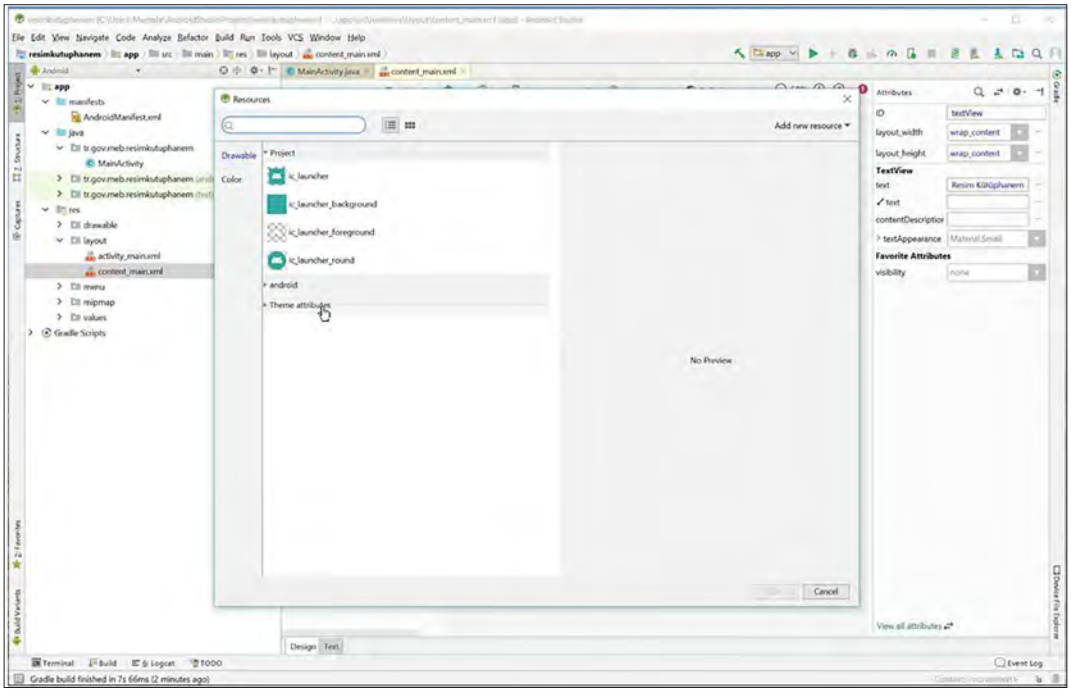
Ekran Görüntüsü 4.192

Tasarım ekranına bir TextView eklenecektir. Text üzerinde yazılı olan metni sağdaki menüden düzenleyebilirsiniz. Burada Türkçe karakter kullanılabilmektedir. Text içeriğine Resim Kütüphanem yazınız.



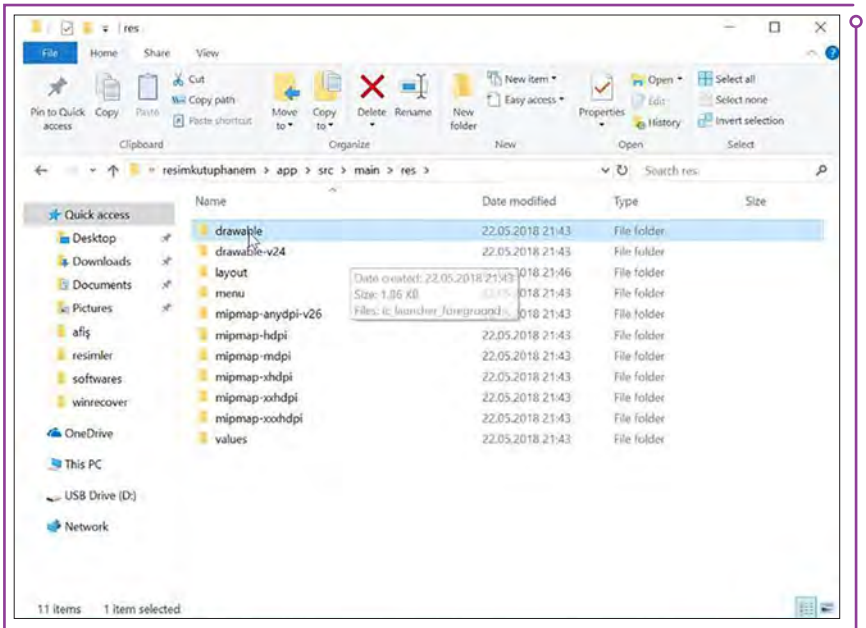
Ekran Görüntüsü 4.193

Daha sonra tasarım ekranına bir Imageview ekleyiniz.



Ekran Görüntüsü 4.194

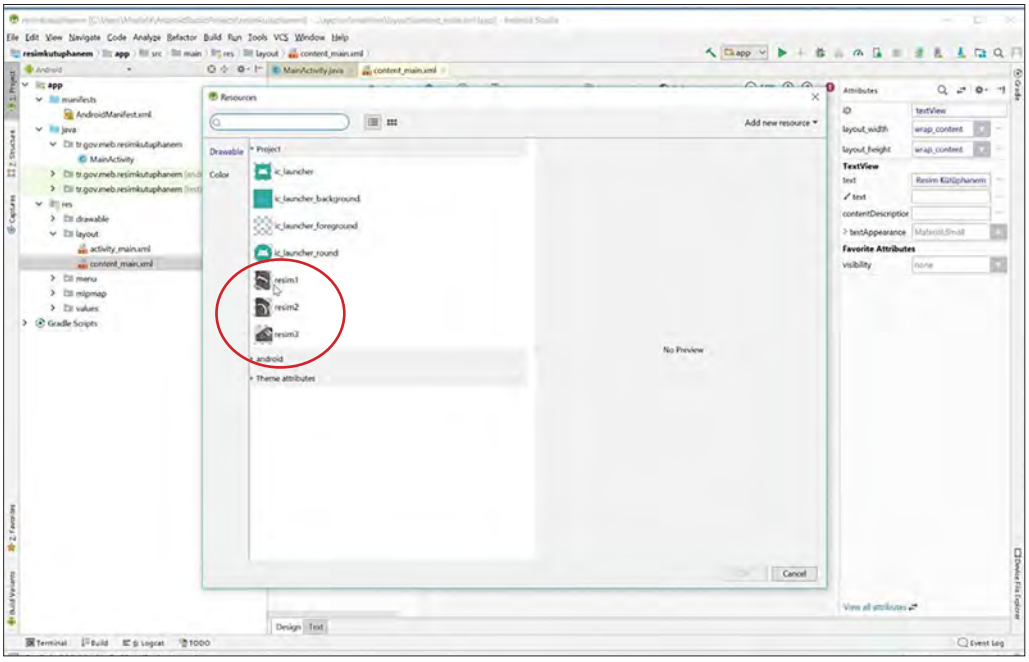
Projenize resim ekleyebilmek için eklenecek resimlerin projenin yer aldığı dizindeki proje klasörü içindeki "drawable" klasörü içine atılması gerekmektedir. (resimkutuphanem/app/src/main/res /drawable)



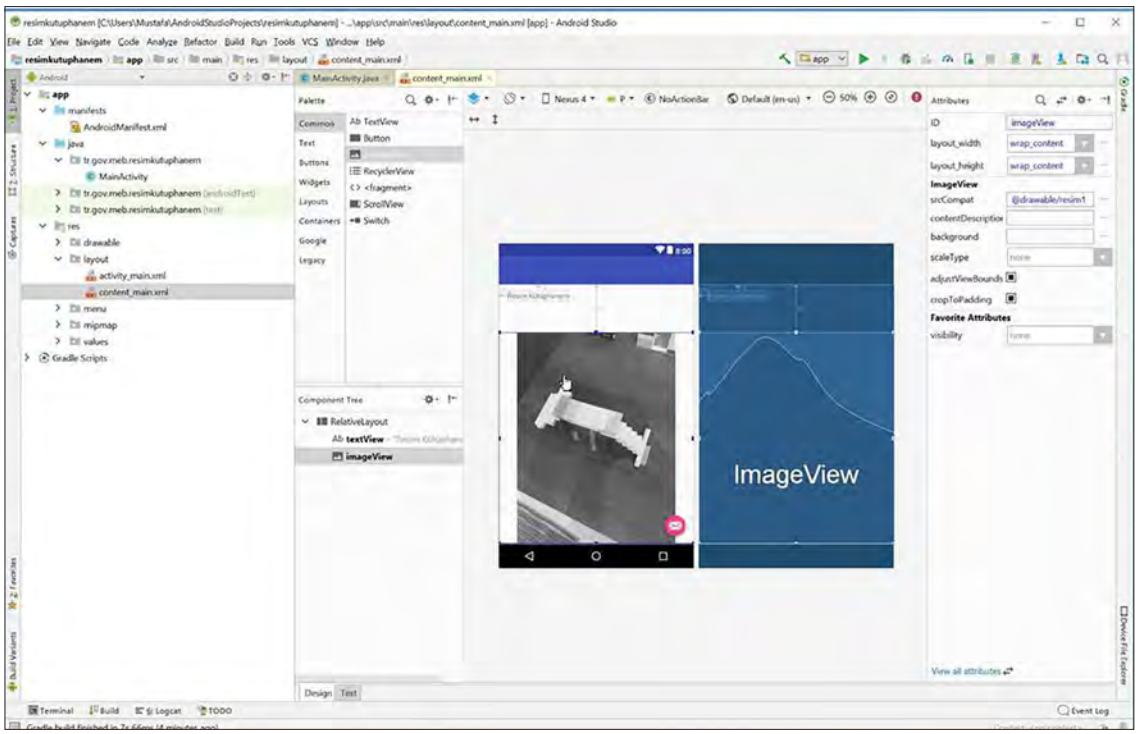
Ekran Görüntüsü 4.195

Burada dikkat edilmesi gereken bir nokta, resimleri adlandırırken rakam ile başlamaması ve boşluk içermemesi gerekmektedir ve çözünürlükleri çok büyük olmamalıdır. Çok büyük olduğunda cep telefonu ekranında görünmeyebilir.

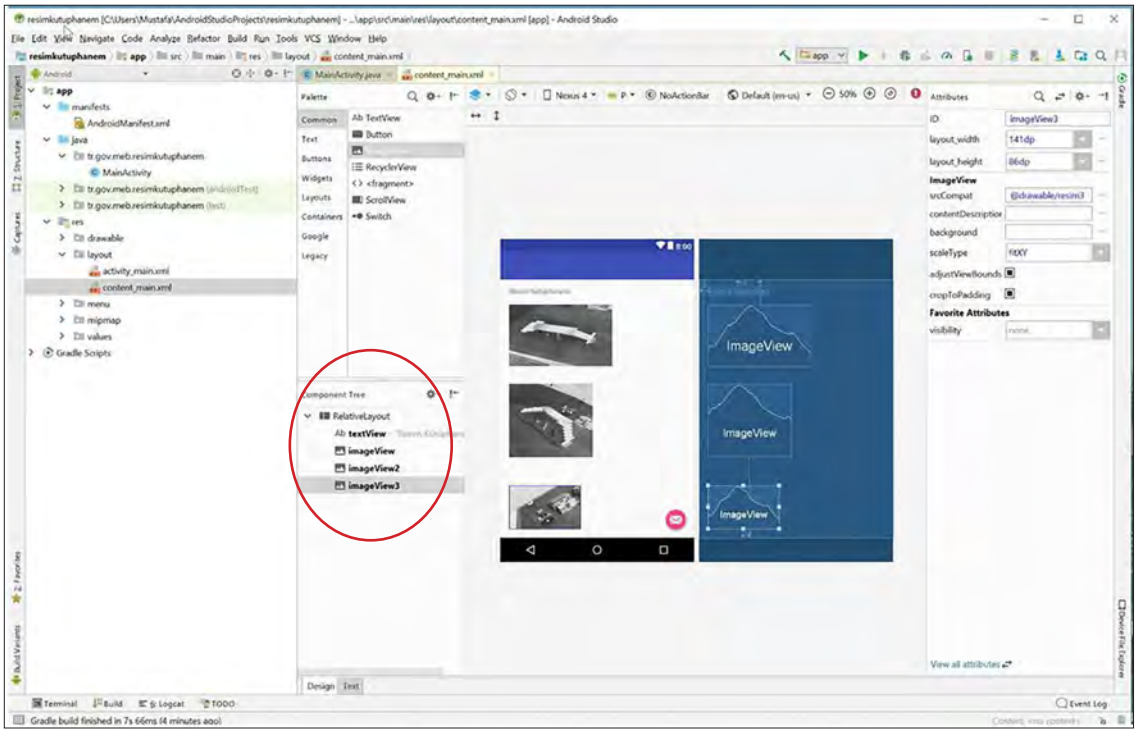
Resimler klasöre eklendikten sonra projeye de eklenebilir hale gelmektedir.



Ekran Görüntüsü 4.196

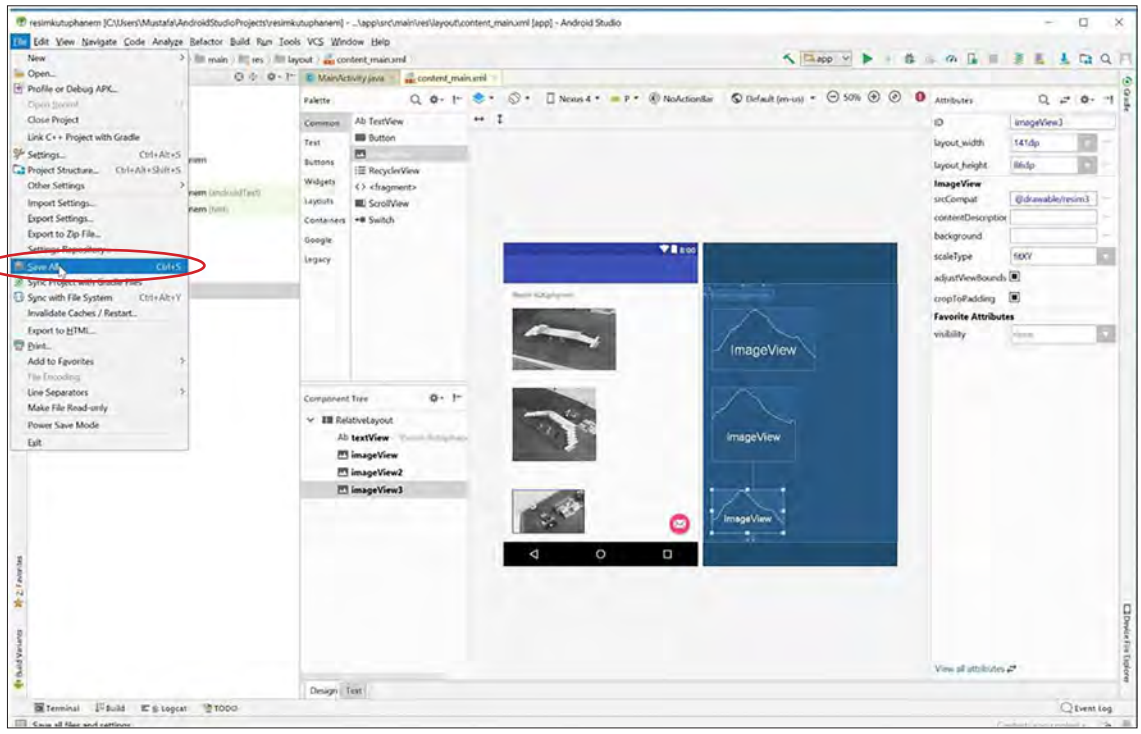


Ekran Görüntüsü 4.197



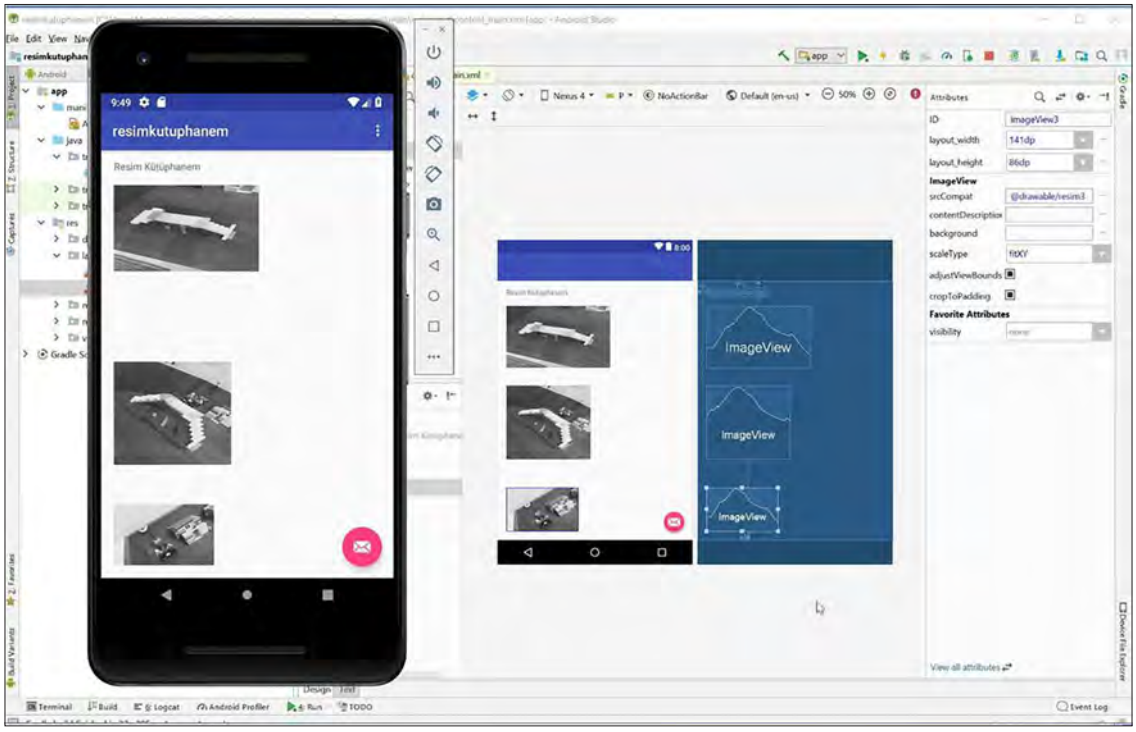
Ekran Görüntüsü 4.198

Resimlerin tamamı eklendikten sonra projenizi File->SaveAll diyerek kaydediniz ve projenizi çalıştırınız.



Ekran Görüntüsü 4.199

Projenizi çalıştırdığınızda emülatördeki görüntü aşağıdaki şekilde olacaktır.

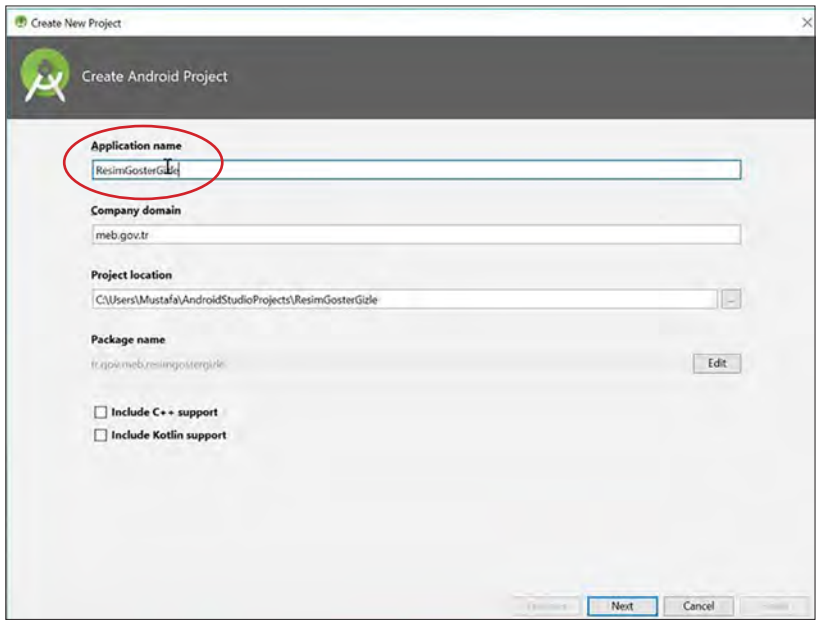


Ekran Görüntüsü 4.200

4.3.5. Resim Göster/Gizle Uygulaması

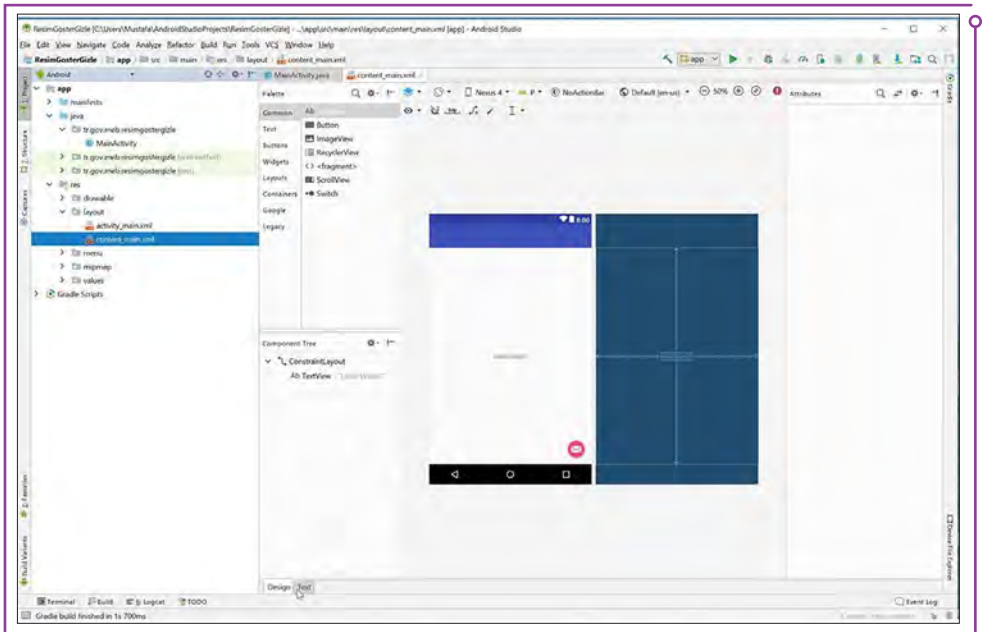
Bu uygulamada ekrana bir ImageView, bir TextView ve bir Button eklenecektir. Kullanıcı düğmeye tıklayarak ImageView içindeki resmi gösterip gizleyecektir. Resim görünür iken düğme başlığı "Gizle", gizli iken de "Göster" olacaktır.

Yeni bir proje oluşturunuz ve ismini ResimGosterGizle yapınız.



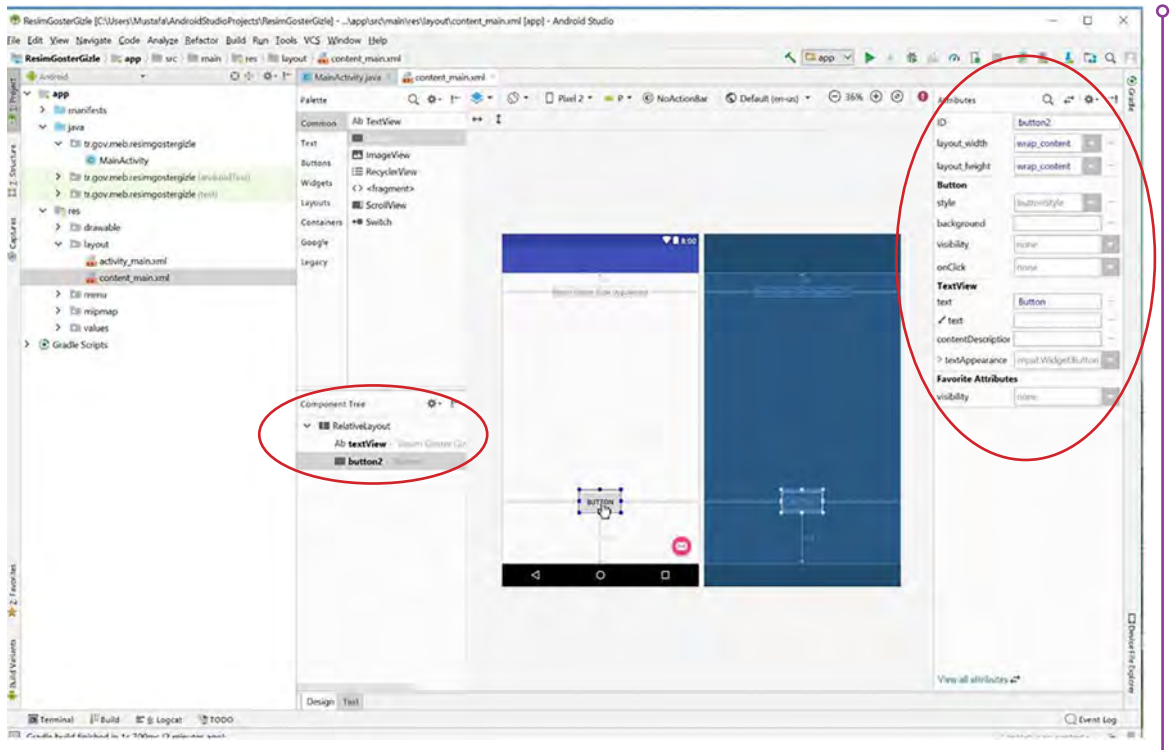
Ekran Görüntüsü 4.201

Basic bir activity seçiniz. Çalışma klasörü program tarafından oluşturulur ve ilgili dosyalar bu klasöre atılır. Content_main.XML dosyanıza geliniz ve helloworld TextView'i (metin kutusunu) siliniz.



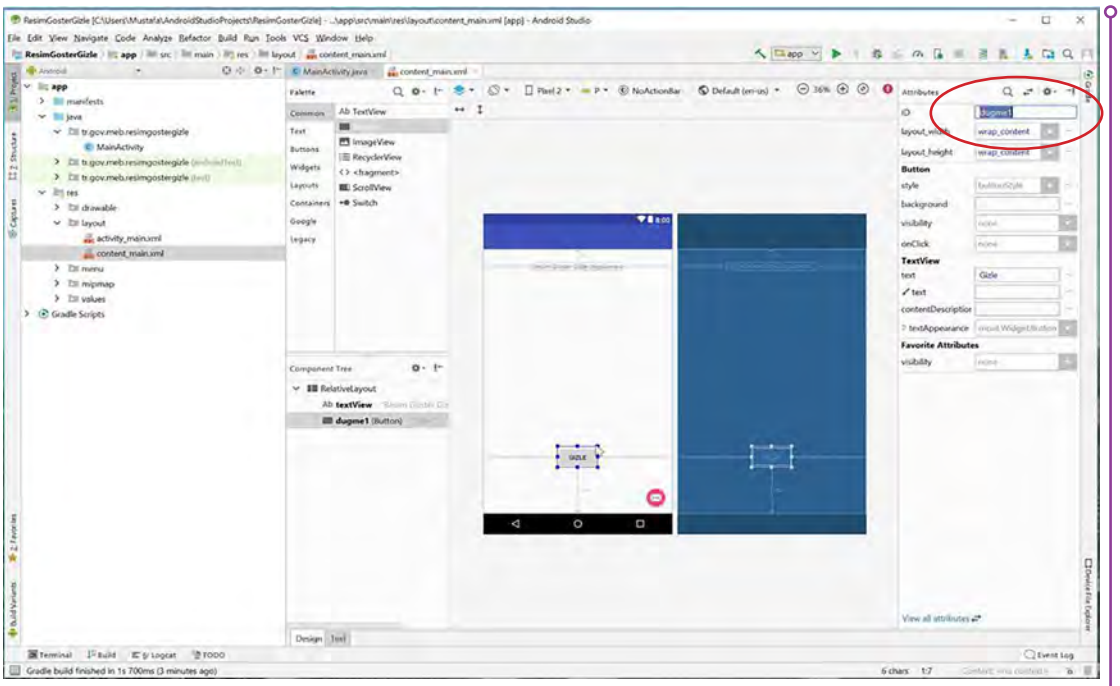
Ekran Görüntüsü 4.202

Resim dosyalarınızı resimgostergizle/app/src/main/res/drawable klasörüne atınız, ConstraintLayout'u RelativeLayout'a çeviriniz. Ardından projenize birer TextView ve Button ekleyiniz. TextView'e Resim Göster Gizle Uygulaması, Button'a "Gizle" yazınız. Aşağıdaki ekran görüntüsünde görüldüğü gibi tasarım ekranınıza (Aktivity) eklediğiniz her öğe daire içine alınan alanda gözükmektedir. Hangi öğenin özelliklerini değiştirmek istiyorsanız bu alanda üzerine tıkladığınızda öğeyle alakalı özellikler sol pencerede açılmaktadır. Bu alandan istediğiniz değişiklikleri yapabilirsiniz.



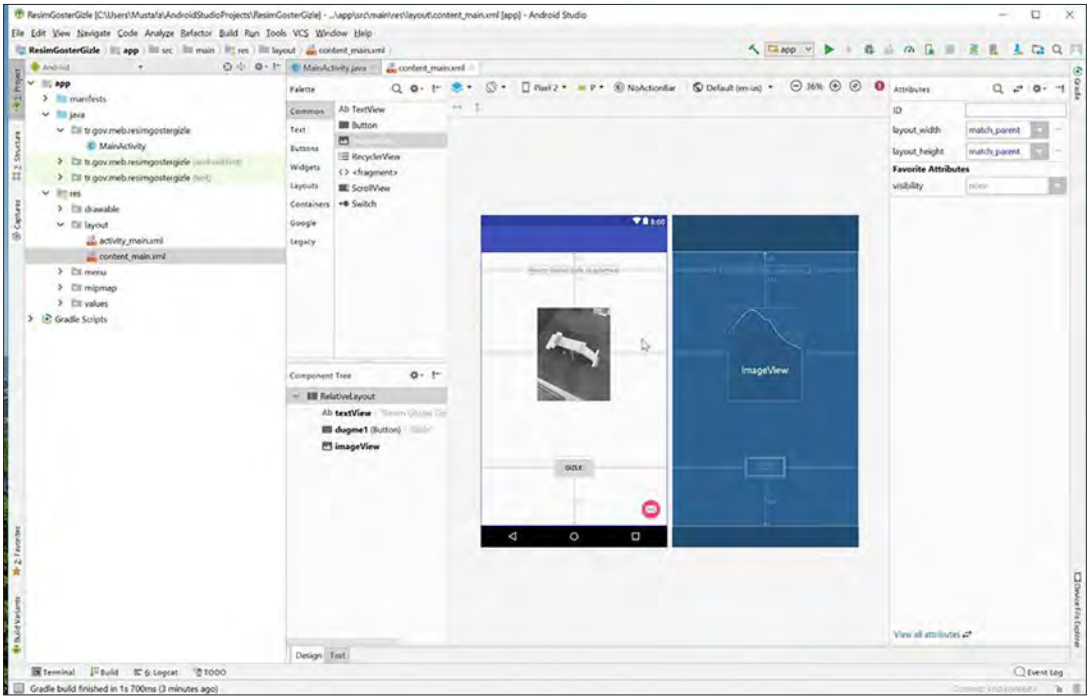
Ekran Görüntüsü 4.203

Eklenen Buton'a ve TextView'e id veriniz. Bu id'ler, daha sonra kullanılacağı için önemlidir. Butonun id'sini dugme1 yapınız.



Ekran Görüntüsü 4.204

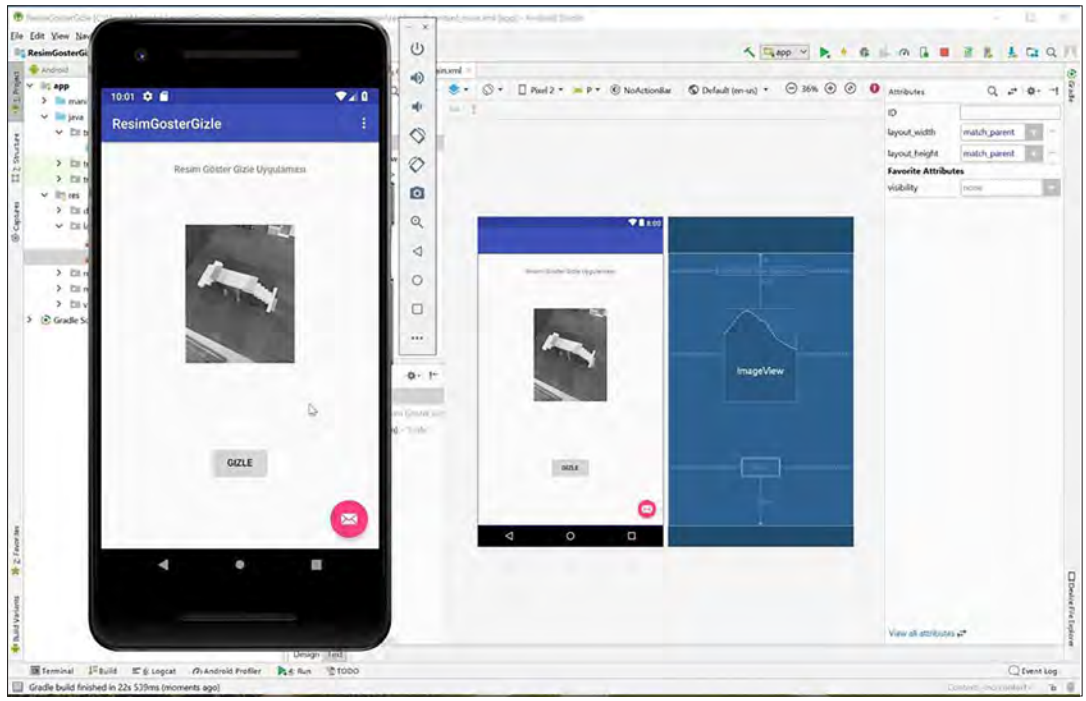
Şimdi de bir image ekleyiniz ve resmi fitXY komutu ile istediğiniz boyuta getiriniz. fitXY komutu, görsel seçildiğinde sağdaki pencerede scaleType özelliklerinin içindedir. Bu özellik içindeki komutlar, görselin boyutlandırılmasında kullanılmaktadır.



Ekran Görüntüsü 4.205

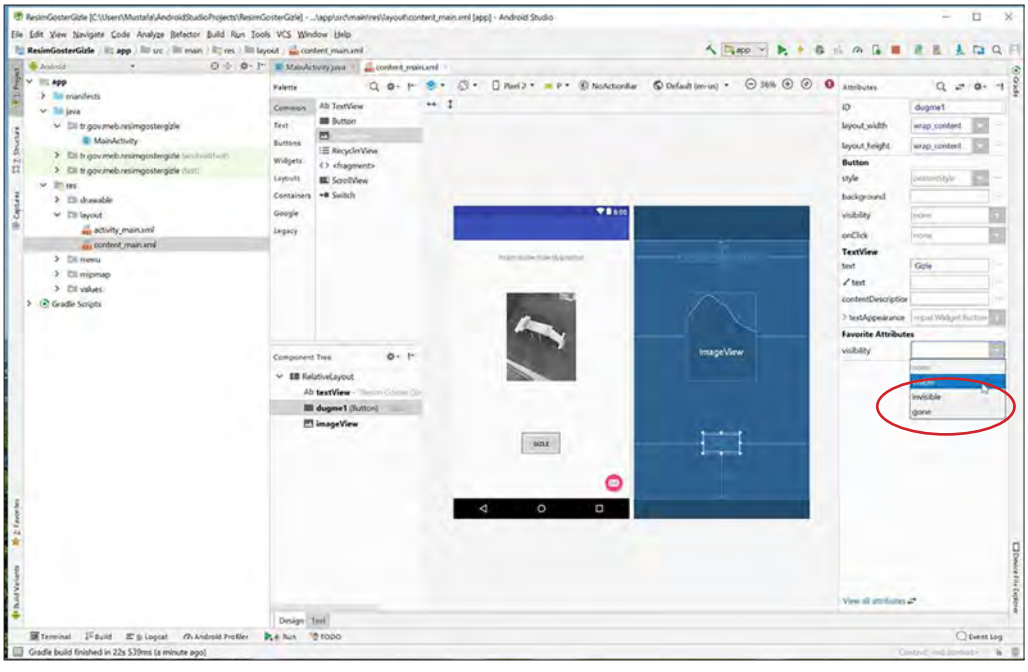
Proje bu şekilde oluşturulduğunda ilgili kodlar XML dosyasına otomatik olarak gelecektir.

Proje çalıştırıldığında aşağıdaki ekran görünecektir.



Ekran Görüntüsü 4.206

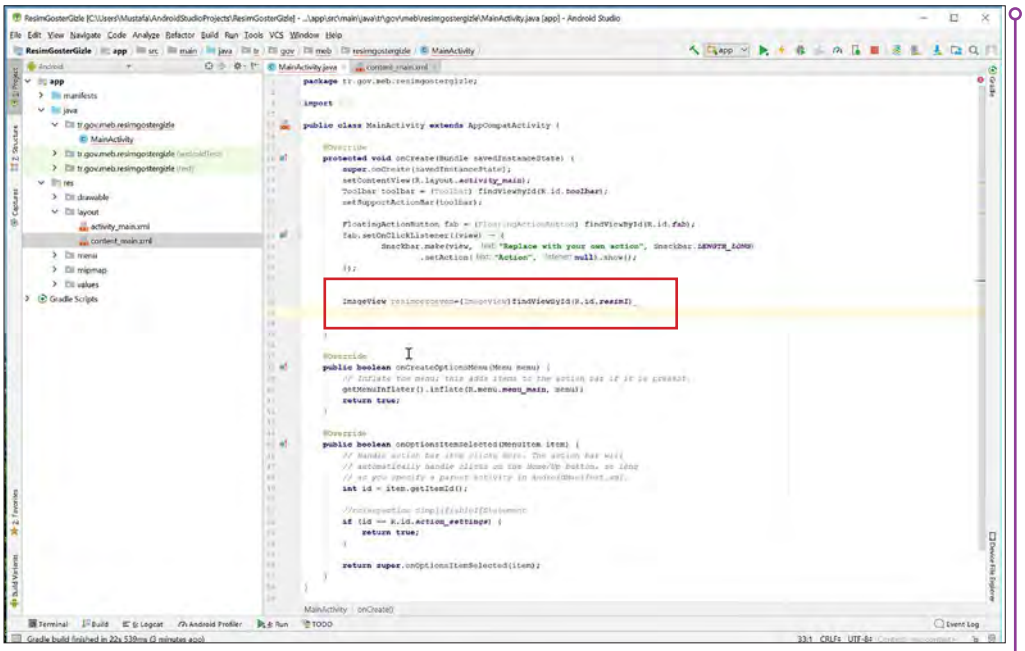
Şu anda gizle Buton'u çalışmıyor, kodlaması yapıldığında çalışacaktır. Kodlama için düğme seçilecek ve düğmede Imageview nesnesi tanıtılacaktır. Buton 'tıklandığında' olayı ile visible değeri invisible değerine çevrilecektir.



Ekran Görüntüsü 4.207

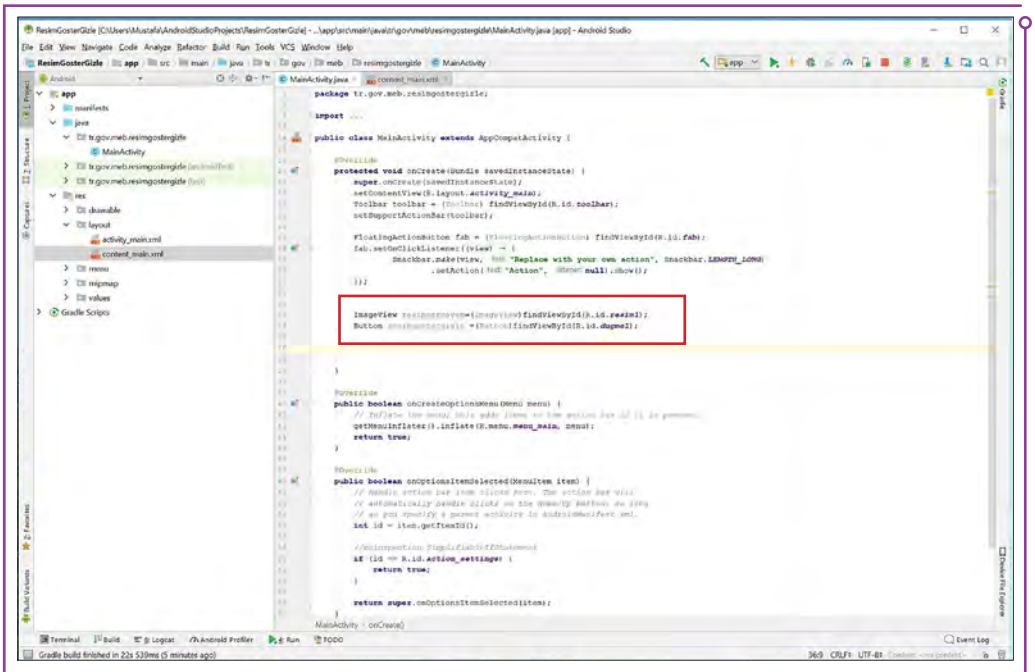
Kodlar, her zamanki gibi mainActivity.java'da yazılacaktır. OnCerate olayının altında bu işlem yapılacaktır. Önce bir Imageview oluşturunuz. Bir boşluğa gelip kodlarınızı yazınız. Yazılan kodları detaylı açıklayalım:

Önce, ne tür bir nesne oluşturulacaksa bunu yazınız. findViewById denilen kod parçası, bütün kaynakların arasından id'sine göre nesneyi bulacaktır.



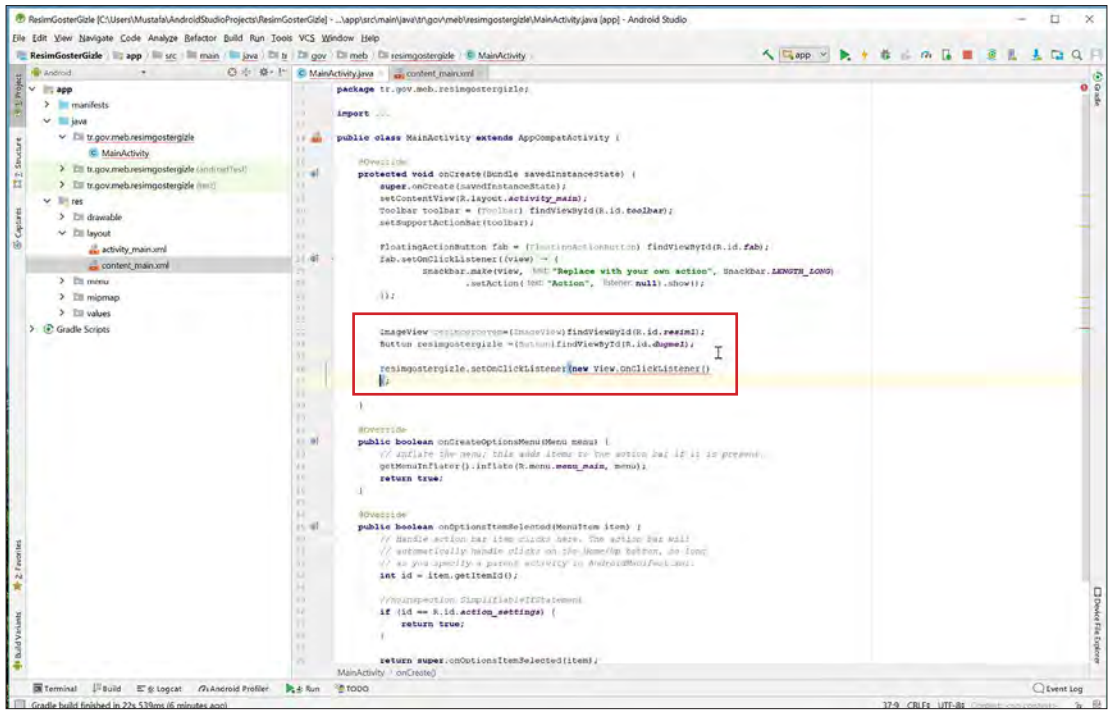
Ekran Görüntüsü 4.208

Oluşturulan buton için kodlarınızı yazınız.



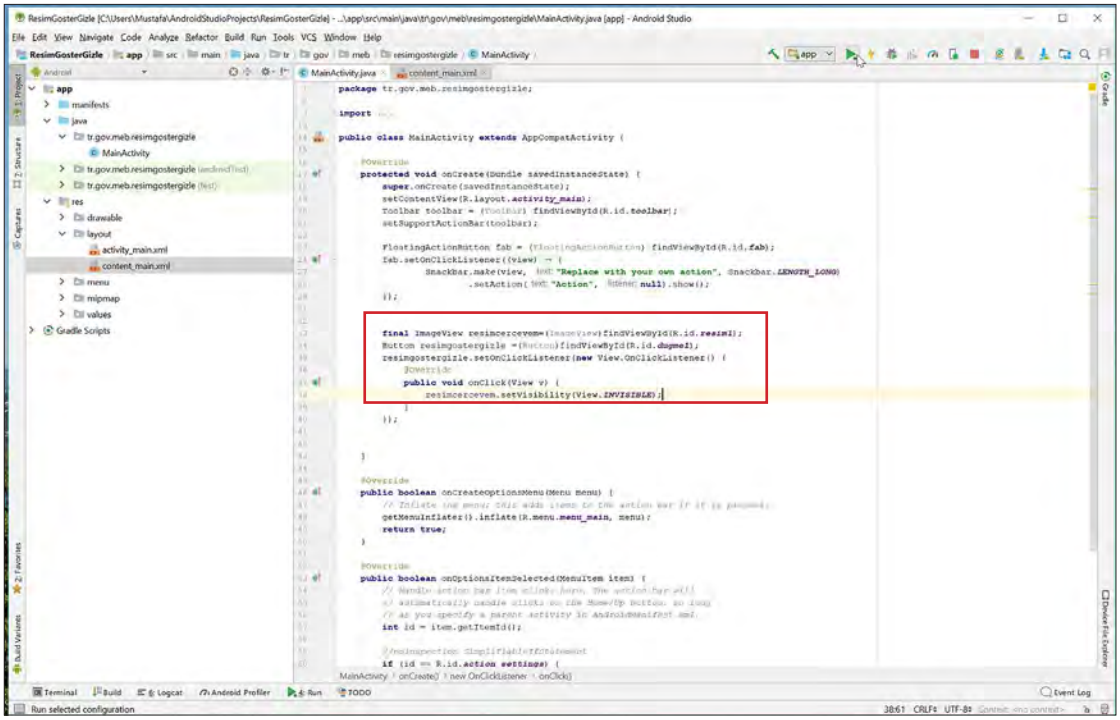
Ekran Görüntüsü 4.209

Şimdi de Buton'a tıklama olayının tetiklenip tetiklenmediğini anlamak için bir class oluşturunuz.



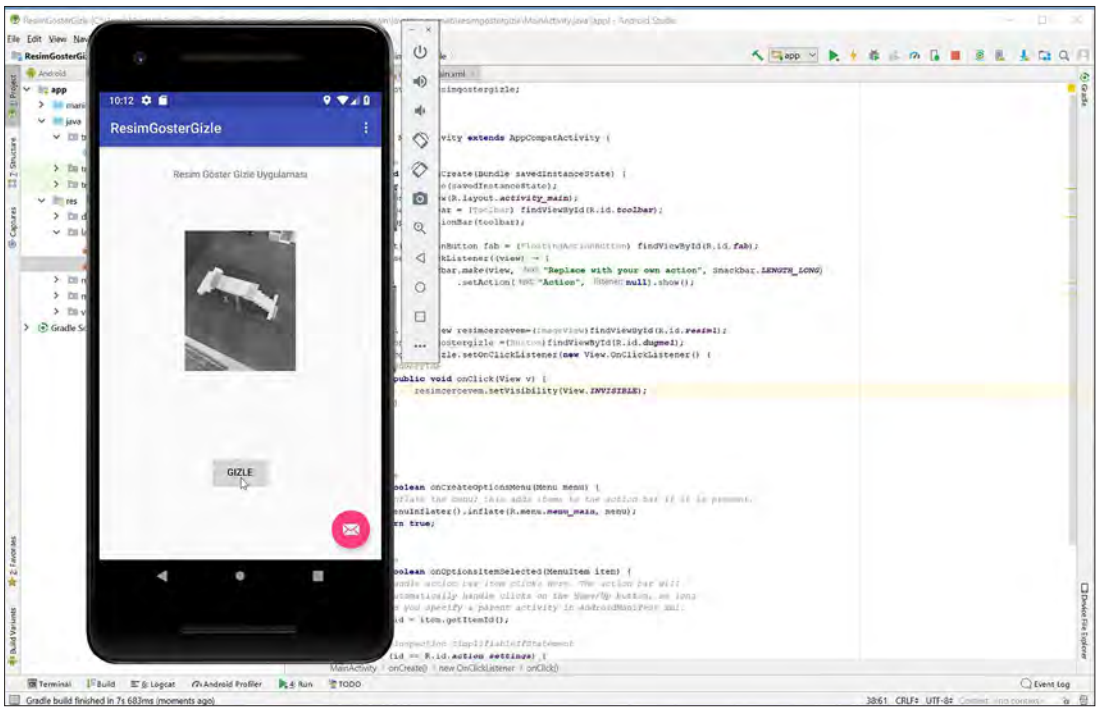
Ekran Görüntüsü 4.210

Butona basıldığında visible özelliğini invisible yapan bir fonksiyon yazılacaktır.



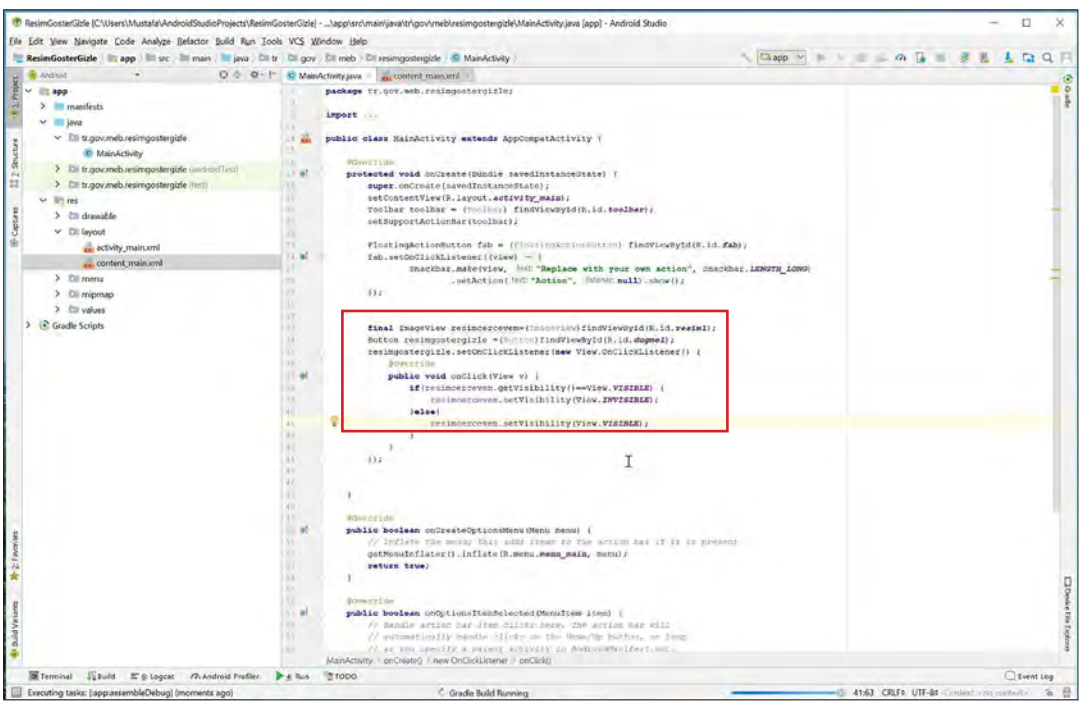
Ekran Görüntüsü 4.211

Proje çalıştırıldığında aşağıdaki görüntü ekrana gelmektedir.



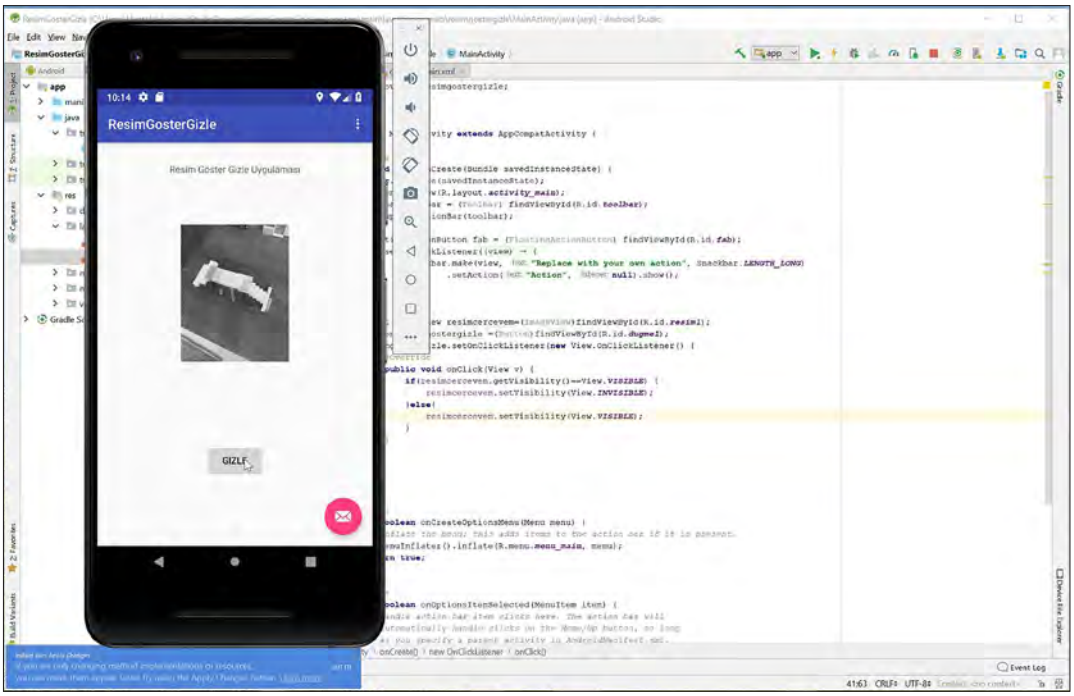
Ekran Görüntüsü 4.212

Gizlenen resmin geri getirilmesi için uygulamaya aşağıdaki kodların eklenmesi gerekmektedir.



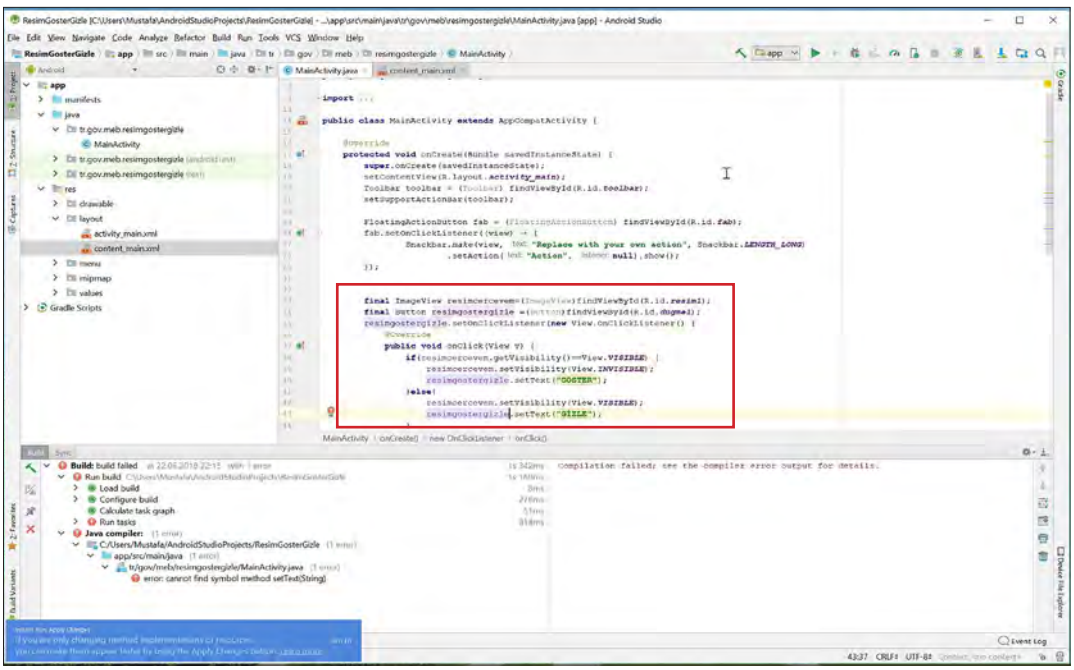
Ekran Görüntüsü 4.213

Kodlar eklendiğinde projenin son hali aşağıdaki gibi olmaktadır.



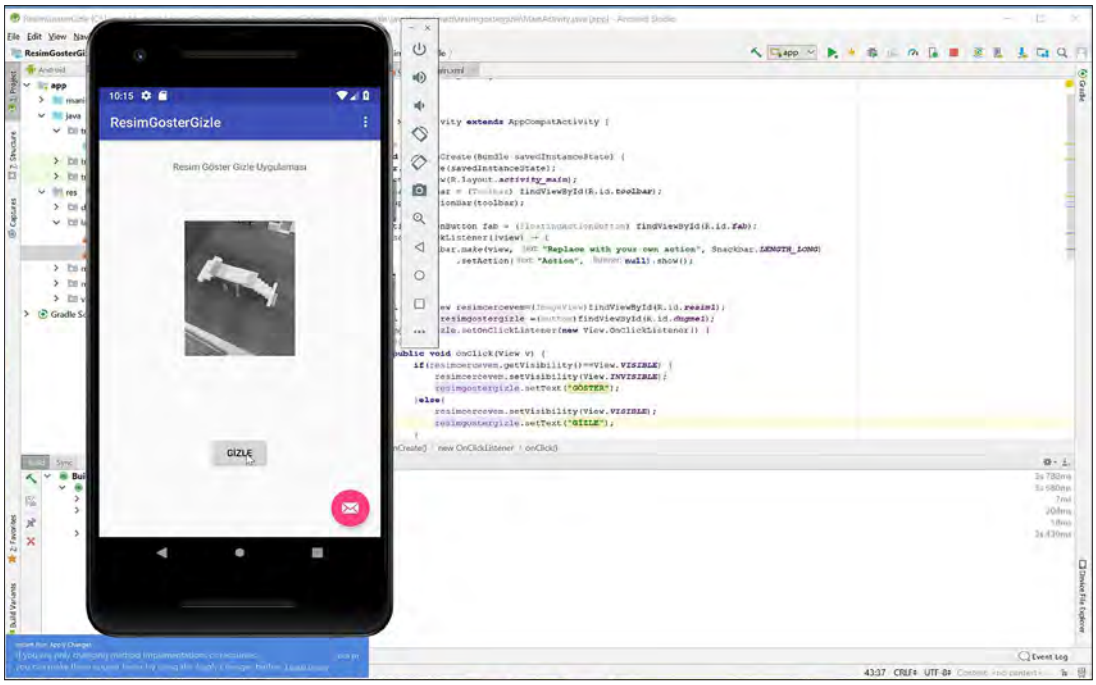
Ekran Görüntüsü 4.214

Şimdi de, resim gizliyse Buton'da Göster, görünürse Gizle yazın. Bunun için komutlar aşağıdaki gibidir.

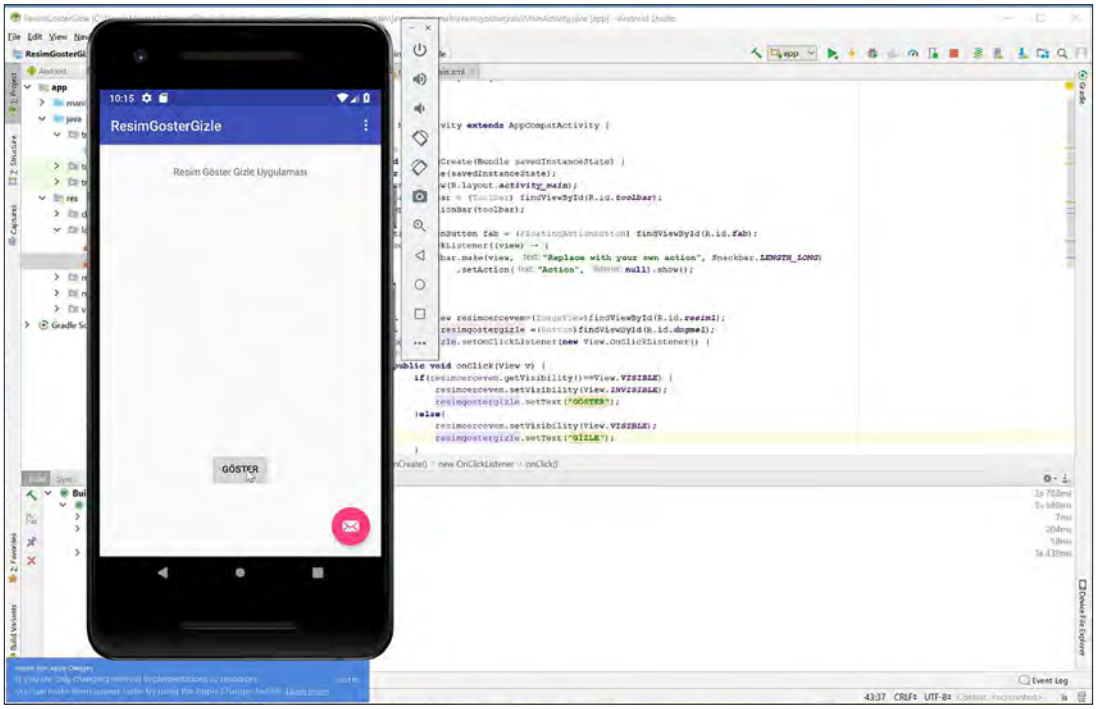


Ekran Görüntüsü 4.215

Resim göster gizle Buton'u hazır hale gelmiştir.



Ekran Görüntüsü 4.216



Ekran Görüntüsü 4.217

XML Kodları

```
<?XML version="1.0" encoding="utf-8"?>
<RelativeLayout XMLns:android="http://schemas.android.com/apk/res/android"
  XMLns:app="http://schemas.android.com/apk/res-auto"
  XMLns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  app:layout_behavior="@string/appbar_scrolling_view_behavior"
  tools:context=".MainActivity"
  tools:showIn="@layout/activity_main">

  <TextView
    android:id="@+id/TextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="30dp"
    android:text="Resim Göster Gizle Uygulaması" />

  <Button
    android:id="@+id/dugme1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="104dp"
    android:text="Gizle" />

  <ImageView
    android:id="@+id/resim1"
    android:layout_width="162dp"
    android:layout_height="205dp"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="123dp"
    android:scaleType="fitXY"
    android:visibility="visible"
    app:srcCompat="@drawable/resim1" />
</RelativeLayout>
```

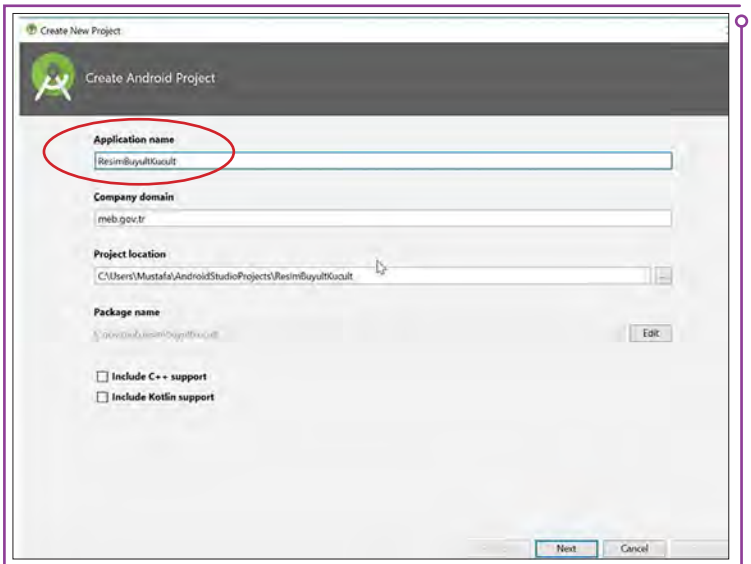
Java Kodları

```
package tr.gov.meb.resimgostergizle;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
     setSupportActionBar(toolbar);

     FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
     fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
           Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                 .setAction("Action", null).show();
        }
     });

     final ImageView resimcercevem=(ImageView)findViewById(R.id.resim1);
     final Button resimgostergizle =(Button)findViewById(R.id.dugme1);
     resimgostergizle.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           if(resimcercevem.getVisibility()==View.VISIBLE) {          // eğer resimcercevem görünür ise
              resimcercevem.setVisibility(View.INVISIBLE); // görünmez yap
              resimgostergizle.setText("GÖSTER");              // resimgostergizle'nin üzerindeki yazıyı GÖSTER yap
           }else{                                                          // değilse
              resimcercevem.setVisibility(View.VISIBLE);     // görünür yap
              resimgostergizle.setText("GİZLE");                //  resimgostergizle'nin üzerindeki yazıyı GİZLE yap
           }
        }
```

```
    });
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {

    int id = item.getItemId();

    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}
}
```

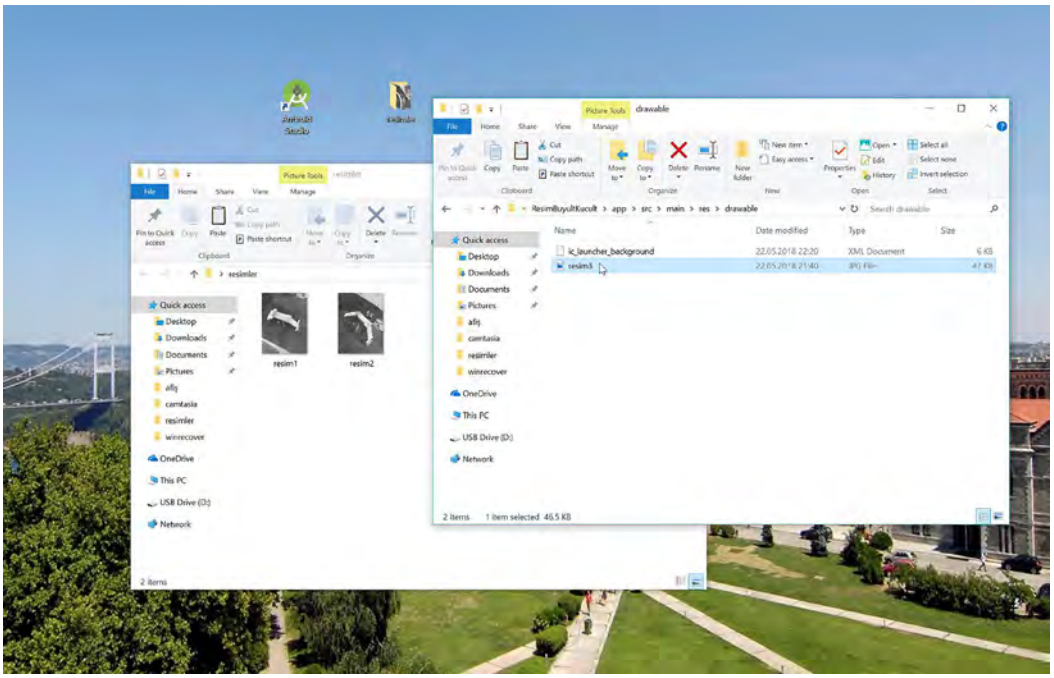
4.3.6. Resim Büyüt/Küçült Uygulaması

Uygulamada resim büyütüp küçültme işlemi yapılacaktır. Projenin adını ResimBuyutKuçult koyunuz. Dosyalar, yine aynı klasörün içinde oluşturulmaktadır.



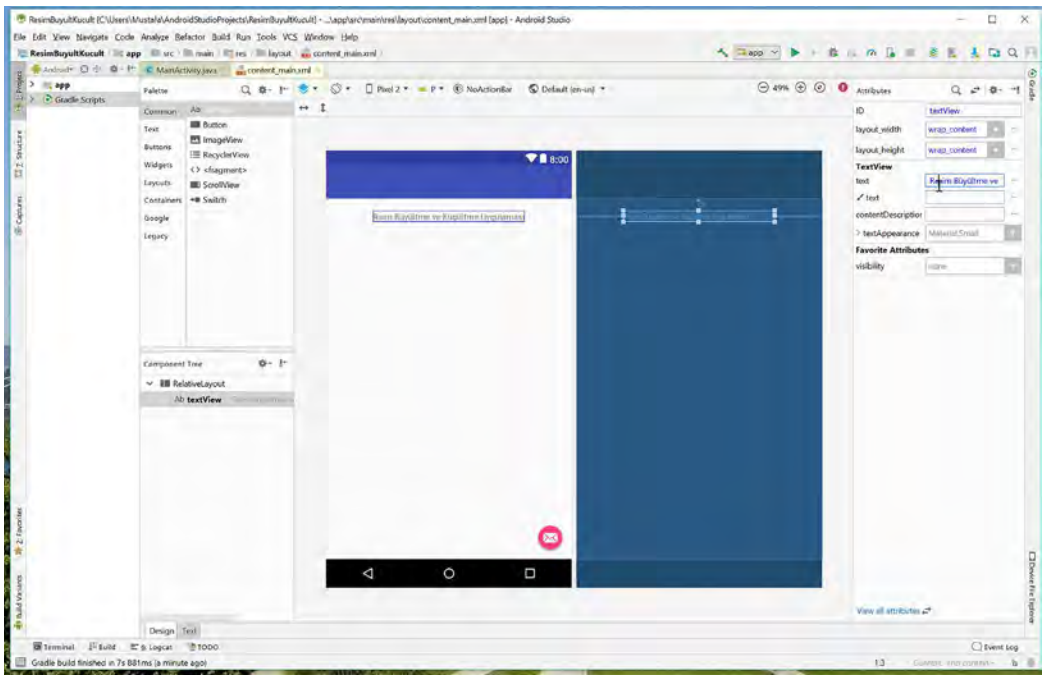
Ekran Görüntüsü 4.218

Resimlerin eklenebilmesi için, yine resmibüyütkücült/app/src/main/res/drawable klasörüne atılması gerekmektedir.



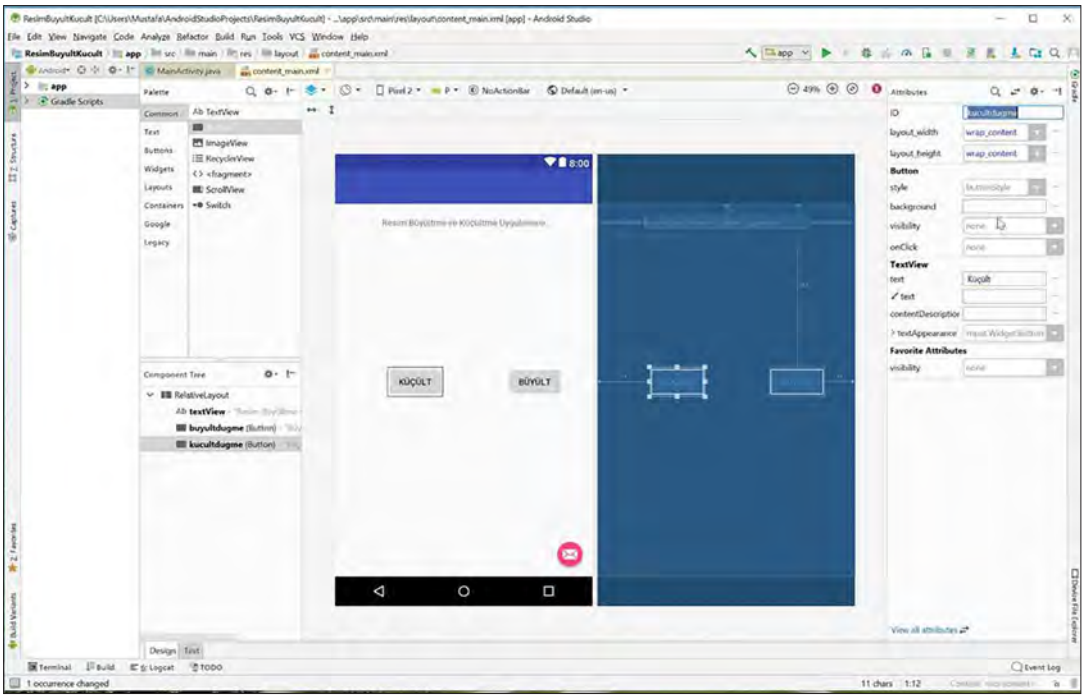
Ekran Görüntüsü 4.219

Android Studio'yu açınız. HelloWorld'u siliniz. Resimlerinizi, daha önceki uygulamalarda olduğu gibi RelativeLayout haline getiriniz. Ardından bir TextView ekleyiniz. "İsmini Resim Büyütüp Küçültme Uygulaması" koyunuz.



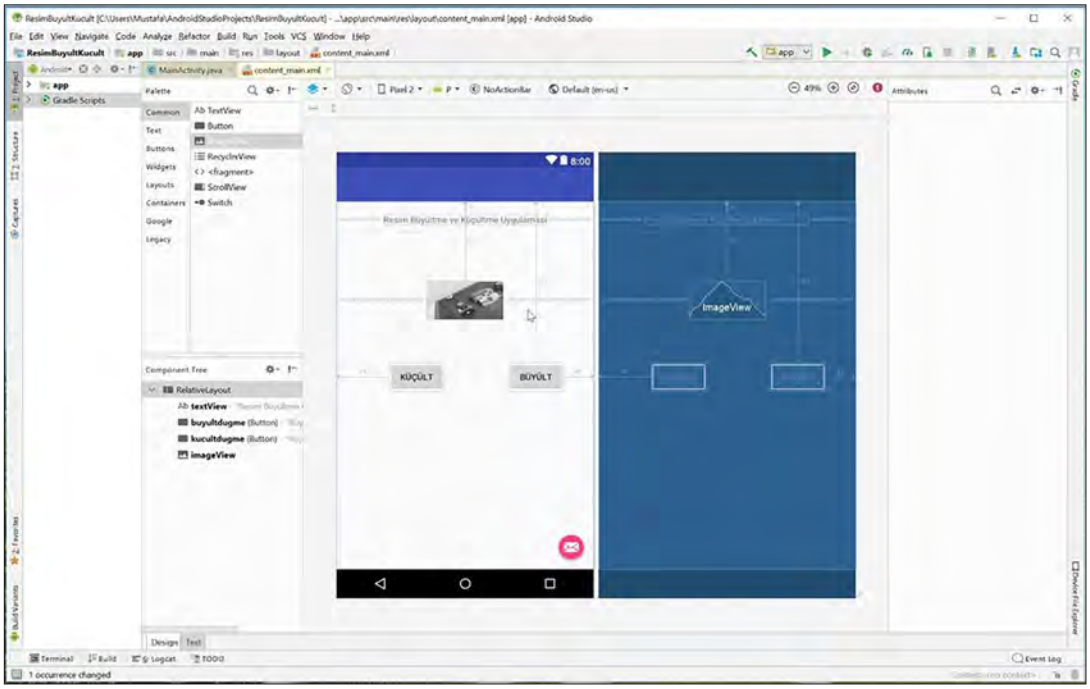
Ekran Görüntüsü 4.220

Ardından 2 adet düğme ekleyiniz. Birinin id'sini buyutdugme, diğerini kucultdugme veriniz.



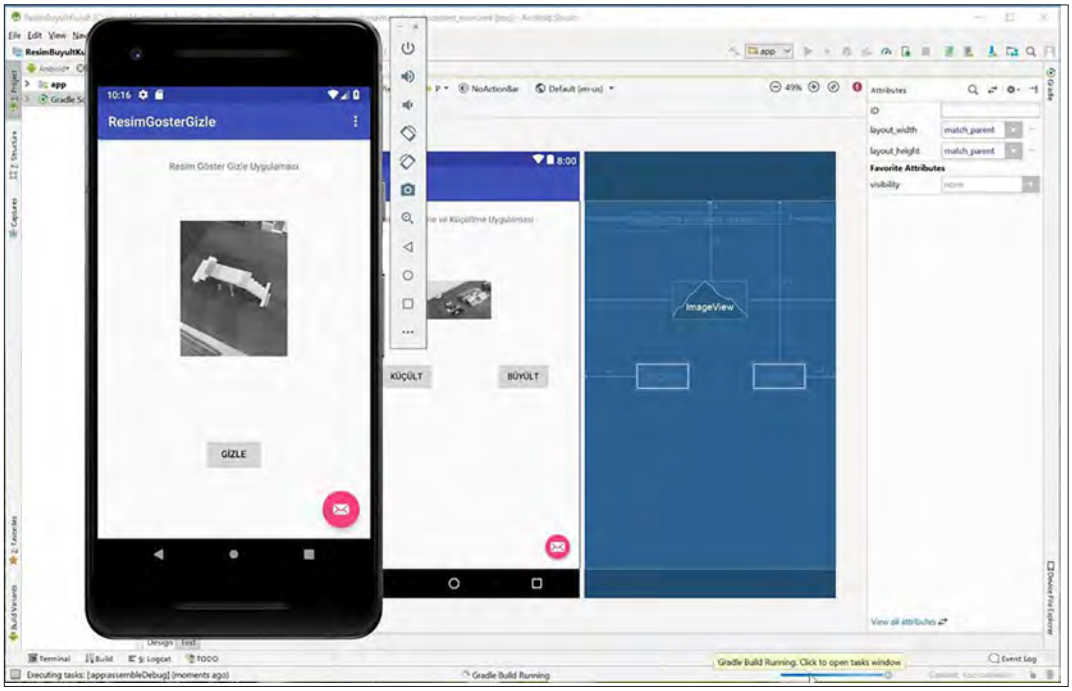
Ekran Görüntüsü 4.221

Daha sonra bir imageview (resim) ekleyiniz ve yerini daha önceki uygulamalarda olduğu gibi fitXY ile ayarlayınız.



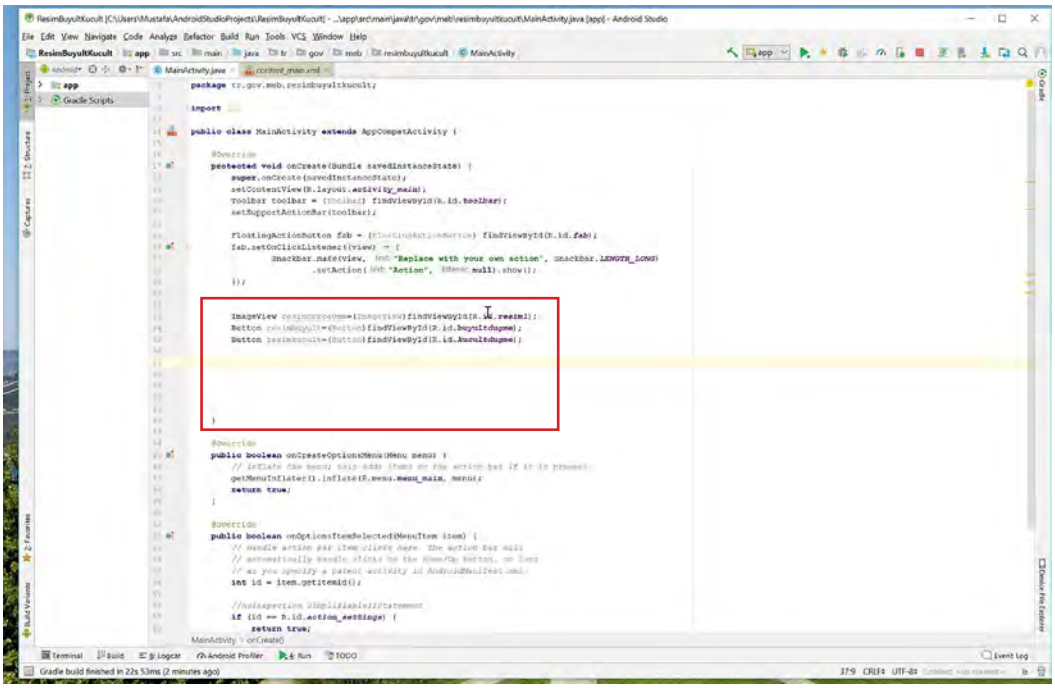
Ekran Görüntüsü 4.222

Programı çalıştırdığınızda emülatörde aşağıdaki görüntü gelmektedir.



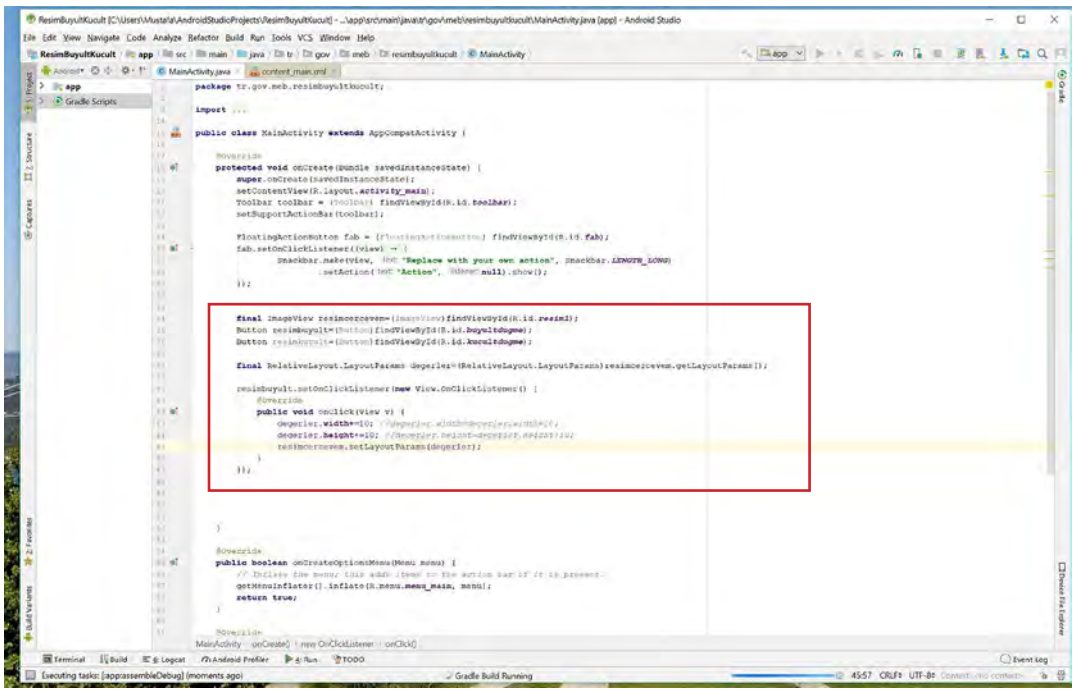
Ekran Görüntüsü 4.223

Programın kodlaması için mainactivity.java'yı açınız ve kodlarınızı yazınız. Öncelikle kodlarla 3 tane nesneyi tanımlayınız.



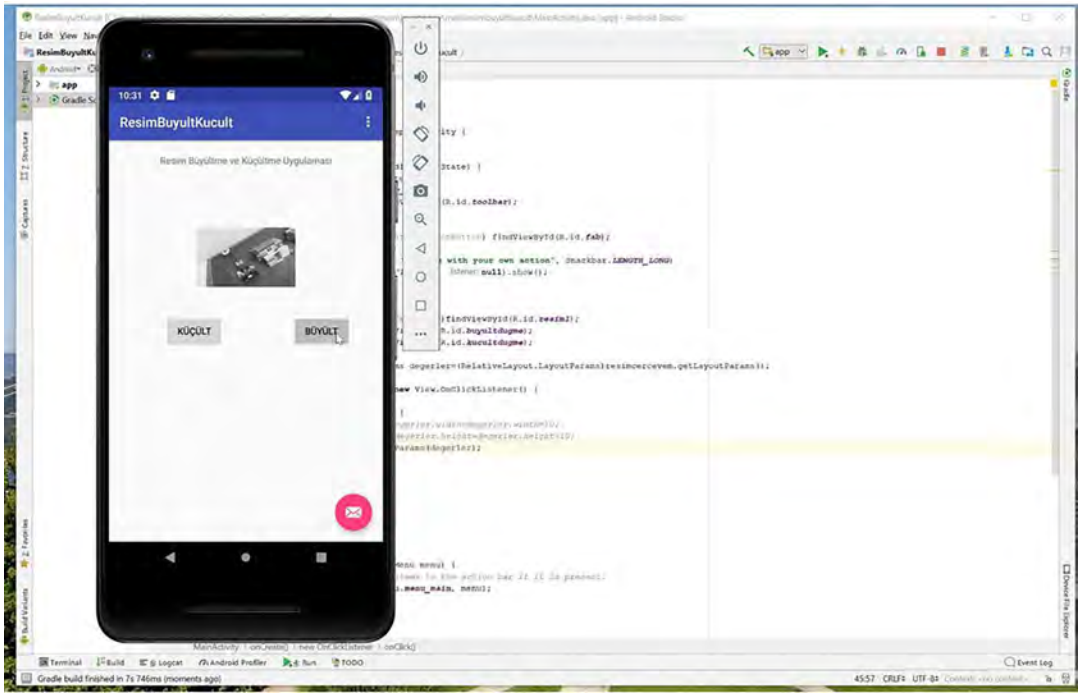
Ekran Görüntüsü 4.224

Resim büyütme ve küçültme için clicklistener, yani tıklama dinleyicisi oluşturulmalıdır.



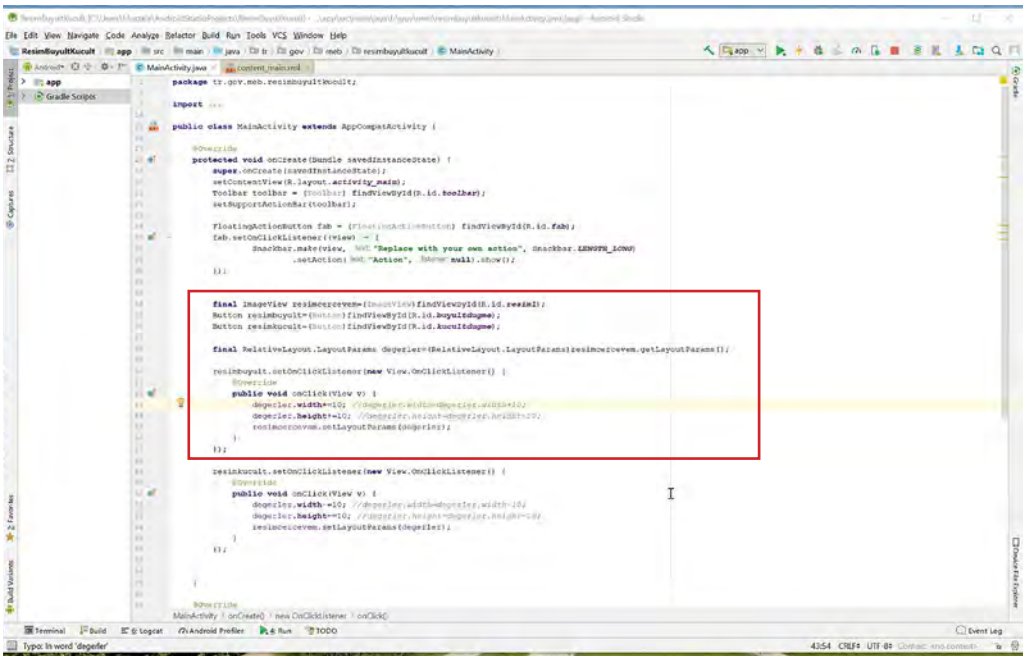
Ekran Görüntüsü 4.225

Proje çalıştırıldığında emülatördeki görüntü aşağıdaki gibi olacak ve büyüt düğmesi çalışacaktır.



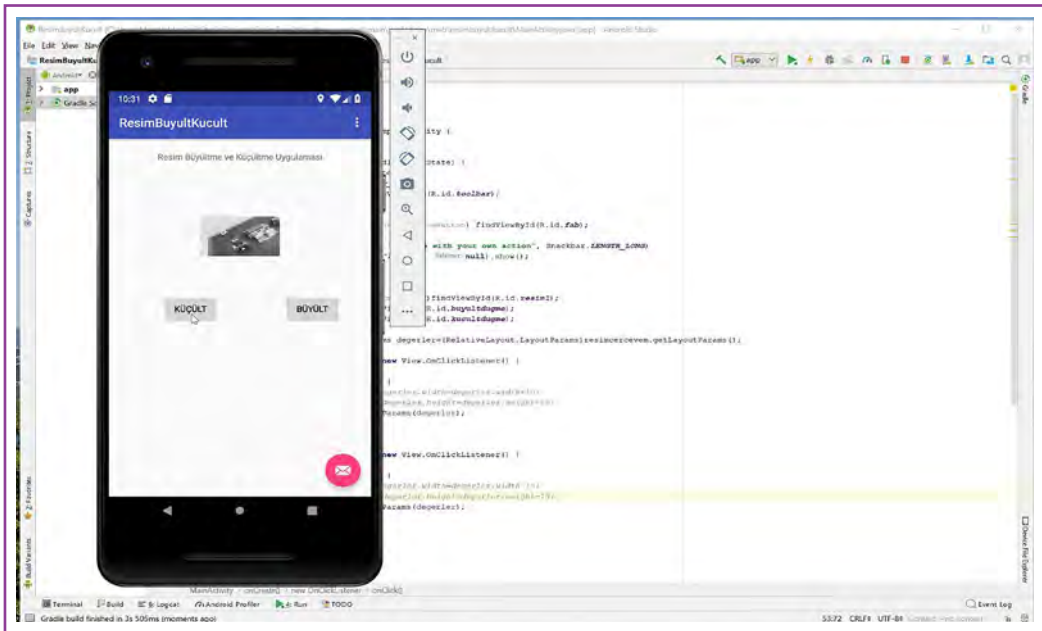
Ekran Görüntüsü 4.226

Küçült düğmesi için de aynı mantık uygulanacaktır.



Ekran Görüntüsü 4.227

Proje çalıştırıldığında küçült düğmesi de çalışacaktır.

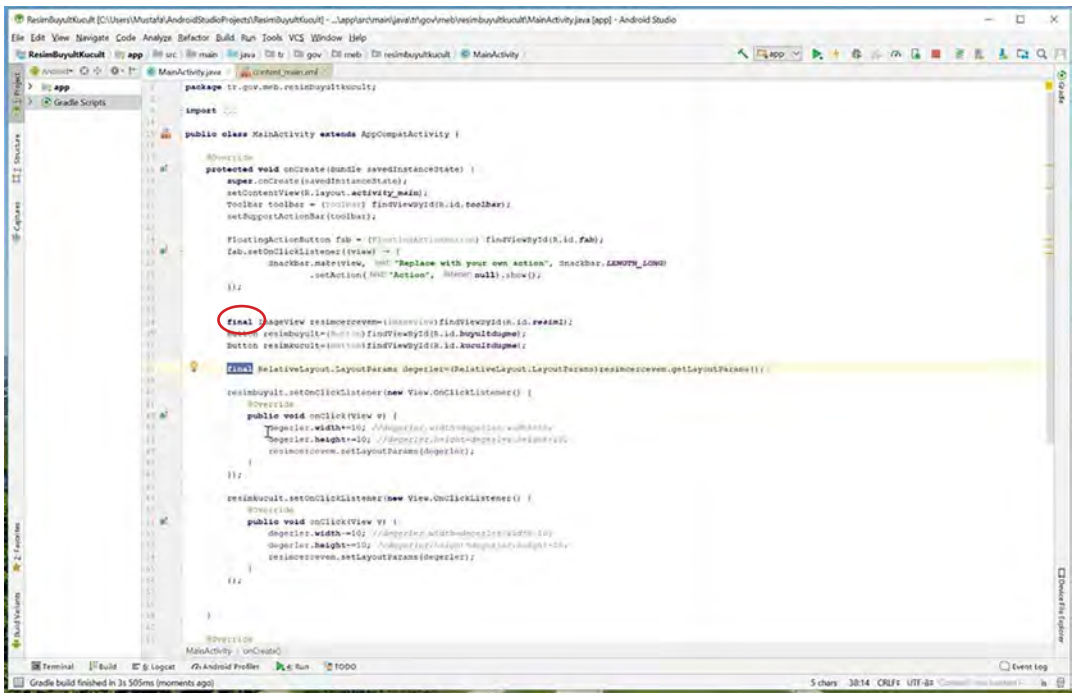


Ekran Görüntüsü 4.228

Burada küçük bir ayrıntı var. Görüldüğü gibi kimi kodların başına "final" yazılmış.

Örneğin "resimcercevem" ve "degerler" nesneleri için final kullanılmış.

Bunun nedeni; eğer bir nesne başka bir nesne içerisindeki bir class içinden çağırılacaksa, bu nesnenin global bir nesneye dönüştürülmesi gerekir. Bu, java'da temel bir mantıktır. 2 nesneyi global olarak tanımlayabilmek için başına final koyulmuştur. Aksi halde nesnelerimiz çağırılamaz.



Ekran Görüntüsü 4.229

XML Kodları

```
<?XML version="1.0" encoding="utf-8"?>
<RelativeLayout XMLns:android="http://schemas.android.com/apk/res/android"
  XMLns:app="http://schemas.android.com/apk/res-auto"
  XMLns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  app:layout_behavior="@string/appbar_scrolling_view_behavior"
  tools:context=".MainActivity"
  tools:showIn="@layout/activity_main">

  <TextView
    android:id="@+id/TextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="20dp"
    android:text="Resim Büyültme ve Küçültme Uygulaması" />

  <Button
    android:id="@+id/buyultdugme"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
```

```
    android:layout_alignParentTop="true"
    android:layout_marginEnd="48dp"
    android:layout_marginTop="263dp"
    android:text="Büyült" />

  <Button
    android:id="@+id/kucultdugme"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignTop="@+id/buyultdugme"
    android:layout_marginStart="84dp"
    android:text="Küçült" />

  <ImageView
    android:id="@+id/resim1"
    android:layout_width="122dp"
    android:layout_height="63dp"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="131dp"
    android:scaleType="fitXY"
    app:srcCompat="@drawable/resim3" />
</RelativeLayout>
```

Java Kodları

```
package tr.gov.meb.resimbuyultkucult;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.RelativeLayout;

public class MainActivity extends AppCompatActivity {

  @Override
```

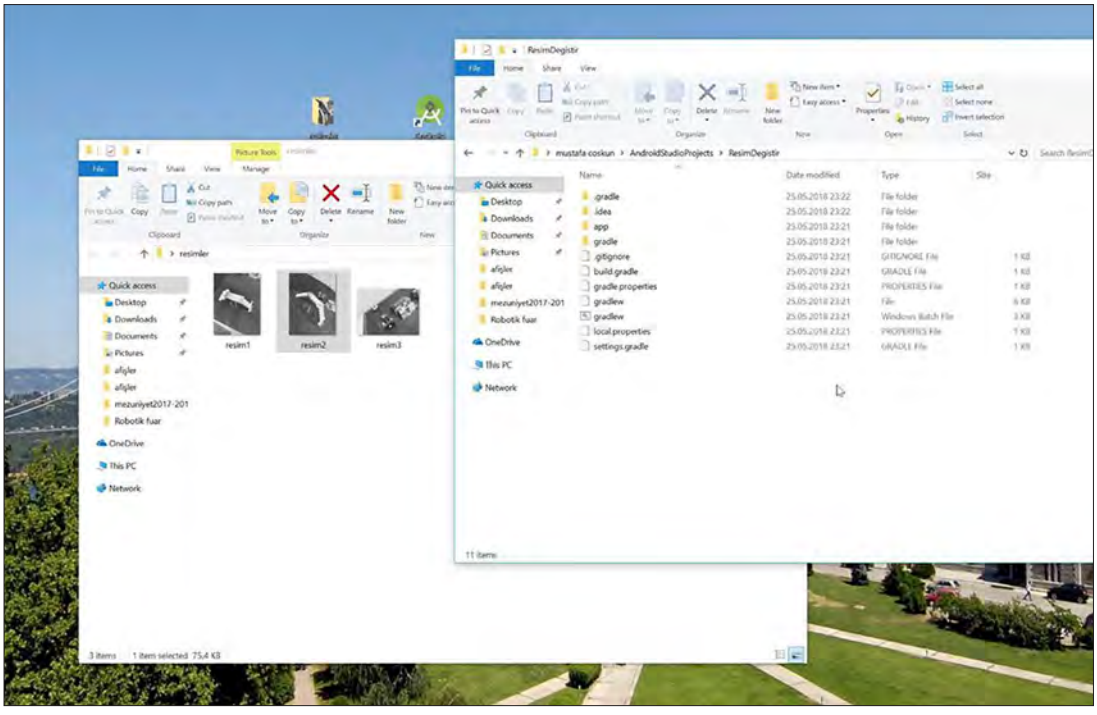
```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
    fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
        }
    });

    final ImageView resimcercevem=(ImageView)findViewById(R.id.resim1);
    Button resimbuyult=(Button)findViewById(R.id.buyultdugme);
    Button resimkucult=(Button)findViewById(R.id.kucultdugme);

    final RelativeLayout.LayoutParams degerler=(RelativeLayout.LayoutParams)resimcercevem.getLayoutParams();

    resimbuyult.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {                          //resimbuyult tıklandığında
            degerler.width+=10;                                // degerler.width=degerler.width+10;
            degerler.height+=10;                               //degerler.height=degerler.height+10;
            resimcercevem.setLayoutParams(degerler);
        }
    });

    resimkucult.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {                          //resimkucult tıklandığında
            degerler.width-=10;                                //degerler.width=degerler.width-10;
            degerler.height-=10;                               //degerler.height=degerler.height-10;
            resimcercevem.setLayoutParams(degerler);
        }
    });
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

4.3.7. Resim Değiştirme Uygulaması

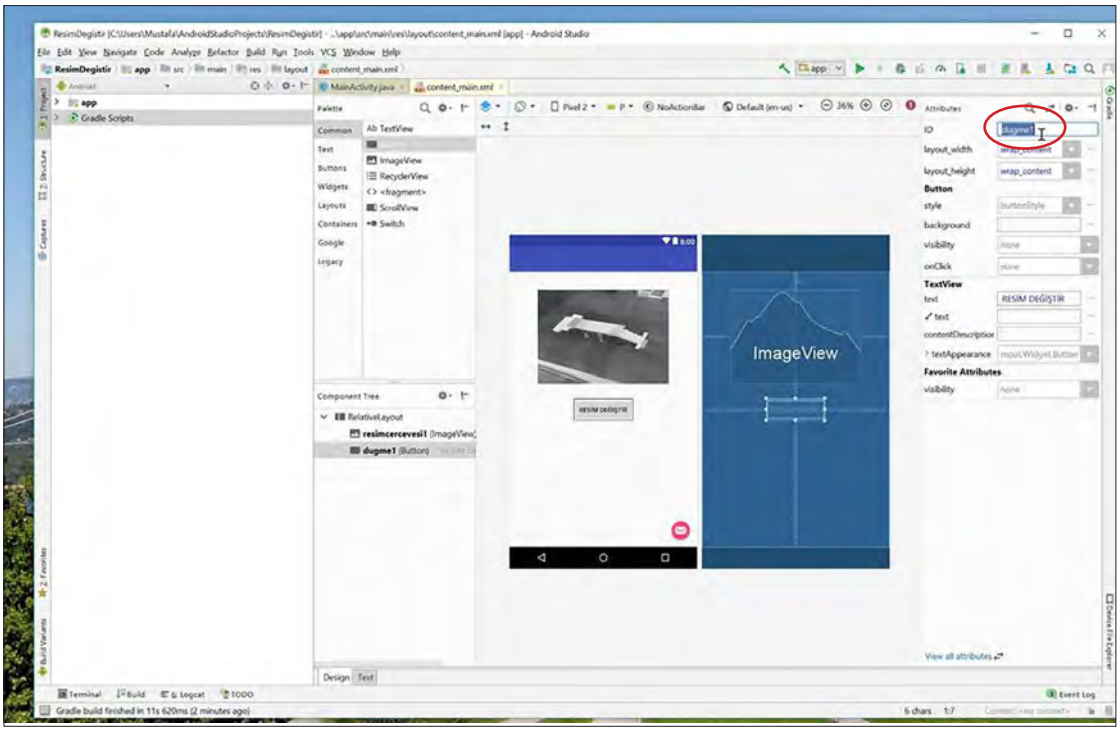
Sıradaki uygulamada resim değiştirme yapılacaktır. Bir düğmeye tıklandığında bir resimden diğerine dönüşmesi sağlanacaktır. Projenin adını ResimDegistir koyunuz. Dosyalar, yine aynı klasörün içinde oluşturuldu.

Resimleri ilgili klasöre (resimdegistir/app/src/main/res /drawable) atınız.



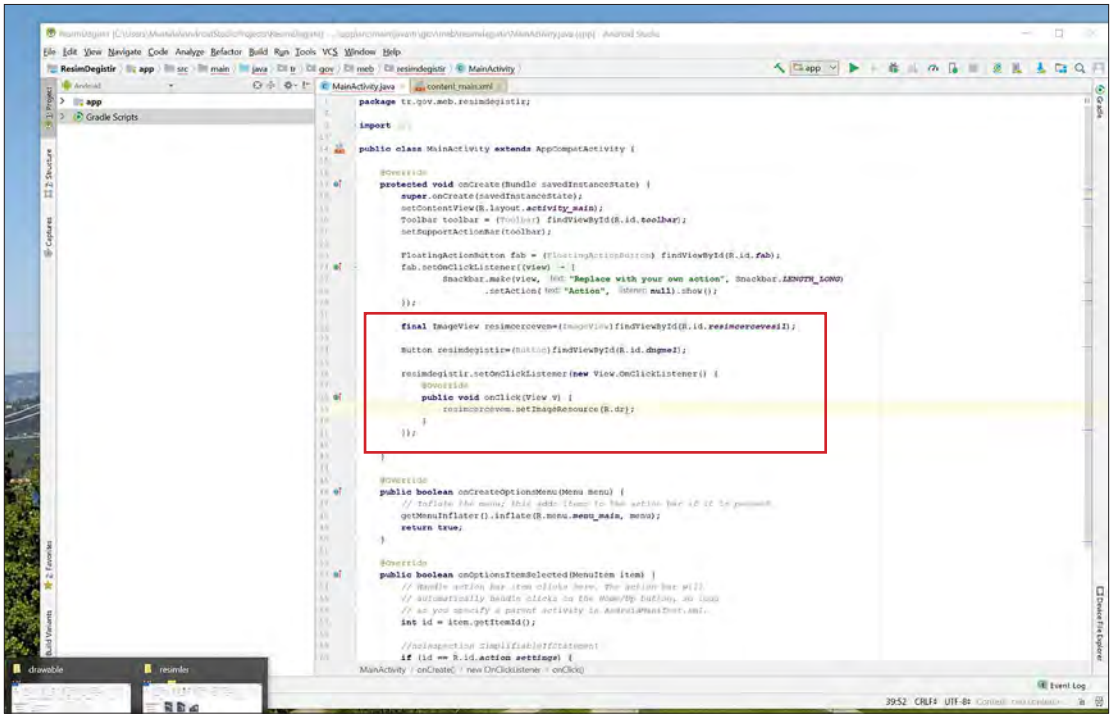
Ekran Görüntüsü 4.230

Android Studio'yu açınız. Her zamanki gibi helloworld'ü siliniz. Resimlerinizi, daha önceki uygulamalarda açıklandığı gibi relative olarak ayarlayınız. Bir ImageView ve Buton ekleyiniz. ImageView'in id'sini resimcercevesi1 koyunuz. Butonun ekranda görünen yazısını Resim Değiştir, id'si ise dugme1 yapınız.



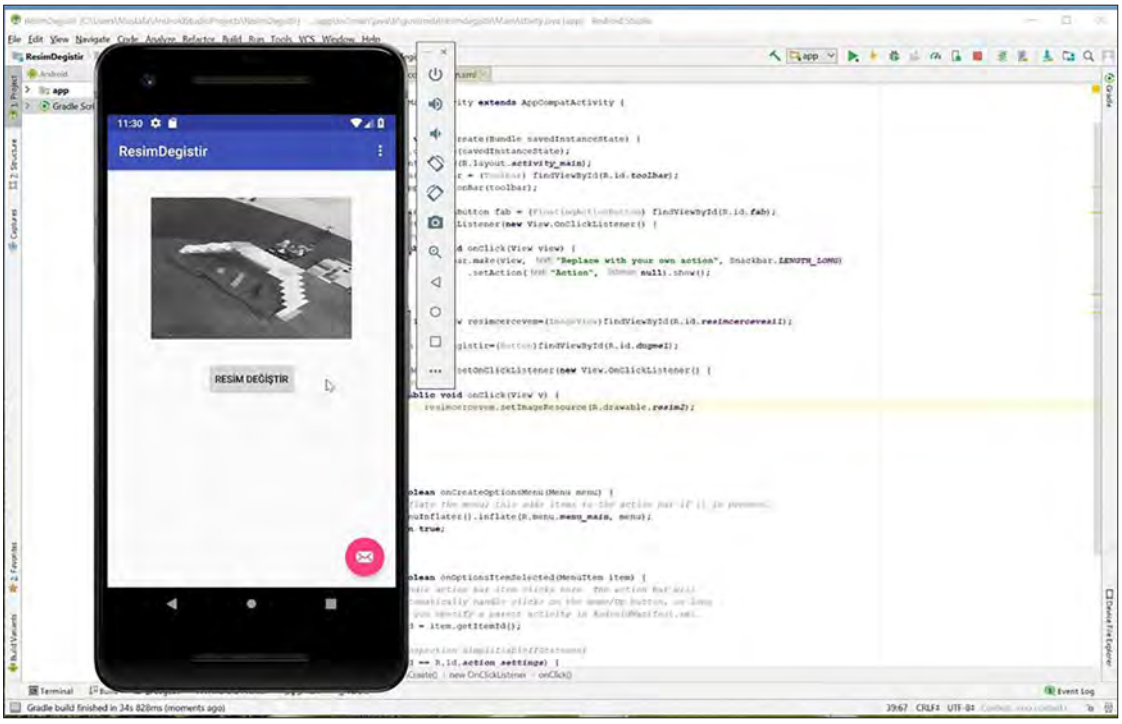
Ekran Görüntüsü 4.231

Kodlarını yazmak için mainactivity.java'ya geliniz ve kodlarınızı yazınız. Öncelikle düğmeye tıklandığında oluşturduğunuz resimcercevem id'sini resim2 yap olan kodu ekleyeniz.



Ekran Görüntüsü 4.232

Proje çalıştırıldığında emülatördeki görüntü aşağıdaki gibidir.



Ekran Görüntüsü 4.233

XML Kodları

```
<?XML version="1.0" encoding="utf-8"?>
<RelativeLayout XMLns:android="http://schemas.android.com/apk/res/android"
  XMLns:app="http://schemas.android.com/apk/res-auto"
  XMLns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  app:layout_behavior="@string/appbar_scrolling_view_behavior"
  tools:context=".MainActivity"
  tools:showIn="@layout/activity_main">

  <ImageView
    android:id="@+id/resimcercevesi1"
    android:layout_width="287dp"
    android:layout_height="205dp"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="40dp"
    android:scaleType="fitXY"
    app:srcCompat="@drawable/resim1" />
```

```
  <Button
     android:id="@+id/dugme1"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_centerInParent="true"
     android:text="RESİM DEĞİŞTİR" />
</RelativeLayout>
```

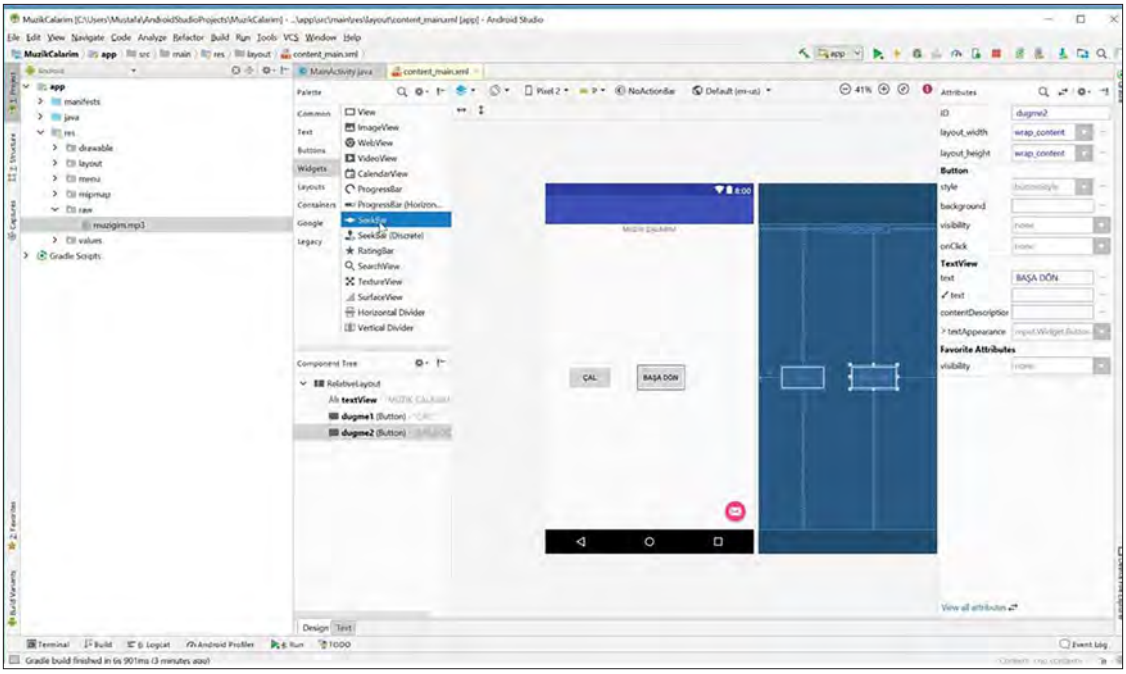
Java Kodları

```
package tr.gov.meb.resimdegistir;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
     setSupportActionBar(toolbar);

     FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
     fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
           Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                 .setAction("Action", null).show();
        }
     });

     final ImageView resimcercevem=(ImageView)findViewById(R.id.resimcercevesi1);

     Button resimdegistir=(Button)findViewById(R.id.dugme1);
```

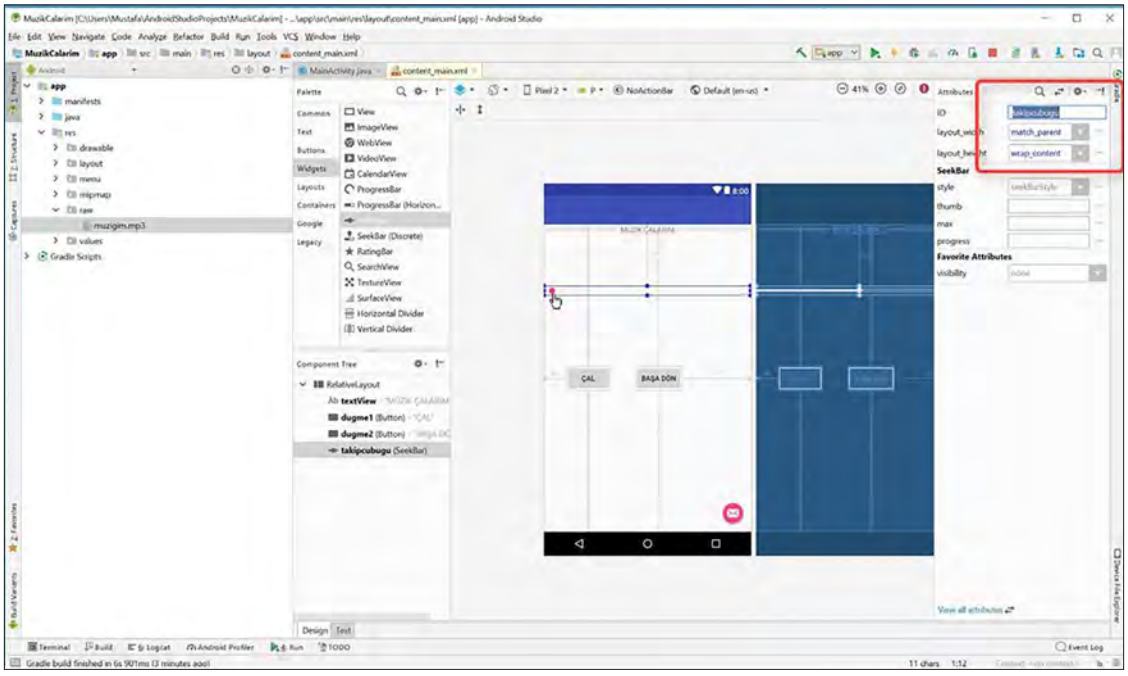
```
    resimdegistir.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {                          //resimdegistir tıklandığında
         resimcercevem.setImageResource(R.drawable.resim2);   //resimcercevem tutucusunun kaynağını resim2 yap
        }
    });

  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
  }

  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.XML.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
      return true;
    }

    return super.onOptionsItemSelected(item);
  }
}
```

4.3.8. Müzik Çalar Uygulaması

Bu uygulamada Progress Bar'lı bir müzik çalar yapılacaktır. Widget'ların arasından SeekBar'ı ekrana sürükleyiniz ve SeekBar'ın adını takipcubugu olarak belirleyiniz.



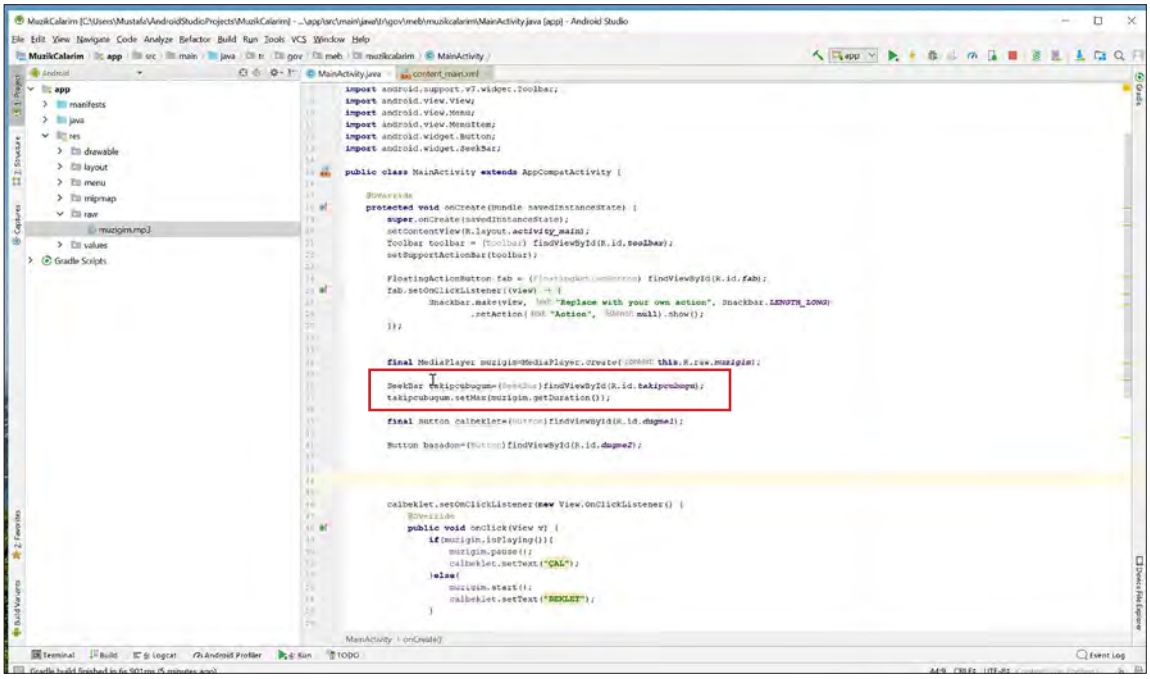
Ekran Görüntüsü 4.234



Ekran Görüntüsü 4.235

SeekBar'ı tanıtmak için MainActivity'ye geliniz ve aşağıdaki kodları yazınız.

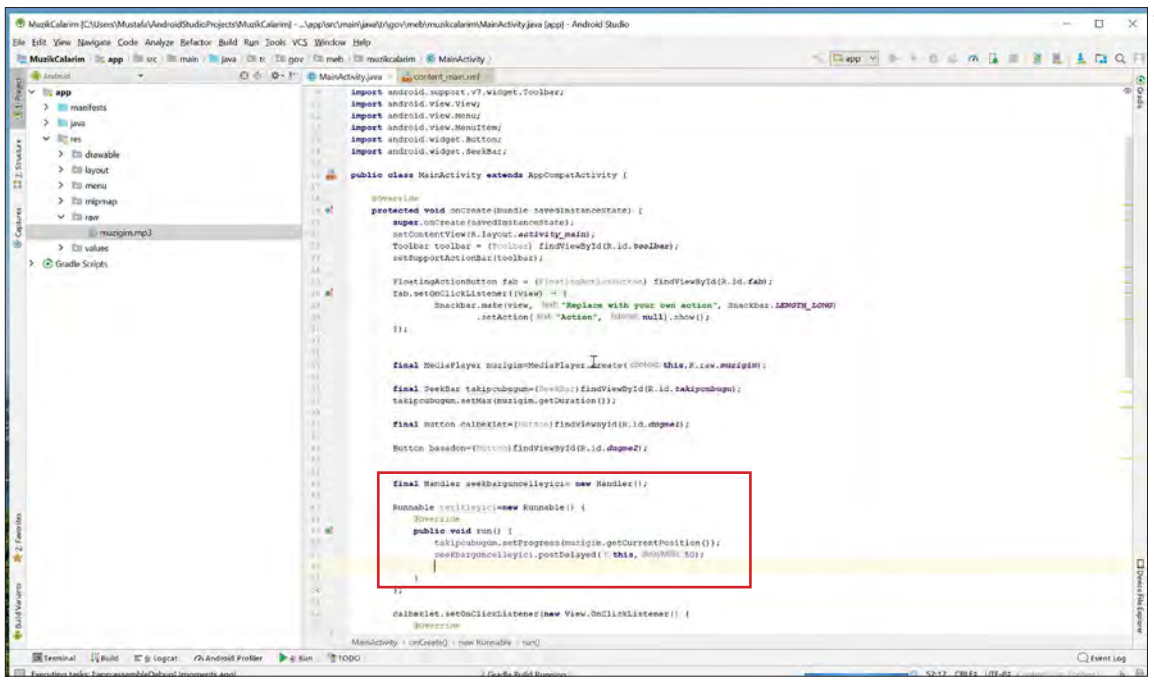
Öncelikle takip çubuğunun max özelliğini ayarlamalısınız. Müziğim nesnesinin duration'u, yani uzunluğu kadar yapınız ve ikisini eşitleyiniz.



Ekran Görüntüsü 4.236

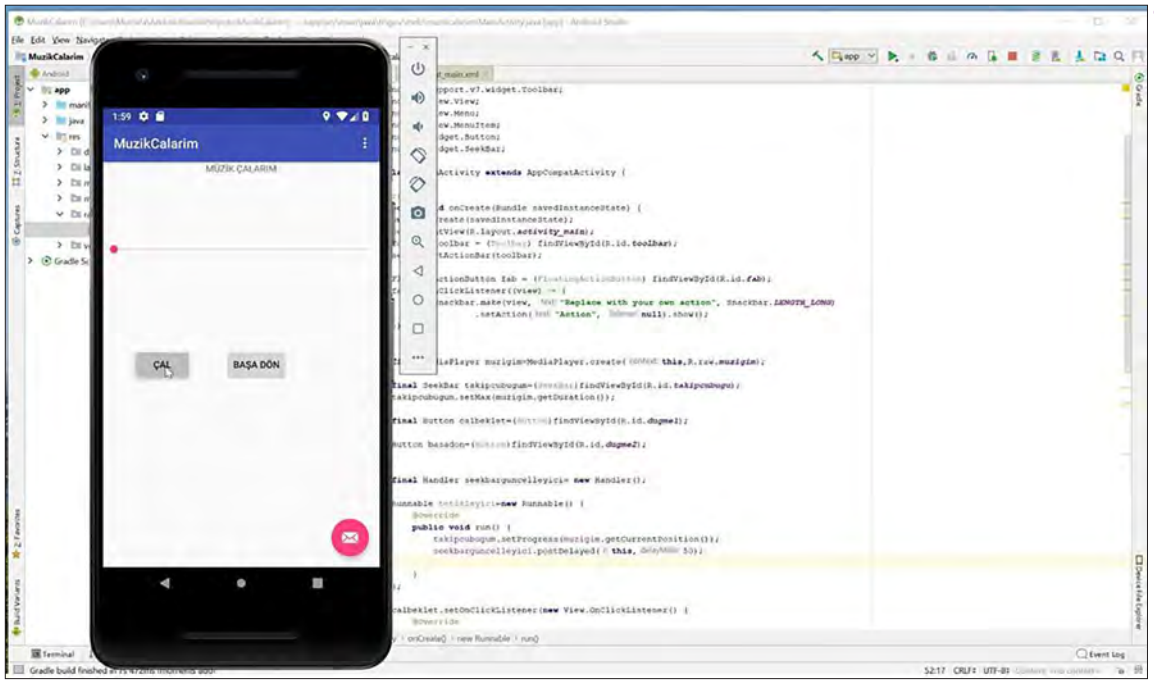
Takip çubuğunun, oluşturduğumuz müzik nesnesini takip edebilmesi için handler ve runable isimli iki nesne oluşturulmalıdır. Handler takip çubuğunu takip edecek, runable ise takip çubuğunu sürekli güncelleyecektir. Handler, nesneleri kontrol etmek için sık kullanılan bir nesnedir ve başa çıkan, üstesinden gelen anlamına gelmektedir.

Peki ekranda sürekli kontrol edilecek olan nedir? Takip çubuğunun progress'i, müziğin o andaki pozisyonu ile eşitlenmesini ve bu runable nesnesi olduğu için her an bu işlemi yapması sağlanmalıdır. Her an bu işlemi tekrarlamak mobil cihazı yoracağı için, bunun yerine handler'ı delay ile geciktirmeliyiz. Böylelikle sürekli değil de 50 milisaniyede bir tetikleme işlemi yapmasını sağlayacağız.



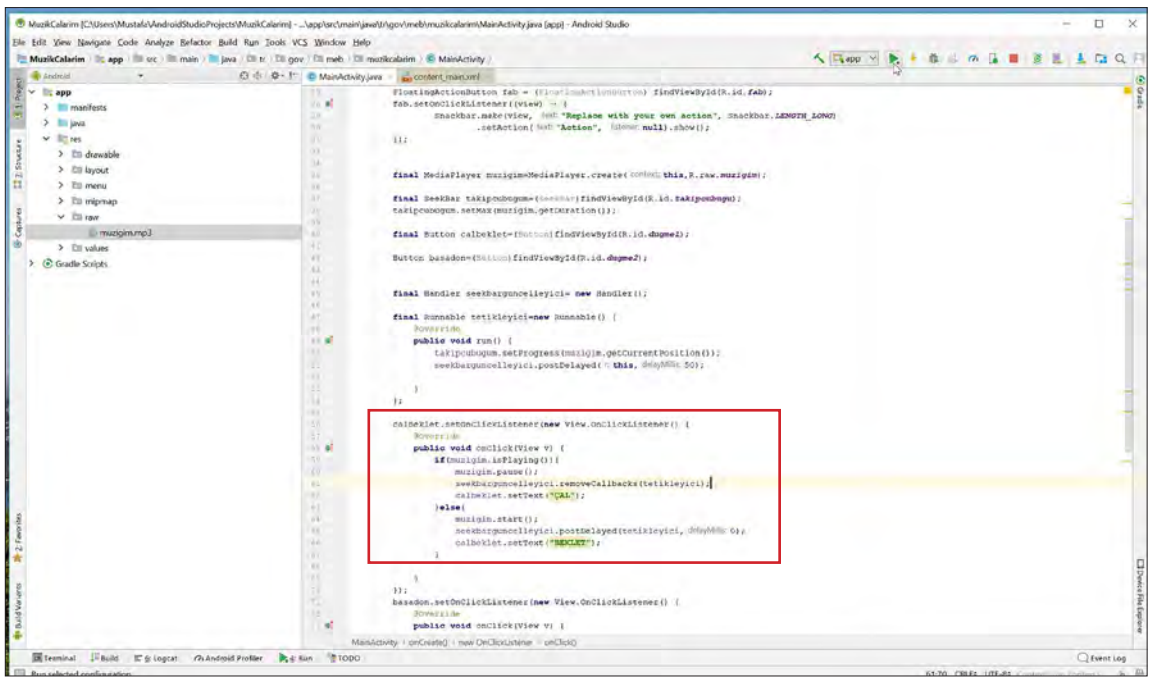
Ekran Görüntüsü 4.237

Bu şekilde programı çalıştırıp tetikleme işlemini yapıyor mu kontrol ediniz. Handler tetiklenmediği için progressbar ilerlemez, runable nesnesi çalışmaz.



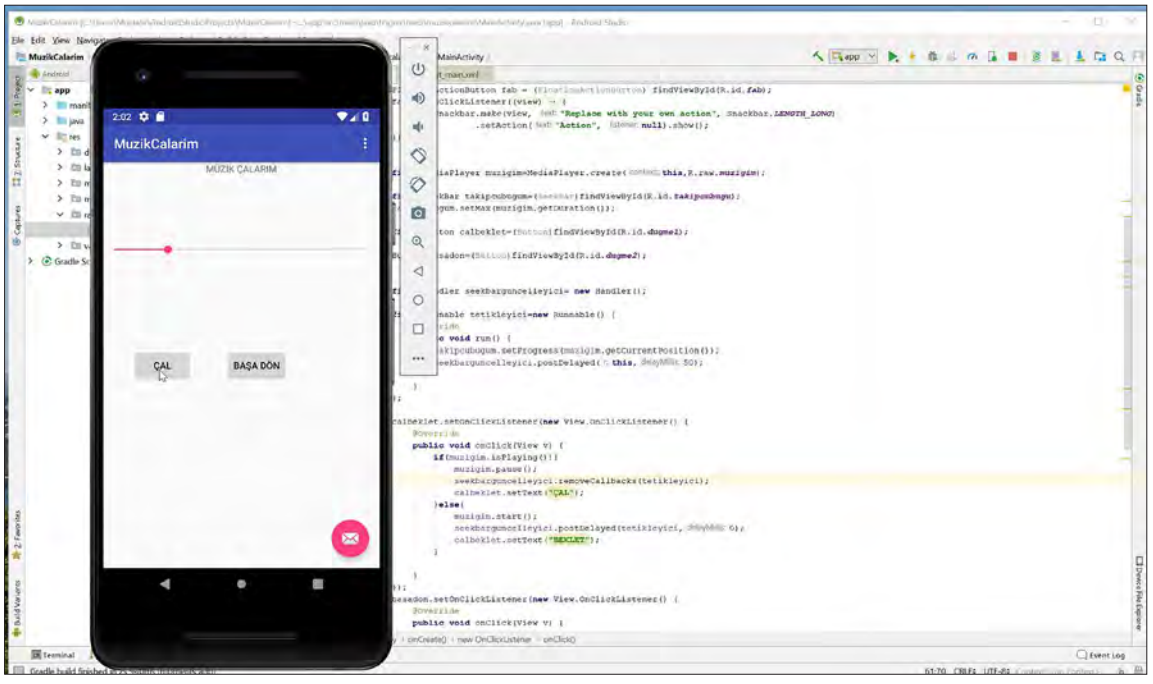
Ekran Görüntüsü 4.238

Müziğin başladığı anda handler'ı tetiklenmesi sağlanmalıdır. Runable durumu overload olana kadar kendisi ilerleyecektir. "Dur" düğmesine tıklandığında da sürekli tetiklemeyi durdurmalısınız.

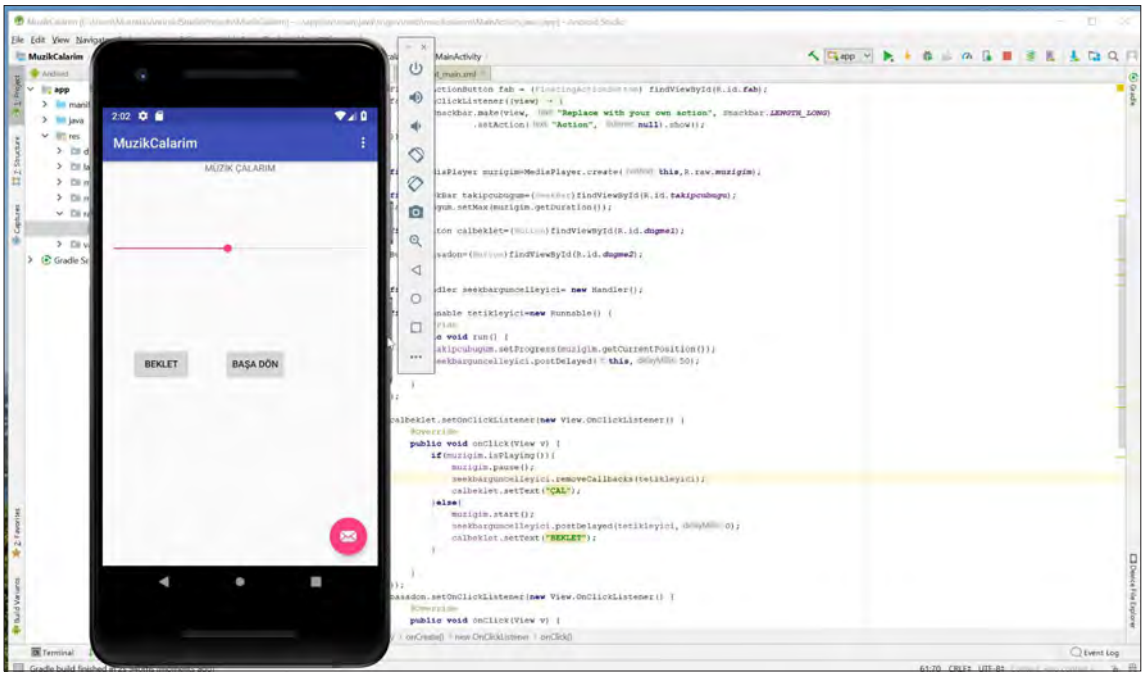


Ekran Görüntüsü 4.239

Uygulama çalıştırıldığında SeekBar da ilerleyecektir.

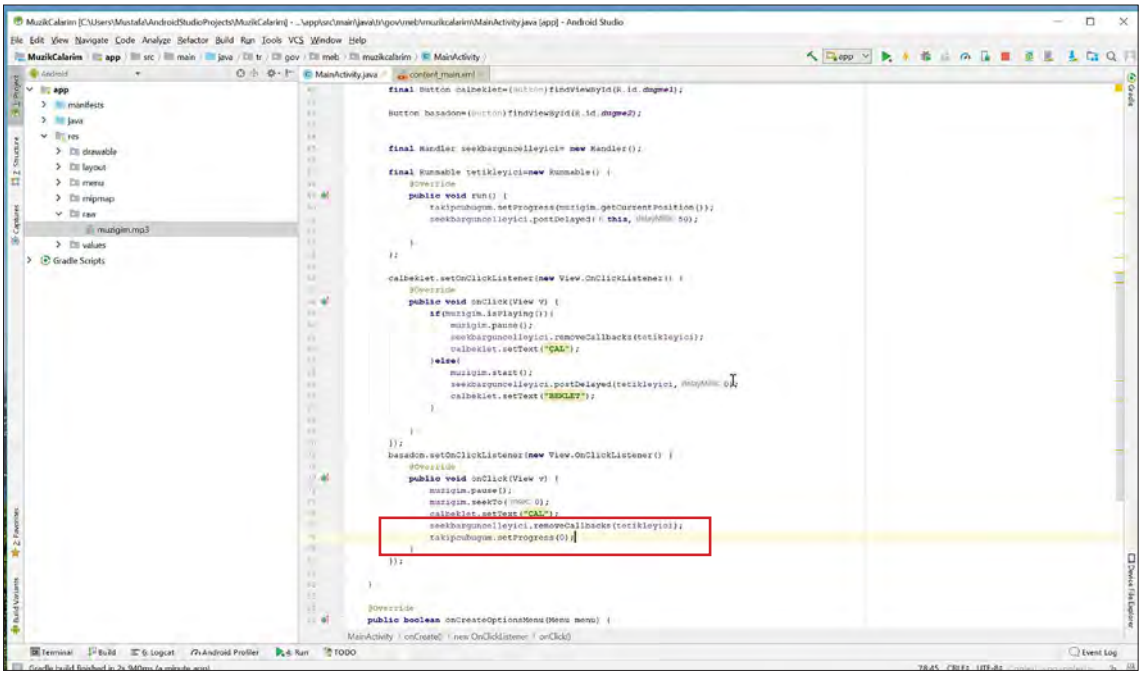


Ekran Görüntüsü 4.240



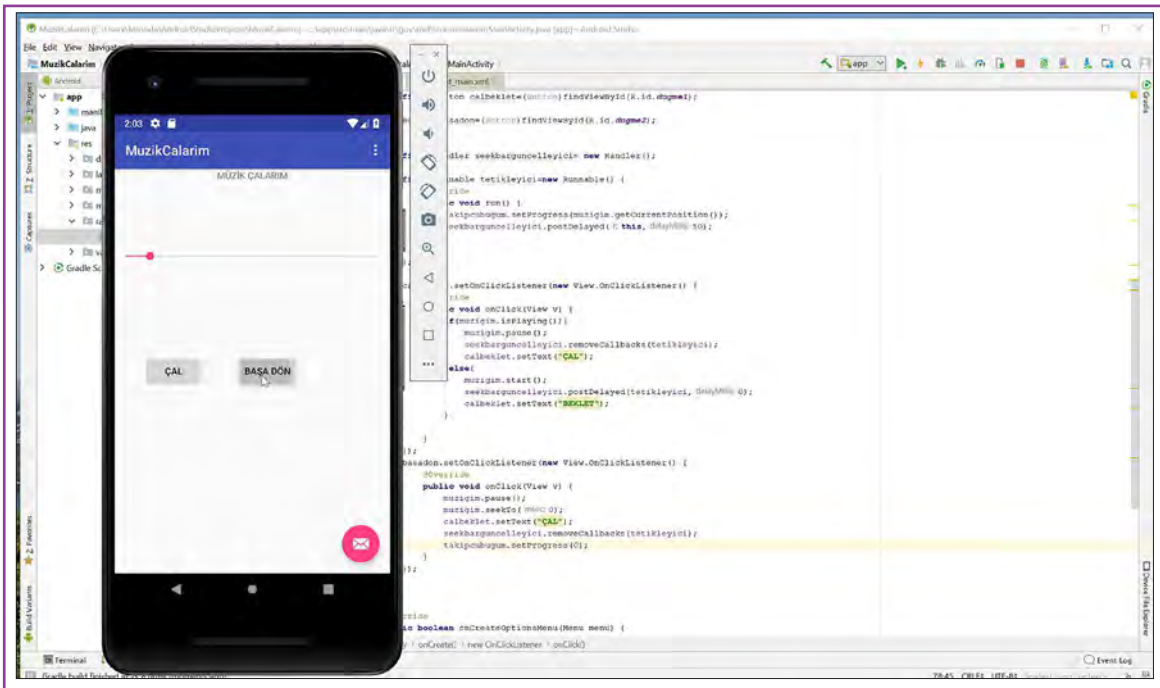
Ekran Görüntüsü 4.241

Bu şekilde Başa Dön düğmesine basıldığında müzik başa döner fakat SeekBar başa dönmez. Bunun için yazılacak komutlar aşağıdaki gibidir.

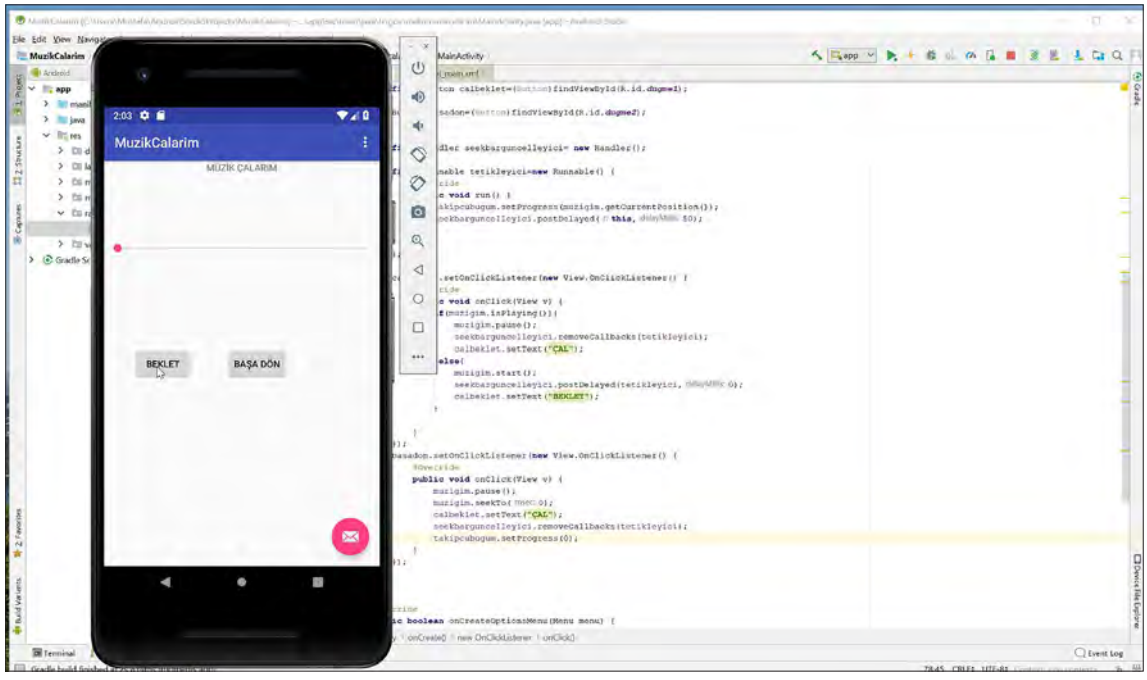


Ekran Görüntüsü 4.242

Uygulama çalıştırıldığında artık SeekBar da güncellenecektir.

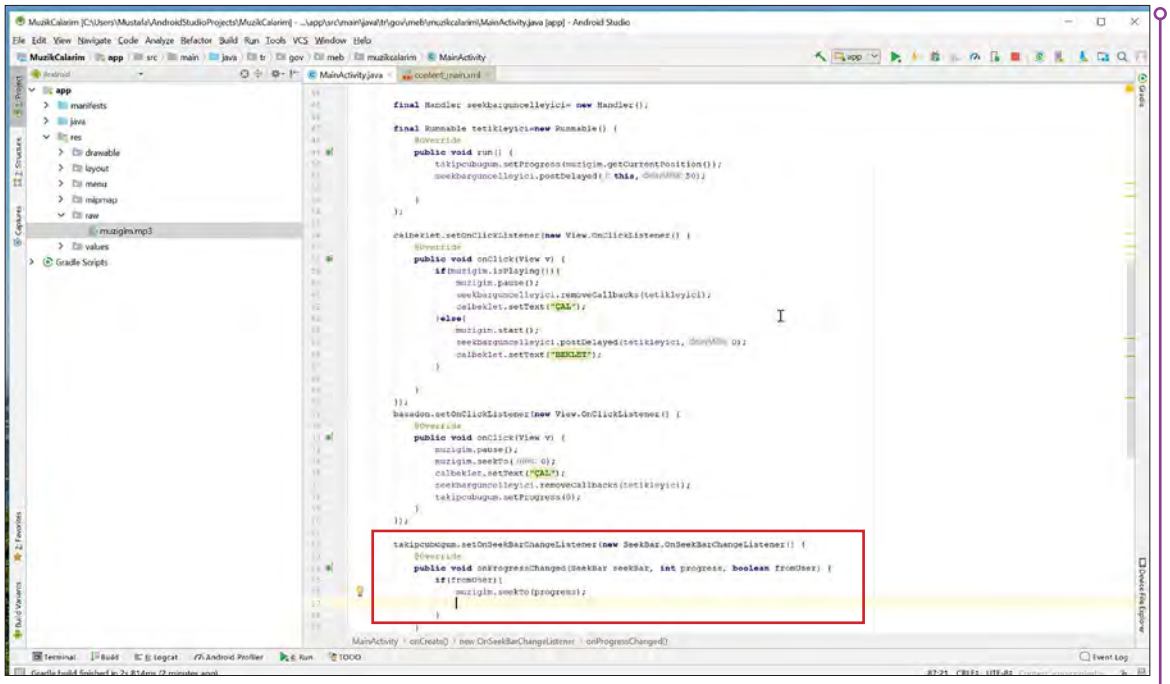


Ekran Görüntüsü 4.243



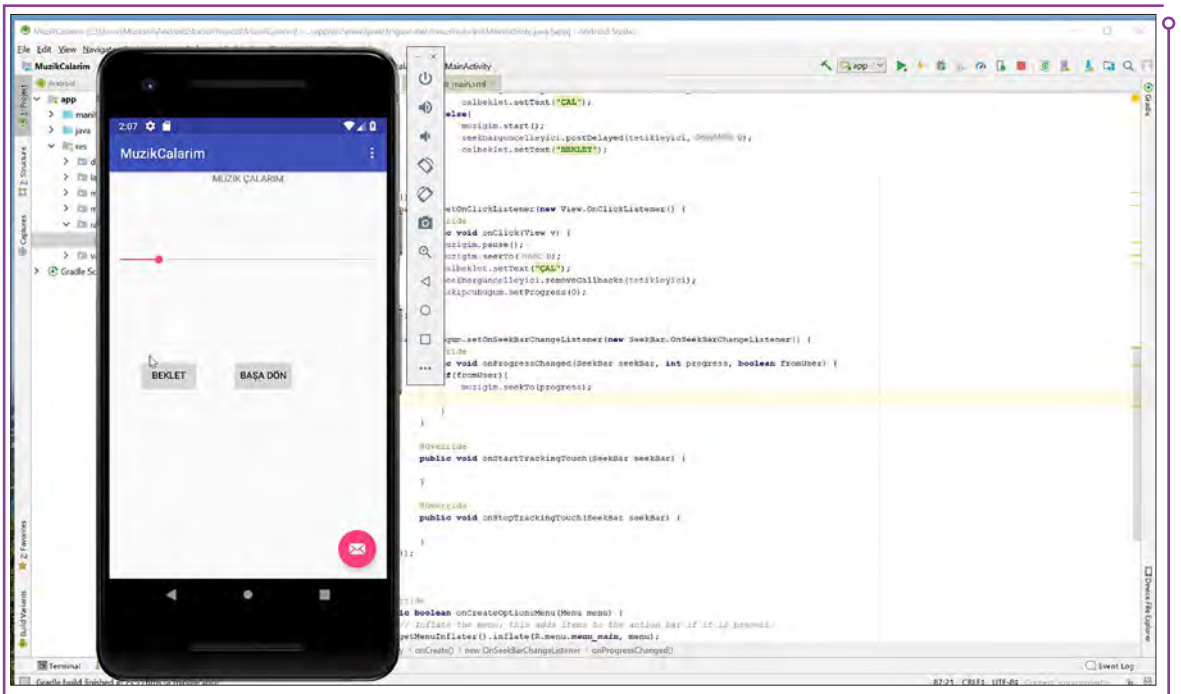
Ekran Görüntüsü 4.244

Kullanıcı SeekBar'ı sürükletip ilerlediğinde, müziğin de aynı şekilde ilerlemesini sağlamak için aşağıdaki kodlar eklenmelidir.



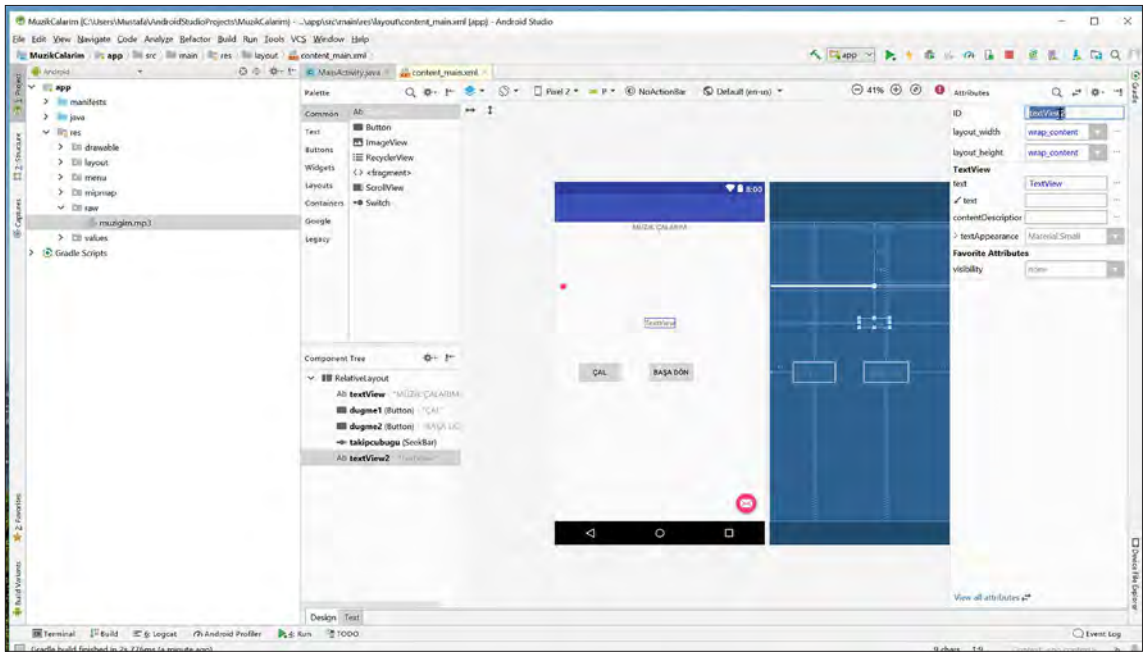
Ekran Görüntüsü 4.245

Artık SeekBar'ı ilerlettiğinizde müzik de ilerler.



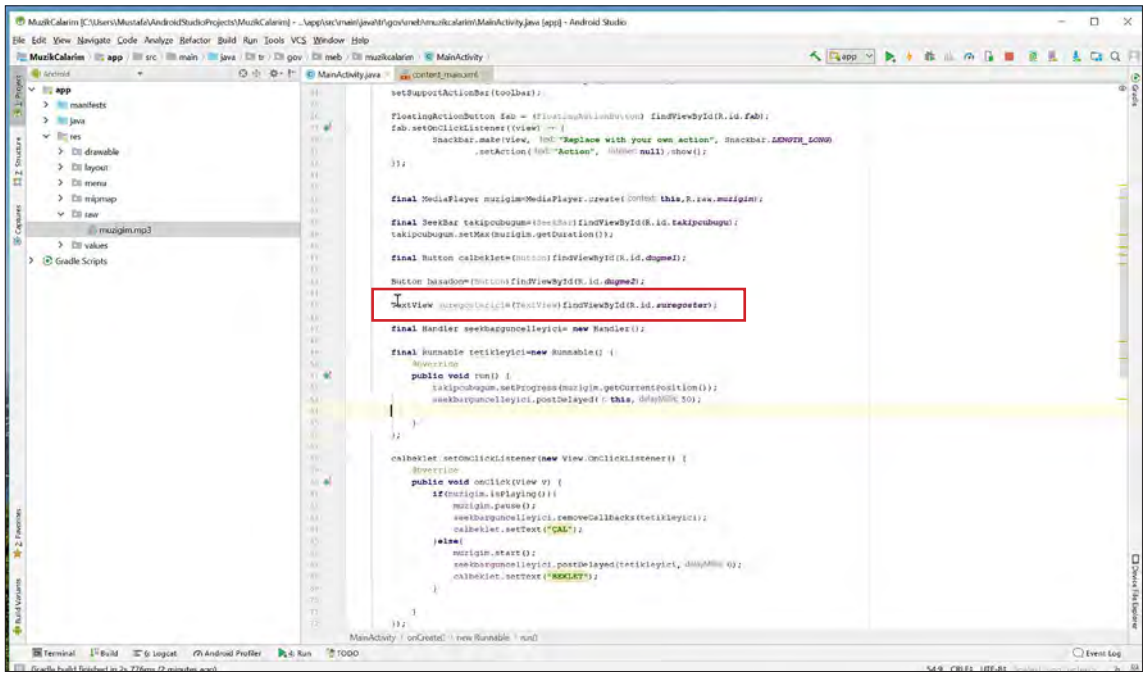
Ekran Görüntüsü 4.246

Süreyi gösterme işlemi için bir TextView ekleyiniz.



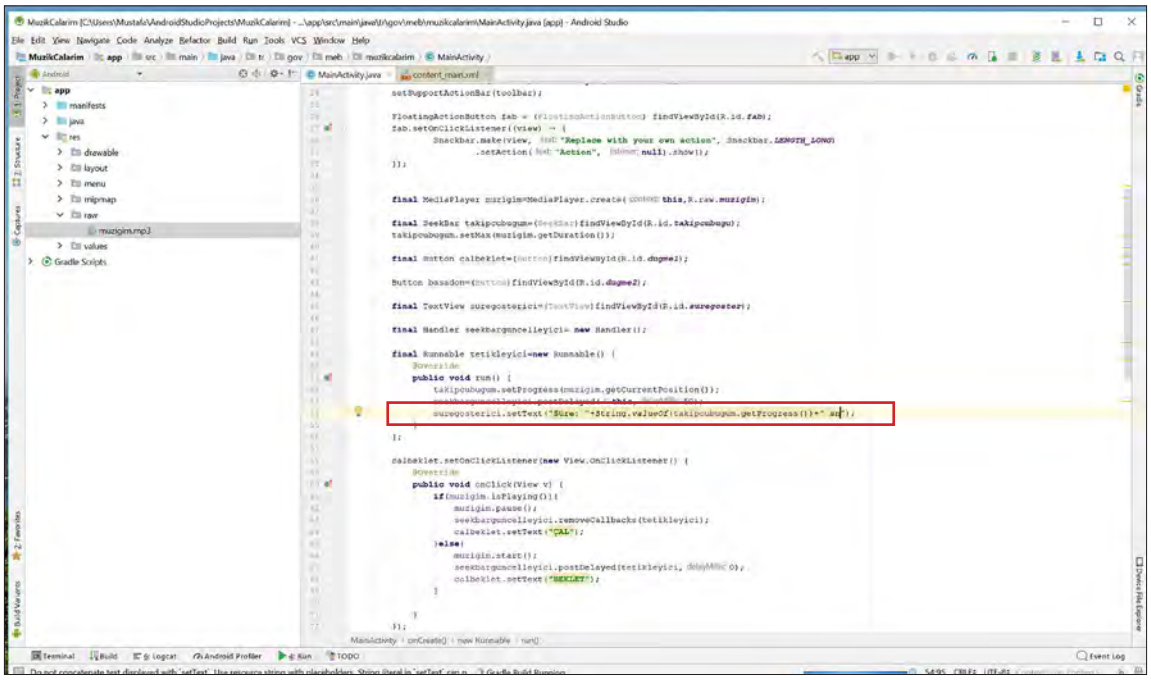
Ekran Görüntüsü 4.247

MainActivity'ye gelip kodlarınızı yazınız. Öncelikle TextView tanıtılmalıdır.



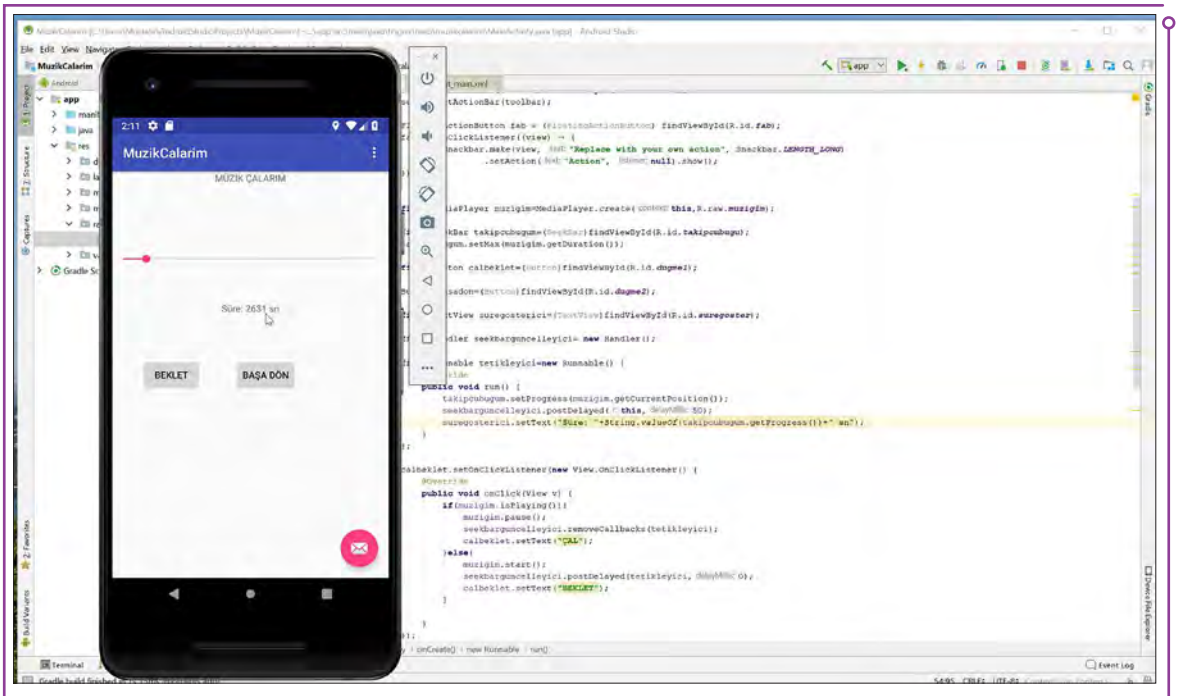
Ekran Görüntüsü 4.248

Takip çubuğunun ilerleme durumunu gösterecek bir süre ayarlayınız. İlerleme durumu değiştiğinde suregoster nesnesi de değişecektir.



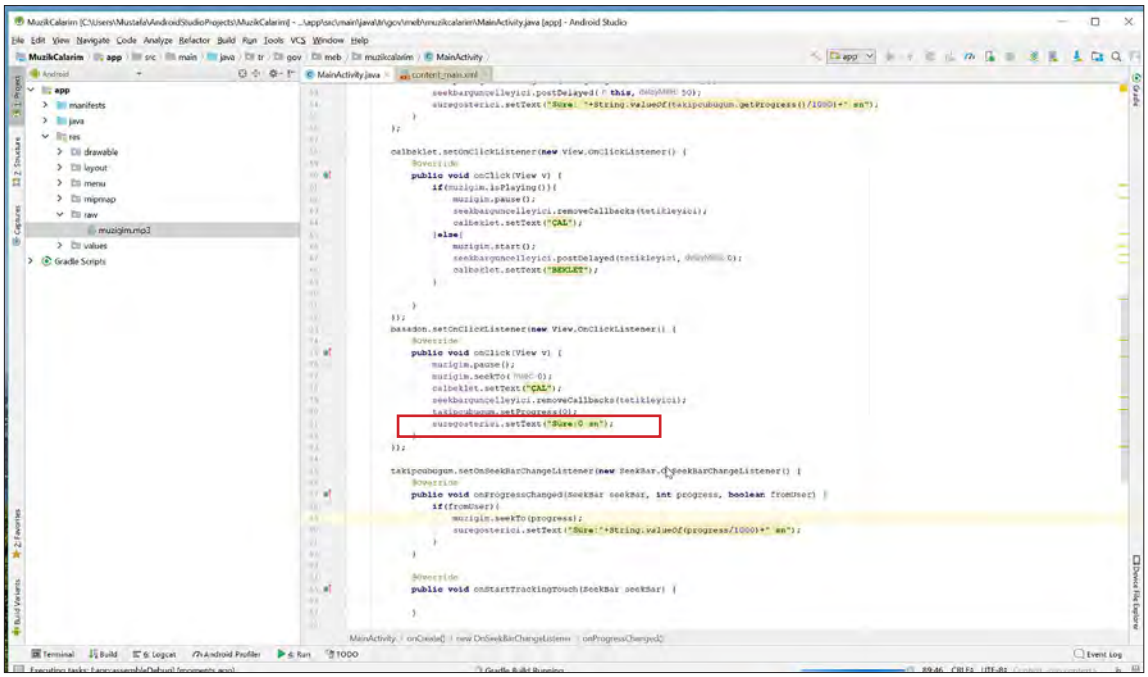
Ekran Görüntüsü 4.249

Bu şekilde oluşan uygulamayı kontrol ediniz.

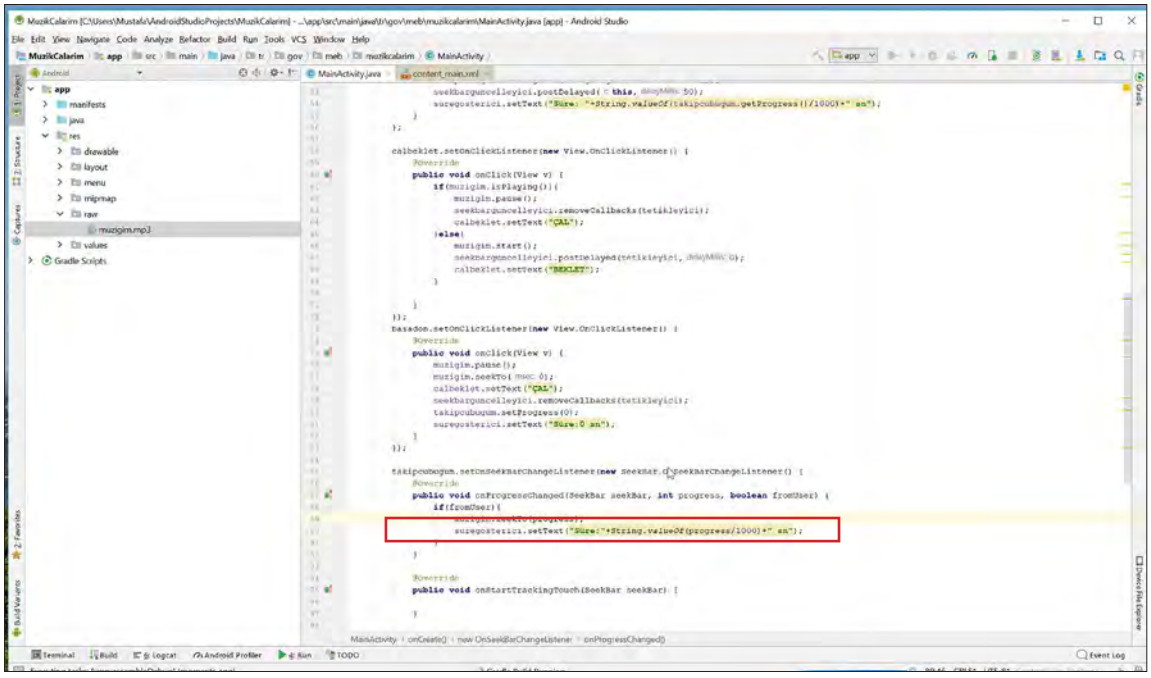


Ekran Görüntüsü 4.250

Şimdi Progress Bar'ı ilerlettiğinizde sürenin de ilerlemesini sağlayınız.

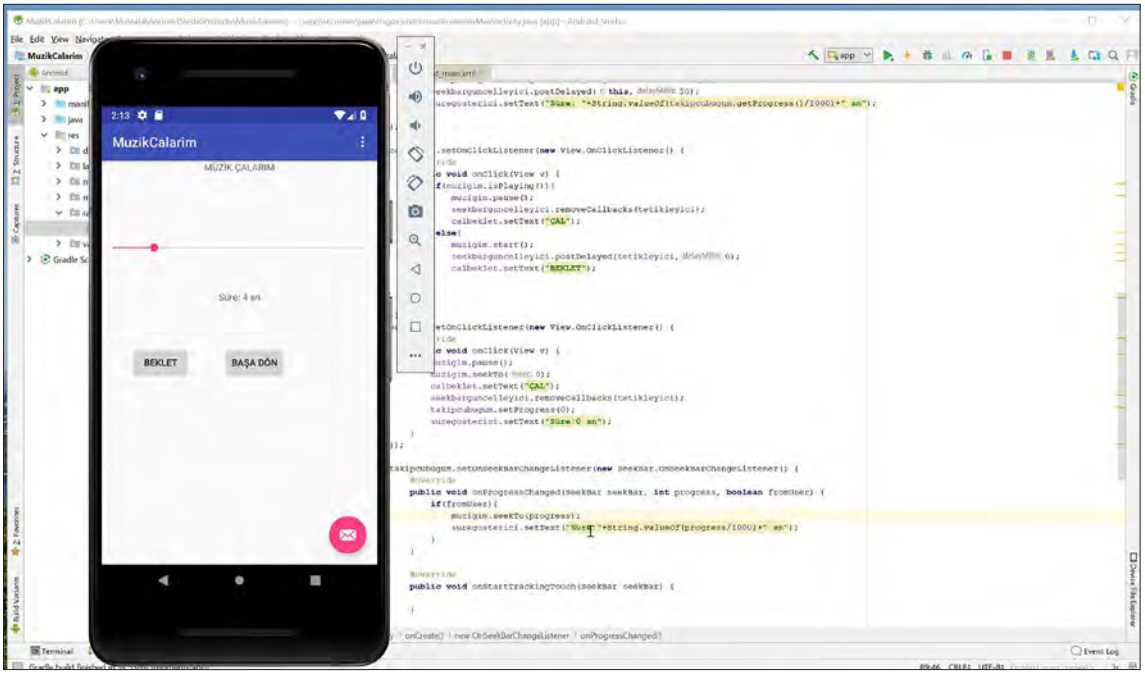


Ekran Görüntüsü 4.251



Ekran Görüntüsü 4.252

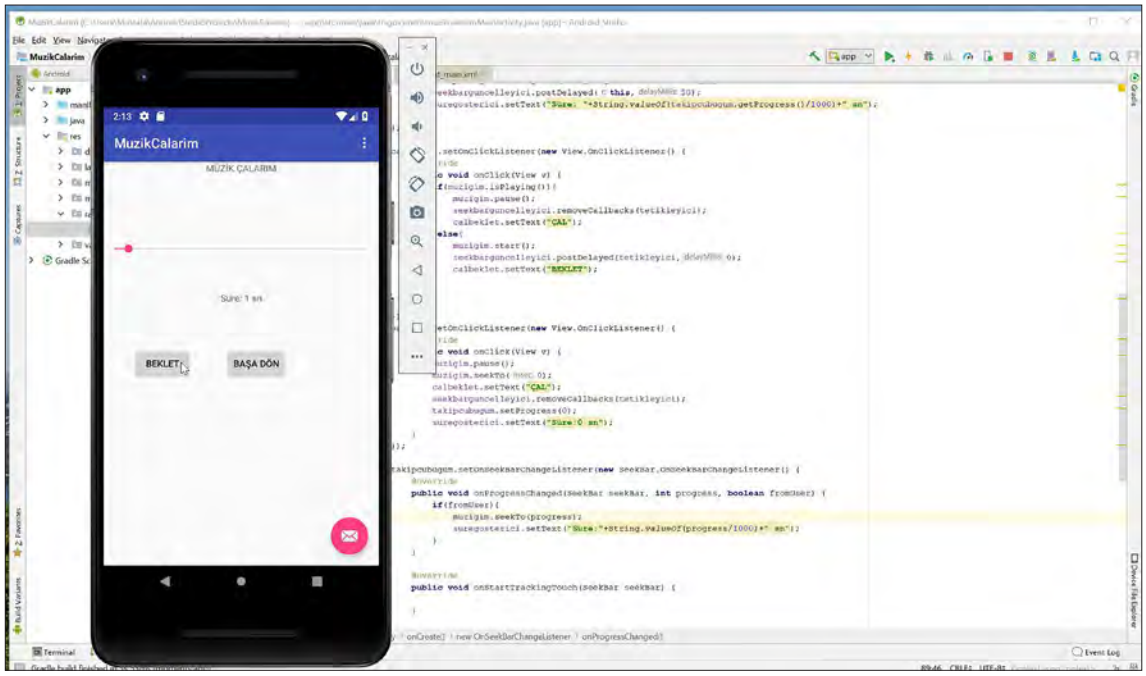
Çalıştırdığınızda aşağıdaki gibi olacaktır.



Ekran Görüntüsü 4.253

Uygulamaya müzik kaçıncı saniyedeyse o saniyeyi ekrana yazdıran kodları ekleyiniz.

Çalıştırdığımızda son ekran aşağıdaki gibi olacaktır.



Ekran Görüntüsü 4.254

XML Kodları

```
<?XML version="1.0" encoding="utf-8"?>
<RelativeLayout XMLns:android="http://schemas.android.com/apk/res/android"
  XMLns:app="http://schemas.android.com/apk/res-auto"
  XMLns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  app:layout_behavior="@string/appbar_scrolling_view_behavior"
  tools:context=".MainActivity"
  tools:showIn="@layout/activity_main">

  <TextView
     android:id="@+id/TextView"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_alignParentTop="true"
     android:layout_centerHorizontal="true"
     android:text="MÜZİK ÇALARIM" />

  <Button
     android:id="@+id/dugme1"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_alignParentStart="true"
     android:layout_centerVertical="true"
     android:layout_marginStart="44dp"
     android:text="ÇAL" />

  <Button
     android:id="@+id/dugme2"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_alignParentEnd="true"
     android:layout_centerVertical="true"
     android:layout_marginEnd="137dp"
     android:text="BAŞA DÖN" />

  <SeekBar
     android:id="@+id/takipcubugu"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:layout_alignParentStart="true"
     android:layout_alignParentTop="true"
     android:layout_marginTop="121dp" />
```

```
  <TextView
      android:id="@+id/suregoster"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="194dp"
      android:text="Süre:" />
</RelativeLayout>
```

Java Kodları

```
package tr.gov.meb.muzikcalarim;

import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.Handler;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
      setSupportActionBar(toolbar);

      FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
      fab.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View view) {
              Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                      .setAction("Action", null).show();
          }
```

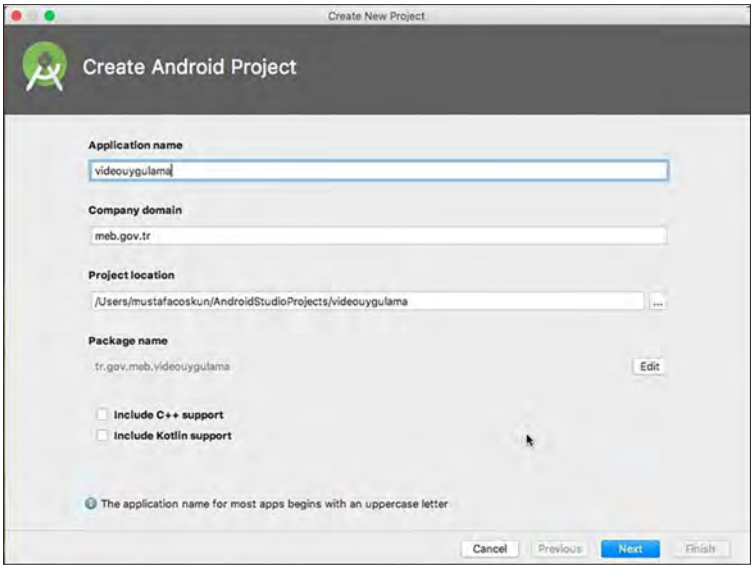
```
    });
    final MediaPlayer muzigim=MediaPlayer.create(this,R.raw.muzigim);
    final SeekBar takipcubugum=(SeekBar)findViewById(R.id.takipcubugu);
    takipcubugum.setMax(muzigim.getDuration());

    final Button calbeklet=(Button)findViewById(R.id.dugme1);

    Button basadon=(Button)findViewById(R.id.dugme2);

    final TextView suregosterici=(TextView)findViewById(R.id.suregoster);

    final Handler seekbarguncelleyici= new Handler();

    final Runnable tetikleyici=new Runnable() {
        @Override
        public void run() {
            takipcubugum.setProgress(muzigim.getCurrentPosition());
            seekbarguncelleyici.postDelayed(this,50);
            suregosterici.setText("Süre: "+String.valueOf(takipcubugum.getProgress()/1000)+"/"+String.valueOf(muzigim.
getDuration()/1000)+" sn");
        }
    };

    calbeklet.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(muzigim.isPlaying()){
                muzigim.pause();
                seekbarguncelleyici.removeCallbacks(tetikleyici);
                calbeklet.setText("ÇAL");
            }else{
                muzigim.start();
                seekbarguncelleyici.postDelayed(tetikleyici,0);
                calbeklet.setText("BEKLET");
            }

        }
    });
    basadon.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            muzigim.pause();
            muzigim.seekTo(0);
            calbeklet.setText("ÇAL");
            seekbarguncelleyici.removeCallbacks(tetikleyici);
```

```
            takipcubugum.setProgress(0);
            suregosterici.setText("Süre:0 sn");
        }
    });

    takipcubugum.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
            if(fromUser){
                muzigim.seekTo(progress);
                suregosterici.setText("Süre:"+String.valueOf(progress/1000)+"/"+String.valueOf(muzigim.getDurati-
on()/1000)+" sn");
            }
        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {

        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {

        }
    });

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.XML.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
```

```
        }

        return super.onOptionsItemSelected(item);
    }
}
```

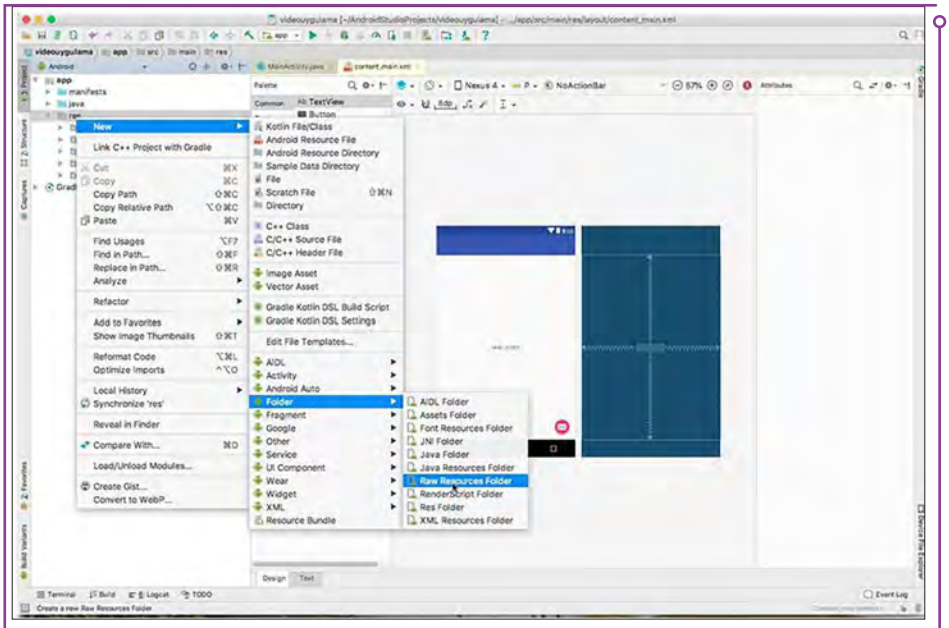
4.3.9. Video Uygulaması

Sıradaki projede bir video uygulaması yapılacaktır. Start Project diyerek projenize videouygulama ismini veriniz ve Basic Activity seçerek çalışmaya başlayınız.

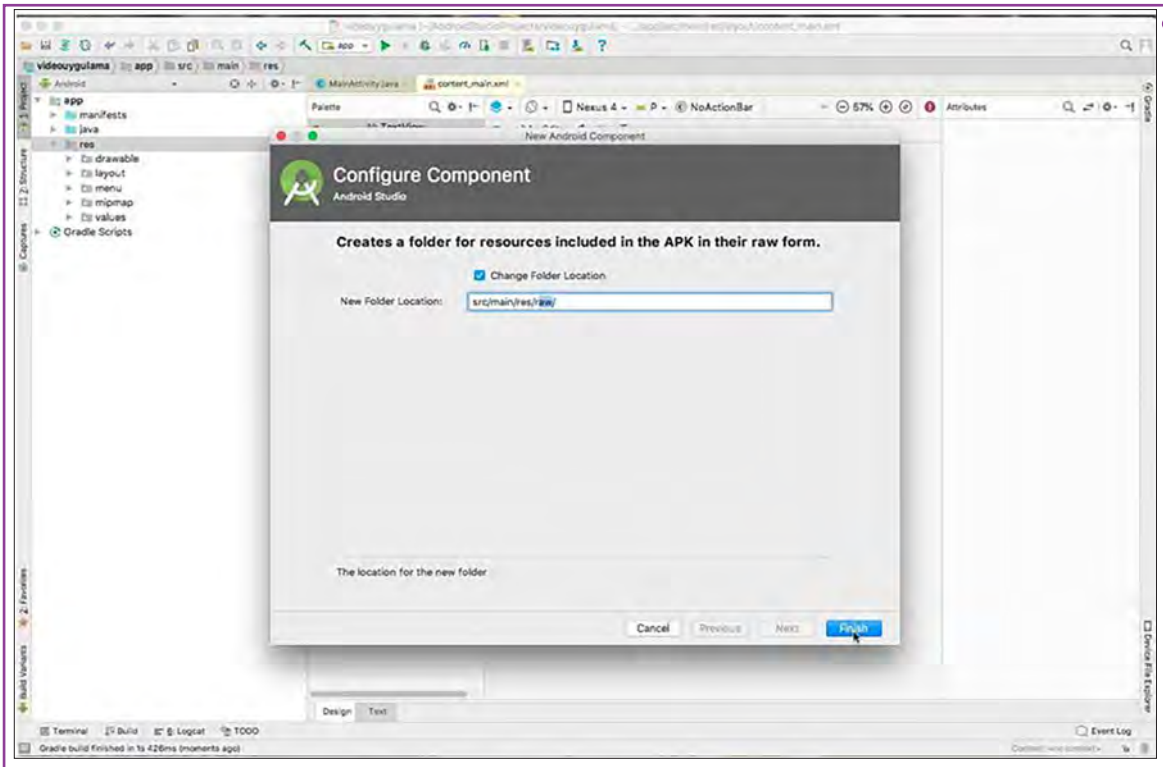


Ekran Görüntüsü 4.255

app klasörü içinde resources (kaynaklar) içinde Row Folder türünde bir klasör oluşturunuz.

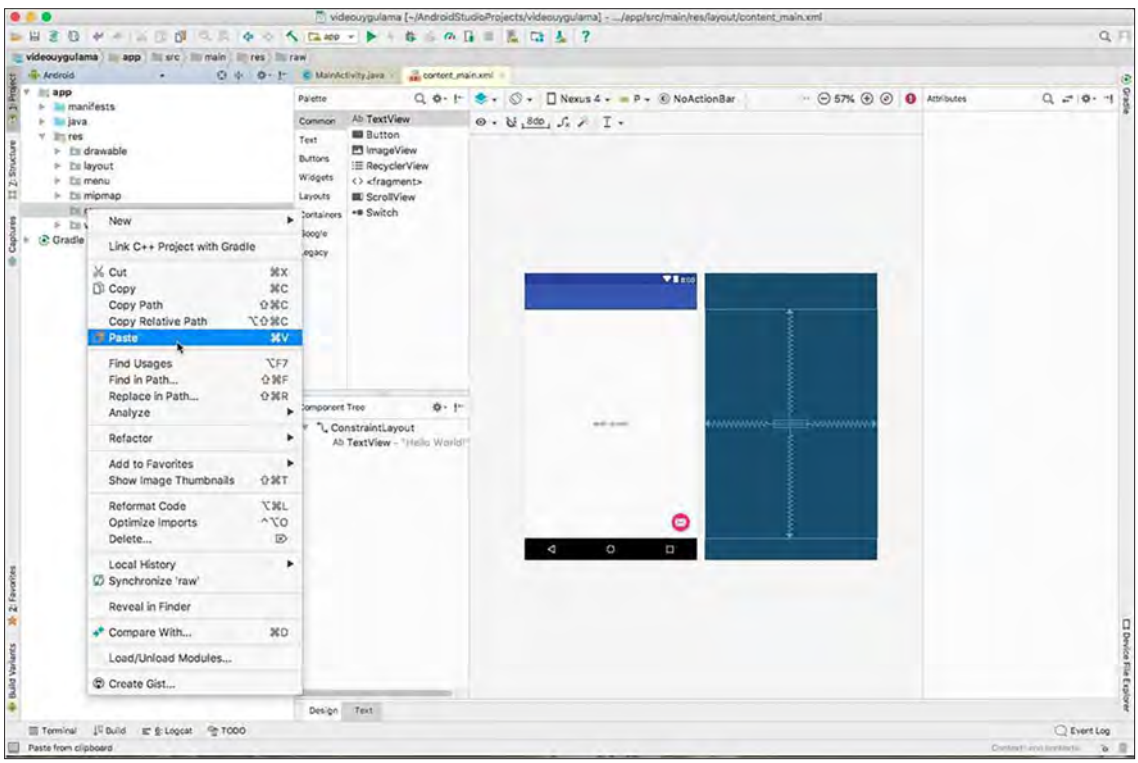


Ekran Görüntüsü 4.256

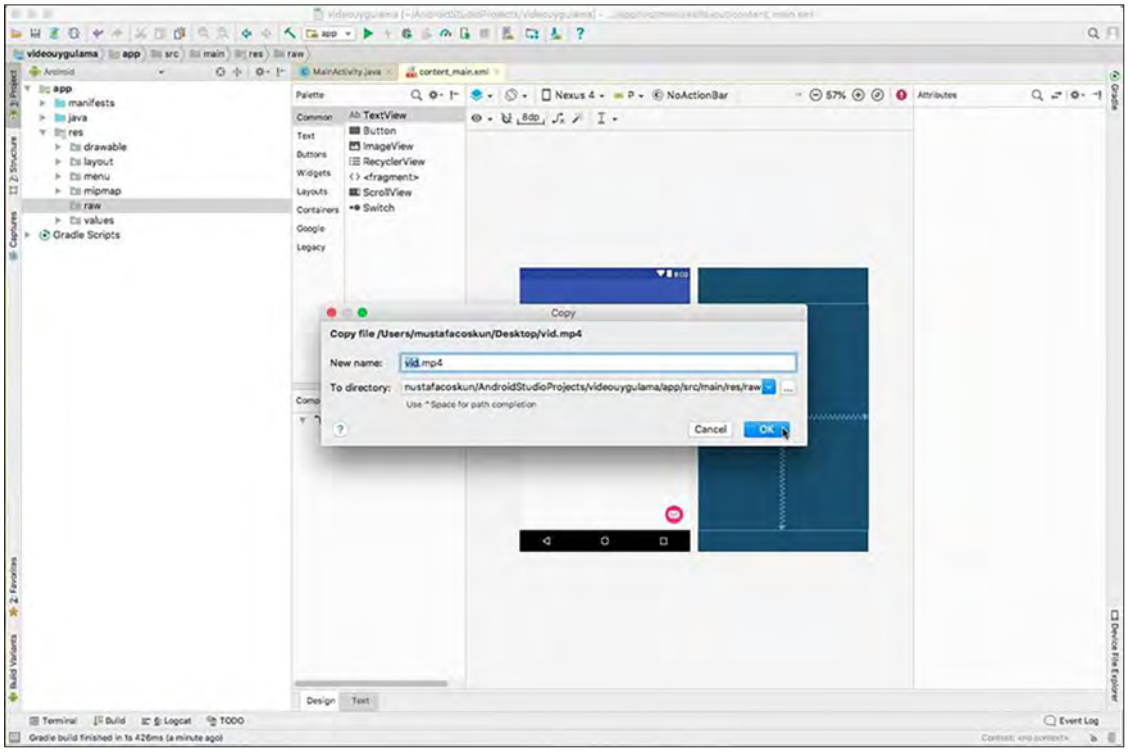


Ekran Görüntüsü 4.257

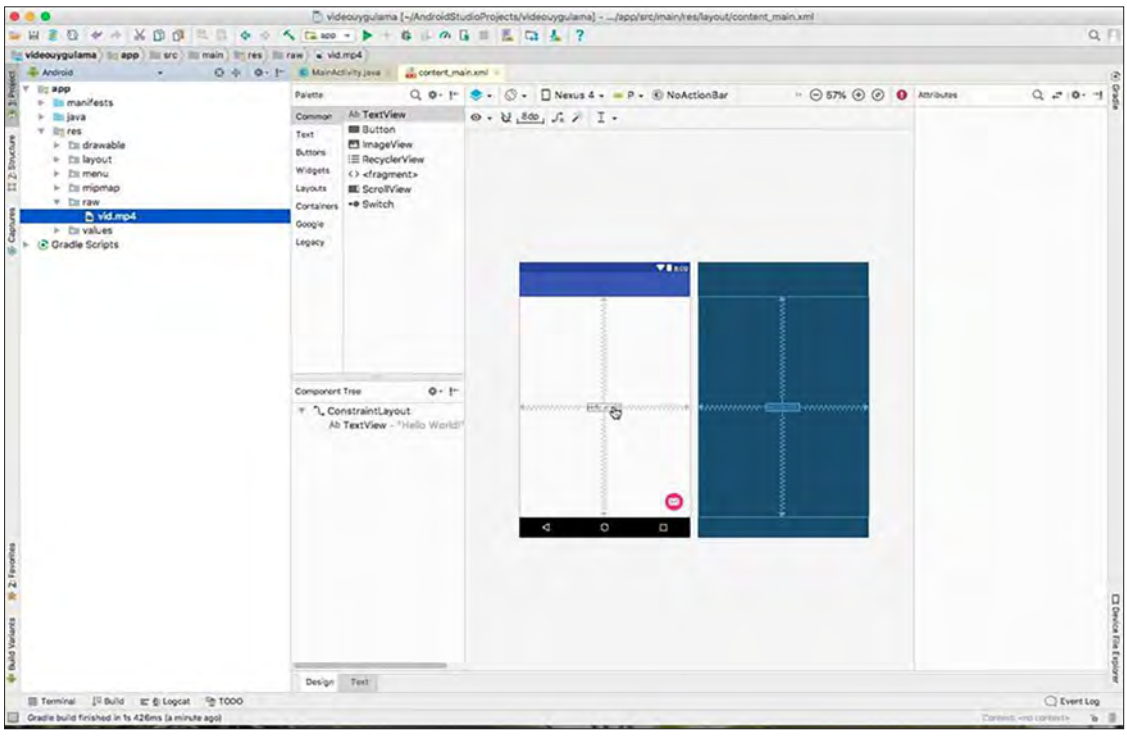
EBA'dan alınan bir video projeye aktarılacaktır. Bu videoyu kopyalayıp projenize yapıştırınız.



Ekran Görüntüsü 4.158

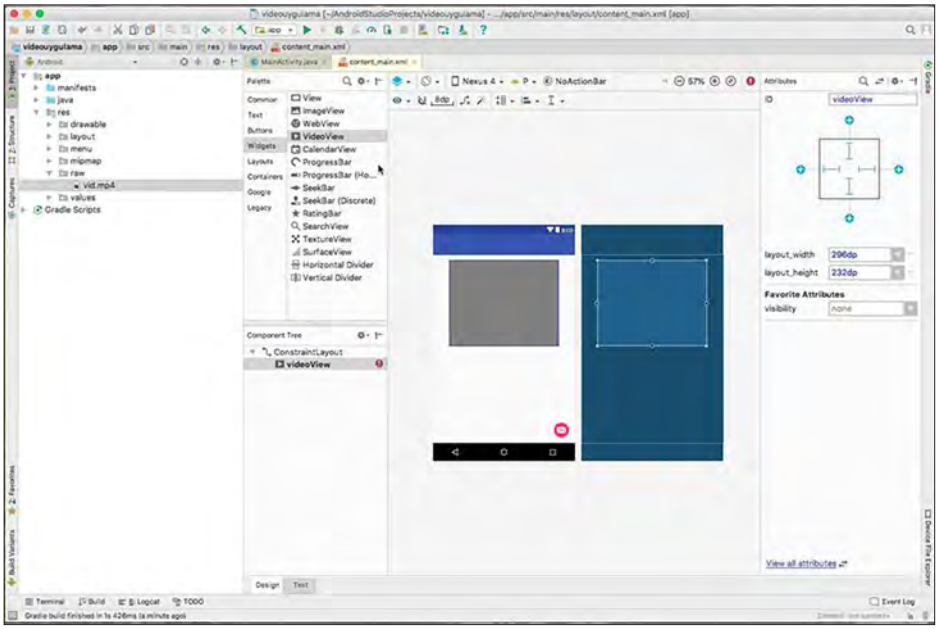


Ekran Görüntüsü 4.259



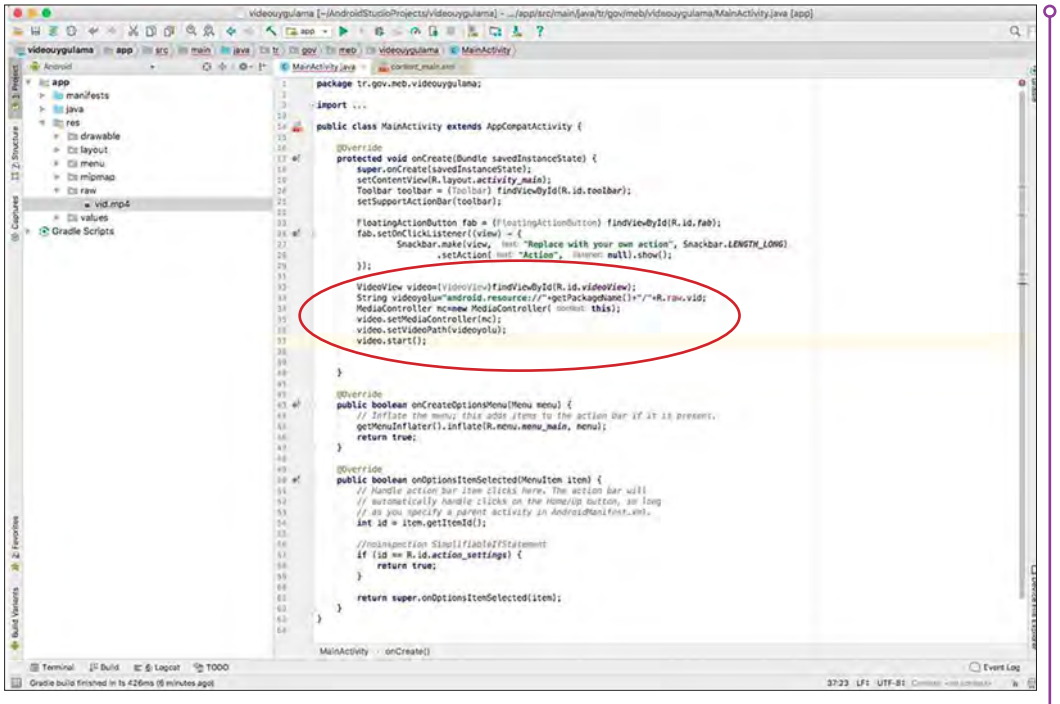
Ekran Görüntüsü 4.260

TextView'i ekrandan kaldırıp, widget'ların altından VideoView nesnesini ekleyiniz.



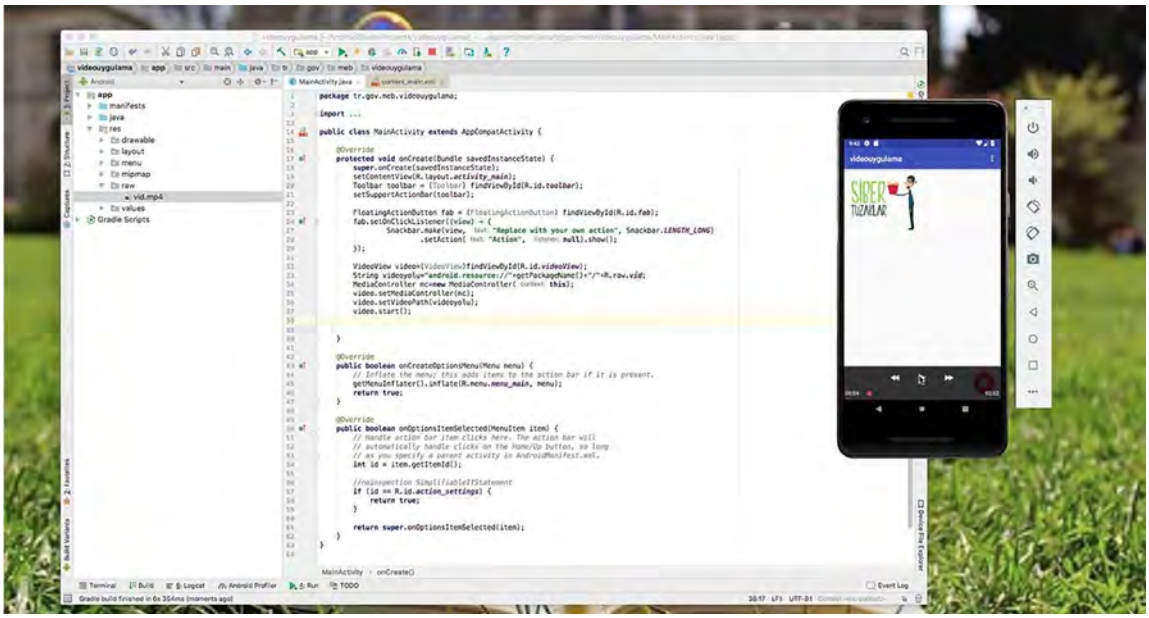
Ekran Görüntüsü 4.261

Şimdi Mainactivity.java'ya gelip kodlamaya başlayınız. Öncelikle videonun yolunu belirleyiniz. VideoView'de video'nun yolunu gösteriniz. Bir string değişken tanımlayanız ve ismini "videoyolu" yapınız. Ardından tırnak içinde bilgisayardaki yerini tarif ediniz. Source'un yerini "get package name" komutu ile alabilirsiniz. Ardından Media Controller nesnesini ve VideoView'i tanımlayınız. Ardından çalıcı (player) tanımlayınız ve videoya start komutu veriniz.



Ekran Görüntüsü 4.262

Projeyi çalıştırdığınızda video çalışır, üzerine tıklandığında durur. Media Controller nesnesi bu yüzden oluşturulmaktadır.



Ekran Görüntüsü 4.263

XML Kodları

```
<?XML version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout XMLns:android="http://schemas.android.com/apk/res/android"
  XMLns:app="http://schemas.android.com/apk/res-auto"
  XMLns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  app:layout_behavior="@string/appbar_scrolling_view_behavior"
  tools:context=".MainActivity"
  tools:showIn="@layout/activity_main">

  <VideoView
     android:id="@+id/videoView"
     android:layout_width="296dp"
     android:layout_height="232dp"
     tools:layout_editor_absoluteX="44dp"
     tools:layout_editor_absoluteY="16dp" />
</android.support.constraint.ConstraintLayout>
```

Java Kodları

```
package tr.gov.meb.videouygulama;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
```

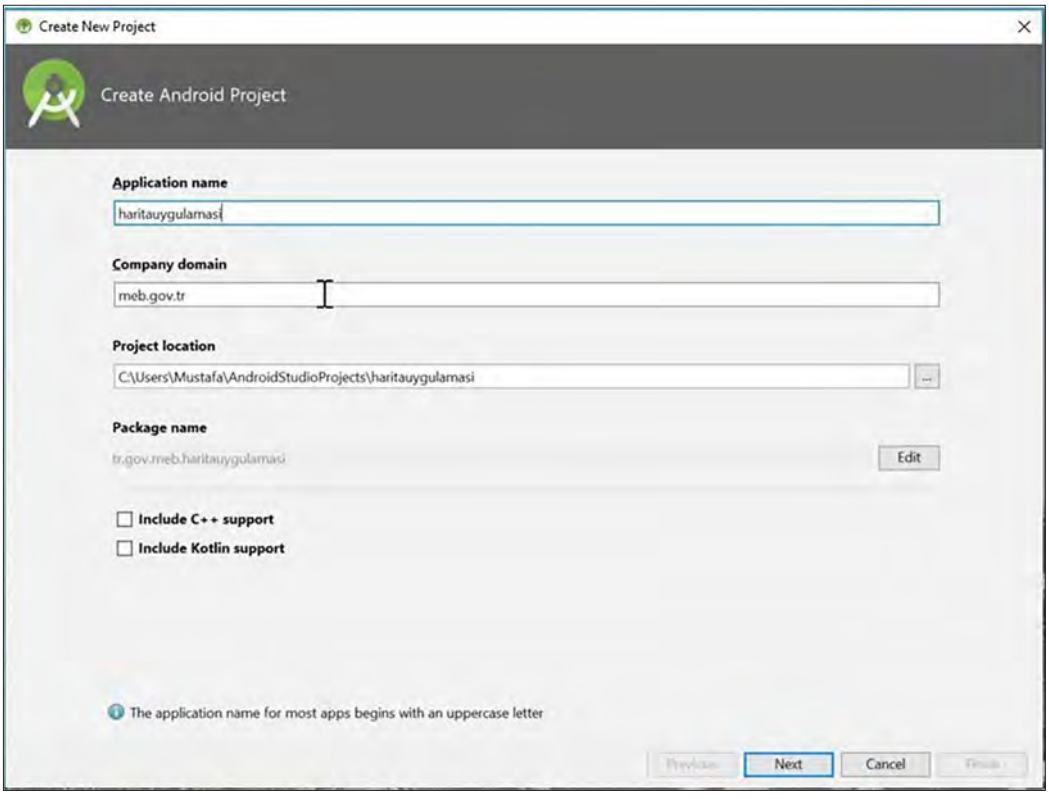
```
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.MediaController;
import android.widget.VideoView;

public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
     setSupportActionBar(toolbar);

     FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
     fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
           Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                 .setAction("Action", null).show();
        }
     });

     VideoView video=(VideoView)findViewById(R.id.videoView); //video gösterici nesne oluştur
     String videoyolu="android.resource://"+getPackageName()+"/"+R.raw.vid;
                                   //görüntü dosyasının yolu paketimin klasöründe “vid” olsun
     MediaController mc=new MediaController(this);
                                   // medya oynatıcı nesnesi oluştur
     video.setMediaController(mc);
                                   //video nesnesini medya oynatıcıya bağla
     video.setVideoPath(videoyolu);
     video.start();
                                   //videoyu başlat

  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
     getMenuInflater().inflate(R.menu.menu_main, menu);
     return true;
  }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

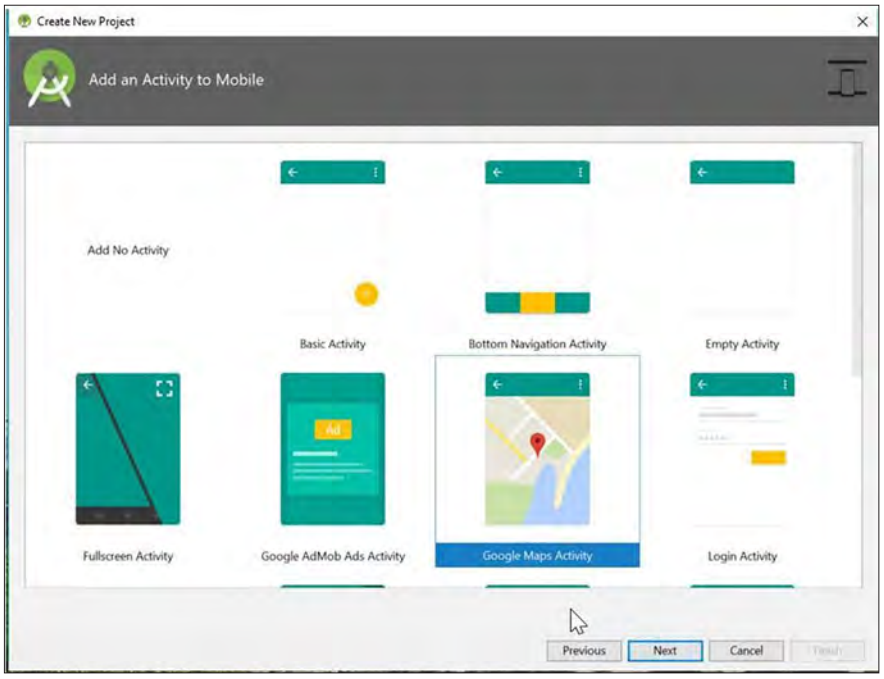
4.3.10. Harita Uygulaması

Sıradaki çalışmada bir map (harita) uygulaması yapılacaktır.

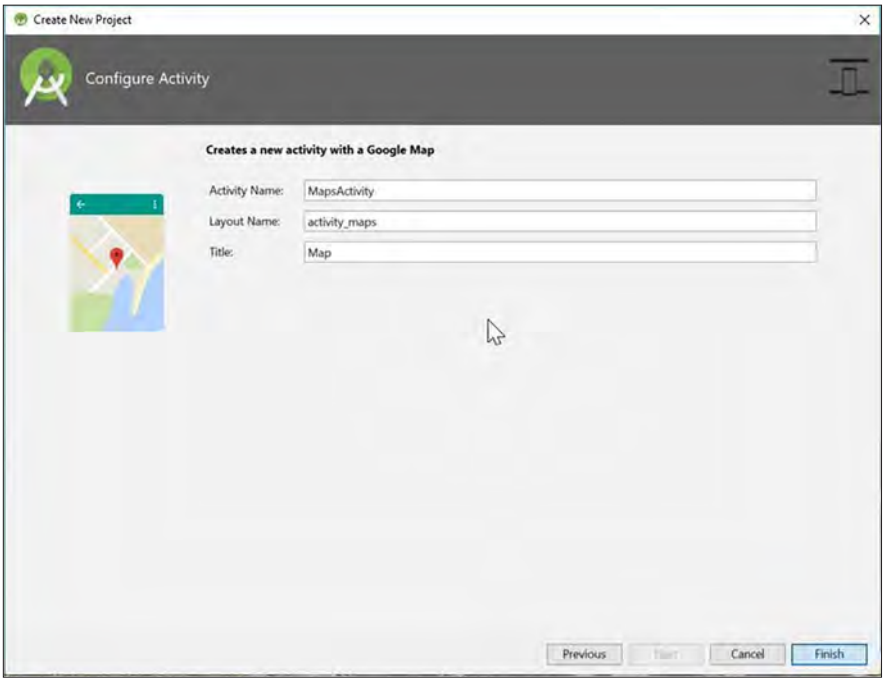
Yeni bir uygulama oluşturunuz ve ismini "haritauygulamasi" koyunuz. Türkçe karakter kullanmamayı unutmayınız. Burada önemli olan nokta uygulamanızın activity'sini Basic Activity değil, Maps Activity olarak seçmemizdir. Çünkü uygulamadaki harita Google Maps üzerinden çağrılacaktır.



Ekran Görüntüsü 4.264



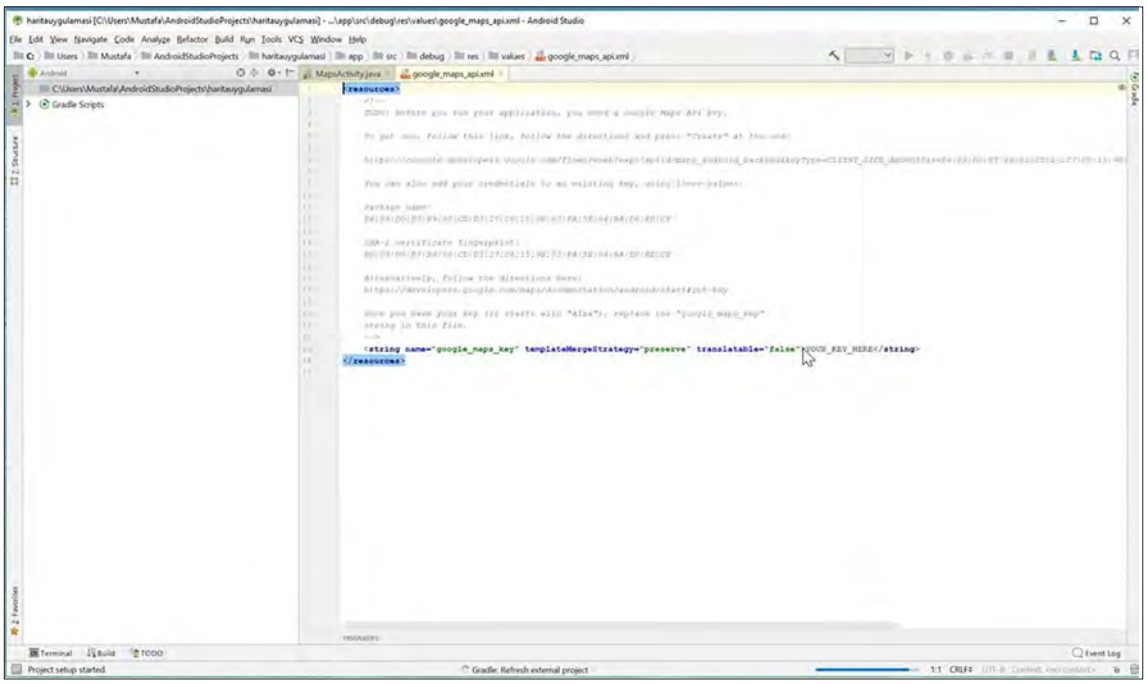
Ekran Görüntüsü 4.265



Ekran Görüntüsü 4.266

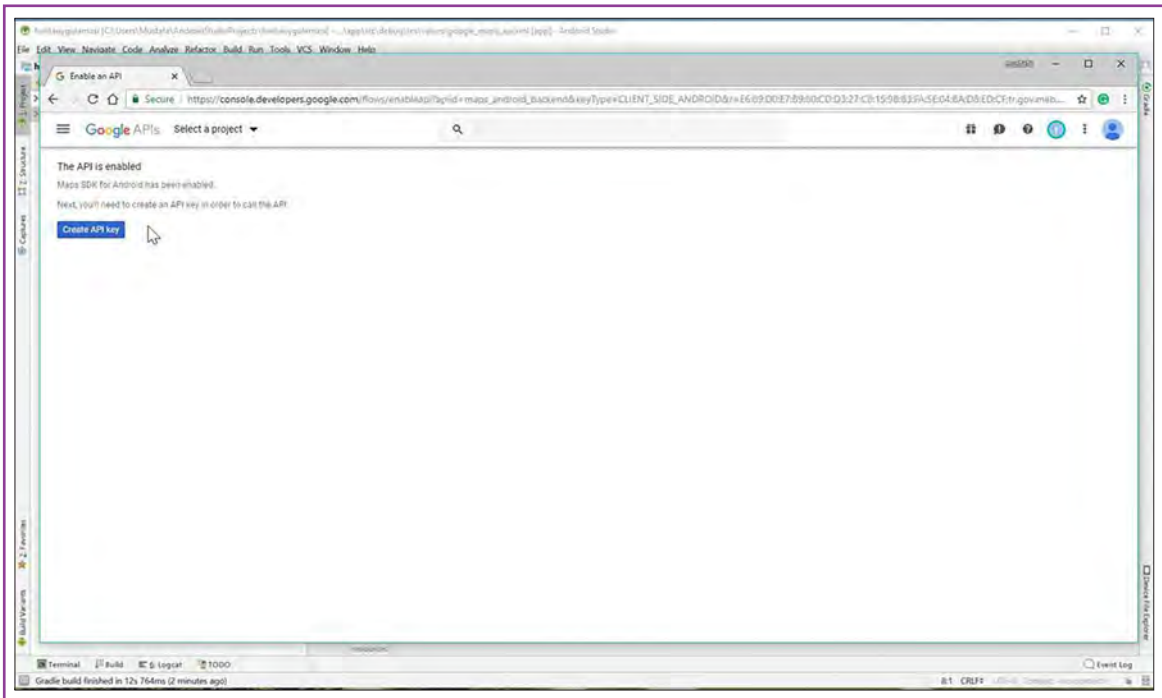
Maps Activity seçildiğinde, XML dosyaları içine ilgili Google Maps API'leri koyulur. ApI; Advance Program Interface, yani farklı uygulamalardan kendi sisteminize veri çekilmesini sağlayan yapıdır.

Burada, Google Map APL'si kullanılacaktır. Bu nedenle Google hesabınızla Google Map APL'sini almak gerekmektedir. Bunun için Ekran Görüntüsü 4.267 de görülen bağlantıyı kopyalayıp tarayıcınızda açınız ve yeni bir proje oluşturunuz.



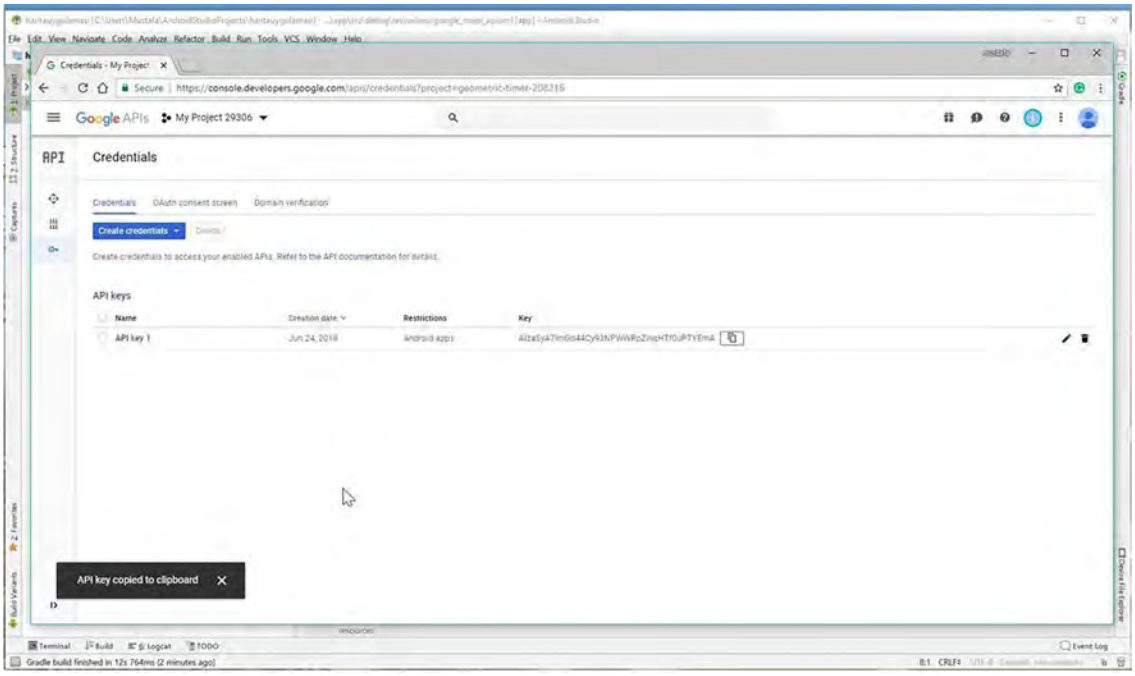
Ekran Görüntüsü 4.267

Projeniz oluştuktan sonra APL'yi de oluşturunuz.



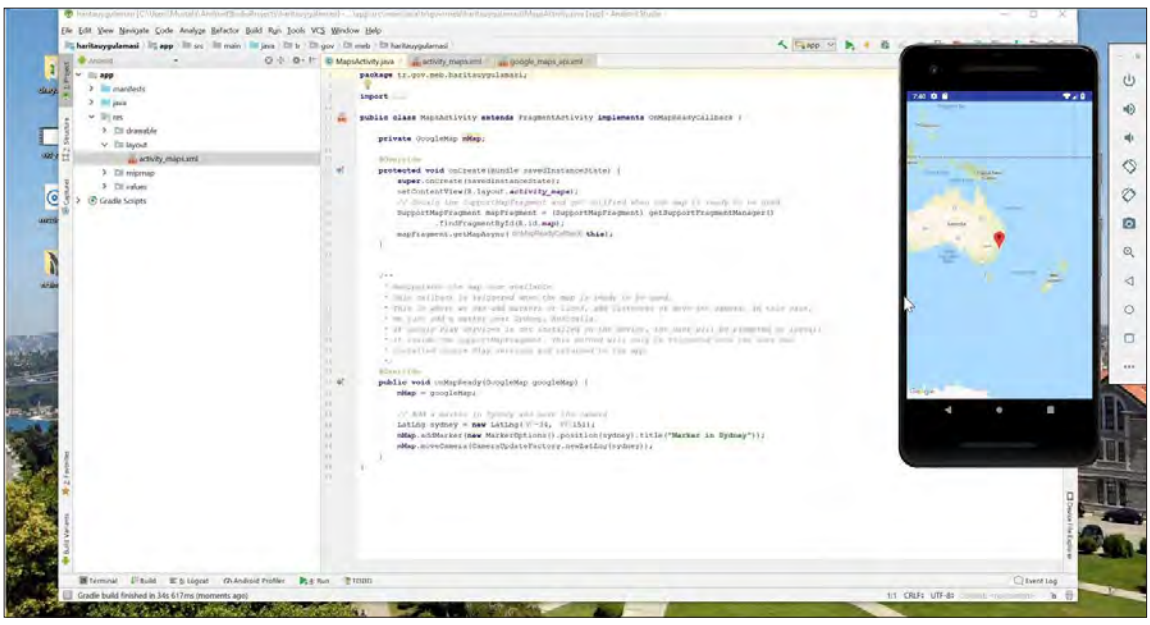
Ekran Görüntüsü 4.268

Oluşturduğunuz APL'yi kopyalayıp projenize (XML) yapıştırınız.



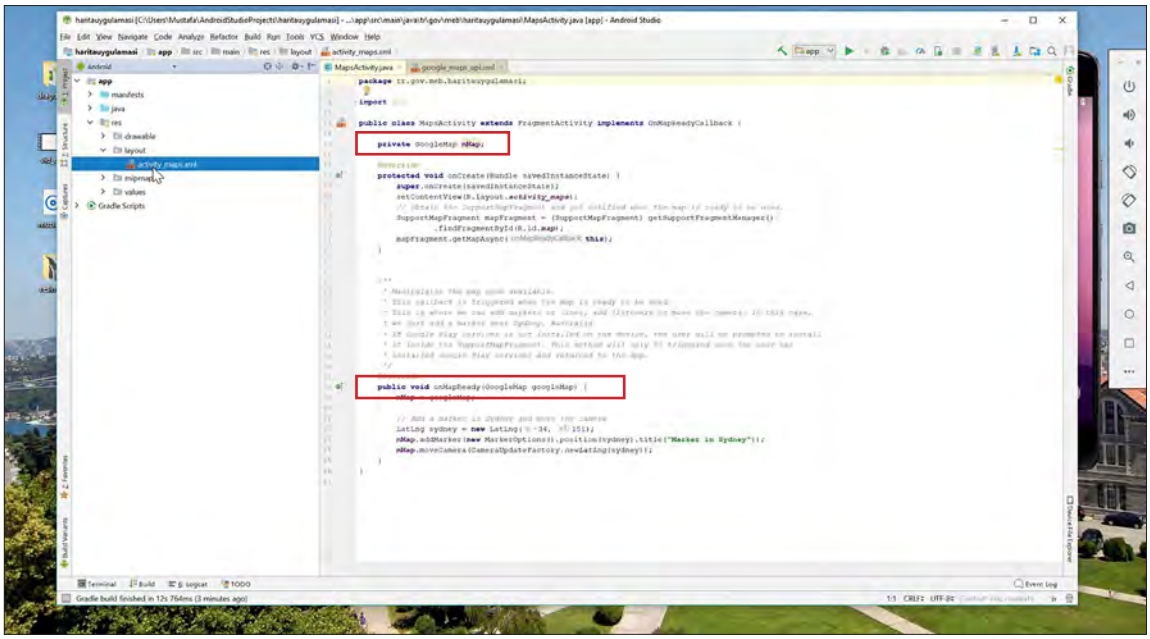
Ekran Görüntüsü 4.269

Activity dosyanızı açınız ve Relative Layaout değil, Fragment Layout seçildiğini gözlemleyiniz. Projeniz çalıştığında Google Maps'in geldiğini göreceksiniz.



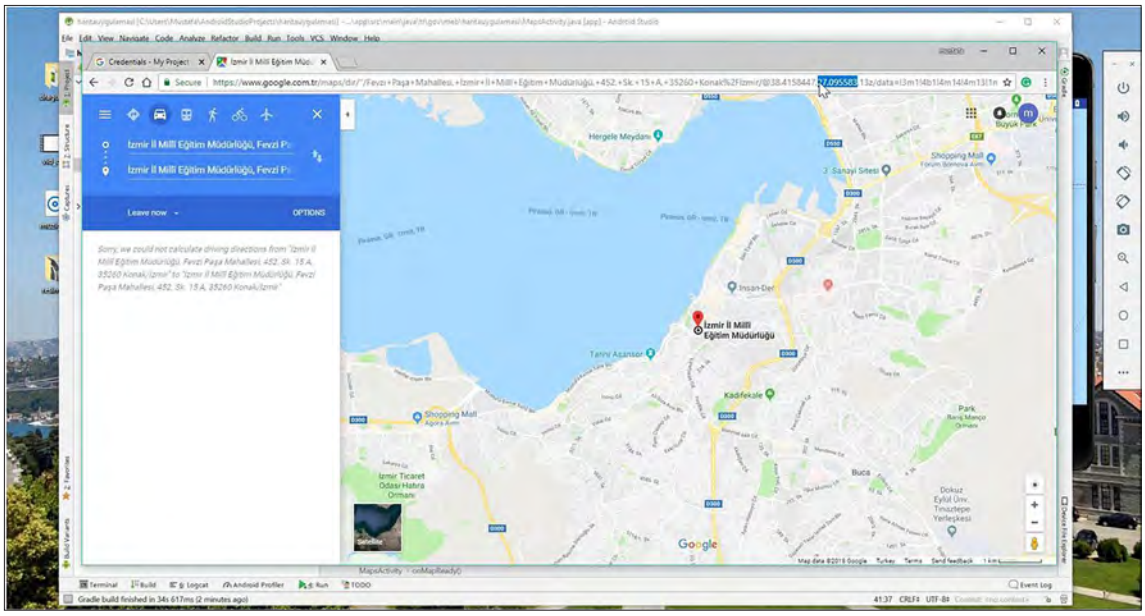
Ekran Görüntüsü 4.270

Otomatik olarak oluşturulan Map nesnesinin adını "benimharitam" koyunuz ve aşağıda da gelen bilgiyi nesneye aktarınız.



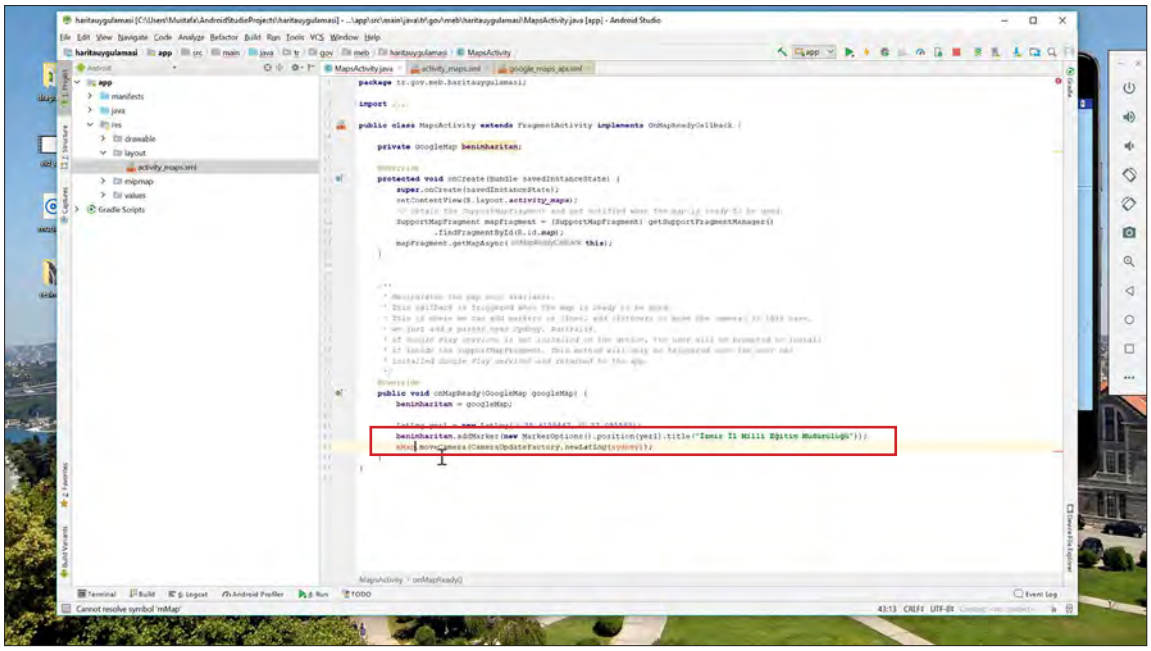
Ekran Görüntüsü 24.271

LatLng diye bir değişken türü tanımlayınız ve ismini de sydney değil, yer1 yapınız. Öncelikle, Google Maps'i açınız ve İzmir İl Milli Eğitim Müdürlüğü'nün enlem ve boylam bilgilerini bulunuz (uygulamanız için farklı bir yer tercih edebilirsiniz).



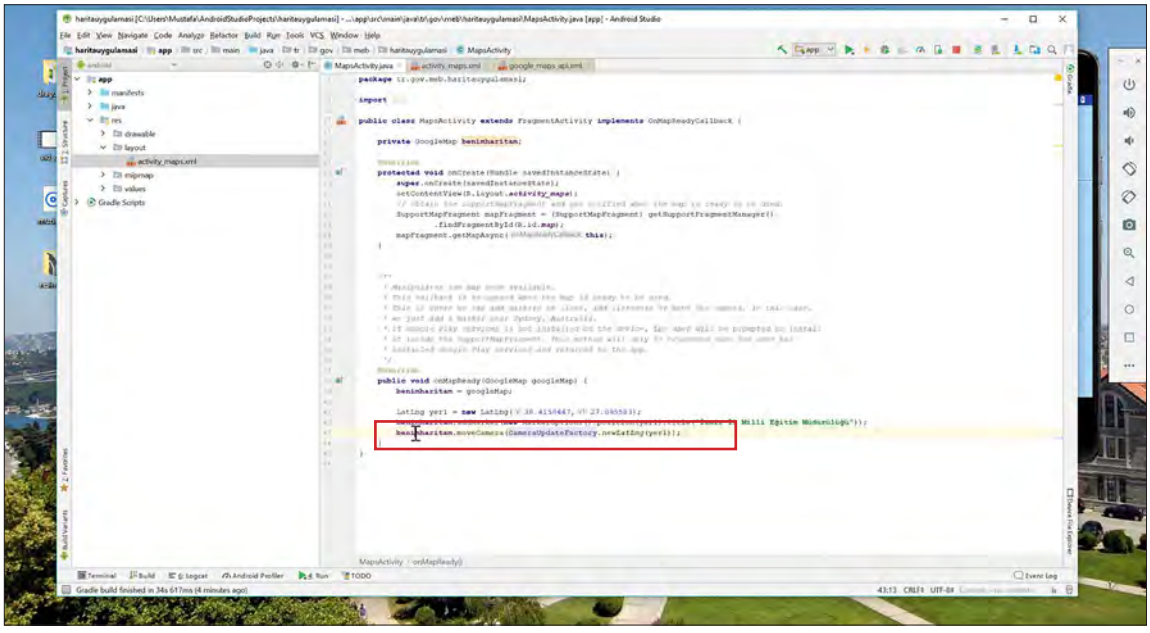
Ekran Görüntüsü 4.272

Bu bilgiyi ve enlem boylam bilgilerini alıp MainActivity'deki kodlara ekleyiniz.



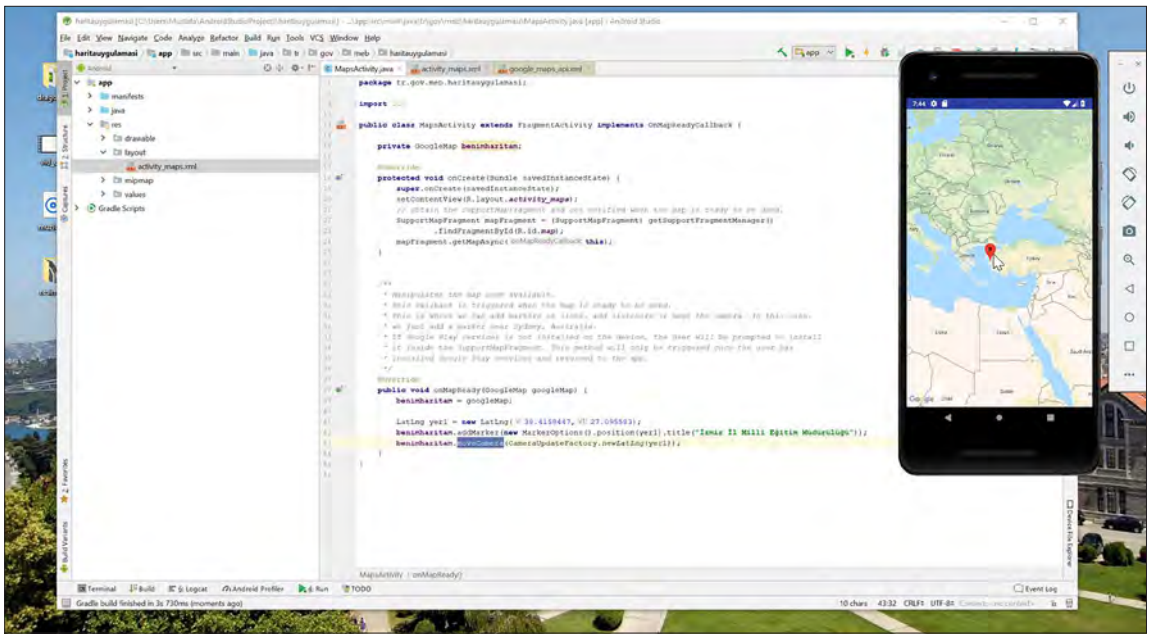
Ekran Görüntüsü 4.273

Şimdi de, 'onMapReady' (Harita hazır olduğunda) fonksiyonunu yazınız.



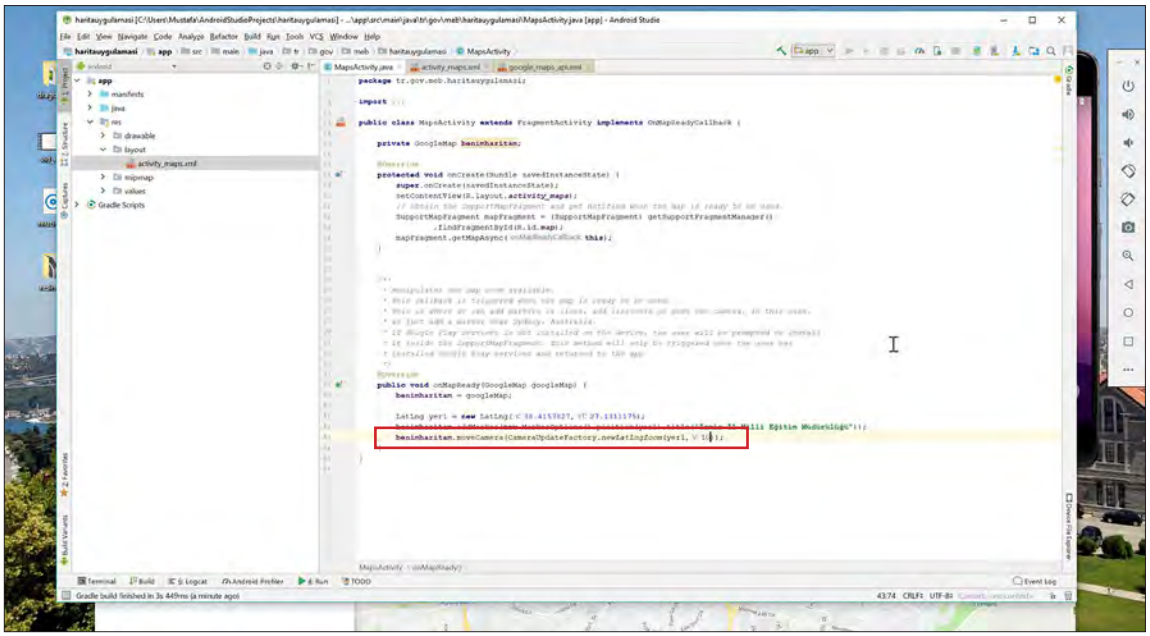
Ekran Görüntüsü 4.274

Projenizi çalıştırdığınızda aşağıdaki ekranı göreceksiniz.



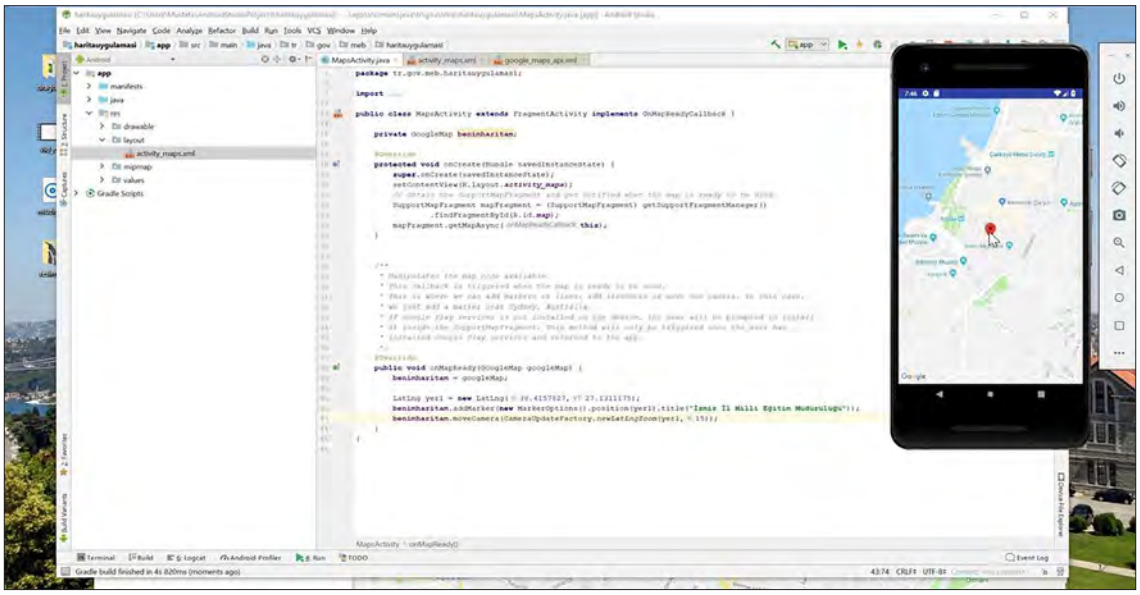
Ekran Görüntüsü 4.275

İzmir İl Milli Eğitim Müdürlüğü'nü ekranda görebilirsiniz. Biraz uzakta çıkacaktır, yakınlaşmasını sağlayabilirsiniz. Bunun için newLatLng fonksiyonu yerine newLatLngzoom fonksiyonu kullanılır ve zoom değeri artırılır.



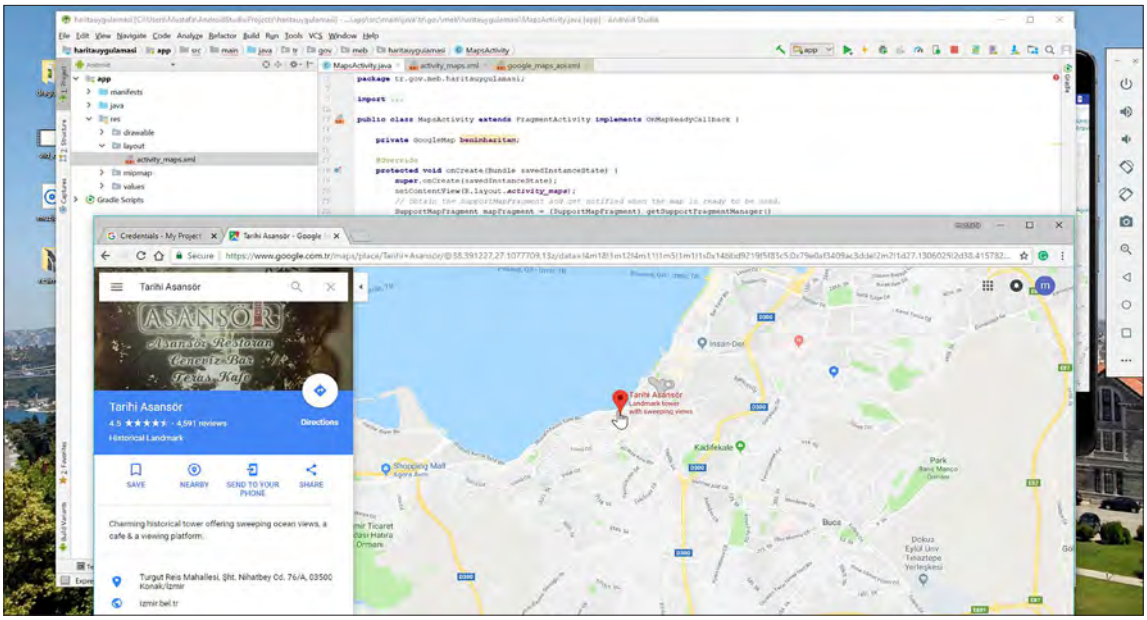
Ekran Görüntüsü 4.276

Bu şekilde çalıştırılınca aşağıdaki gibi görünecektir, belirlenen yer daha yakında olacaktır.



Ekran Görüntüsü 4.277

Artık haritanızda istediğiniz bir yeri belirleyip gösterebilirsiniz.



Ekran Görüntüsü 4.278

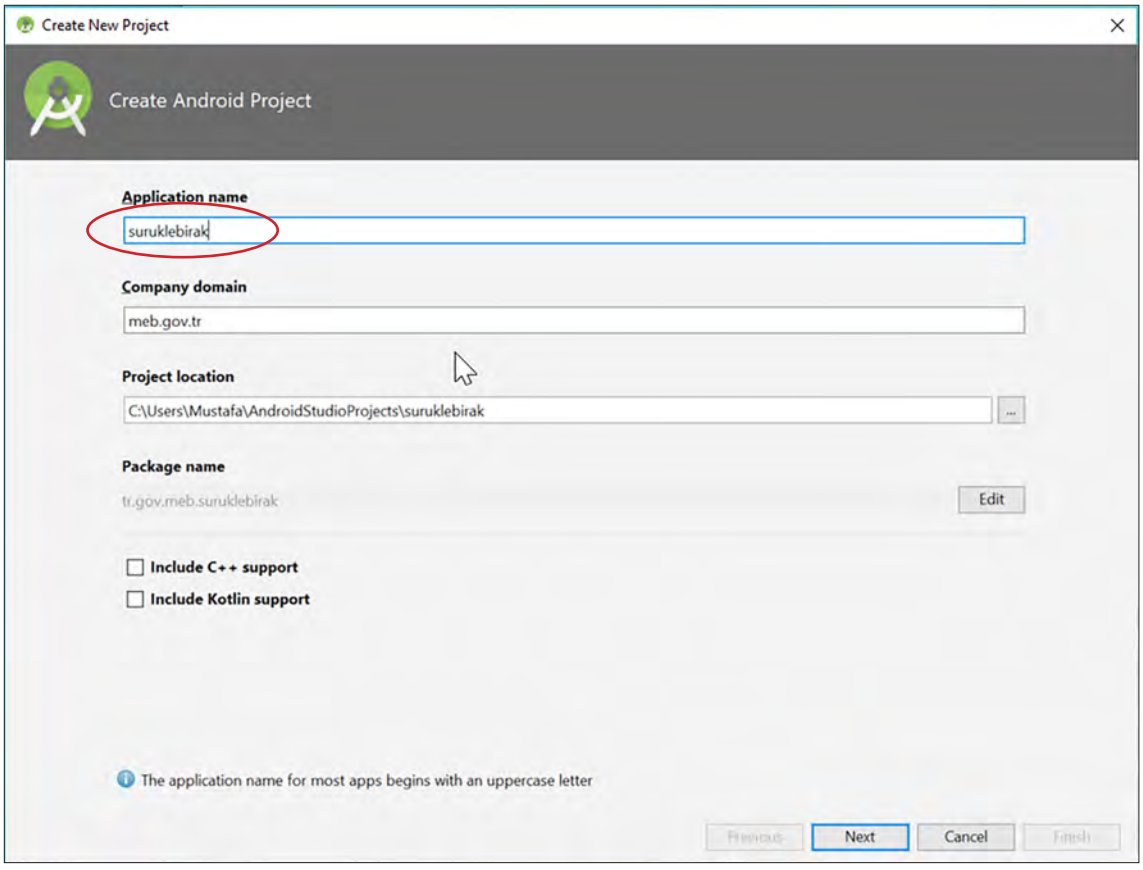
Java Kodları

```
package tr.gov.meb.haritauygulamasi;

import android.support.v4.app.FragmentActivity;
import android.os.Bundle;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

  private GoogleMap benimharitam;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_maps);
     // Obtain the SupportMapFragment and get notified when the map is ready to be used.
     SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
           .findFragmentById(R.id.map);
     mapFragment.getMapAsync(this);
  }
  /**
   * Manipulates the map once available.
   * This callback is triggered when the map is ready to be used.
   * This is where we can add markers or lines, add listeners or move the camera. In this case,
   * we just add a marker near Sydney, Australia.
   * If Google Play services is not installed on the device, the user will be prompted to install
   * it inside the SupportMapFragment. This method will only be triggered once the user has
   * installed Google Play services and returned to the app.
   */
  @Override
  public void onMapReady(GoogleMap googleMap) {
     benimharitam = googleMap;                    //benimharitam isimli googleMap harita nesnesi oluştur

     LatLng yer1 = new LatLng(38.4157827,27.1311175);
                    //yer1 isimli enlemboylam nesnesi oluştur ve ilk değerlerini ata
     LatLng yer2 = new LatLng(38.4018217,27.1130066);
                    //yer2 isimli enlemboylam nesnesi oluştur ve ilk değerlerini ata
```

```
    benimharitam.addMarker(new MarkerOptions().position(yer1).title("İzmir İl Milli Eğitim Müdürülüğü"));
                                        //haritada başlığı belirtilen şekilde yer1 adresinde bir işaret oluştur
    benimharitam.addMarker(new MarkerOptions().position(yer2).title("Tarihi Asansör"));
                                        //haritada başlığı belirtilen şekilde yer2 adresinde bir işaret oluştur
    benimharitam.moveCamera(CameraUpdateFactory.newLatLngZoom(yer1,10));
                                        //ekranı yer1 adresine 10 değerinde büyütme (zoom) yaparak ortala
  }
}
```

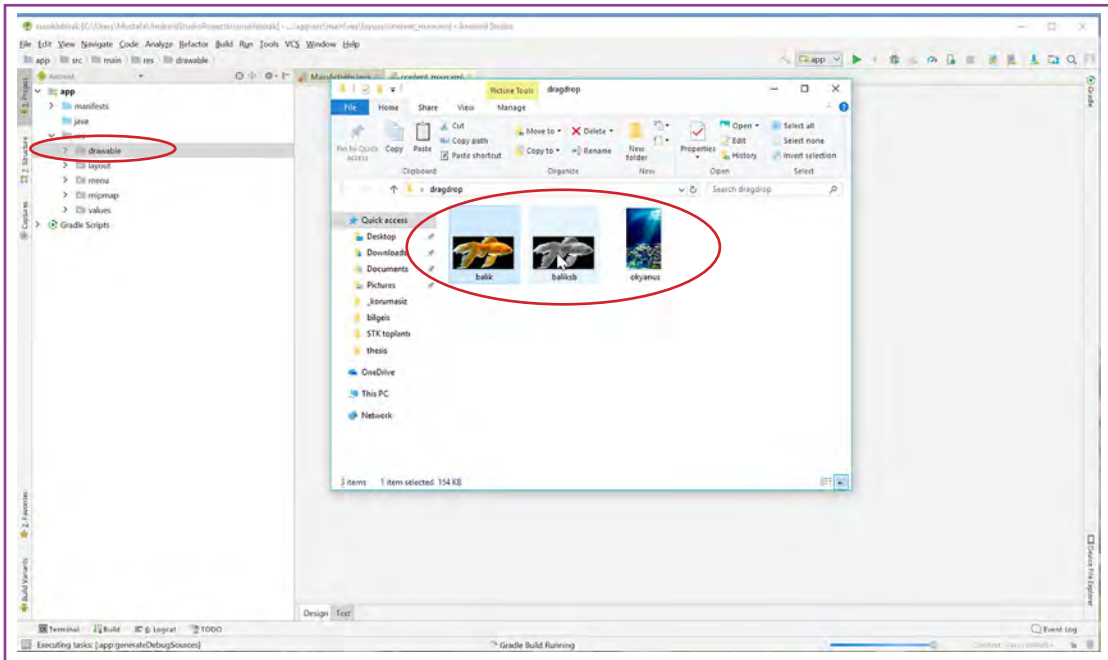
4.3.11. Sürükle Bırak Uygulaması

Bu uygulamada bir ImageView yani ekrandaki resim sürüklenerek diğer bir ImageView üzerine ekran sınırları içerisinde yerleştirilecektir. Öncelikle surukleBirak isimli Activity olarak Basic Activity seçip yeni bir proje oluşturunuz.



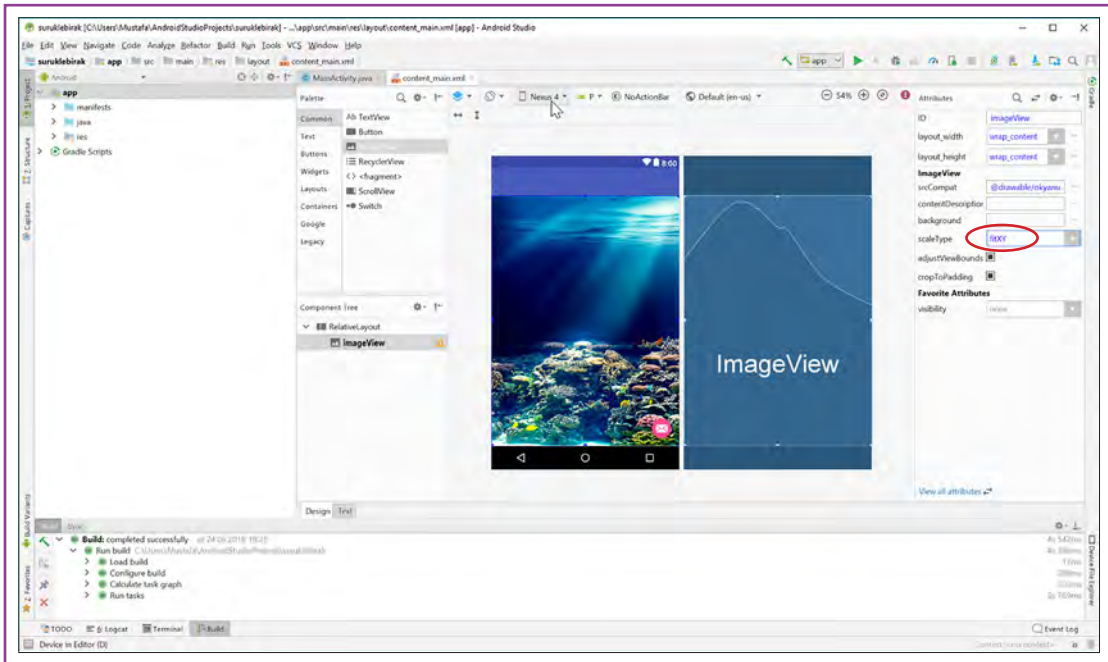
Ekran Görüntüsü 4.279

Projede kullanacağınız 3 resmi projenin yer aldığı "suruklebirak/app/src/main/res/drawable" klasörüne kopyala-yapıştır diyerek ekleyiniz. Okyanus resmi arkaplan resmi, renkli olan balık resmi sürüklenen resim ve siyah beyaz olan balık resmi sabit resim olarak kullanılacaktır.



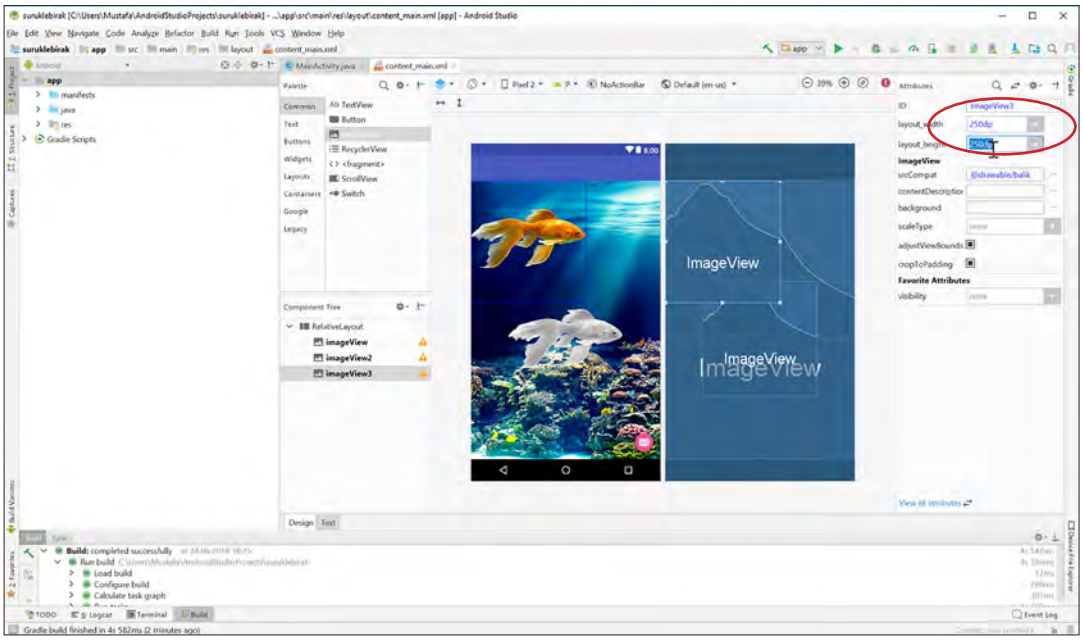
Ekran Görüntüsü 4.280

Resimleri proje klasörüne ekledikten sonra projedeki Hello Word'ü silip, Layout özelliğini Relative-Layout olarak değiştiriniz. Daha sonra ilk ImageView olarak okyanus resmini ekleyiniz. Okyanus resmi arkaplan olarak gözükeceğinden scaleType olarak fitXY seçiniz.



Ekran Görüntüsü 4.281

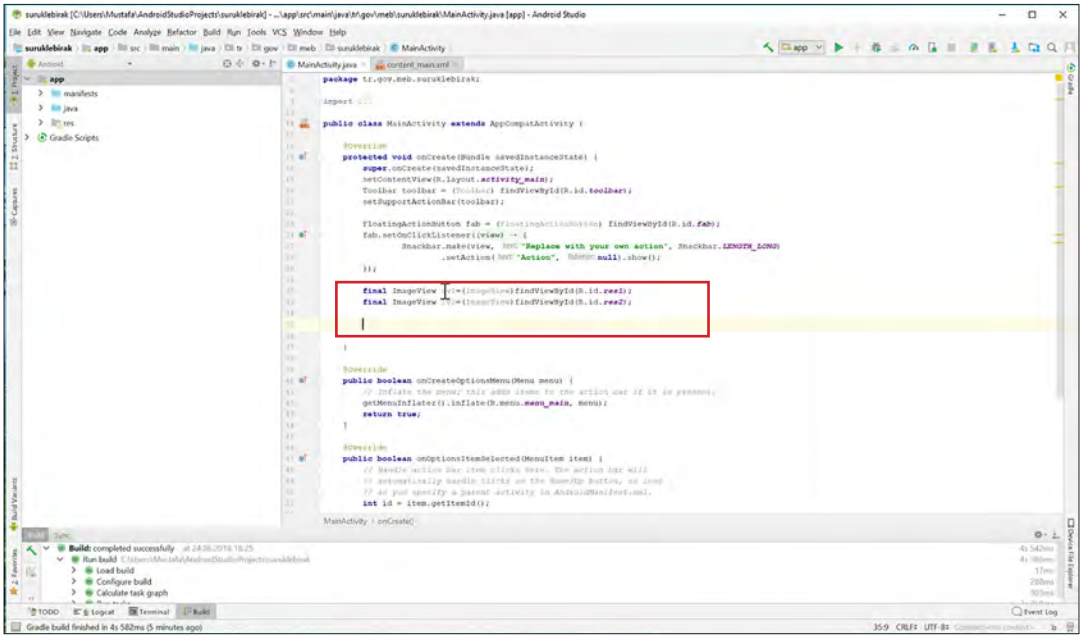
Balık resimlerini eklemek için iki tane daha ImageView ekleyiniz. Sürükle-bırak uygulaması yapacağınız için iki balık resminin boyutlarının aynı olması gerekmektedir. Layout_width and layout_height değerlerini iki balık için de 250-250 dp yapınız. Renkli balık resminin id'sini res1, diğer balık resminin id'sini res2 yapınız.



Ekran Görüntüsü 4.282

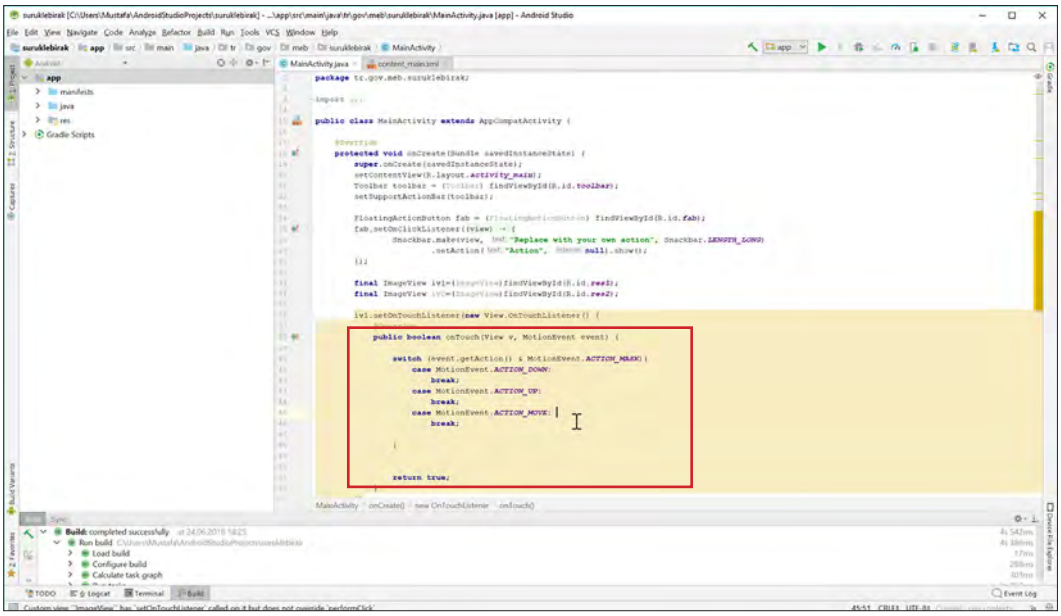
Design kısmında tüm resimlerinizi ekledikten ve bu resimlerin tasarım ayarlarını yaptıktan sonra MainActivity.java dosyasını açarak kodları yazmaya başlayınız.

Renkli ve siyah beyaz balık resim için id'ler vererek imageview nesneleri oluşturunuz.



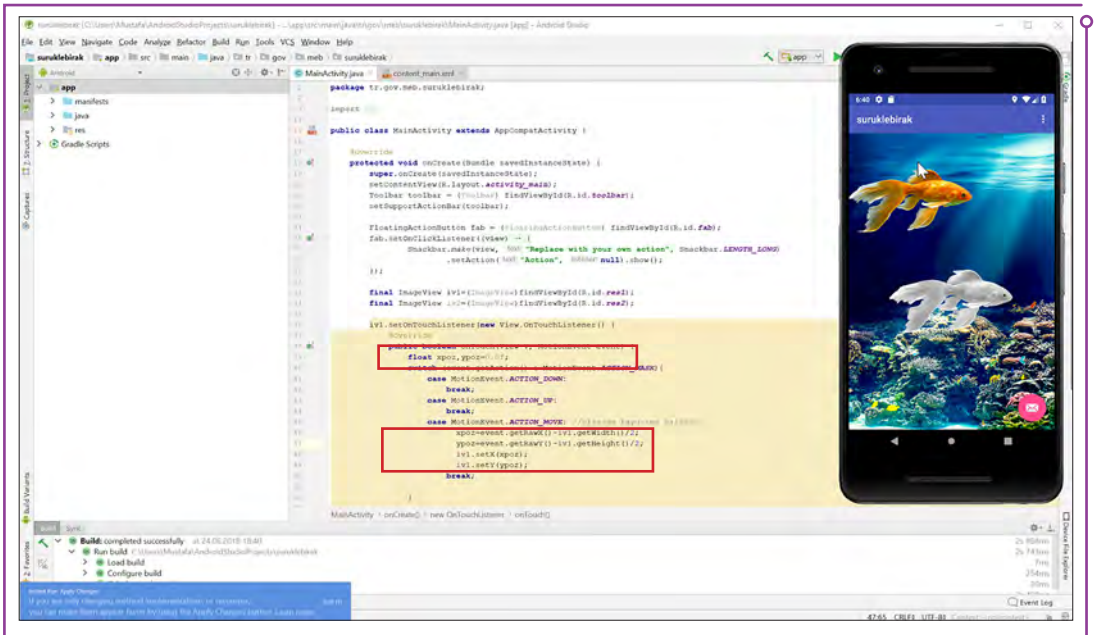
Ekran Görüntüsü 4.283

Sürüklenecek balık resmine dokunma ve sürükleme olayını kontrol eden kodları ekleyiniz. Burada switch kodu ile koşulları yazmanız gerekmektedir. Nesnenin hareketini kontrol etmek için MotionEvent kullanılır. Bu uygulamada MotionEvent olarak ekrana dokunma, ekrandan dokunmayı kesme ve sürükleme durumları kontrol edilecektir.



Ekran Görüntüsü 4.284

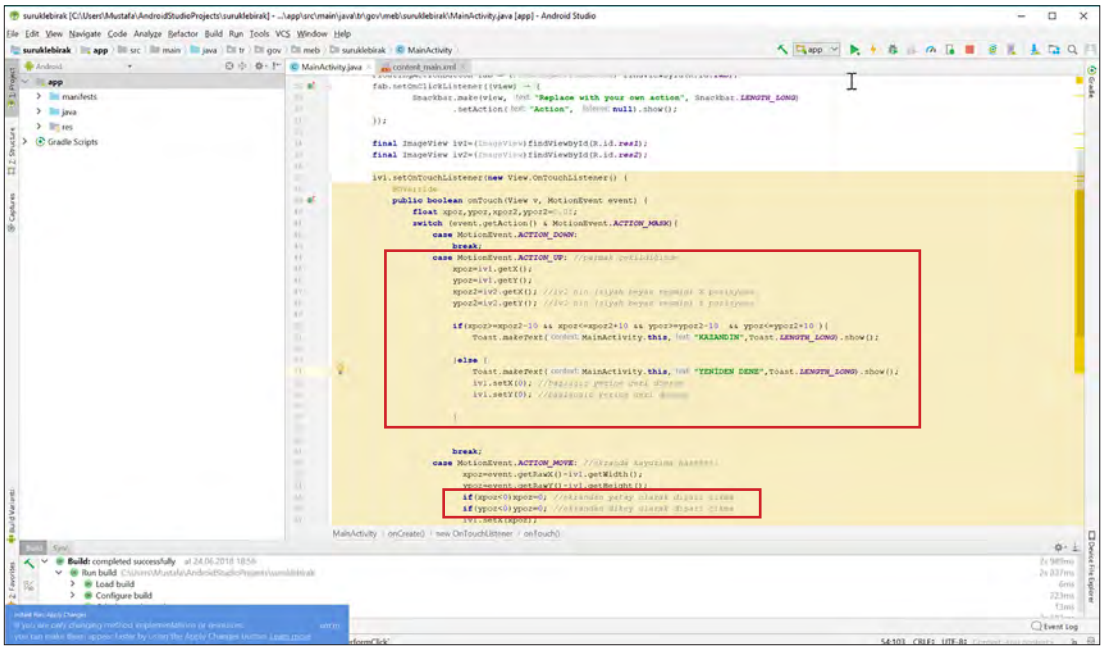
Fare işaretçisinin X ve Y değerlerini almak için xpos ve ypos float değişkenleri oluşturunuz. Daha sonra bu değerleri sürüklenecek balık resminizin X ve Y değerlerine eşitleyiniz. Bu şekilde ekranda parmağınızla renkli balık resmini istediğiniz yere sürükleyebilirsiniz.



Ekran Görüntüsü 4.285

Uygulama bu şekilde çalıştığında sürüklediğiniz balık nesnesi ekranın dışına taşınabilecektir. Bunu engellemek için gerekli kodların eklenmesi gerekmektedir. Daha sonra renkli balık nesnesinin siyah-beyaz balık nesnesine denk gelip gelmediğini öğrenmek için bir kontrol konması gerekmektedir. Bu kontrol parmak ekrandan çekildiğinde yani MotionEvent Action.Up olduğunda kod bloğunun içinde yer almalıdır. İki balığın konumlarının aynı yerde olup olmadığını anlamak için sabit balık nes-

nesinin de X ve Y değerlerini alan değişkenler atanmalıdır. Değişkenler atandıktan sonra eşitliği kontrol eden kodları yazınız. Eşit değilse renkli balık X ve Y değerleri 0 olan konuma geri dönecek ve tekrar dene yazacak, eşitse kazandın yazacak şekilde kodlarınızı ekleyiniz.



Ekran Görüntüsü 4.286

Tüm kodlarınız tamamlandıktan sonra projeniz yandaki gibi olacaktır. Renkli balık nesnesi siyah beyaz balık nesnesine tam olarak bırakıldığında orada kalacak ve "kazandın" yazısı ekranda görünecek; tam üzerine gelmediğinde ise önceki konumuna gidip ekranda "yeniden dene" yazısı görünecektir.

Ekran Görüntüsü 4.287

XML kodlar

```
<RelativeLayout XMLns:android="http://schemas.android.com/apk/res/android"
  XMLns:app="http://schemas.android.com/apk/res-auto"
  XMLns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  app:layout_behavior="@string/appbar_scrolling_view_behavior"
  tools:context=".MainActivity"
  tools:showIn="@layout/activity_main">

  <ImageView
     android:id="@+id/imageView"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_alignParentStart="true"
     android:layout_alignParentTop="true"
     android:scaleType="fitXY"
     app:srcCompat="@drawable/okyanus" />

  <ImageView
     android:id="@+id/res2"
     android:layout_width="250dp"
     android:layout_height="250dp"
     android:layout_alignParentBottom="true"
     android:layout_centerHorizontal="true"
     android:layout_marginBottom="136dp"
     app:srcCompat="@drawable/baliksb" />

  <ImageView
     android:id="@+id/res1"
     android:layout_width="250dp"
     android:layout_height="250dp"
     android:layout_alignParentStart="true"
     android:layout_alignParentTop="true"
     app:srcCompat="@drawable/balik" />
</RelativeLayout>
```

Java kodları

```
package tr.gov.meb.suruklebirak;

import android.media.Image;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
```

```
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.MotionEvent;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ImageView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
     setSupportActionBar(toolbar);

     FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
     fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
           Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                 .setAction("Action", null).show();
        }
     });

     final ImageView iv1=(ImageView)findViewById(R.id.res1);        // iv1 isimli bir ImaveView nesnesi oluştur
     final ImageView iv2=(ImageView)findViewById(R.id.res2);        // iv2 isimli bir ImaveView nesnesi oluştur

     iv1.setOnTouchListener(new View.OnTouchListener() {            // iv1'e tıklanıp tıklanmadığını kontrol et
        @Override
        public boolean onTouch(View v, MotionEvent event) {         // tıklandığında yapılacaklar bloğu
           float xpoz,ypoz,xpoz2,ypoz2=0.0f;
           switch (event.getAction() & MotionEvent.ACTION_MASK){
              case MotionEvent.ACTION_DOWN:
                 break;
              case MotionEvent.ACTION_UP:                            // parmak çekildiğinde
                 xpoz=iv1.getX();
                 ypoz=iv1.getY();
                 xpoz2=iv2.getX();                                   //iv2 nin (siyah beyaz resmin) X pozisyonu
                 ypoz2=iv2.getY();                                   // iv2 nin (siyah beyaz resmin) Y pozisyonu

                 if(xpoz>=xpoz2-10 && xpoz<=xpoz2+10 && ypoz>=ypoz2-10  && ypoz<=ypoz2+10 ){
```

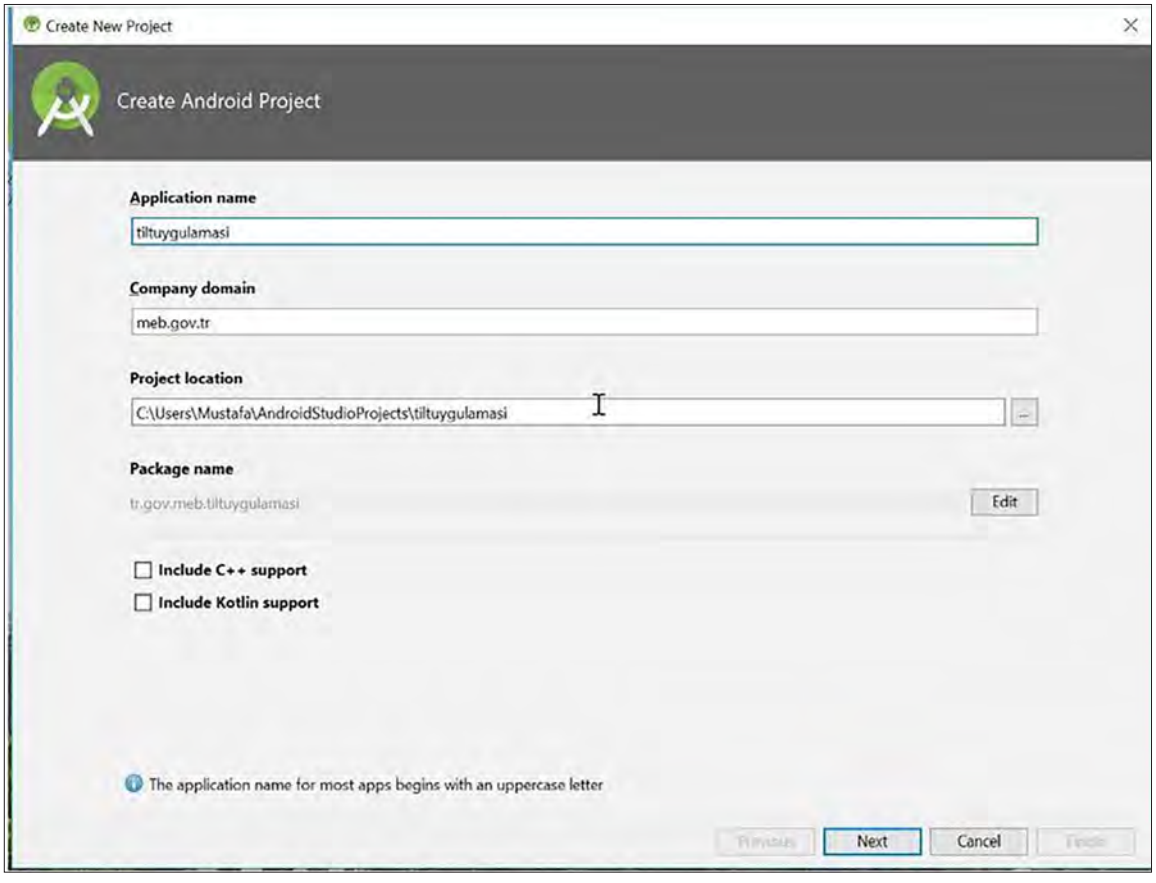
```
                Toast.makeText(MainActivity.this,"KAZANDIN",Toast.LENGTH_LONG).show();

            }else {
                Toast.makeText(MainActivity.this,"YENİDEN DENE",Toast.LENGTH_LONG).show();
                iv1.setX(0);                    //başlangıç yerine geri dönsün
                iv1.setY(0);                    //başlangıç yerine geri dönsün

            }

            break;
        case MotionEvent.ACTION_MOVE:   //ekranda kaydırıma hareketi
            xpoz=event.getRawX()-iv1.getWidth();
            ypoz=event.getRawY()-iv1.getHeight();
            if(xpoz<0)xpoz=0;            //ekrandan yatay olarak dışarı çıkma
            if(ypoz<0)ypoz=0;            //ekrandan dikey olarak dışarı çıkma
            iv1.setX(xpoz);
            iv1.setY(ypoz);
            break;
    }

    return true;
  }
});

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
     int id = item.getItemId();

    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}
}
```

4.3.12. Sensör Uygulamaları

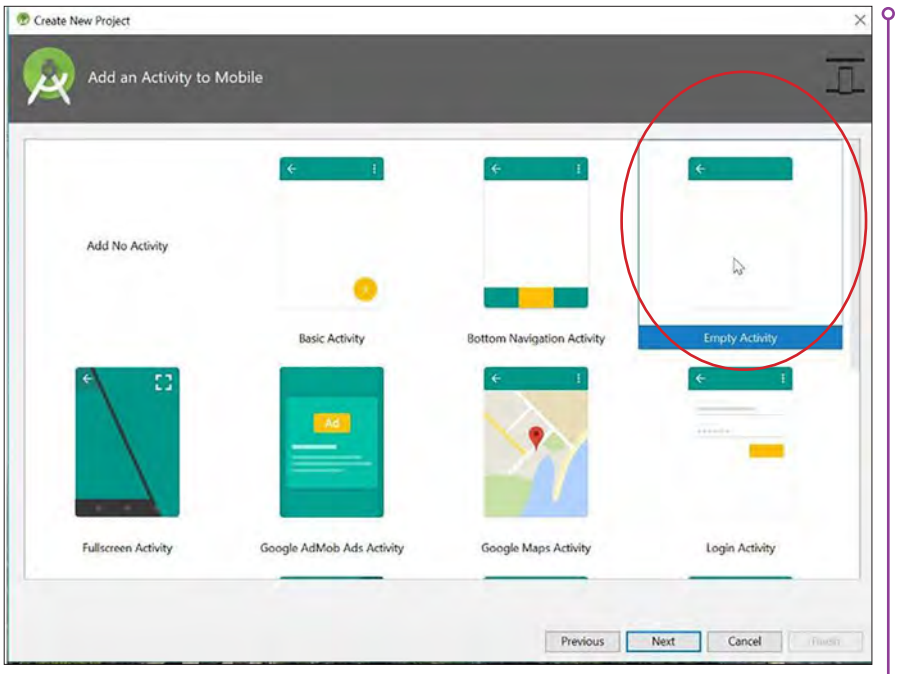
İki adet hareket sensörü uygulaması (Tilt - Eğim ve Shake - Sallama) yapılacaktır. Android'de hareket sensörü uygulamaları, accelerometer (ivme ölçer) ve gyroscop (jiroskop) ile yapılabilmektedir. Her iki sensör de hareket sensörü olmakla birlikte, gyroscop rotasyonu, accelerometer ise ivmelenmeyi kontrol etmektedir. Gyroscop, accelerometer'dan daha yeni bir teknolojidir, Android işletim sisteminin daha yeni versiyonları ile çalışmaktadır ve daha pahalı cihazlar gerektirmektedir. Ayrıca, gyroscop daha sensitive (hassas) uygulamalar yapılmasına imkan tanımasına rağmen, accelerometer'ye göre cihazın pil gücünü daha fazla kullanmaktadır. İki sensörün de kendi içinde avantaj ve dezavantajları bulunmakta, çocuklara yönelik uygulamalarda daha çok accelerometer tercih edilmektedir.

a) Tilt (Eğim) Sensörü

Bu çalışmada accelerometer ile bir tilt sensörü uygulaması yapılacaktır. İlk olarak her zaman olduğu gibi Android projenizi oluşturunuz ve ismini "tiltuygulamasi" olarak belirleyiniz.

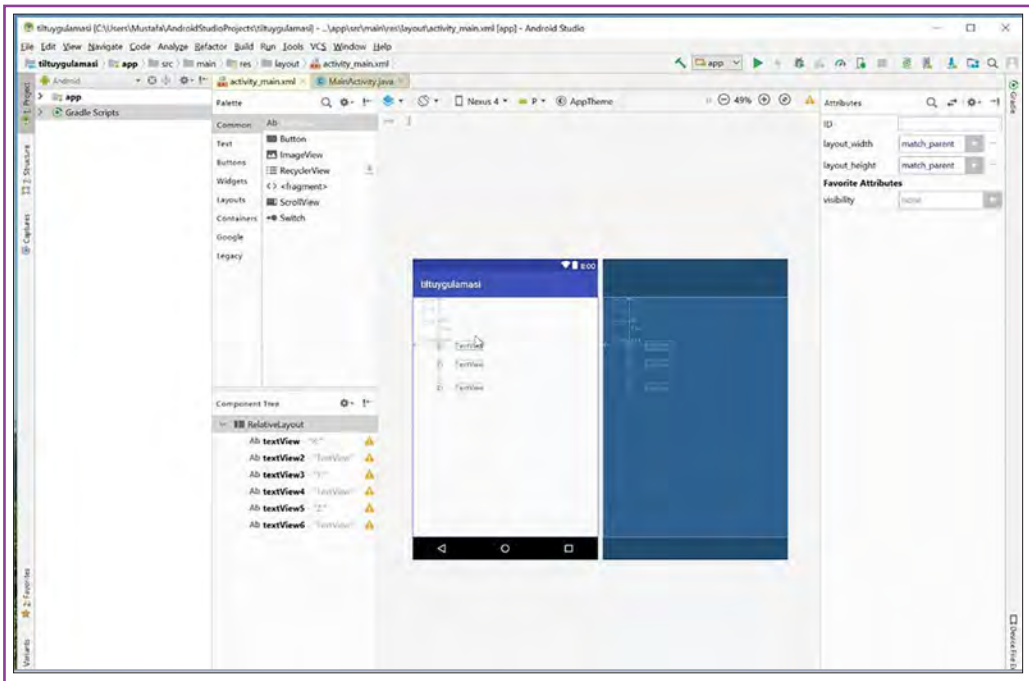


Ekran Görüntüsü 4.288



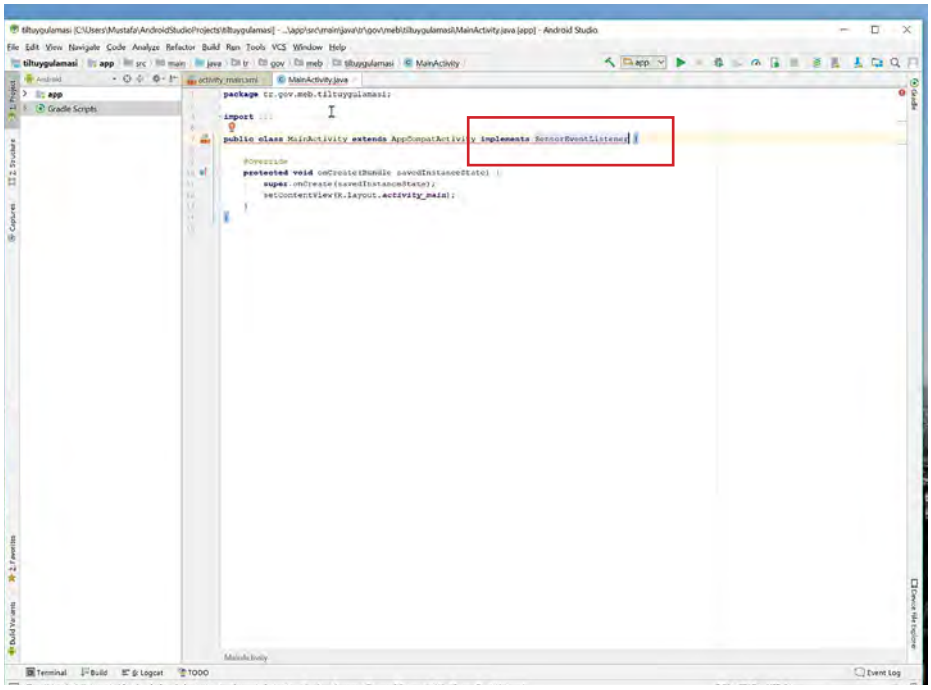
Ekran Görüntüsü 4.289

Her zaman yaptığınız gibi ilk olarak XML dosyanızı açınız ve tasarınınızı yapınız. Hello World'ü kaldırınız ve Content Layout'u Relative Layout'a çeviriniz. Ekrana 6 adet TextView ekleyiniz ve üzerinde X, Y ve Z yazmasını sağlayınız.



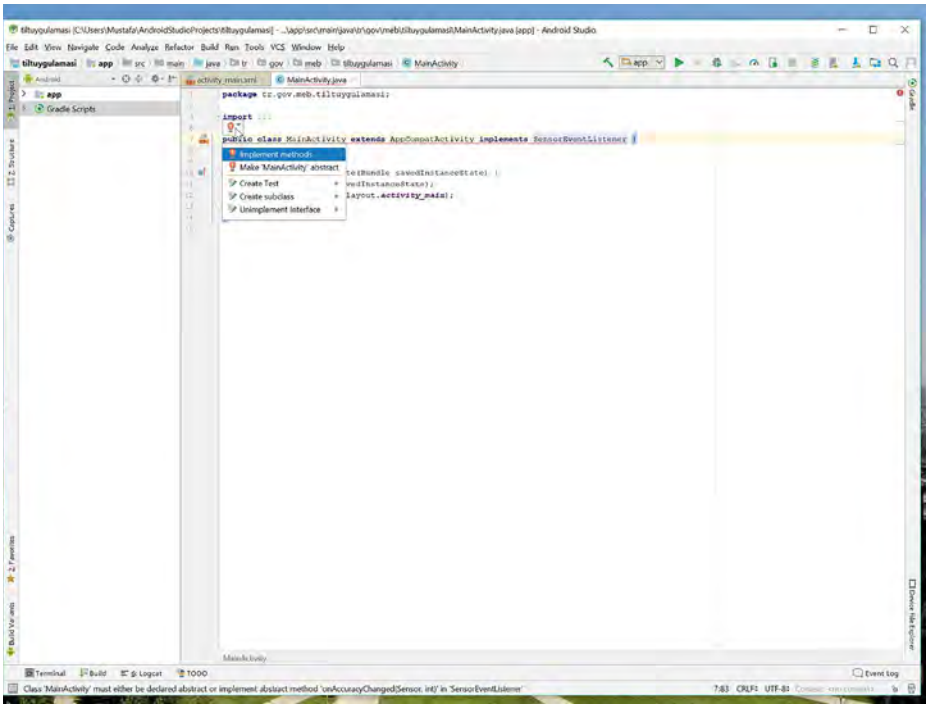
Ekran Görüntüsü 4.290

Tasarım kısmı tamamlandıktan sonra MainActivty.java dosyanıza geliniz ve kodlarınızı yazmaya başlayınız. İlk olarak public class'ına sensoreventlistener uygulaması (implement) eklenecektir.

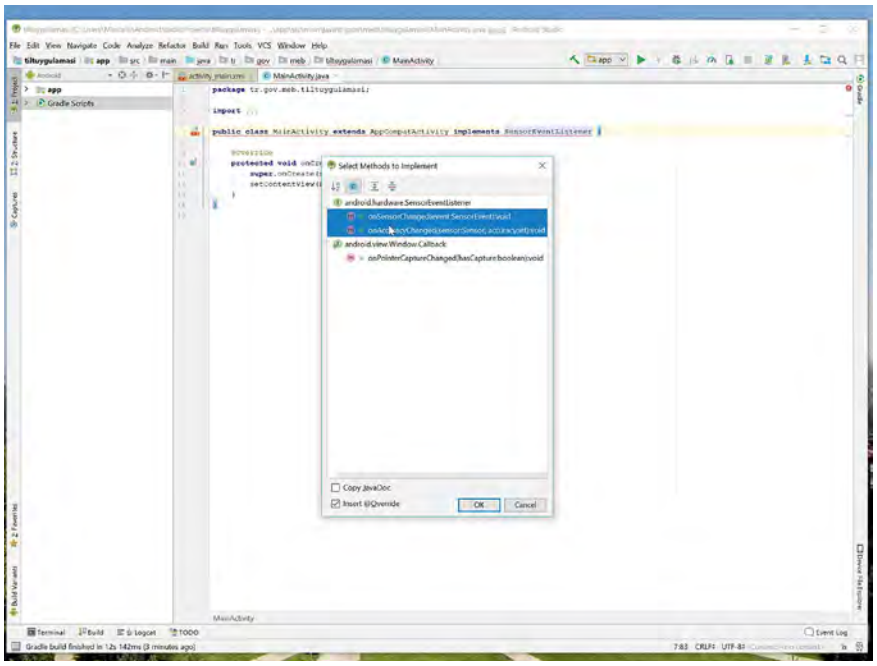


Ekran Görüntüsü 4.291

Bu kodu yazdıktan sonra bir uyarı penceresi (Alt+ Enter ile de açılabilir) gelmektedir. Burada, uygulamanın implement (çalıştırma) metodlarını ekleyebilirsiniz. Sensör değiştiğinde fonksiyonunu eklendiğinde yeni bir pencere açılır. Bu pencerede, sensör değiştiğinde ve doğruluk değeri değiştiğinde yöntemlerini ekleyiniz.

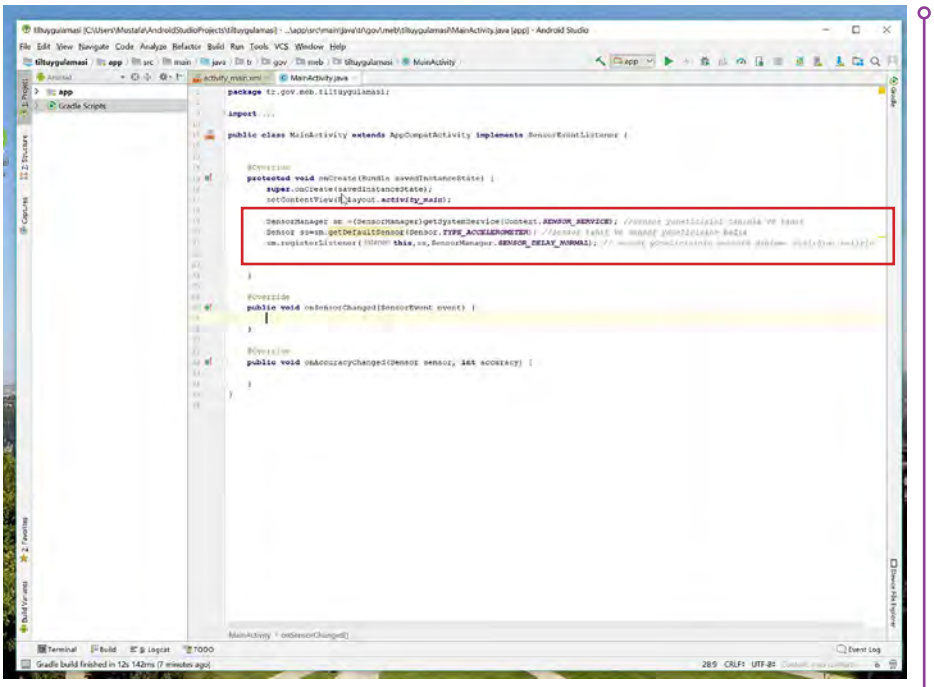


Ekran Görüntüsü 4.292



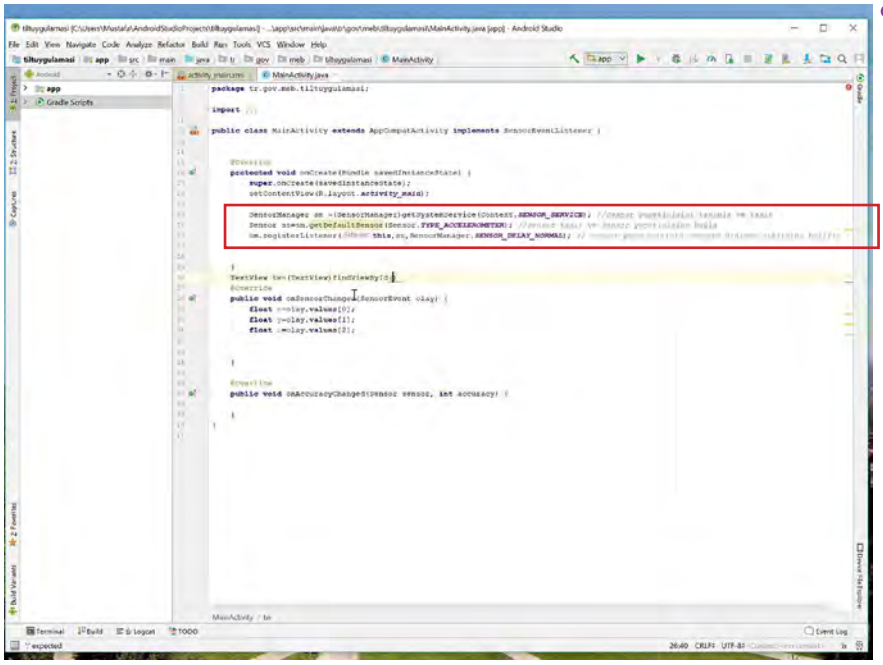
Ekran Görüntüsü 4.293

Sensör nesnesi ve sensör manager olmak üzere iki nesne kullanılacaktır, bu iki nesneyi tanıtıp tanımlayınız.



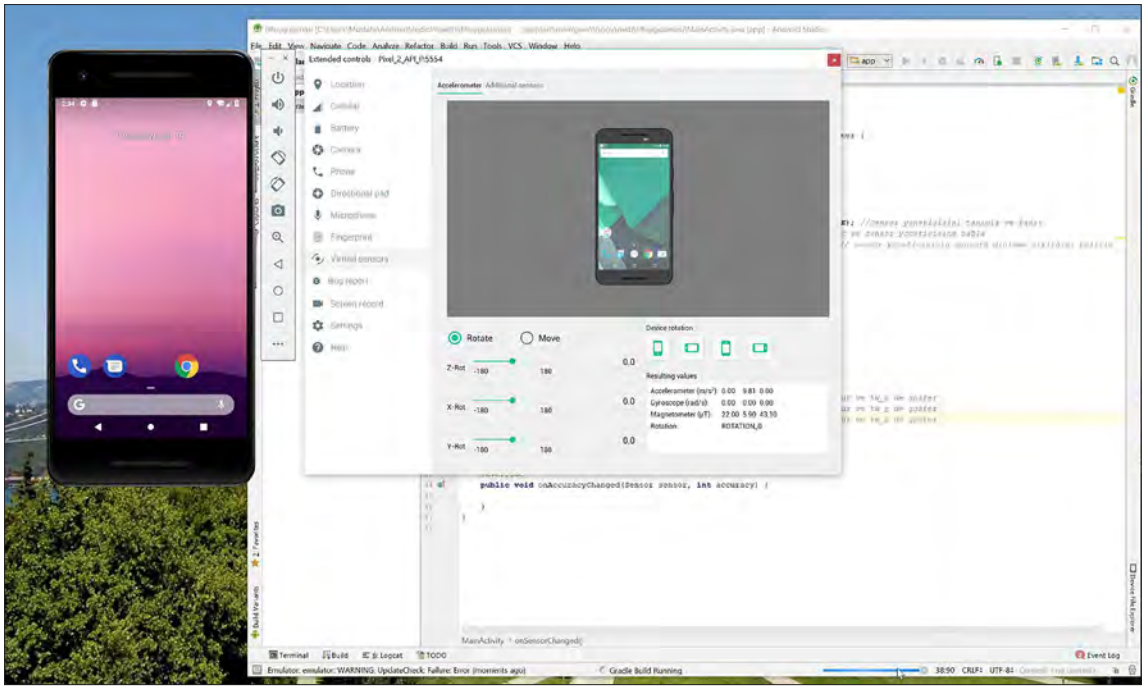
Ekran Görüntüsü 4.294

Sensör değiştiğinde, yani veriler gelmeye başladığında yapılması gerekenlerin belirlenmesi gerekmektedir.



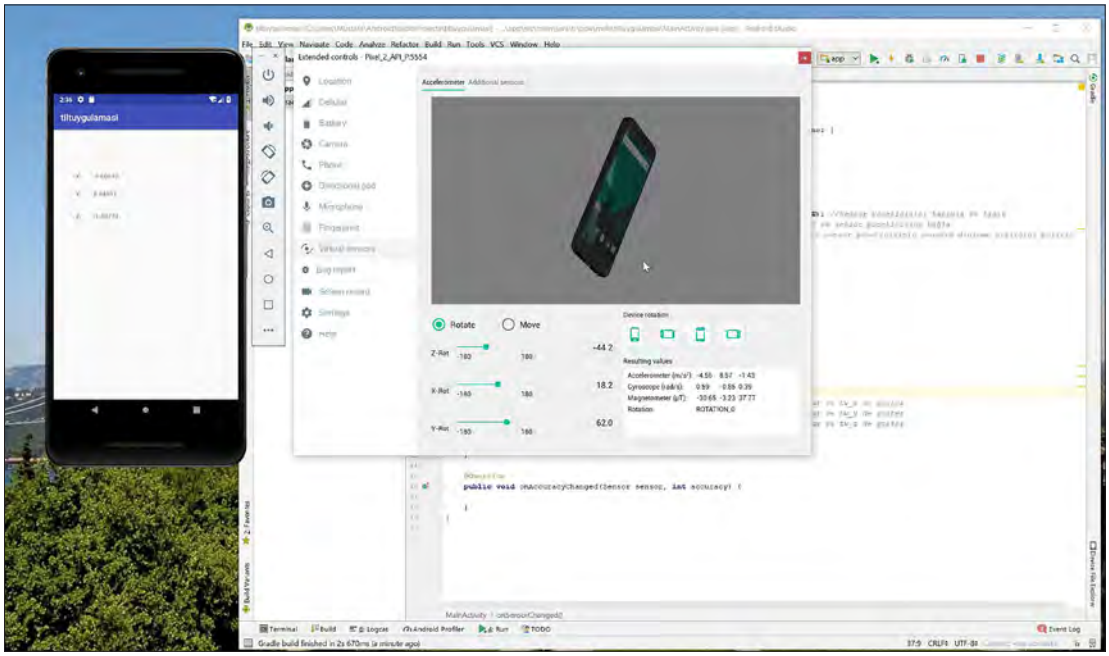
Ekran Görüntüsü 4.295

Emülatörde test etmek için emülatörün hareket ettirilmesi gerekmektedir. Bunun için emülatör üzerinde Virtual Sensor seçeneğini seçip cihazı oynatabilirsiniz:



Ekran Görüntüsü 4.296

Cihazı hareket ettirdiğinizde x, y ve z değerlerinin değiştiğini görebilirsiniz:



Ekran Görüntüsü 4.297

XML Kodları

```
<?XML version="1.0" encoding="utf-8"?>
<RelativeLayout XMLns:android="http://schemas.android.com/apk/res/android"
   XMLns:app="http://schemas.android.com/apk/res-auto"
   XMLns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">

   <TextView
      android:id="@+id/TextView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentStart="true"
      android:layout_alignParentTop="true"
      android:layout_marginStart="50dp"
      android:layout_marginTop="94dp"
      android:text="X:" />

   <TextView
      android:id="@+id/TextView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentStart="true"
```

```
    android:layout_alignTop="@+id/TextView"
    android:layout_marginStart="89dp"
    android:text="TextView" />

  <TextView
    android:id="@+id/TextView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignStart="@+id/TextView"
    android:layout_marginTop="134dp"
    android:text="Y:" />

  <TextView
    android:id="@+id/TextView4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignStart="@+id/TextView2"
    android:layout_alignTop="@+id/TextView3"
    android:text="TextView" />

  <TextView
    android:id="@+id/TextView5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignStart="@+id/TextView"
    android:layout_marginTop="184dp"
    android:text="Z:" />

  <TextView
    android:id="@+id/TextView6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignStart="@+id/TextView2"
    android:layout_alignTop="@+id/TextView5"
    android:text="TextView" />
</RelativeLayout>
```

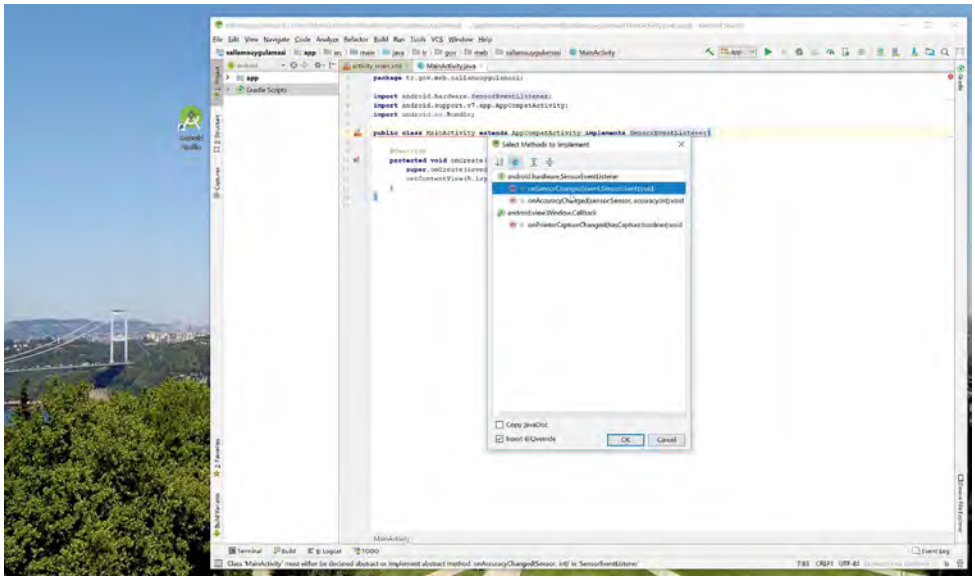
Java Kodları

```
package tr.gov.meb.tiltuygulamasi;

import android.content.Context;
import android.hardware.Sensor;
```

```
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements SensorEventListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        SensorManager sm =(SensorManager)getSystemService(Context.SENSOR_SERVICE); //Sensör yöneticisini tanımla ve tanıt
        Sensor ss=sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER); //sensor tanıt ve sensör yöneticisine bağla
        sm.registerListener(this,ss,SensorManager.SENSOR_DELAY_NORMAL); // sensör yöneticisinin sensörü dinleme sıklığını belirle

    }

    @Override
    public void onSensorChanged(SensorEvent olay) {
        float x=olay.values[0];
        float y=olay.values[1];
        float z=olay.values[2];

        TextView tw_x=(TextView)findViewById(R.id.TextView2);
        TextView tw_y=(TextView)findViewById(R.id.TextView4);
        TextView tw_z=(TextView)findViewById(R.id.TextView6);

        tw_x.setText(String.valueOf(x)); //x değişkenin değerini Stringe dönüştür ve tw_x de göster
        tw_y.setText(String.valueOf(y)); //y değişkenin değerini Stringe dönüştür ve tw_y de göster
        tw_z.setText(String.valueOf(z)); //z değişkenin değerini Stringe dönüştür ve tw_z de göster

    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }
}
```

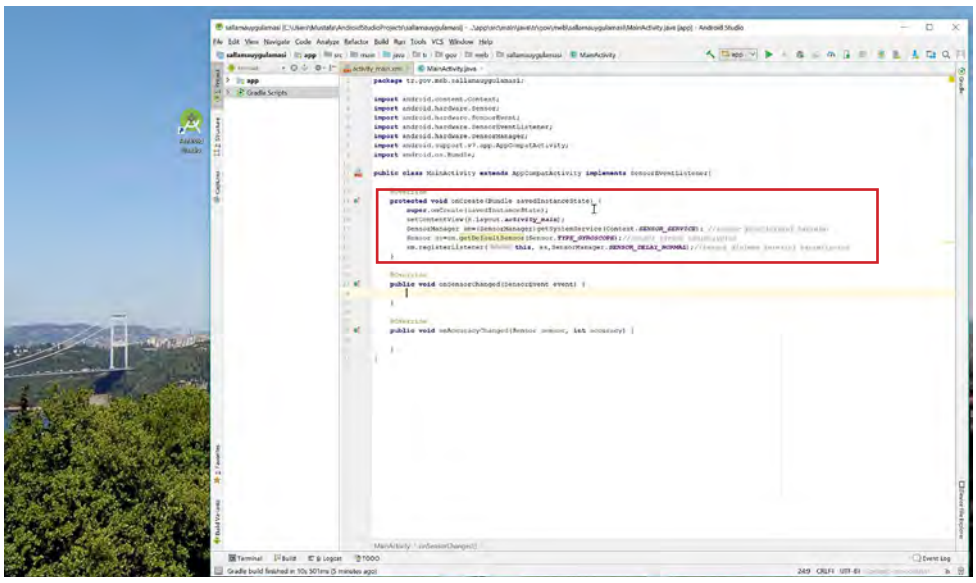
b) Shake (Sallama) Sensörü

Cihaz sallandığında ekranın rengini değiştiren, durduğunda tekrar eski rengine çeviren bir shake (sallama) uygulaması yapılacaktır. Projenizin ismi SallamaUygulamasi olsun ve Empty (Boş) bir proje açınız. Aynı mantıkla bir sensör olay yöneticisi tanımlanacaktır. Yine XML dosyanıza geliniz, Hello World'ü kaldırınız, Relative Layout'u ayarlayınız ve id'sini "cerceve" koyunuz. Ekranınıza cihazın sallanıp sallanmadığını gösterecek bir TextView ekleyiniz ve içini boşaltınız. Java dosyanıza geliniz ve bir önceki tilt uygulamasında olduğu gibi sensör olay dinleyicisini çalıştıran ve bu sensör olay dinleyicisinin metotlarını çalıştıran implement'i ekleyiniz.



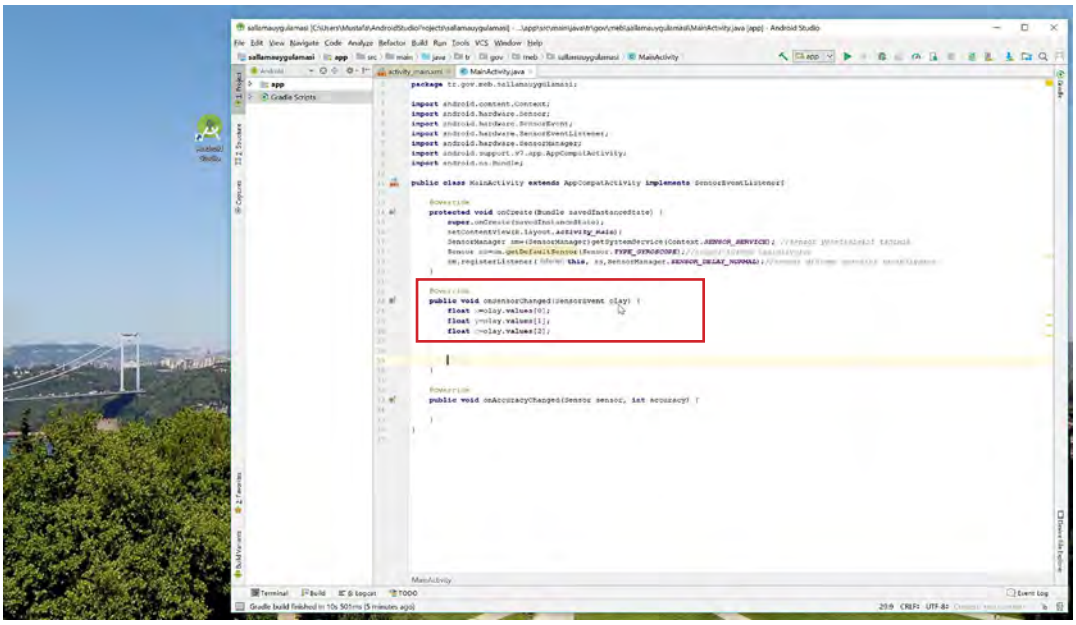
Ekran Görüntüsü 4.298

Tilt uygulamasında yaptığınız gibi oncreate'in altına sensörünüzü ve sensör yöneticinizi tanımlayınız. Burada sallama işlemi gyroscop sensörü ile kontrol edilecektir.



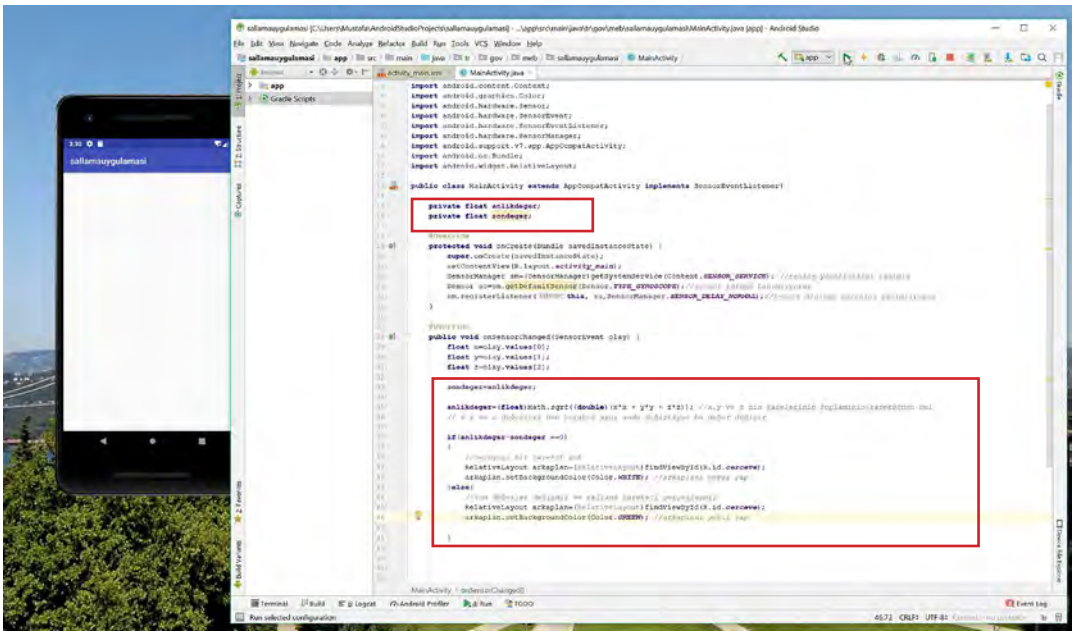
Ekran Görüntüsü 4.299

Sensörünüzü ve onu dinleyen uygulamanızı (implement) tanımlamış oldunuz. Şimdi durum değiştiğinde (cihaz sallandığında) ne olacağı belirlenecektir. Bunun için cihazın x, y ve z konumlarını alınması gerekir.



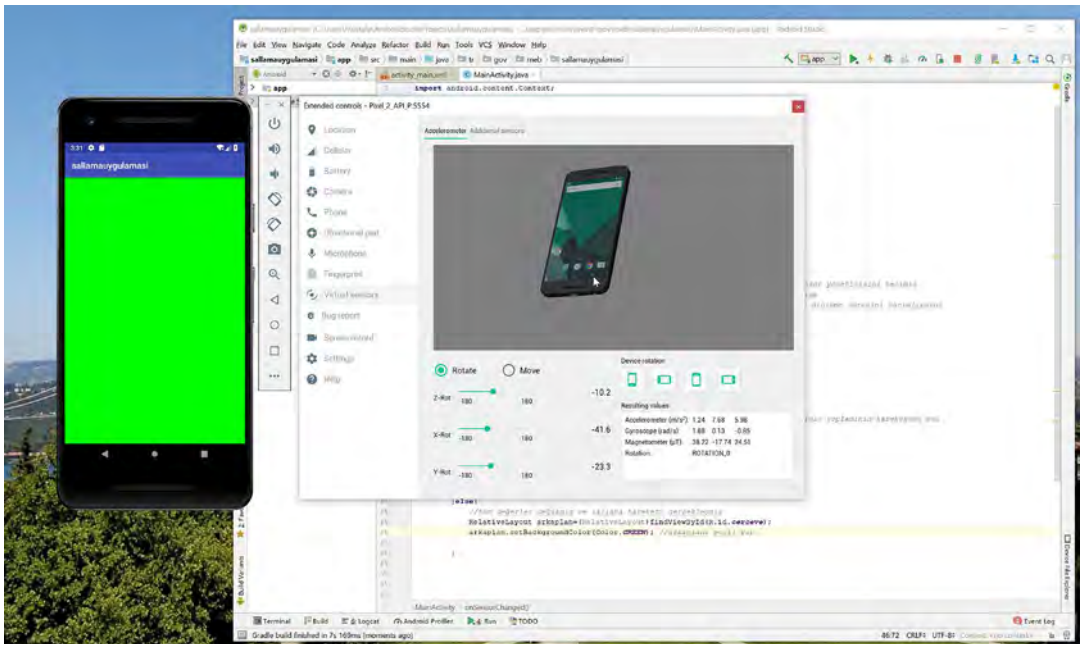
Ekran Görüntüsü 4.300

x,y ve z'nin durumuna göre (değerlerdeki hareketliliğe, değişime göre), üçünün de değerleri değişirse cihazın sallandığı anlaşılacaktır. Cihaz sallandığında arka plan yeşil, durdurulduğunda beyaz olacaktır. Değişkenlerinizi global olarak tanımlayınız. Burada bir matematiksel fonksiyon (değerlerin karelerinin toplamının karekökünü alan) kullanılacaktır. Önce bu kodları yazalım ve daha sonra açıklayalım:

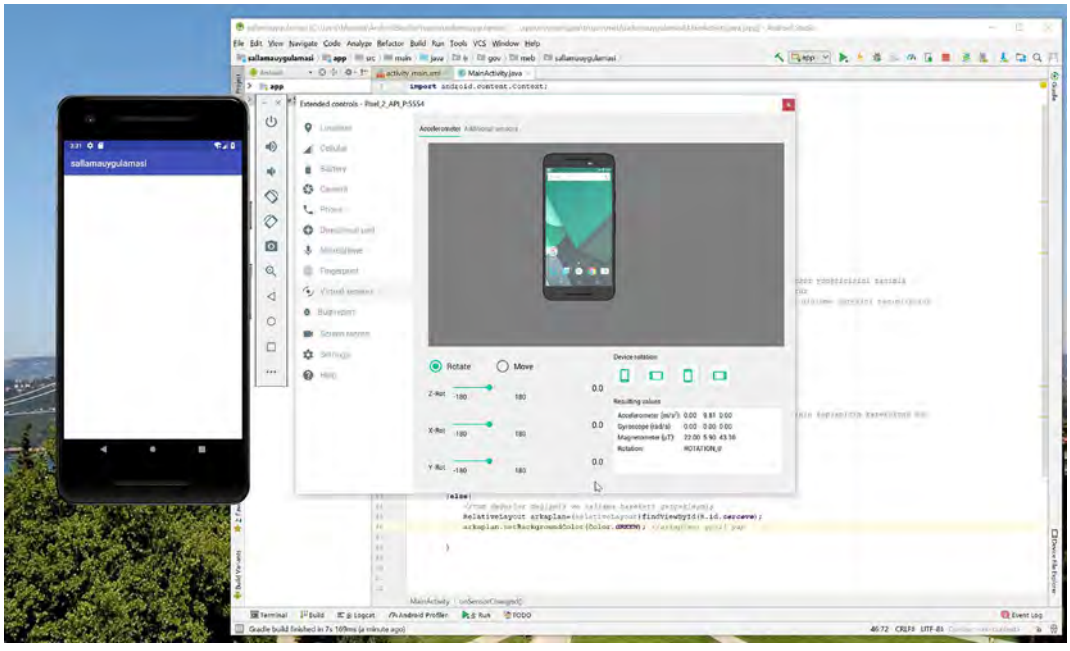


Ekran Görüntüsü 4.301

Projenizi çalıştırınız ve cihazı salladığınızda arka plan renginin değiştiğini, bırakıldığında eski haline geldiğini gözlemleyiniz.

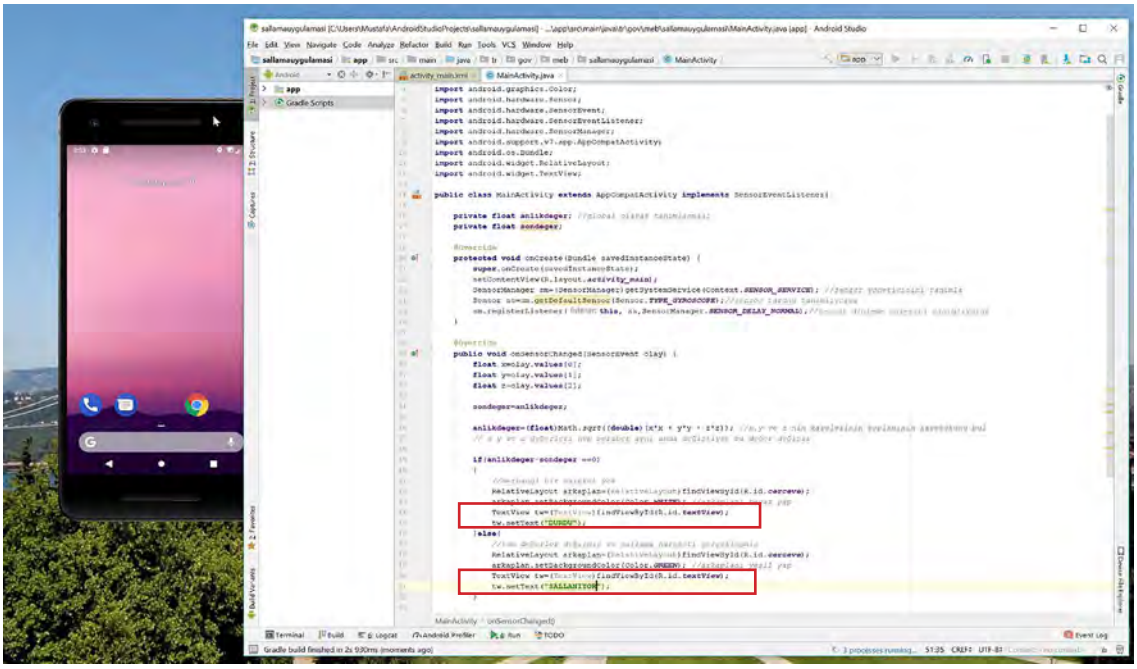


Ekran Görüntüsü 4.302



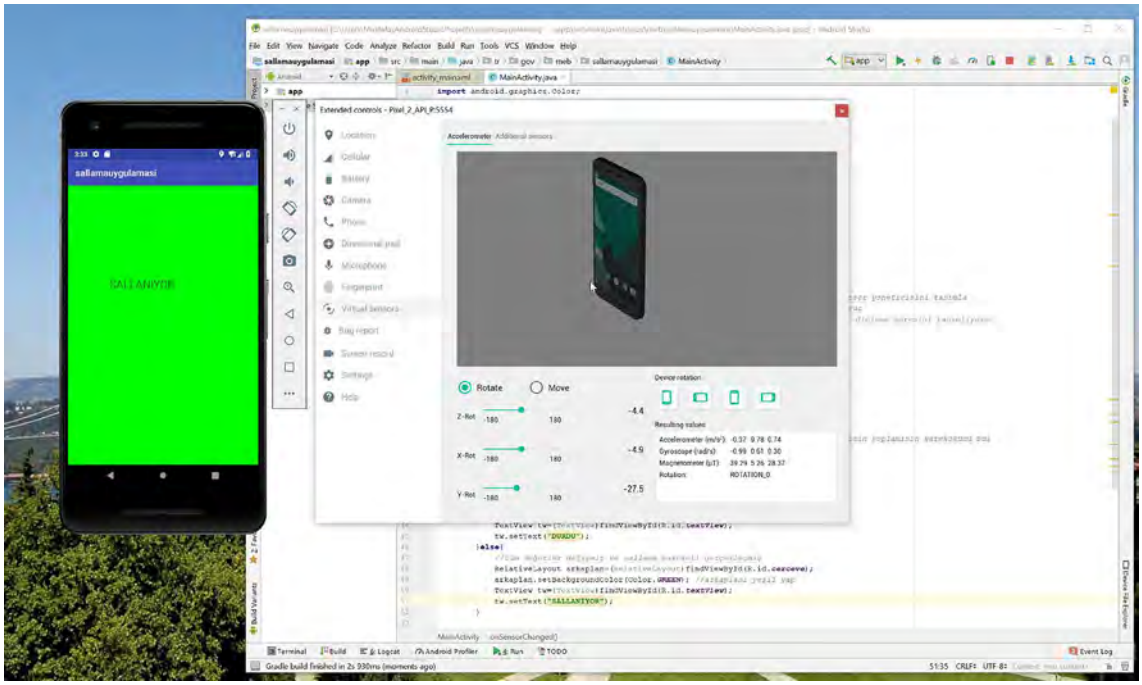
Ekran Görüntüsü 4.303

Son olarak, eklenen TextView'de cihazın durumuna göre "sallanıyor" veya "durdu" yazmasını sağlayan kodları ekleyiniz.



Ekran Görüntüsü 4.304

Yine çalıştırınız ve TextView'inizdeki değişimi de gözlemleyiniz.



Ekran Görüntüsü 4.305

XML Kodları

```
<?XML version="1.0" encoding="utf-8"?>
<RelativeLayout XMLns:android="http://schemas.android.com/apk/res/android"
    XMLns:app="http://schemas.android.com/apk/res-auto"
    XMLns:tools="http://schemas.android.com/tools"
    android:id="@+id/cerceve"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/TextView"
        android:layout_width="231dp"
        android:layout_height="80dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="197dp"
        android:textSize="24sp" />
</RelativeLayout>
```

Java Kodları

```
package tr.gov.meb.sallamauygulamasi;

import android.content.Context;
import android.graphics.Color;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.RelativeLayout;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements SensorEventListener{

    private float anlikdeger; //global olarak tanımlanmalı
    private float sondeger;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        SensorManager sm=(SensorManager)getSystemService(Context.SENSOR_SERVICE); //sensör yöneticisini tanımla
```

```
    Sensor ss=sm.getDefaultSensor(Sensor.TYPE_GYROSCOPE); //sensör türünü tanımlıyoruz
    sm.registerListener(this, ss,SensorManager.SENSOR_DELAY_NORMAL); //sensör dinleme süresini tanımlıyoruz
  }

  @Override
  public void onSensorChanged(SensorEvent olay) {
    float x=olay.values[0];
    float y=olay.values[1];
    float z=olay.values[2];

    sondeger=anlikdeger;

    anlikdeger=(float)Math.sqrt((double)(x*x + y*y + z*z)); //x,y ve z nin karelerinin toplamının karekökünü bul
    // x y ve z değerleri hep beraber aynı anda değiştiyse bu değer değişir

    if(anlikdeger-sondeger ==0)
    {
      //herhangi bir hareket yok
      RelativeLayout arkaplan=(RelativeLayout)findViewById(R.id.cerceve);
      arkaplan.setBackgroundColor(Color.WHITE); //arkaplanı beyaz yap
      TextView tw=(TextView)findViewById(R.id.TextView);
      tw.setText("DURDU");
    }else{
      //tüm değerler değişmiş ve sallama hareketi gerçekleşmiş
      RelativeLayout arkaplan=(RelativeLayout)findViewById(R.id.cerceve);
      arkaplan.setBackgroundColor(Color.GREEN); //arkaplanı yeşil yap
      TextView tw=(TextView)findViewById(R.id.TextView);
      tw.setText("SALLANIYOR");
    }

  }

  @Override
  public void onAccuracyChanged(Sensor sensor, int accuracy) {

  }
}
```

5. DOSYA İŞLEMLERİ

5.1. Xcode Dosya İşlemleri

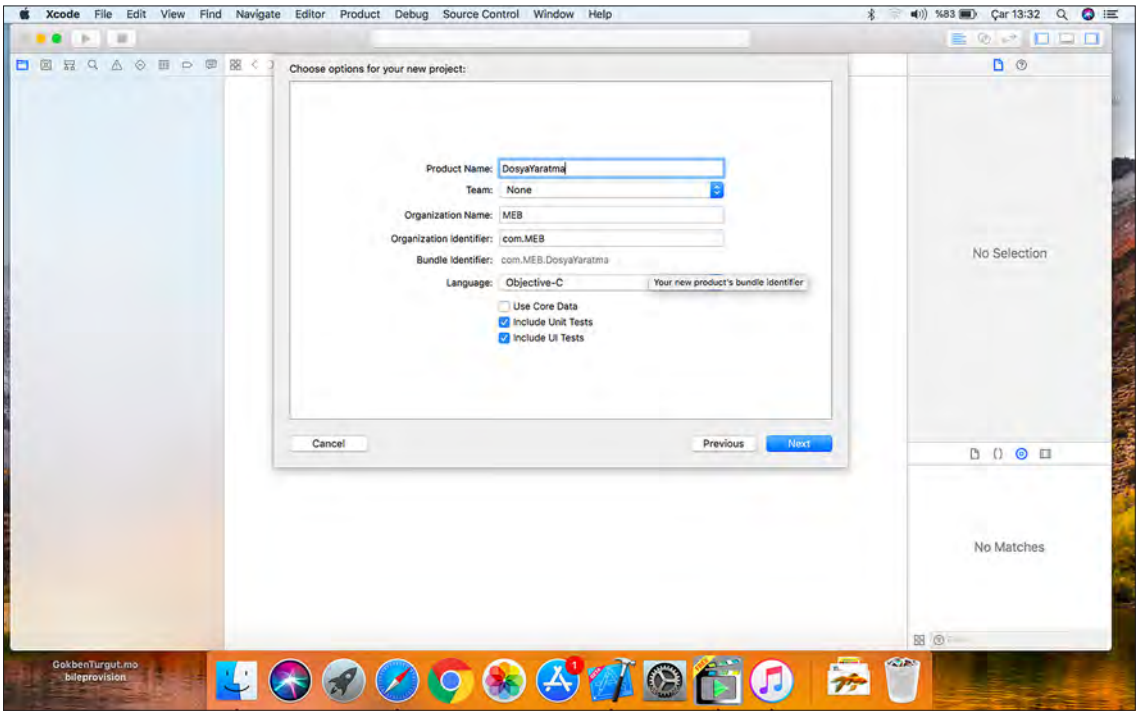
5.1.1. Dosya Oluşturma

Xcode projelerinde .plist uzantılı dosyalar oluşturularak veritabanı gerektirmeyen basit uygulamalarda bilgiler kullanıcının cihazına kaydedilmektedir. Plist dosyaları masaüstü ve web programlamacılığında kullanılan text (metin) dosyalarından biraz daha farklı olarak gelişmiş bir dizi ortamı sağlamaktadır. Alınan kayıtlar plist dosyalarına kaydedilirken çeşitli diziler (dictionary, array vb.) oluşturularak içine yazılabilmektedir. Bu verilere daha sonra tekrar erişim istendiğinde de bir diziden bir eleman çekme işlemi kadar basit yöntemlerle dosya okunabilmekte, değiştirilebilmekte ve yazılabilmektedir.

Uygulama içerisinde cihaza kaydedilecek plist dosyası yaratma işlemi elle manuel olarak yapılabildiği gibi kodla da proje içerisinden oluşturulabilmektedir. Bu bölümde iki farklı yöntemle dosya oluşturma işlemleri gösterilecektir.

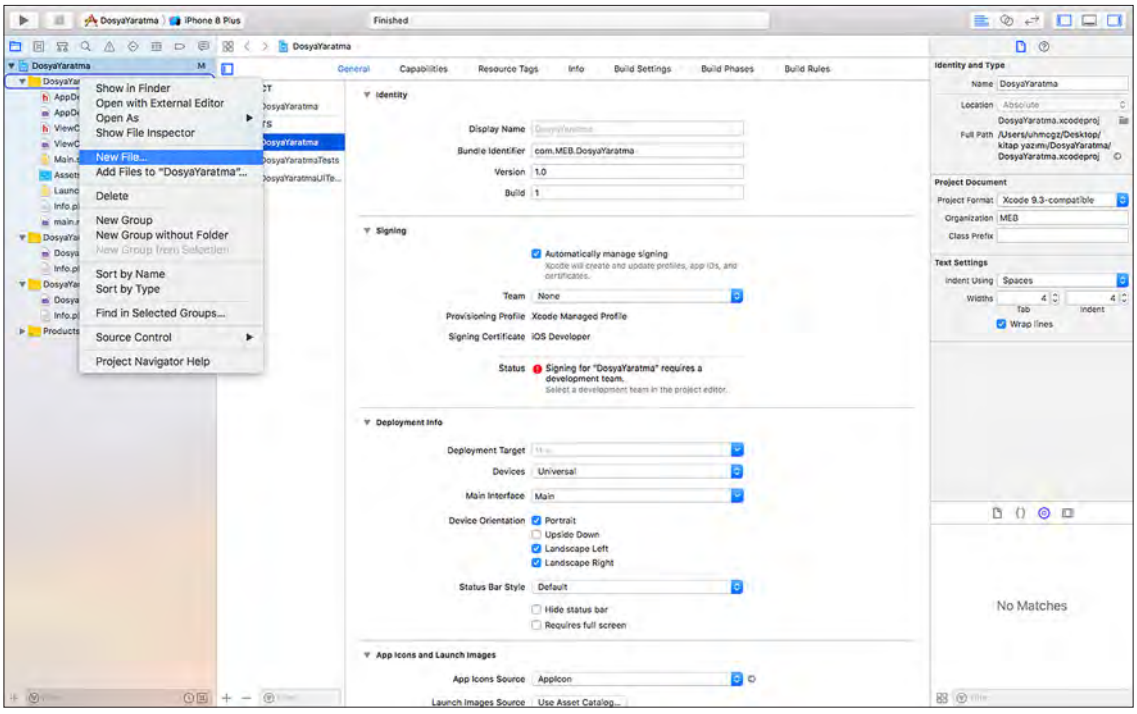
Manuel Dosya Oluşturma İşlemi

DosyaYaratma isimli bir Xcode projesi açılır ve diske kaydedilir.

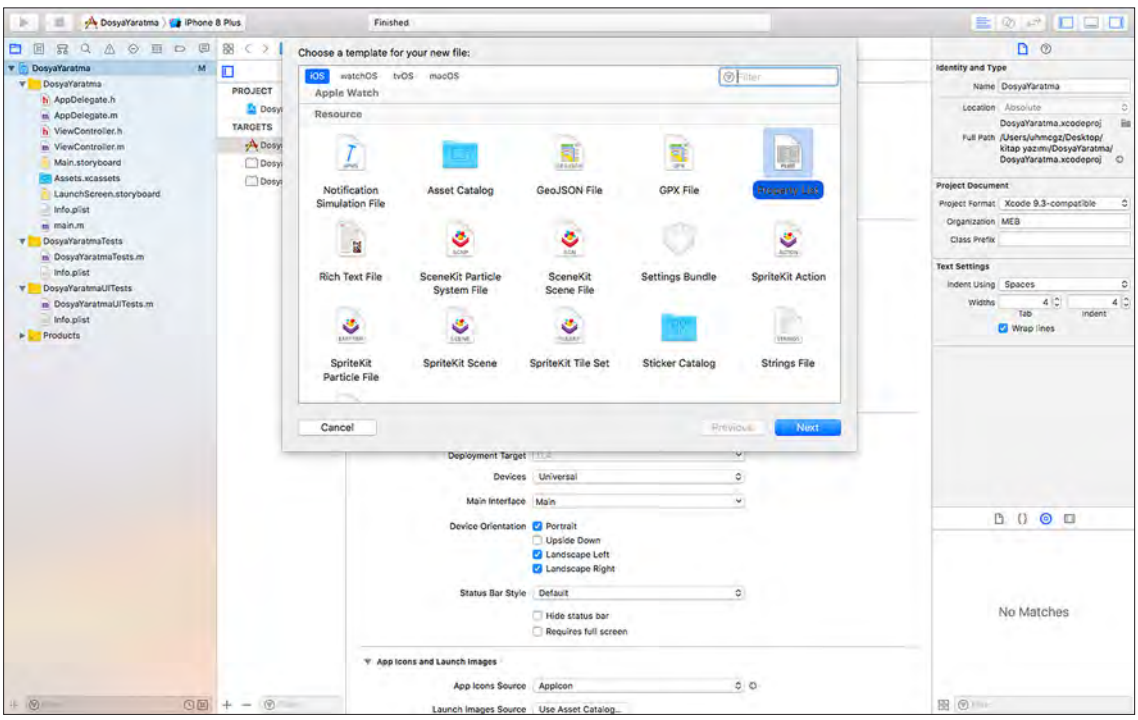


Ekran Görüntüsü 5.1

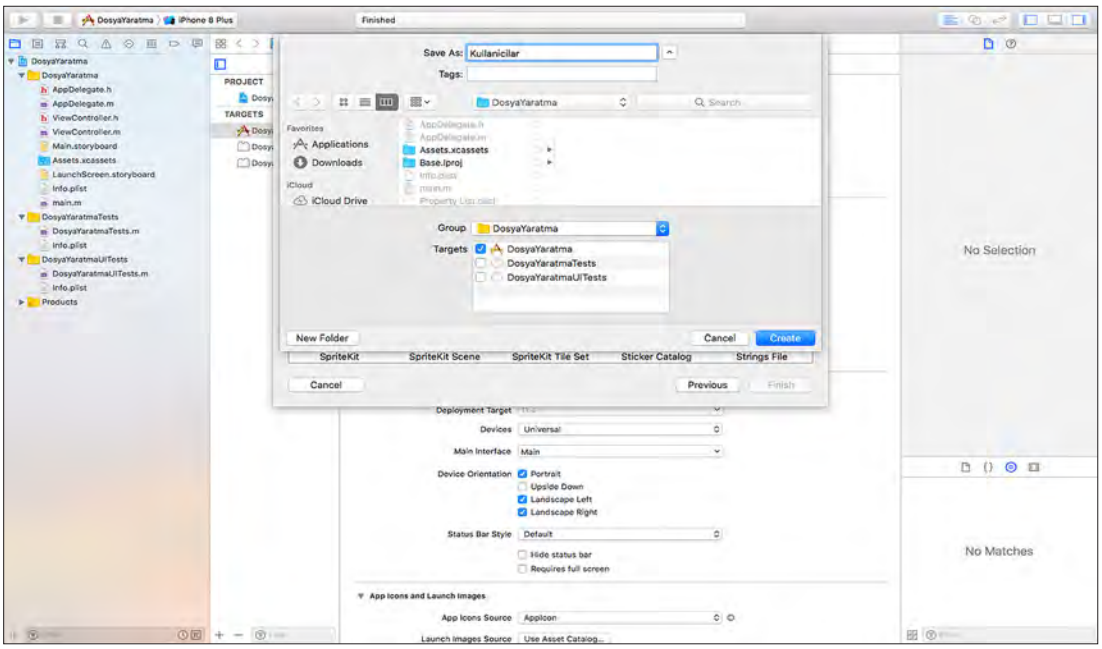
Sol bölmede proje klasörü üzerinde sağ tıklanarak "New File" seçeneği tıklanır. Açılan pencerede alt bölümlerden Property List seçilerek Next düğmesine tıklanır. Yine açılan dosyayı projeye kaydetme penceresinde plist dosyasına "Kullanicilar" ismi verilerek Create düğmesine tıklanır.



Ekran Görüntüsü 5.2

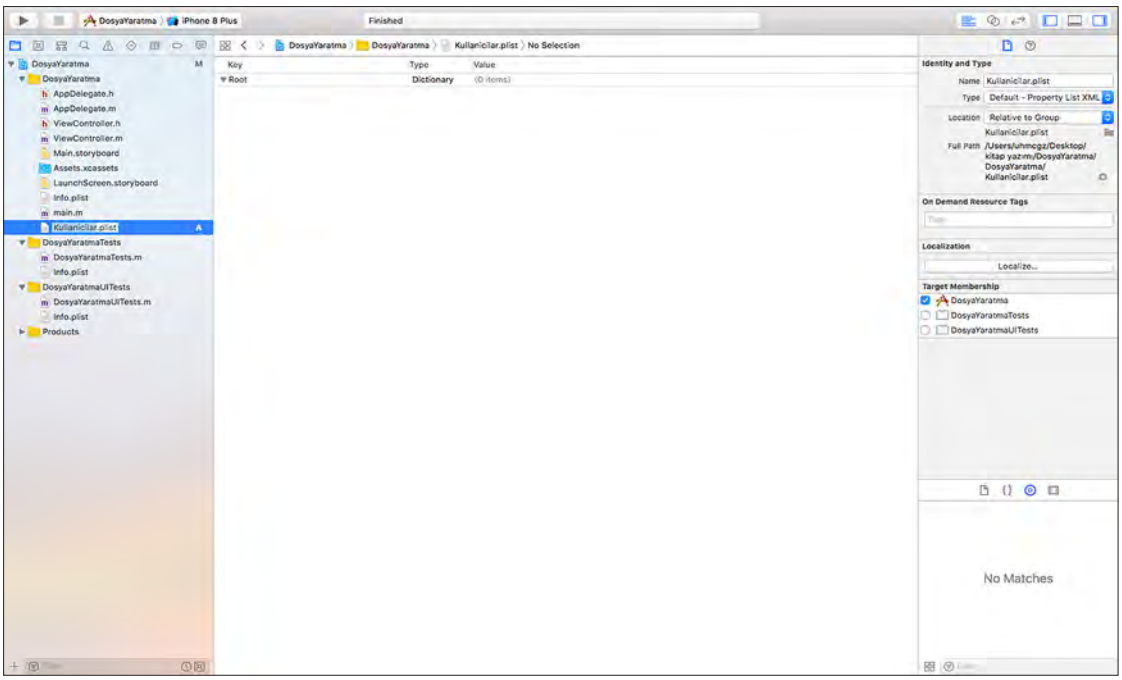


Ekran Görüntüsü 5.3



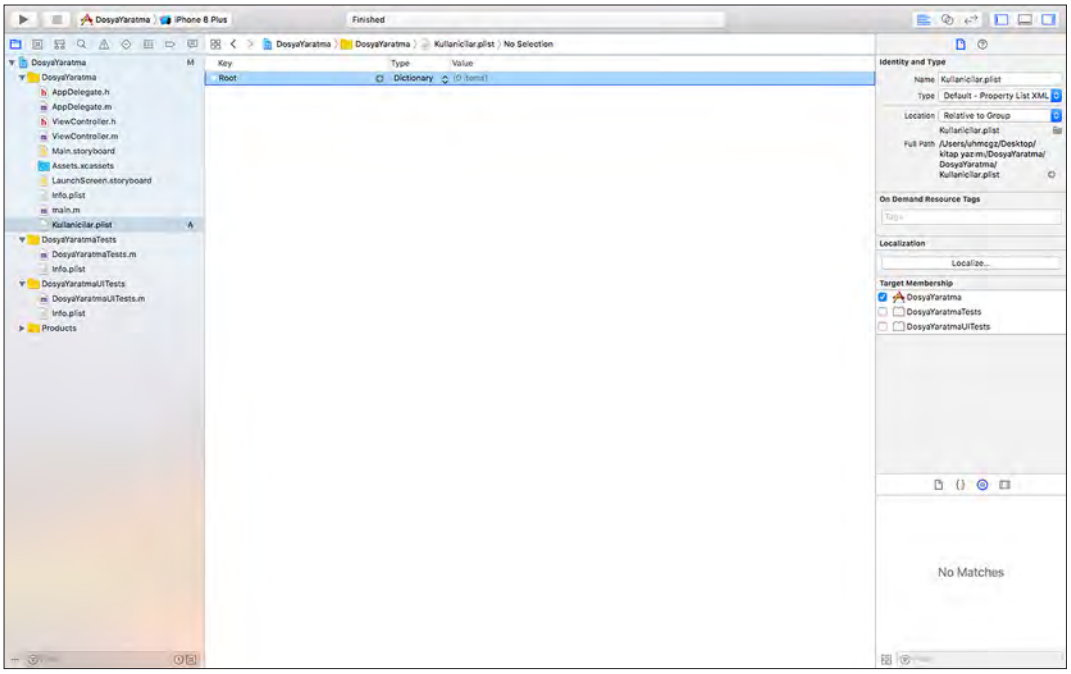
Ekran Görüntüsü 5.4

Dosyayı oluşturma işleminden sonra sol bölmedeki dosyalar kısmından Kullanicilar.plist dosyayı görülecektir. Dosyanın projede açıldığındaki görüntüsü aşağıdaki gibi olacaktır.

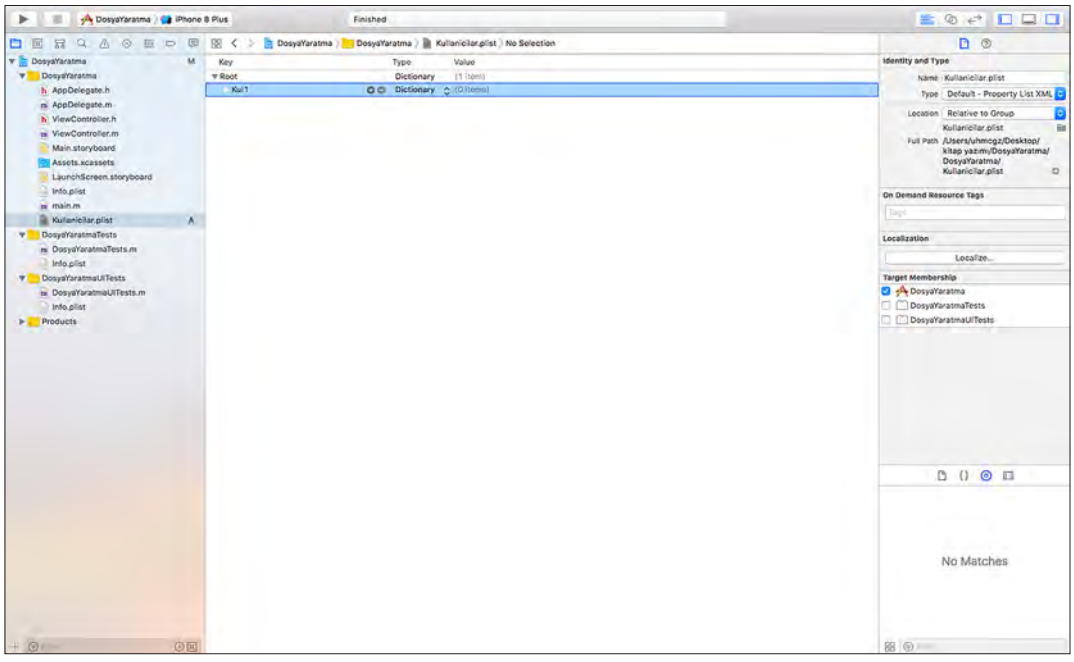


Ekran Görüntüsü 5.5

Dosyaya elle kayıt (veri) eklemek için Root tıklanır ve beliren + (artı) işaretine tıklanarak kayıtlar Kay1, Kay2, Kay3 şeklinde eklenir. Eklenen bu alanlar içeriğinde ad, soyad ve şifre gibi birden fazla bilgi saklayacağı için Type'ı Dictionary olarak seçilir.

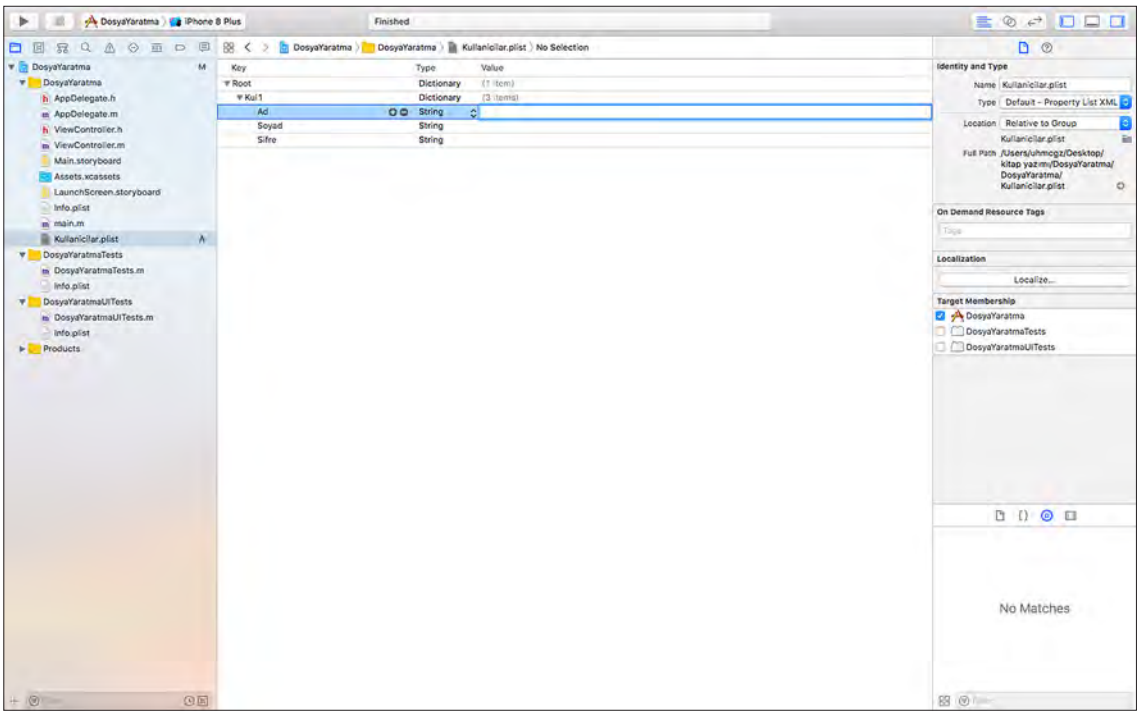


Ekran Görüntüsü 5.6



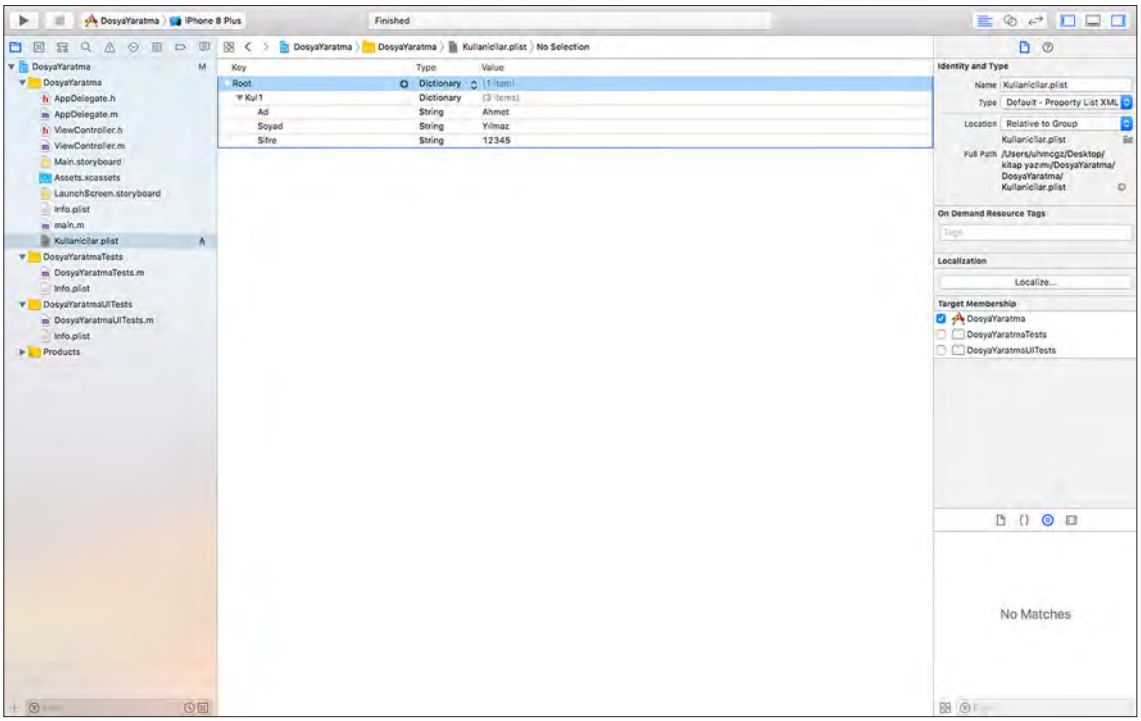
Ekran Görüntüsü 5.7

Oluşturulan Dictionary tipindeki Kul1 alanına ad, soyad ve sifre alanları eklenir. Burada dikkat edilmesi gereken önemli nokta, Kul1'in altında ad, soyad, sifre alanlarının oluşturulmasıdır. Bunun için de Kul1 seçili iken solunda bulunan ok işareti tıklanarak aşağı yöne bakması sağlandıktan sonra + (artı) işaretine tıklanarak sırasıyla ad, soyad ve sifre alanları oluşturulur. Bu alanlar metin tipinde tek kayıt alacağı için Type'ı String olarak seçilir.



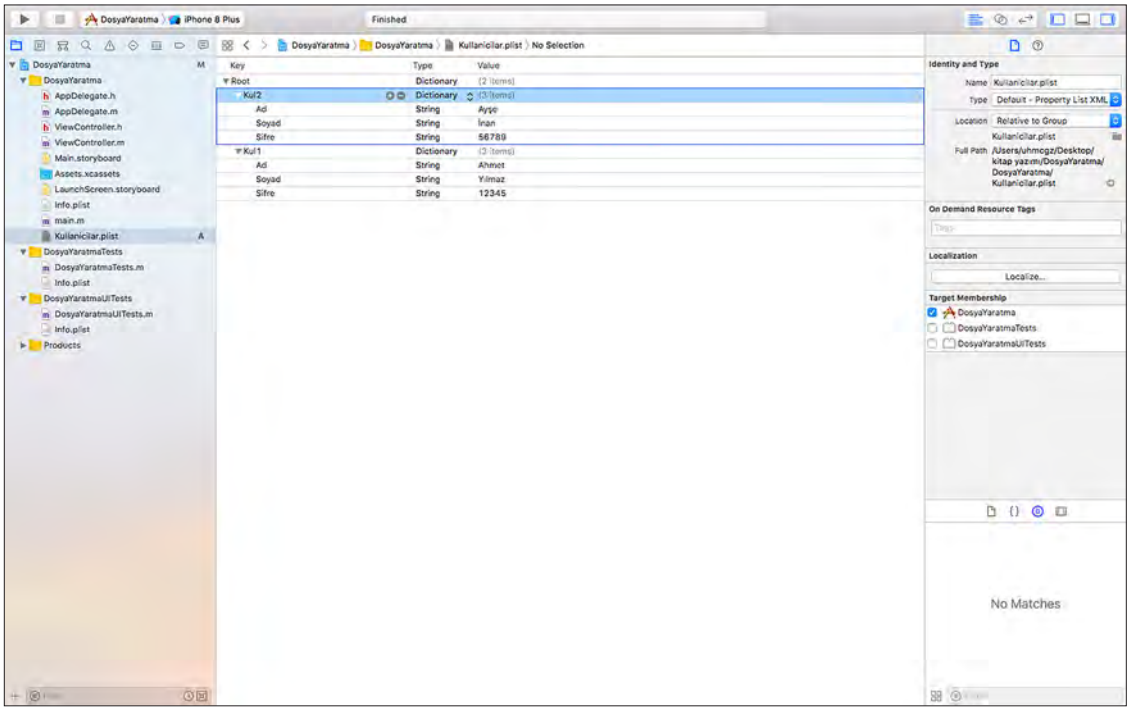
Ekran Görüntüsü 5.8

Kayıtlar elle girilmesi isteniyorsa ad, soyad ve şifre alanlarının karşısına (Value'ya) tıklanarak değer girilir.



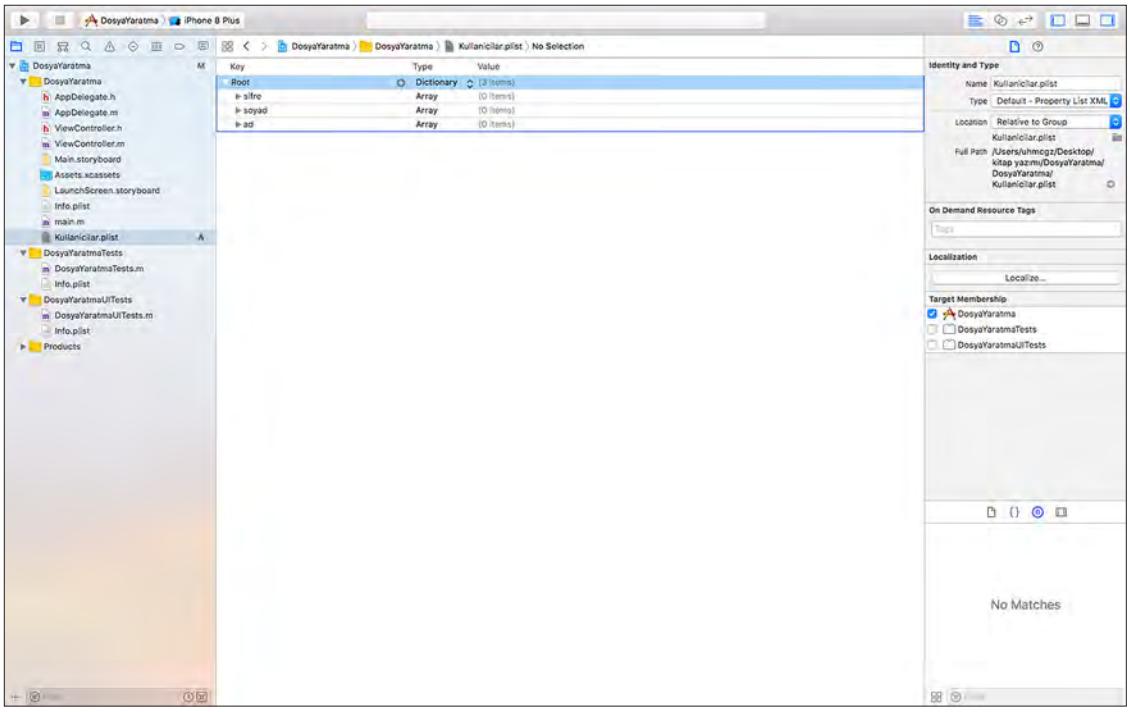
Ekran Görüntüsü 5.9

Benzer şekilde Kul2, Kul3 şeklinde Dictionary tipinde alanlar oluşturarak yeni kullanıcı kayıtları eklenebilir.

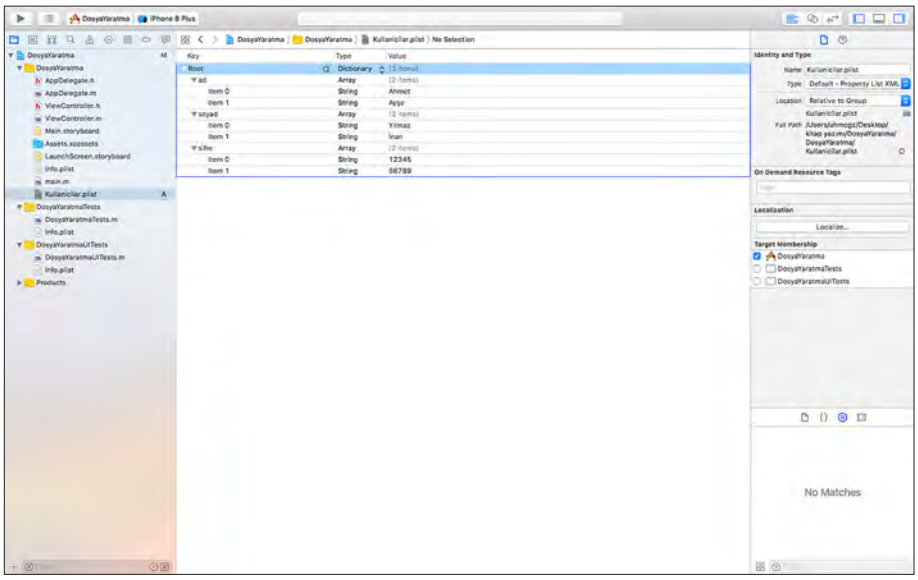


Ekran Görüntüsü 5.10

Kayıtlar istenirse dizi tipinde alanlar oluşturularak farklı şekillerde de oluşturulabilir. Örneğin aşağıdaki gibi bir yöntemle ad, soyad ve şifre alanları Array tipinde seçilerek altına kayıtlar Item 0, Item 1, Item 2 şeklinde kaydedilebilir.



Ekran Görüntüsü 5.11



Ekran Görüntüsü 5.12

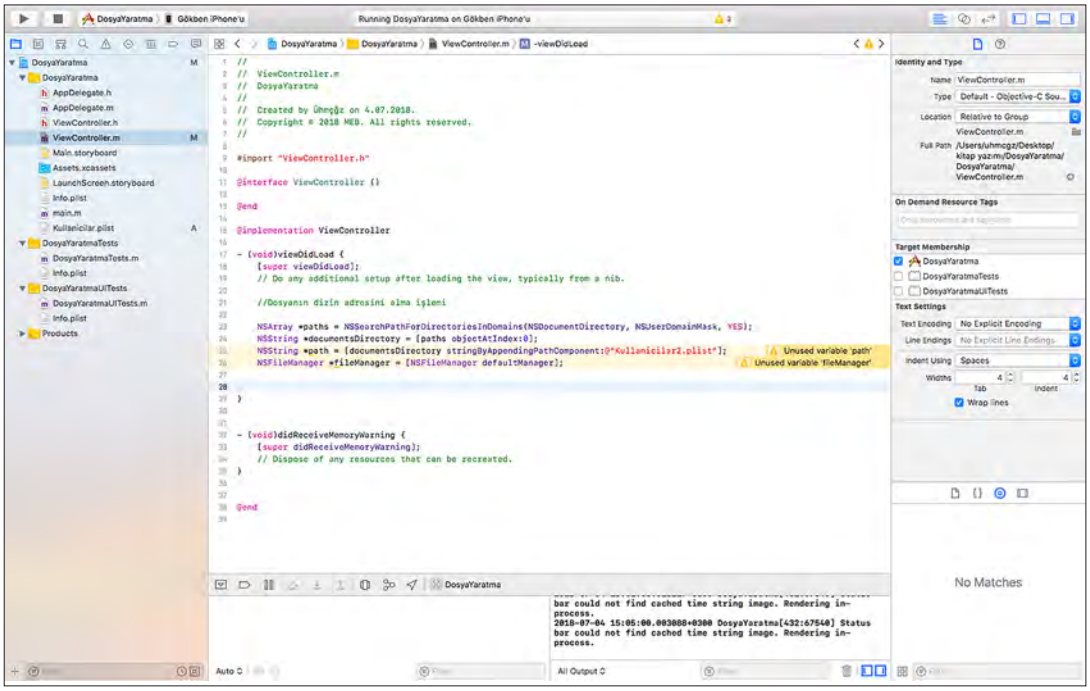
Verileri (kayıtları) dosyaya kodlama ile kaydederken veya okurken geliştiricinin belirlemiş olduğu Plist düzenine göre işlemler yapılır. Geliştiricinin ihtiyaç duyacağı dosya kayıt gereksinimine göre yukarıdaki Plist alanları değişkenlik gerektirebilir.

Kodla Dosya Oluşturma İşlemi

Aynı Xcode proje üzerinden devam edildiğinde ViewController.m dosyası açılarak ViewDidLoad içerisine Plist dosyası oluşturacak kodlar yazılır.

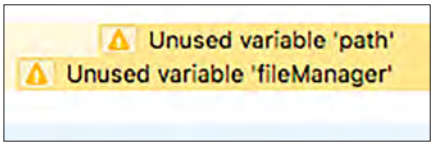
Öncelikle Plist dosyasının kaydedileceği dizinin adres yolu (paths) bir dizi değişkeni içerisine kaydedilir.

```
- (void)viewDidLoad {
[super viewDidLoad];
// Do any additional setup after loading the view, typically from a nib.

//Dosyanın dizin adresini alma işlemi

NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);

// NSArray tipinde bir paths değişkeni yaratılır ve uygulama dosyasının cihaz içindeki klasör adres satırı içerisne kaydedilir

NSString *documentsDirectory = [paths objectAtIndex:0];
// documentsDirectory isimli string tipinde yaratılan değişkenin içerisine paths dizisindeki ilk değer (sıfırıncı kayıttaki value) kaydedilir.

NSString *path = [documentsDirectory stringByAppendingPathComponent:@"Kullanicilar2.plist"];
// path isimli string tipinde yaratılan değişkenin içerisine dosyanın ismi belirtilerek dosyanın tam yolu kaydedilir.

NSFileManager *fileManager = [NSFileManager defaultManager];
// Dosya işlemlerini yapan bir değişken yaratılır
```



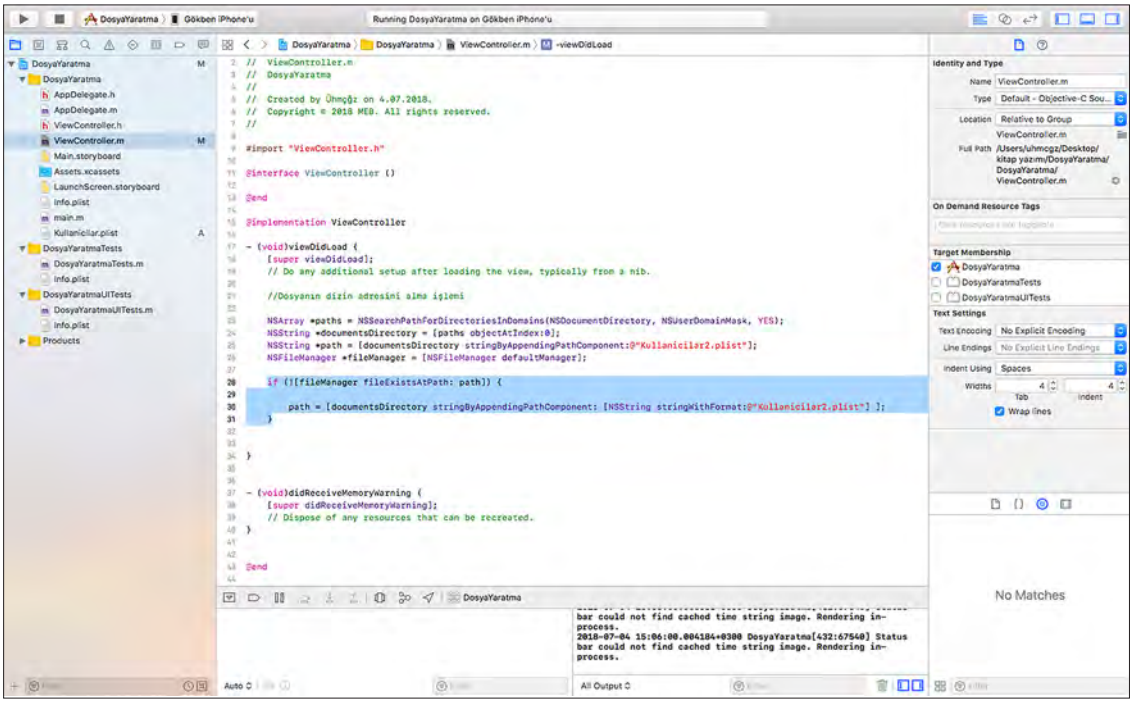
Ekran Görüntüsü 5.13

İfadesi yaratılan değişkenlerin kullanılmadığı uyarısını vermektedir. path ve fileManager değişkenleri aşağıda belirtilen kodlarda kullanıldığında bu uyarılar otomatik olarak kaybolacaktır.

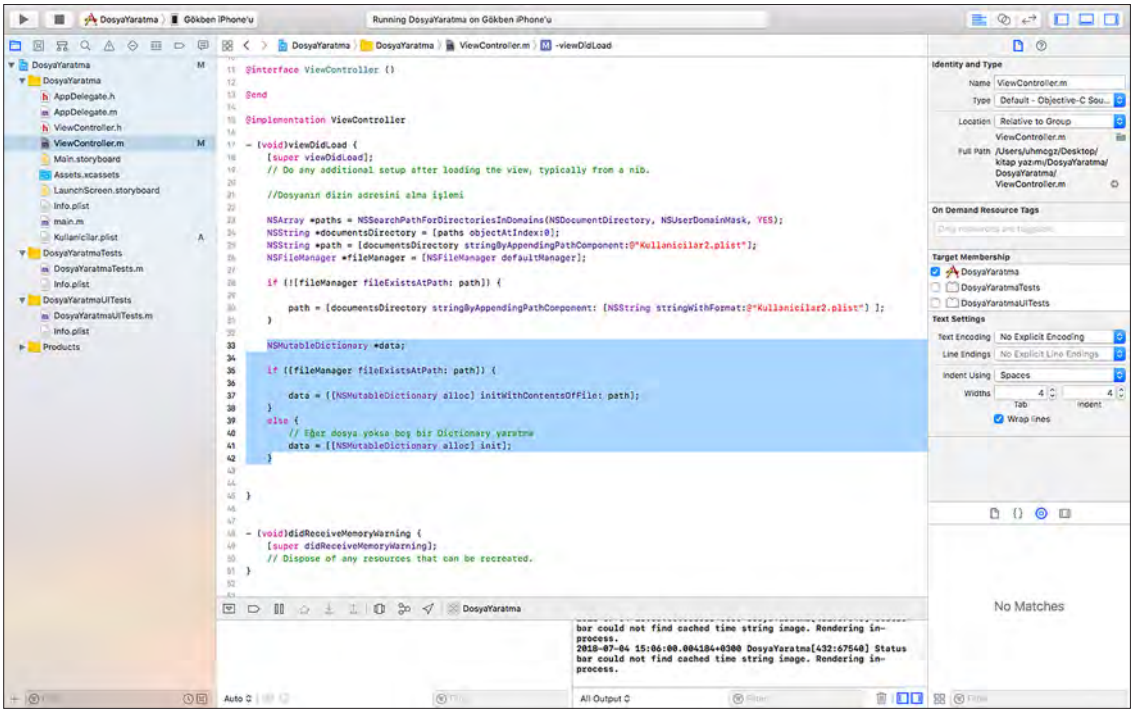


Ekran Görüntüsü 5.14

```
if (![fileManager fileExistsAtPath: path]) {
path = [documentsDirectory stringByAppendingPathComponent: [NSString stringWithFormat:@"Kullanicilar2.plist"]
];
}

NSMutableDictionary *data;  // data isimli bir Dictionary tipinde değişken oluşturulur.

if ([fileManager fileExistsAtPath: path]) {

// eğer ilgili plist dosyası varsa data plist dosyasına kaydedilecek alan olarak belirtilir.
data = [[NSMutableDictionary alloc] initWithContentsOfFile: path];
}
else {        // Eğer dosya yoksa boş bir Dictionary yaratılır

data = [[NSMutableDictionary alloc] init];
}
```



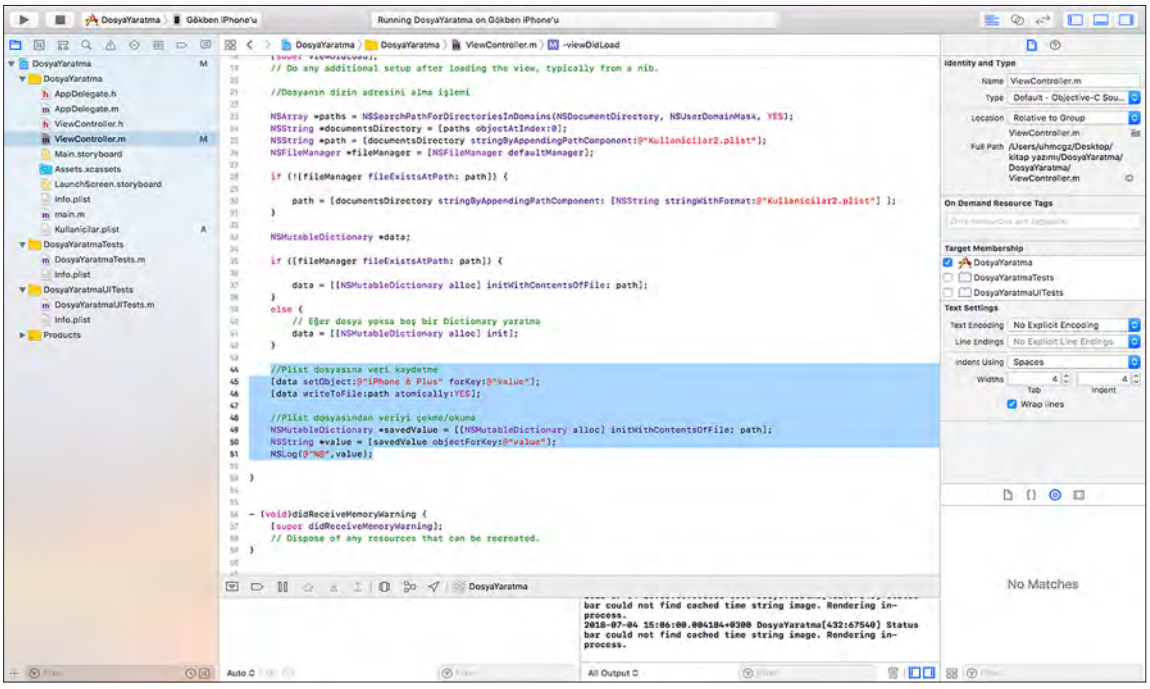
Ekran Görüntüsü 5.15



Ekran Görüntüsü 5.16

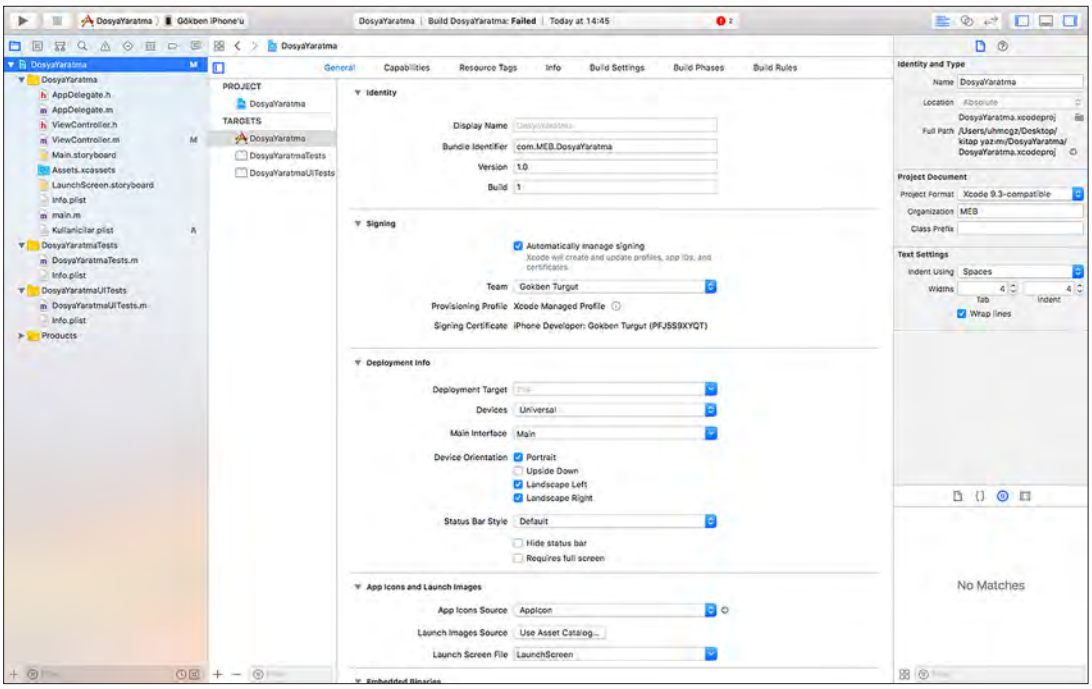
Deneme amacıyla oluşturulan Kullanicilar2.plist dosyasına bir kayıt eklemek ve okumak için aşağıdaki kodlar eklenir.

```
//Plist dosyasına veri kaydetme
[data setObject:@"iPhone 6 Plus" forKey:@"value"];
[data writeToFile:path atomically:YES];

//Plist dosyasından veriyi çekme/okuma
NSMutableDictionary *savedValue = [[NSMutableDictionary alloc] initWithContentsOfFile: path];
NSString *value = [savedValue objectForKey:@"value"];
NSLog(@"%@",value);
```



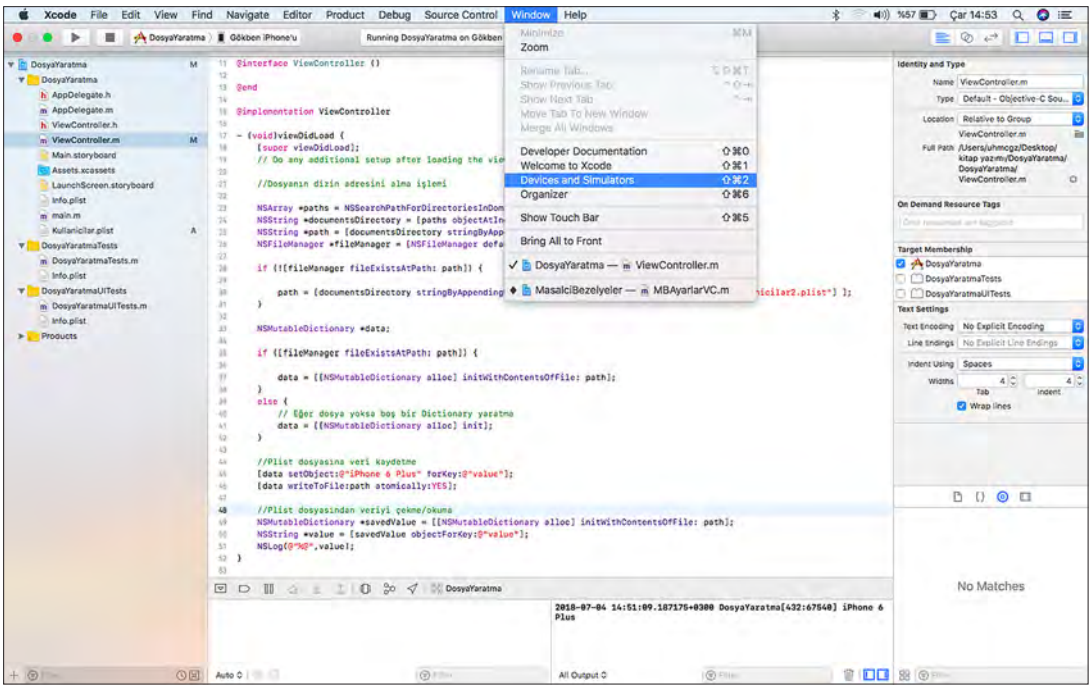
Ekran Görüntüsü 5.17

İşlemlerin yapıldığını kontrol etmek için cihaz içeriğinin görüntülenmesi gerekmektedir. Öncelikle proje Development Team girilerek bir iPhone cihazında çalıştırılır. Bunun için proje ayarlar ekranından kullanıcı seçilir.



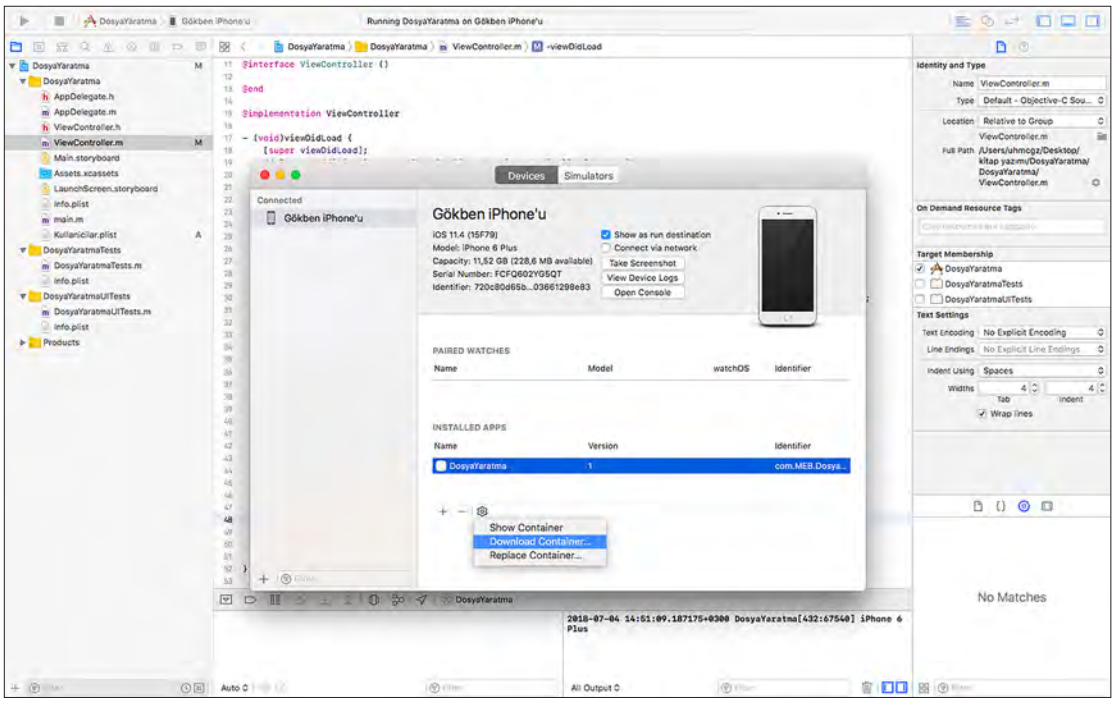
Ekran Görüntüsü 5.18

Uygulama bilgisayara bağlanan kayıtlı iPhone cihazında çalıştırılır. Ardından Xcode ekranından Window menüsü, Devices and Simulators seçeneği tıklanır.

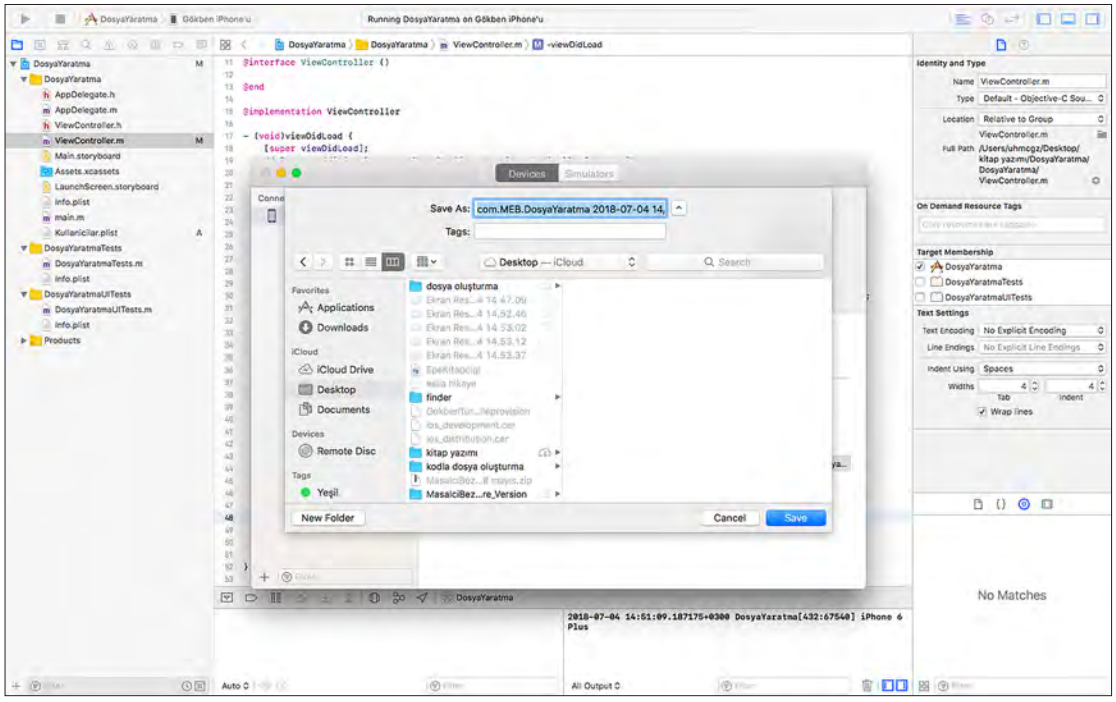


Ekran Görüntüsü 5.19

Ekrana gelen pencereden uygulama ismi tıklanarak alt kısmında bulunan ayarlar simgesine tıklanır. Download Container seçilerek bilgisayara uygulama dosyaları indirilir.



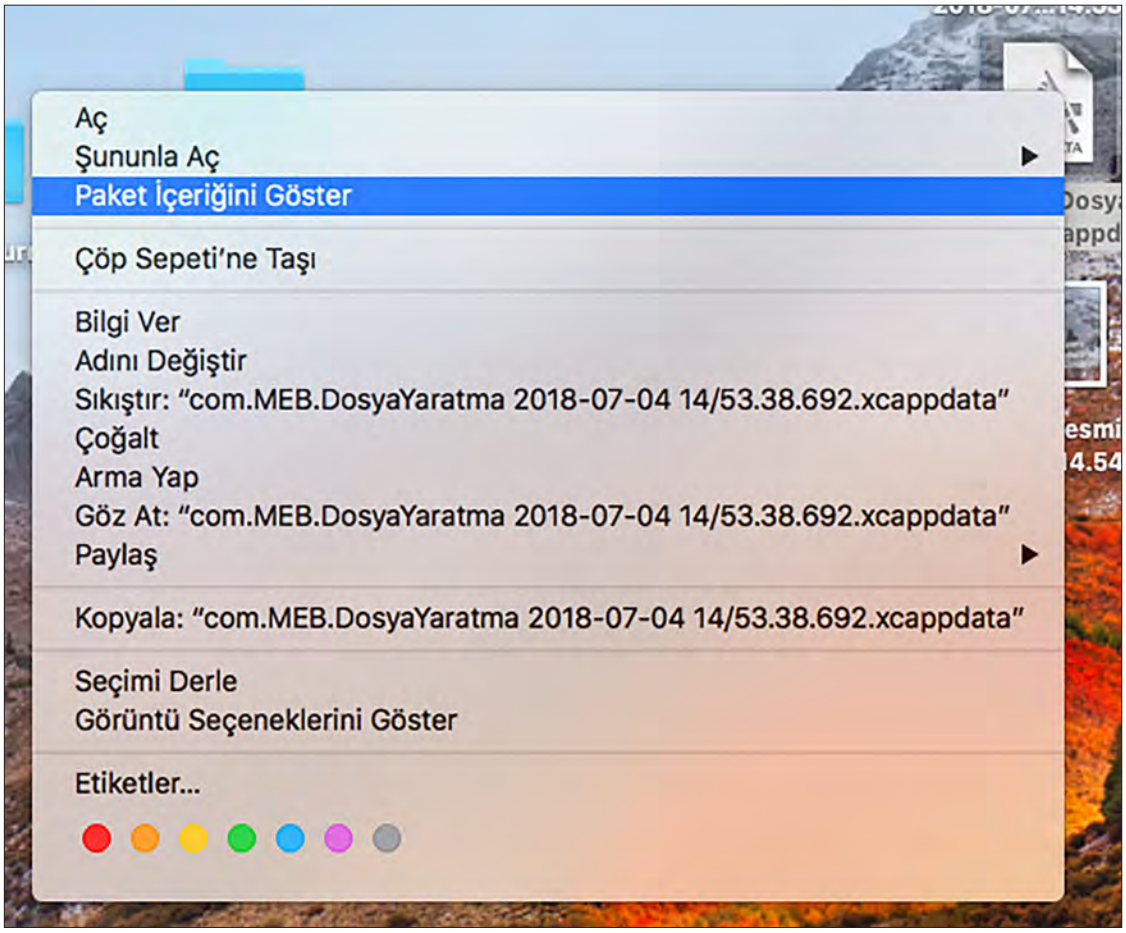
Ekran Görüntüsü 5.20



Ekran Görüntüsü 5.21

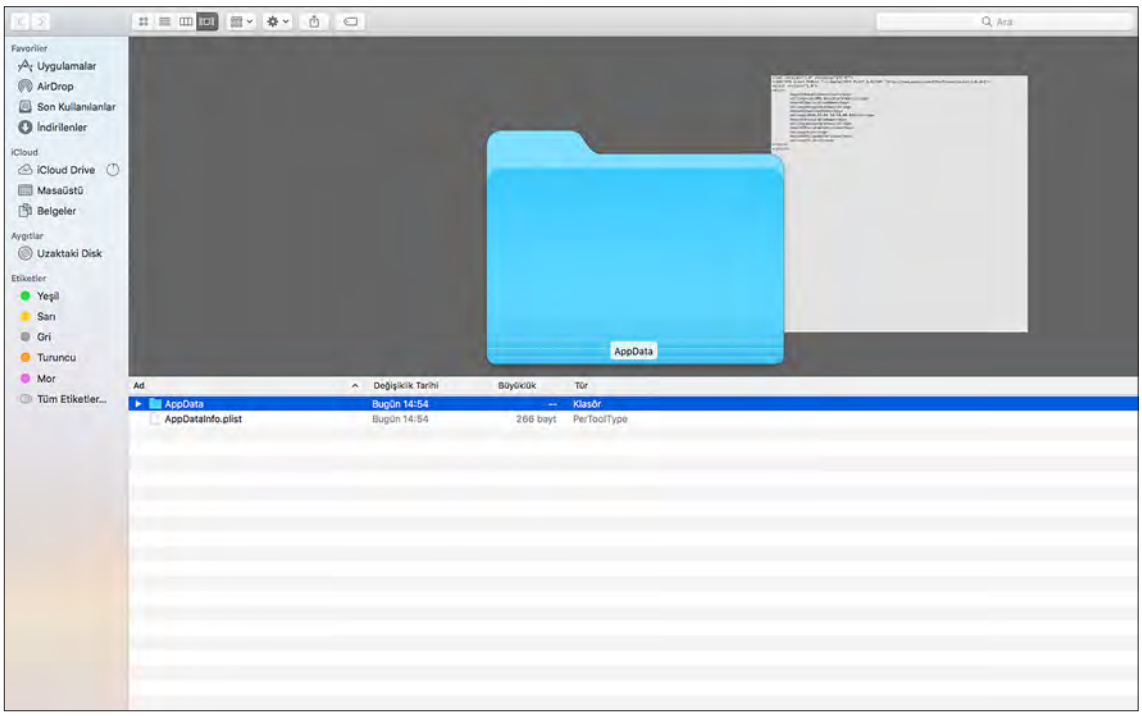
Masaüstüne gelen dosyanın üzerinde sağ tıklanarak Paket İçeriğini Göster seçilir.

Ekran Görüntüsü 5.22

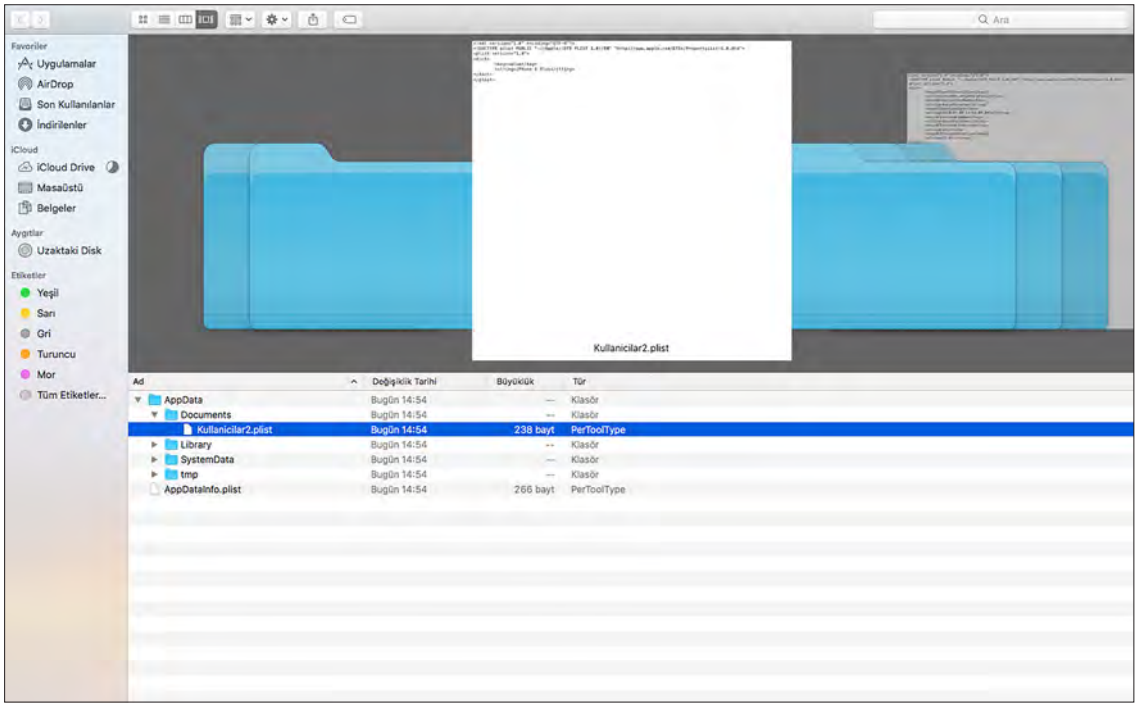


Ekran Görüntüsü 5.23

Açılan pencerede AppData klasörü açılır ve Documents klasörü de çift tıklanarak Kullanicilar2.plist dosyası görüntülenir.



Ekran Görüntüsü 5.24



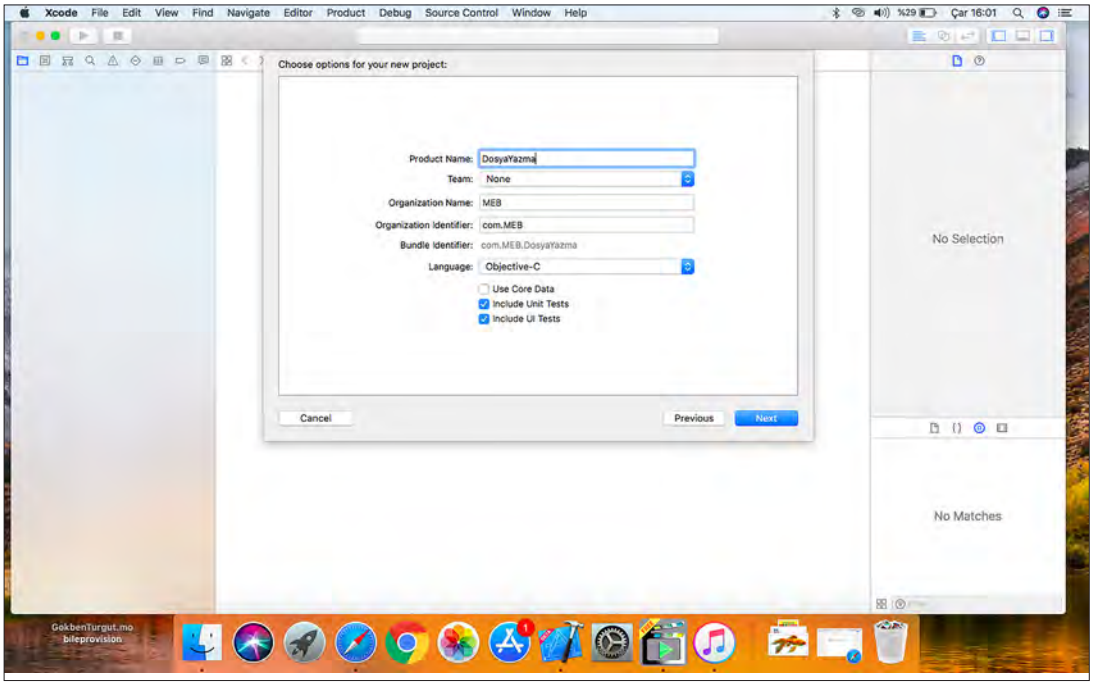
Ekran Görüntüsü 5.25

Plist dosyasının içeriği text dosyası olarak görüntülendiğinde aşağıdaki gibi olacaktır.

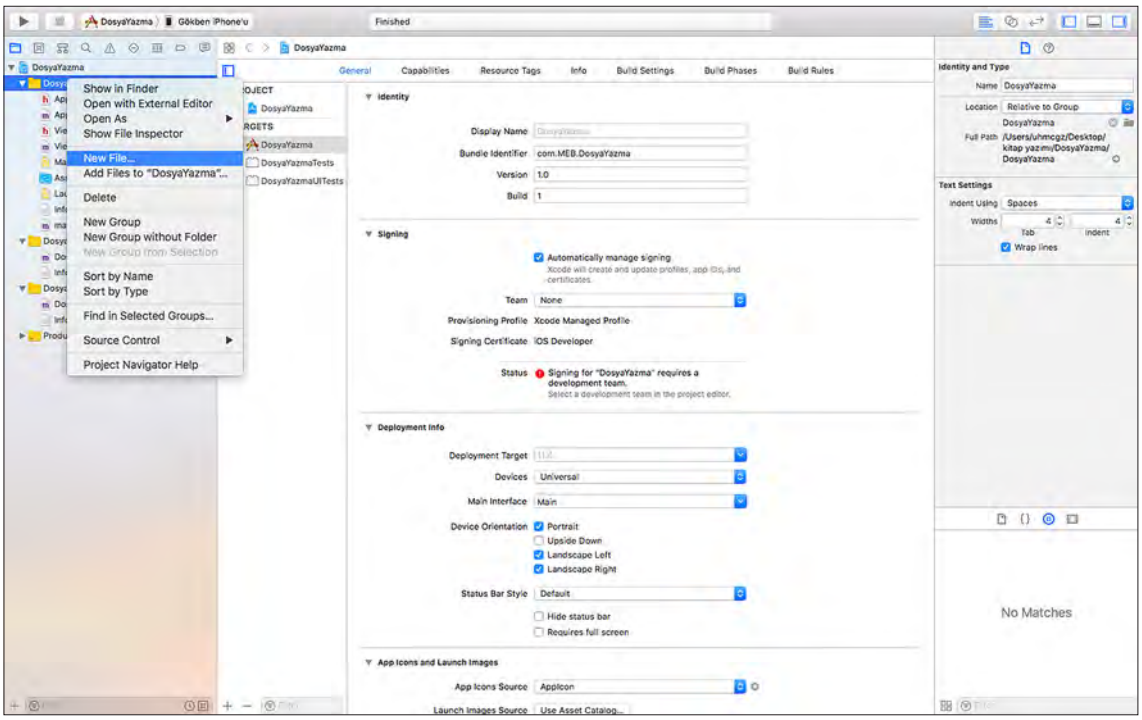
Ekran Görüntüsü 5.26

5.1.2. Dosyaya Yazma

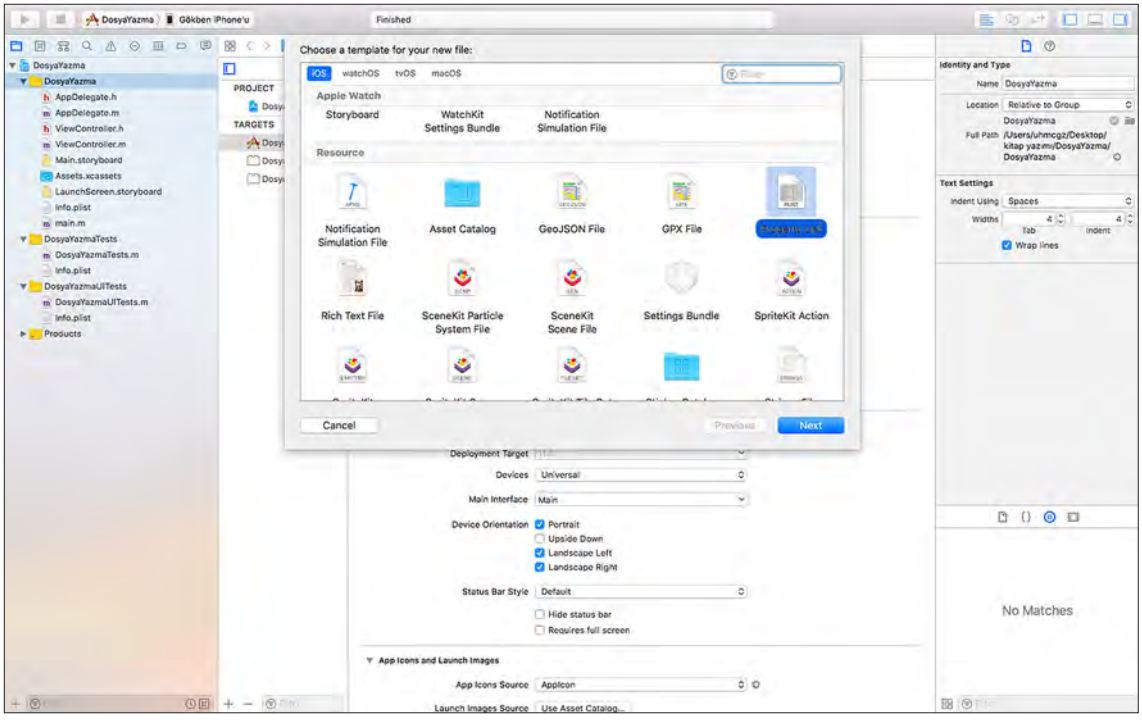
DosyaYazma isimli bir Xcode projesi oluşturulur. DosyaYaratma uygulamasında olduğu gibi bir Plist dosyası ve ilgili alanlar oluşturulur.



Ekran Görüntüsü 5.27

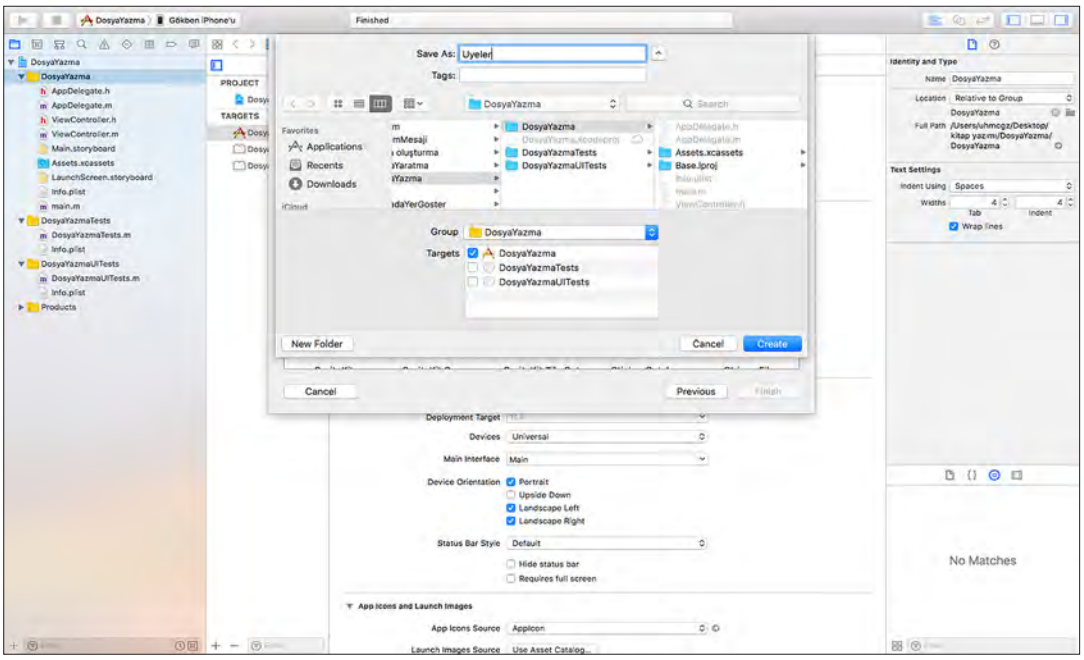


Ekran Görüntüsü 5.28



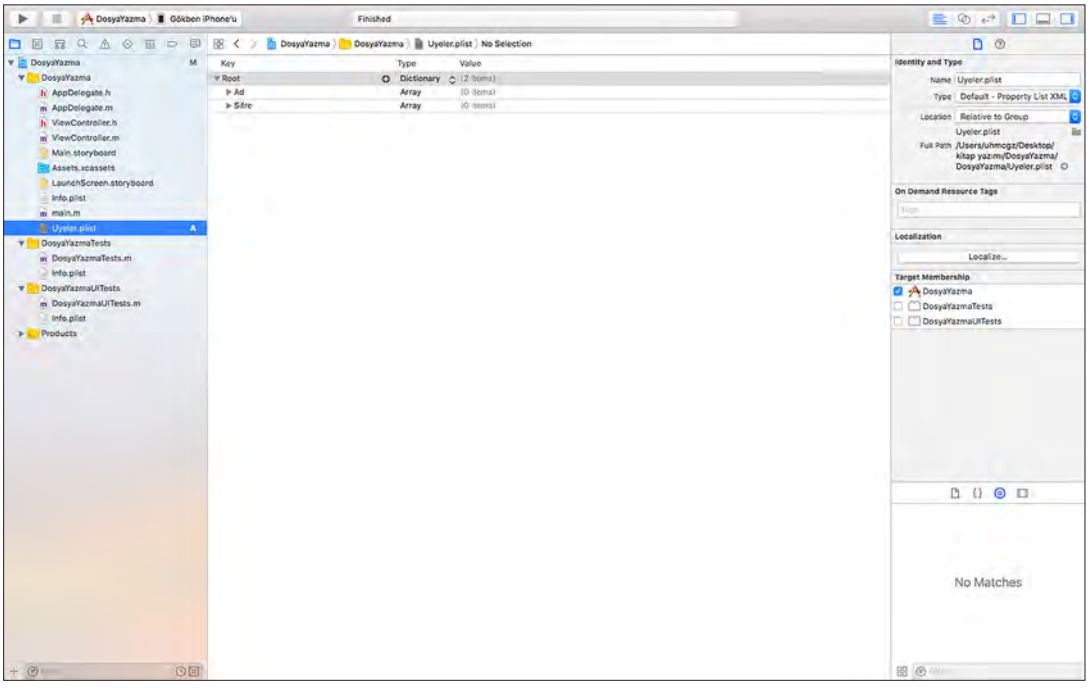
Ekran Görüntüsü 5.29

Plist dosyasına Uyeler ismi verilerek projeye kaydedilir.



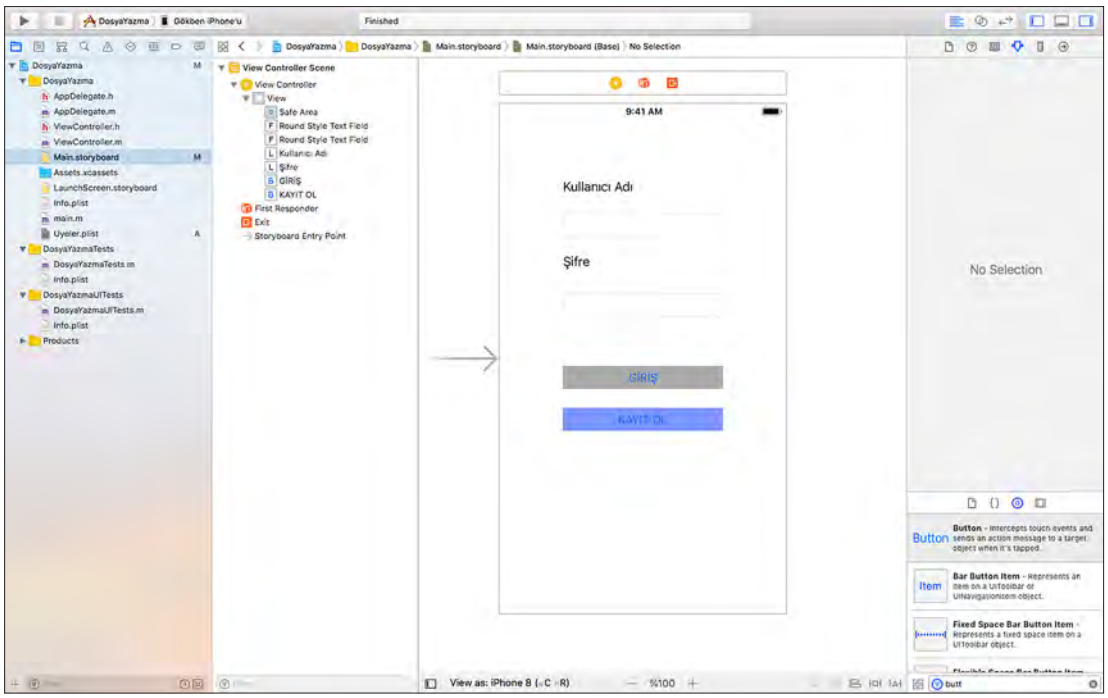
Ekran Görüntüsü 5.30

Dosyanın içerisine Ad ve Sifre isimli Array tipinde iki alan oluşturulur. Kaydedilen üyeler ad ve sifre alanları altına Item 0, 1, 2 şeklinde kaydedilecektir.



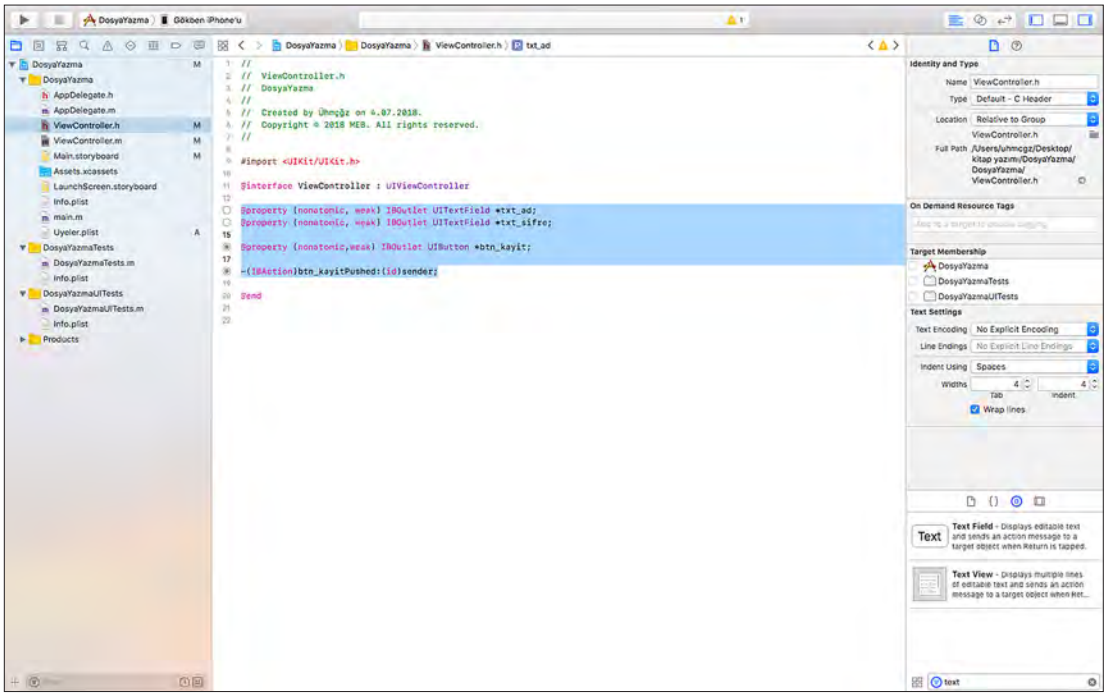
Ekran Görüntüsü 5.31

Storyboard açılarak Uye kaydı ve girişi yapılacak ekran tasarlanır. İki Label, iki TextField ve iki Button kullanılmıştır.



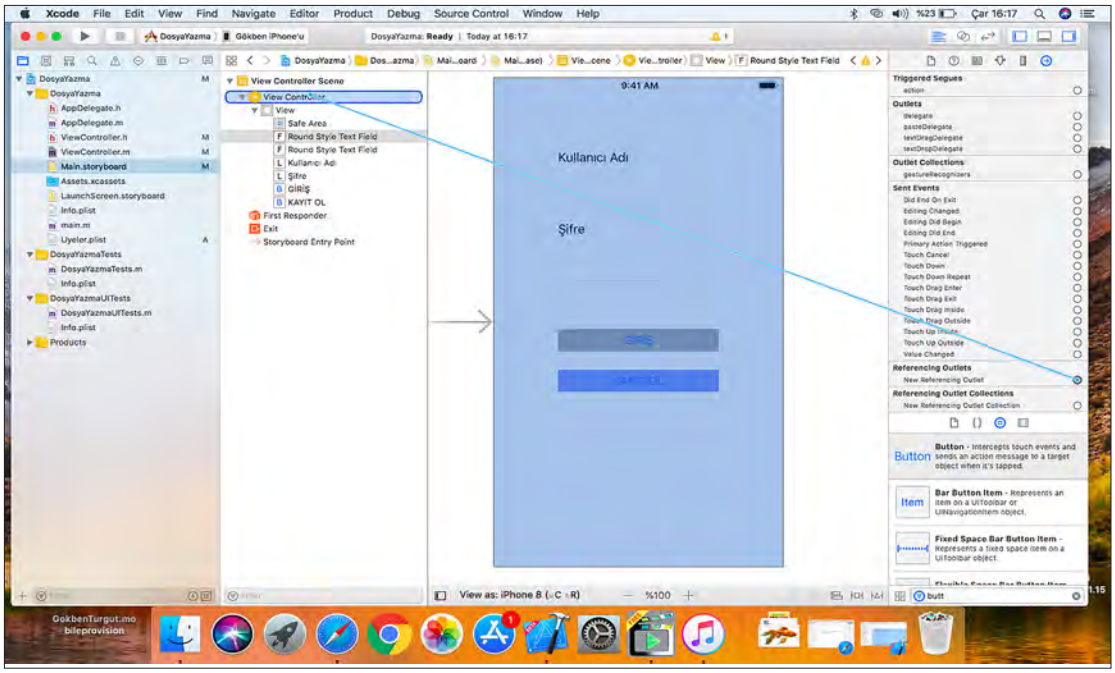
Ekran Görüntüsü 5.32

Oluşturulan nesnelerin ve düğme tıklama Action'ının ViewController.h dosyasında tanımlama işlemleri yapılır.

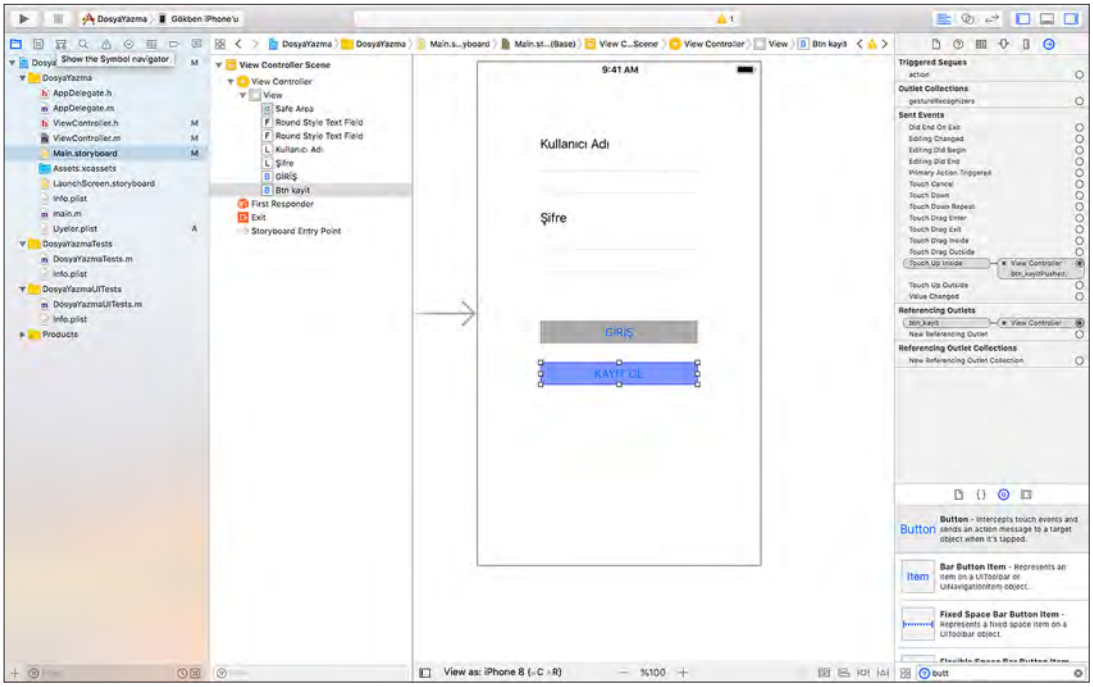


Ekran Görüntüsü 5.33

ViewController.h dosyasında yazılan action fonksiyon ismi ve nesnelerin isimleri Main.storyboard dosyasında ilgili nesnelerle ilişkilendirilir.



Ekran Görüntüsü 5.34



Ekran Görüntüsü 5.35

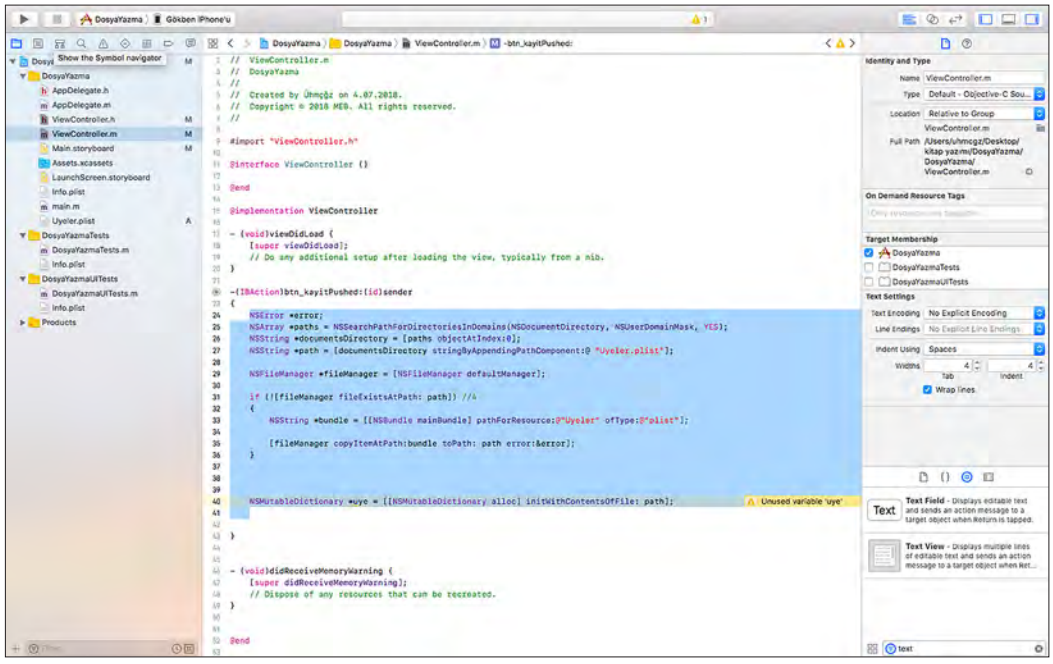
Toplam 4 ilişki kurma işlemi sırasıyla yapılır. İki adet TextField için birer işlem, Kayıt Ol düğmesi için de iki işlem gerçekleştirilir.

Ardından ViewController.m dosyası açılarak Kayıt Ol düğmesi tıklandığında yapılacak işlemler aşağıdaki Action fonksiyonu içerisine yazılır.

```
-(IBAction)btn_kayitPushed:(id)sender
{


}
```

Aşağıdaki kodlar bir önceki DosyaYaratma uygulamasında yazdığımız kodların benzeridir ve sadece plist dosyasının ismi değiştirilmiştir.

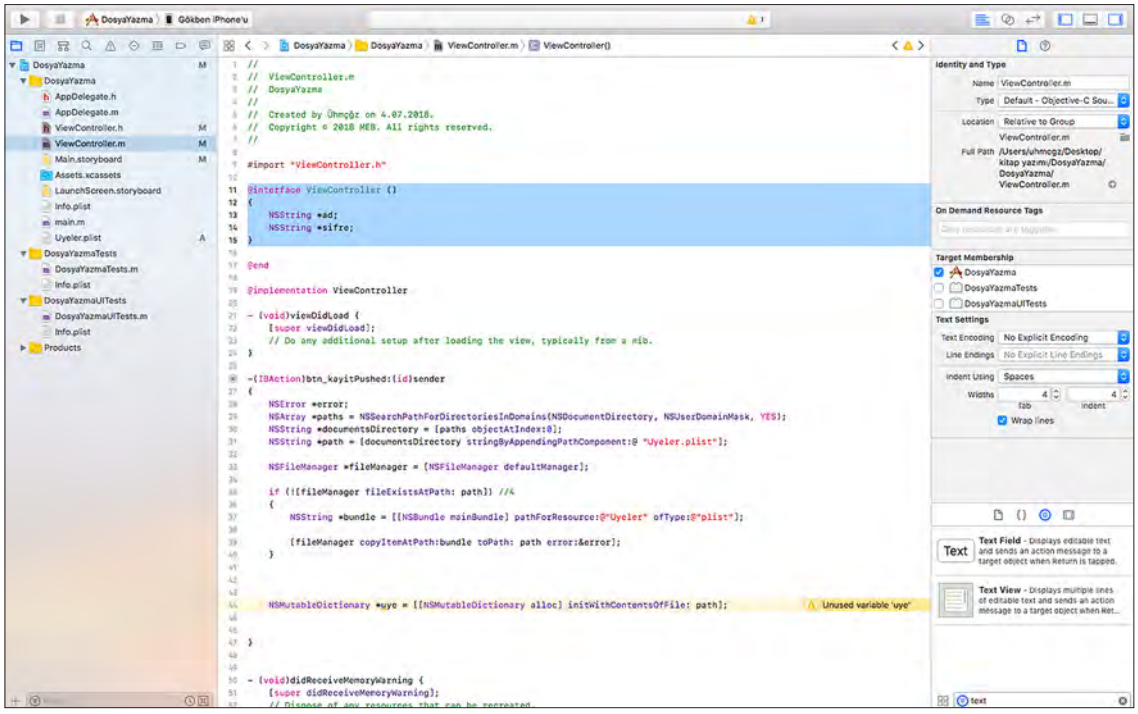
```
NSError *error;
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
NSString *documentsDirectory = [paths objectAtIndex:0];
NSString *path = [documentsDirectory stringByAppendingPathComponent:@ "Uyeler.plist"];

NSFileManager *fileManager = [NSFileManager defaultManager];

if (![fileManager fileExistsAtPath: path]){
NSString *bundle = [[NSBundle mainBundle] pathForResource:@"Uyeler" ofType:@"plist"];
[fileManager copyItemAtPath:bundle toPath: path error:&error];
}

NSMutableDictionary *uye = [[NSMutableDictionary alloc] initWithContentsOfFile: path];
```



Ekran Görüntüsü 5.36

Uygulama çalıştırılarak kullanıcı ilgili alanlarına bilgilerini girdikten sonra KAYIT OL düğmesine tıklayacak. Bundan dolayı, TextField'lardaki kullanıcıdan alınan verilerin String tipinde bir değişkene kaydedilmesi gerekmektedir. Bu sebeple dosyanın üst kısmında NSString tipinde ad ve sifre isimli iki değişken yaratılır.

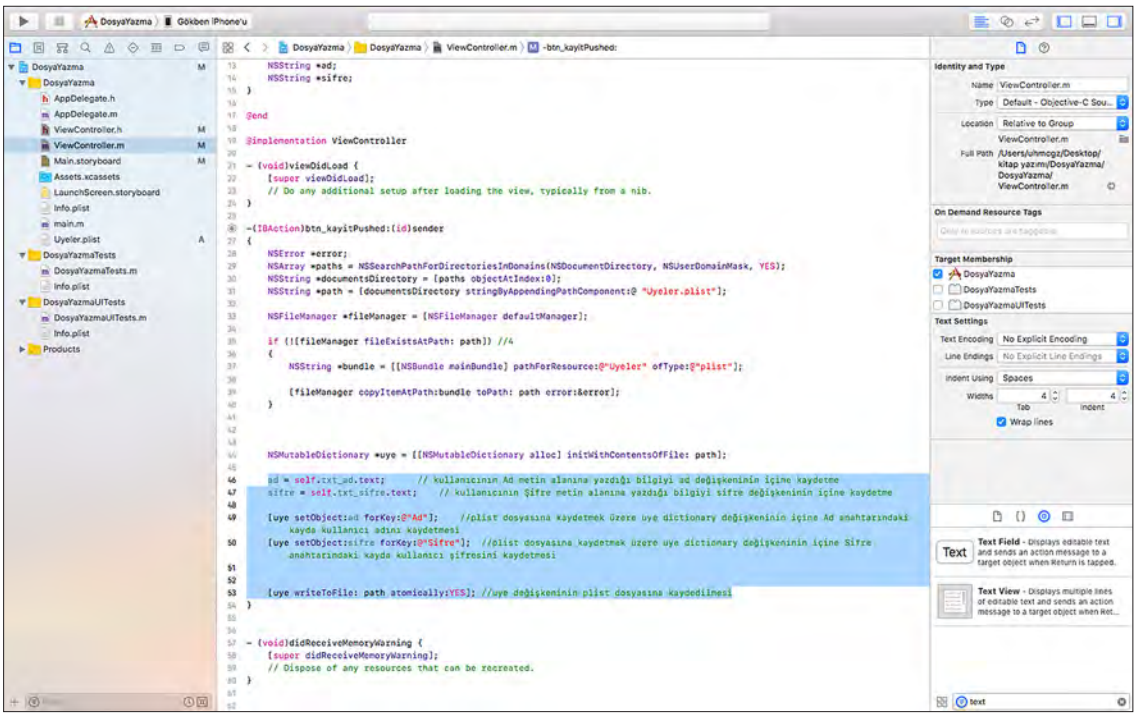
```
@interface ViewController ()
{
NSString *ad;
NSString *sifre;
}
```



Ekran Görüntüsü 5.37

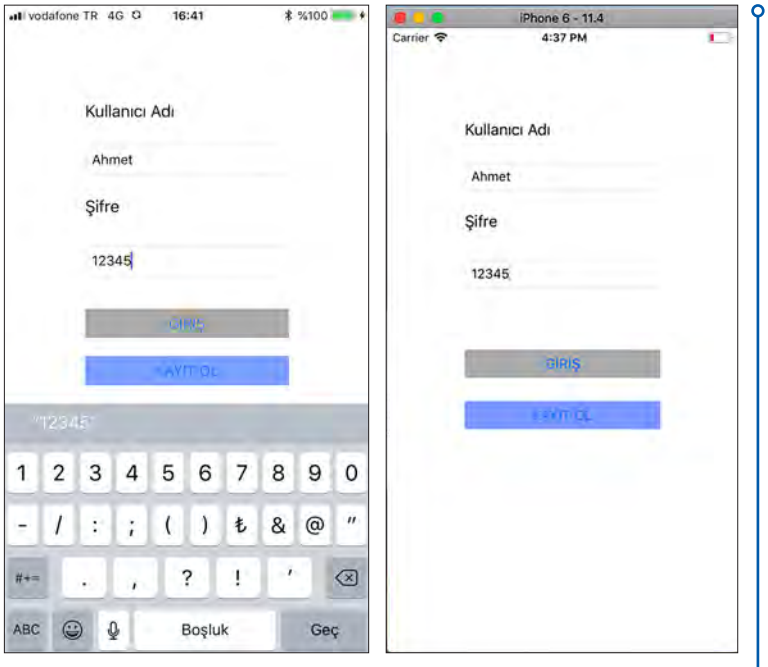
Değişkenlerin içerisine TextField'lardan gelen veriler kaydedilerek aşağıdaki kod dizini ile plist dosyasına üye kaydı gerçekleştirilir.

```
ad = self.txt_ad.text;      // kullanıcının Ad metin alanına yazdığı bilgiyi ad değişkeninin içine kaydetme

sifre = self.txt_sifre.text;    // kullanıcının Şifre metin alanına yazdığı bilgiyi sifre değişkeninin içine kaydetme

[uye setObject:ad forKey:@"Ad"];    //plist dosyasına kaydetmek üzere uye dictionary değişkeninin içine Ad anahtarındaki kayda kullanıcı adını kaydetmesi

[uye setObject:sifre forKey:@"Sifre"];  //plist dosyasına kaydetmek üzere uye dictionary değişkeninin içine Sifre anahtarındaki kayda kullanıcı şifresini kaydetmesi

[uye writeToFile: path atomically:YES];   //uye değişkeninin plist dosyasına kaydedilmesi
```



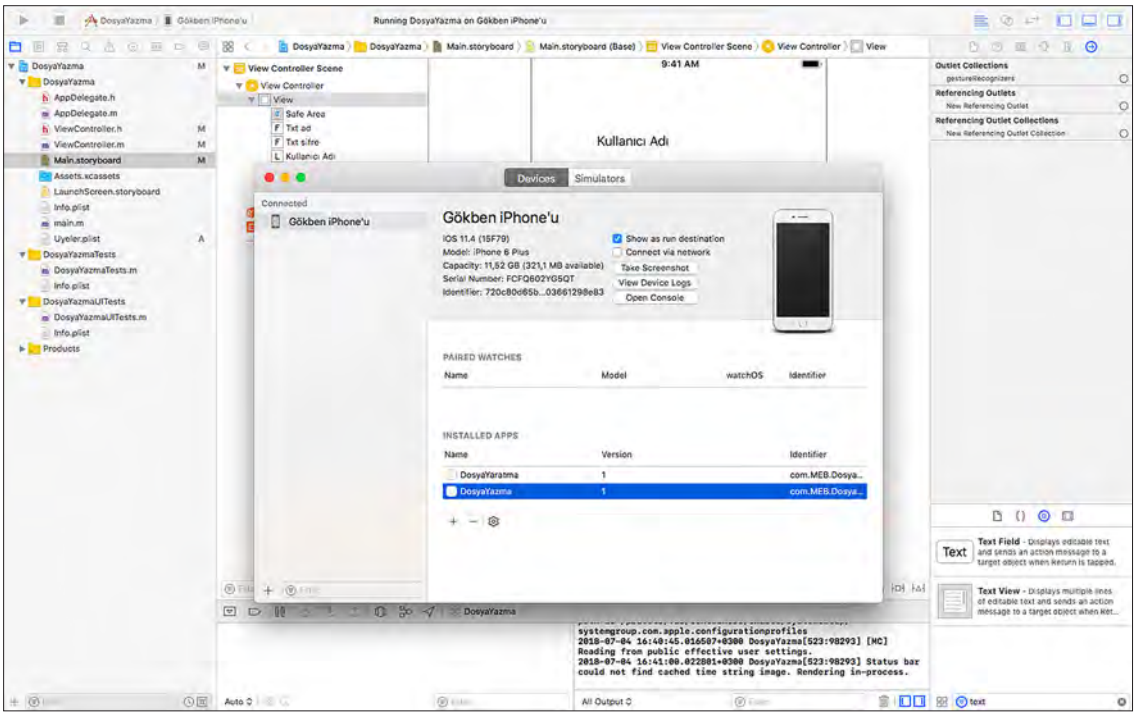
Ekran Görüntüsü 5.38

Uygulama çalıştırıldığında gerçekleştirilen veri/üye kayıtlarının görülebilmesi için iPhone cihazında çalıştırılması gerekmektedir. Bunun için projenin ayarlar bölümünden Team seçilerek, uygulama bilgisayara bağlı, kayıtlı bir cihazda çalıştırılır. Ekran görüntüsü cihaz ekranında ve simülatörde aşağıdaki gibi olacaktır.



Ekran Görüntüsü 5.39

iPhone cihazında uygulama çalıştırıldığında ve ad, şifre bilgileri girildikten sonra KAYIT OL düğmesine tıklanır. Ardından Xcode ekranında Windows menüsünden Devices and Simulators seçeneği açılır. Açılan pencerede uygulama ismi tıklanarak ayarlar simgesinden Download Container açılır. Ekrana gelen kayıt penceresinde sol taraftan masaüstü seçilerek Save düğmesi tıklanır. Masaüstünde oluşan appdata dosyasının üzerinde sağ tıklanarak içeriği görüntülenir.



Ekran Görüntüsü 5.40

Uyeler.plist dosyası açılarak kullanıcı olarak yapılan kayıt olma işlemi, metin dosyasında aşağıdaki gibi görüntülenir.

Uyeler.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Ad</key>
        <string>Ahmet</string>
        <key>Sifre</key>
        <string>12345</string>
</dict>
</plist>
```

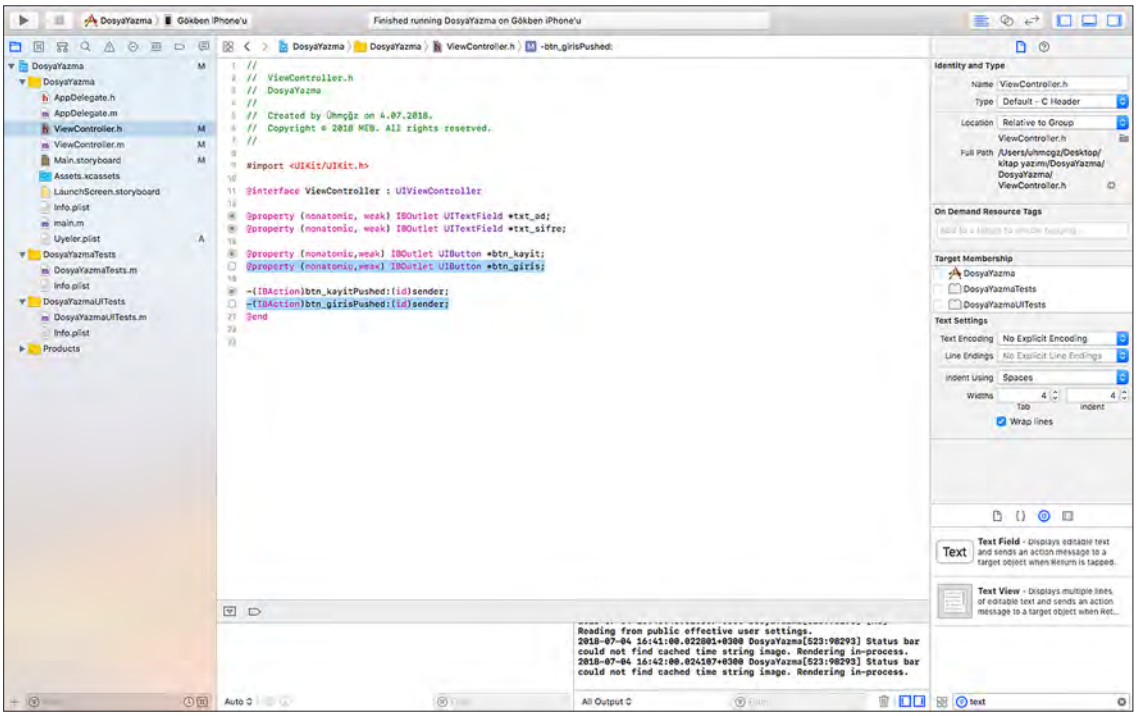
Ekran Görüntüsü 5.41

5.1.3. Dosya Okuma

DosyaYazma Xcode projesinden devam edilerek, kullanıcı TextField'lara üye bilgilerini girdiğinde, eğer Uyeler.plist dosyasında kayıtlı ilgili üye varsa ve şifre uyumlu ise kullanıcı ekranında Hoşgeldiniz Uye İsmi yazdırılacaktır.

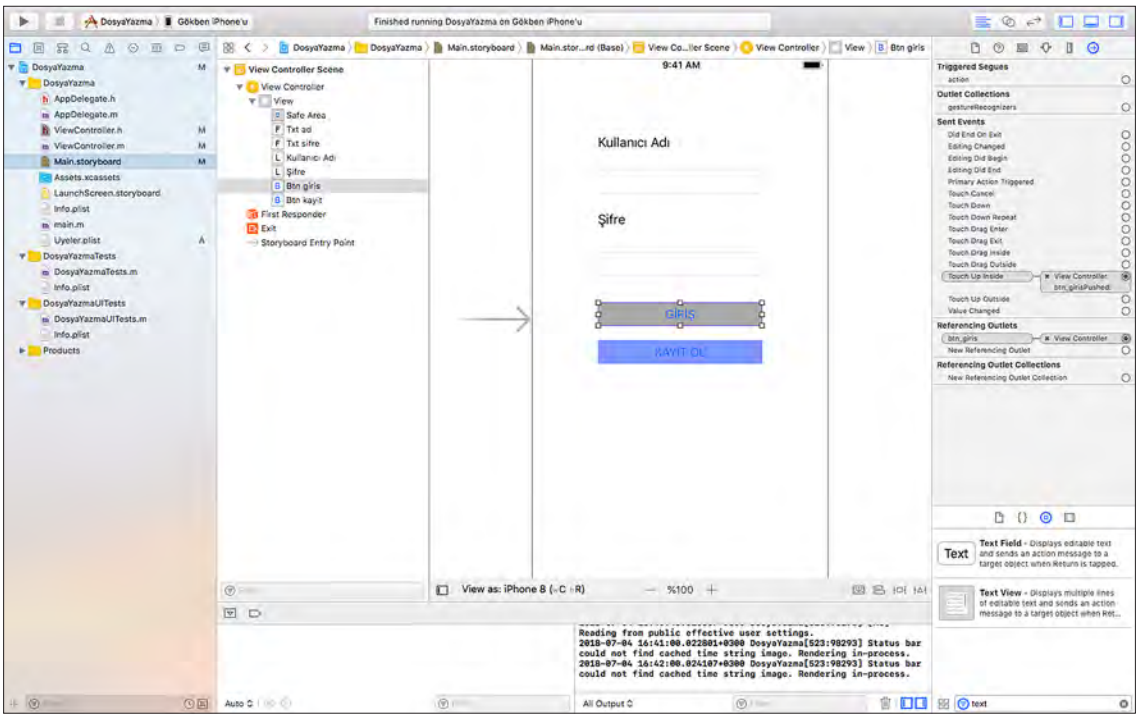
Öncelikle GİRİŞ düğmesi için ViewController.h dosyasında tanımlama yapılır.

```
@property (nonatomic,weak) IBOutlet UIButton *btn_giris;
-(IBAction)btn_girisPushed:(id)sender;
```

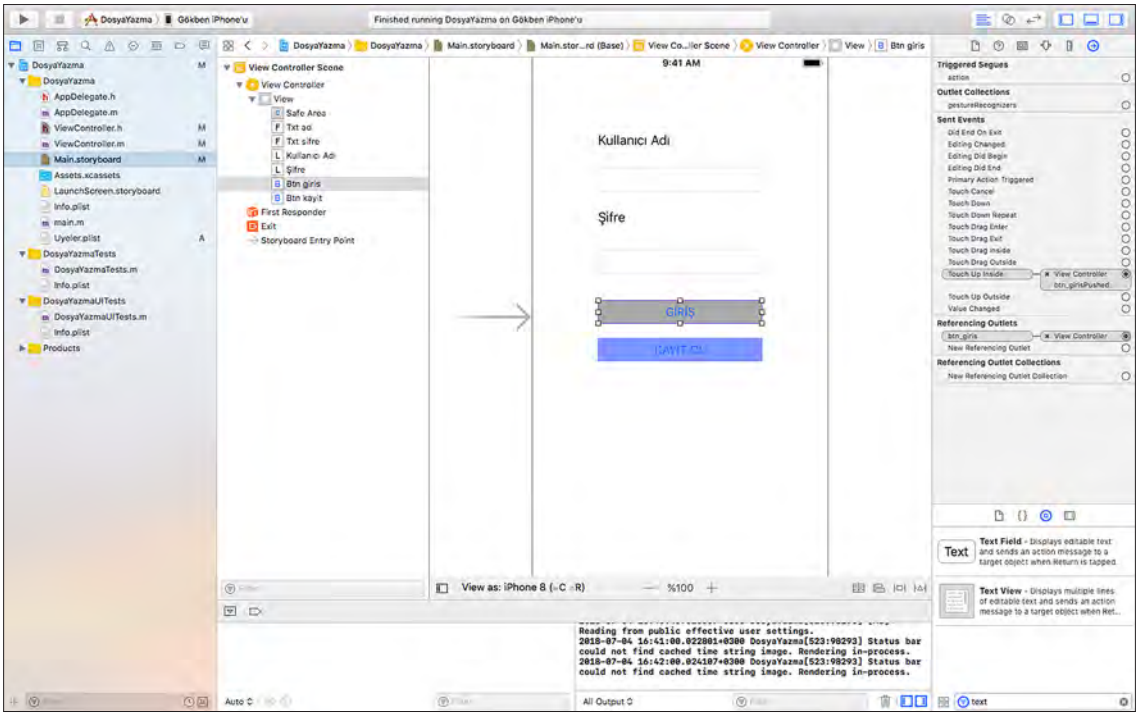


Ekran Görüntüsü 5.42

Main.storyboard ekranında GİRİŞ düğmesinin ilişkilendirmeleri yapılır.



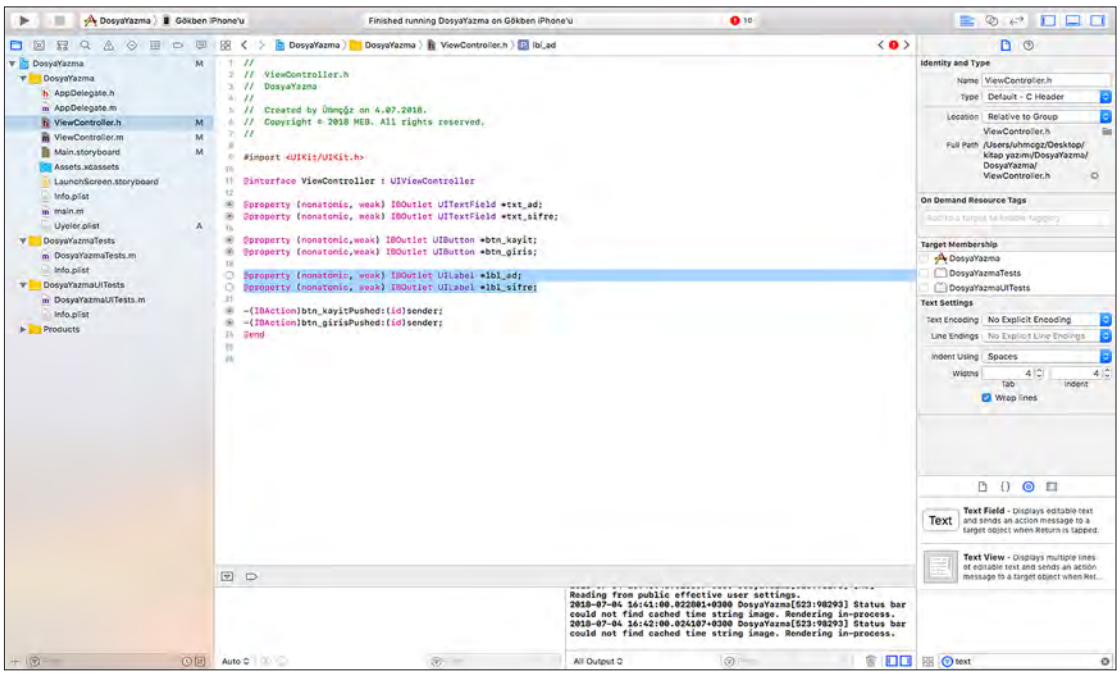
Ekran Görüntüsü 5.43



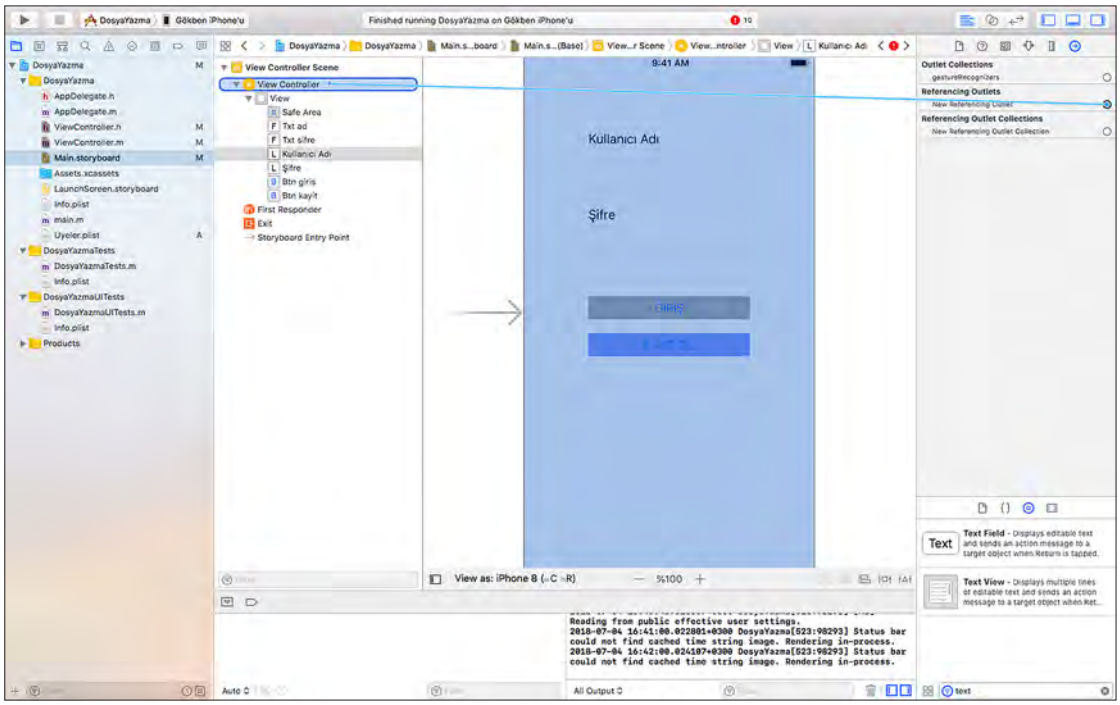
Ekran Görüntüsü 5.44

Kullanıcı girişi gerçekleştiğinde ekrandaki Label'lar ile ilgili işlemler yapılabilmesi için ViewController.h dosyasında tanımlamalar yapılır. Main.stroyboard'da da nesneler ve isimleri arasında ilişkilendirmeler gerçekleştirilir.

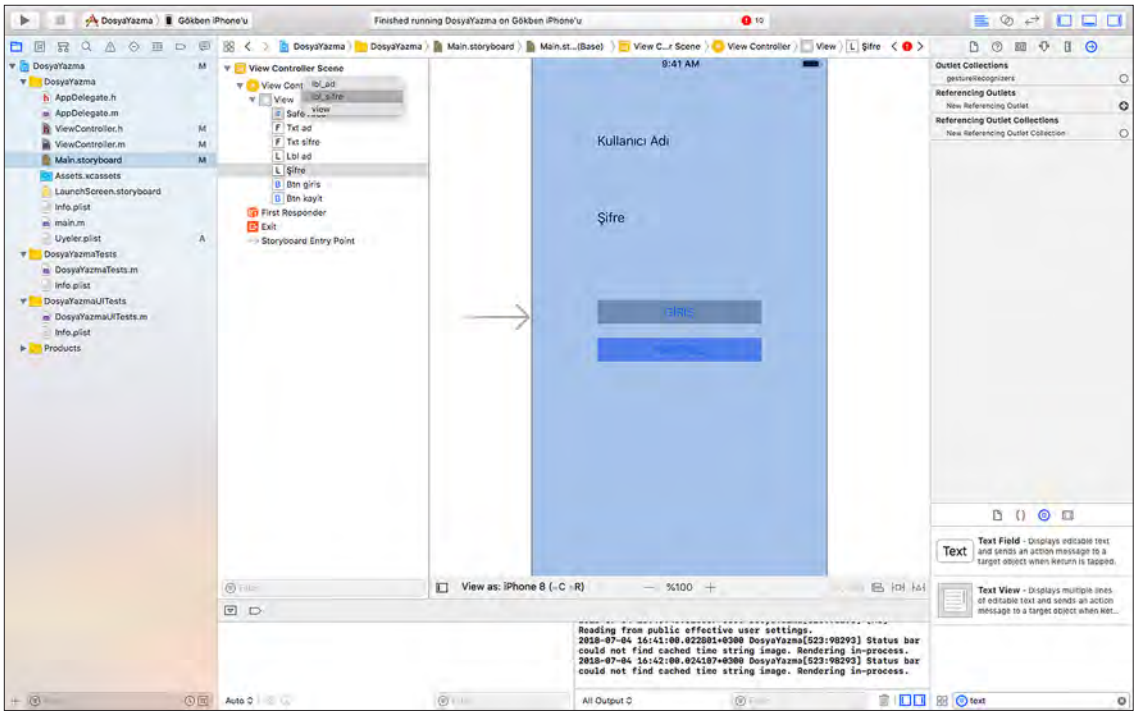
```
@property (nonatomic, weak) IBOutlet UILabel *lbl_ad;
@property (nonatomic, weak) IBOutlet UILabel *lbl_sifre;
```



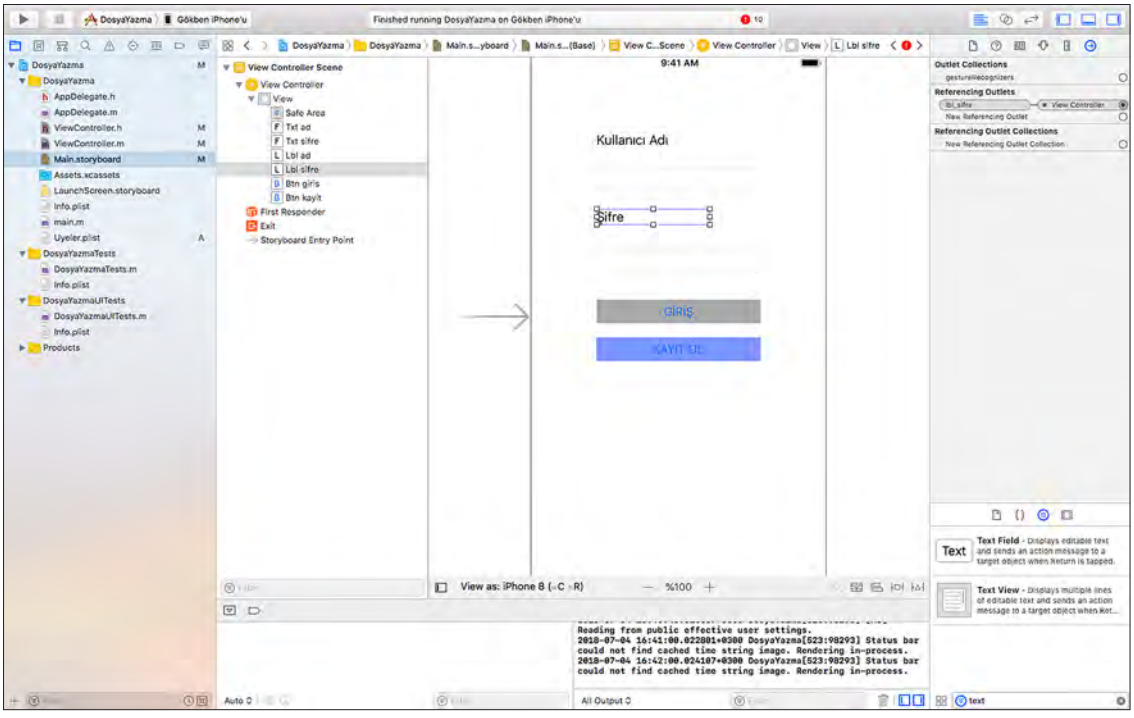
Ekran Görüntüsü 5.45



Ekran Görüntüsü 5.46



Ekran Görüntüsü 5.47



Ekran Görüntüsü 5.48

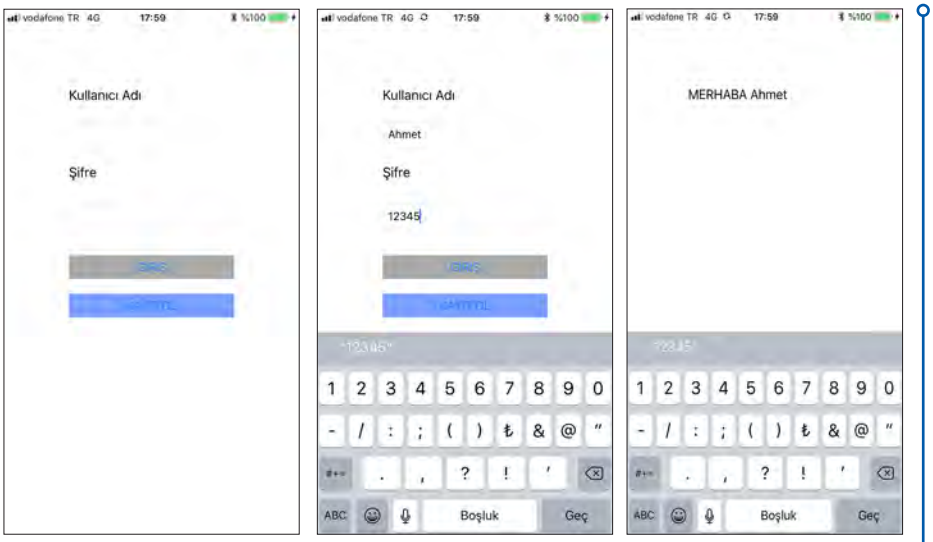
Ardından ViewController.m dosyasında GİRİŞ düğmesi tıklandığında gerçekleşecek işlemleri belirten Action fonksiyonu yazılır.

```
-(IBAction)btn_girisPushed:(id)sender
{


}
```

Action fonksiyonunun içerisine de aşağıdaki kodlar eklenir.

```
NSString *girisAd = self.txt_ad.text;      //girisAd değişkeni yaratılarak içerisine TextField'dan gelen ad bilgisinin kaydedilmesi
NSString *girisSifre = self.txt_sifre.text; //girisSifre değişkeni yaratılarak içerisine TextField'dan gelen şifre bilgisinin kaydedilmesi
NSError *error;
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
NSString *documentsDirectory = [paths objectAtIndex:0];
NSString *path = [documentsDirectory stringByAppendingPathComponent:@ "Uyeler.plist"];

NSFileManager *fileManager = [NSFileManager defaultManager];

if (![fileManager fileExistsAtPath: path]){
NSString *bundle = [[NSBundle mainBundle] pathForResource:@"Uyeler" ofType:@"plist"];
[fileManager copyItemAtPath:bundle toPath: path error:&error];
}
// kayitliKullanicilar isimli Dictionary tipinde bir değişken oluşturma ve içeriğine plist dosyasındaki verileri kaydetme
NSMutableDictionary *kayitliKullanicilar = [[NSMutableDictionary alloc] initWithContentsOfFile:path];
ad = [kayitliKullanicilar objectForKey:@"Ad"];
sifre = [kayitliKullanicilar objectForKey:@"Sifre"];

if ([ad isEqual:girisAd]) { // Eğer kullanıcıdan alınan ad bilgisi kayıtlı ad bilgileri ile uyuşuyorsa
if ([sifre isEqual:girisSifre]){
self.txt_ad.alpha = 0; //ekranda TextField nesnelerini görünmez hale getirme
self.txt_sifre.alpha = 0;
self.lbl_ad.text = [NSString stringWithFormat:@"MERHABA %@",ad];  // ilk Label'ın içeriğini değiştirme
self.btn_giris.alpha = 0;  //ekrandaki butonları, şifre Label'ını ve TextField'ları görünmez hale getirme
self.btn_kayit.alpha = 0;
self.lbl_sifre.alpha = 0;
self.txt_sifre.alpha = 0;
self.txt_ad.alpha = 0;
}
}else{      // Eğer kayıtlı bir kullanıcı yoksa uyarı penceresi oluşturma

UIAlertView *alert = [[UIAlertView alloc] initWithTitle: @"Yanlış Kullanıcı"
                                        message: @"Bu kullanıcıya ait bir kayıt bulunmamaktadır!"
                                       delegate: nil
                              cancelButtonTitle:@"OK"
                              otherButtonTitles:nil];
[alert show];
}
```

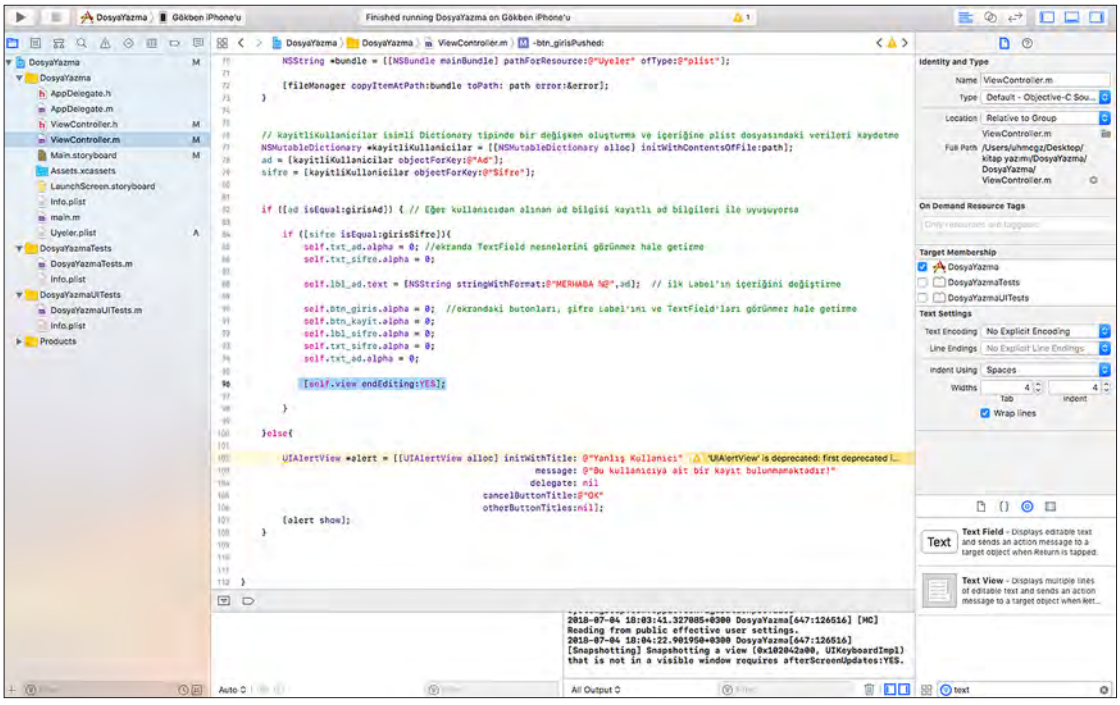
Uygulama cihazda çalıştırıldığında ekran görüntüsü aşağıdaki gibi olacaktır.



Ekran Görüntüsü 5.49

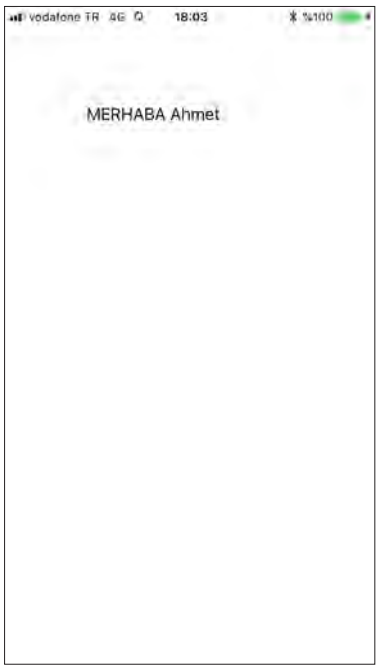
Kullanıcı girişi başarılı bir şekilde yapıldığında ekran klavyesinin kaybolması için ilgili yere aşağıdaki kod eklenir:

```
[self.view endEditing:YES];
```

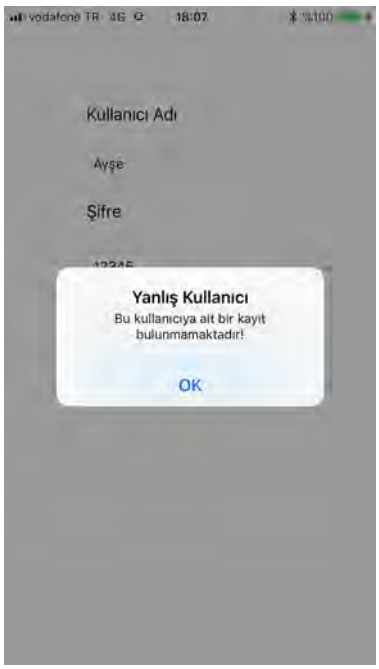


Ekran Görüntüsü 5.50

Bu işlemden sonra ekran görüntüsü aşağıdaki gibi olacaktır. Eğer yanlış bir kullanıcı girişi işlemi yapılırsa kullanıcıya uyarı penceresi açılacaktır



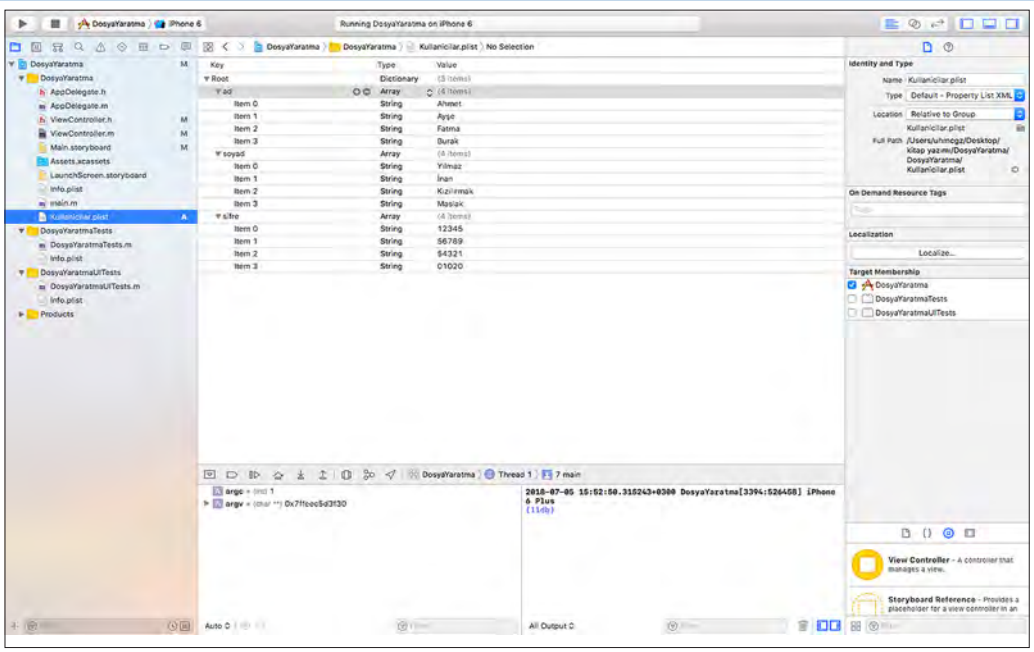
Ekran Görüntüsü 5.51



Ekran Görüntüsü 5.52

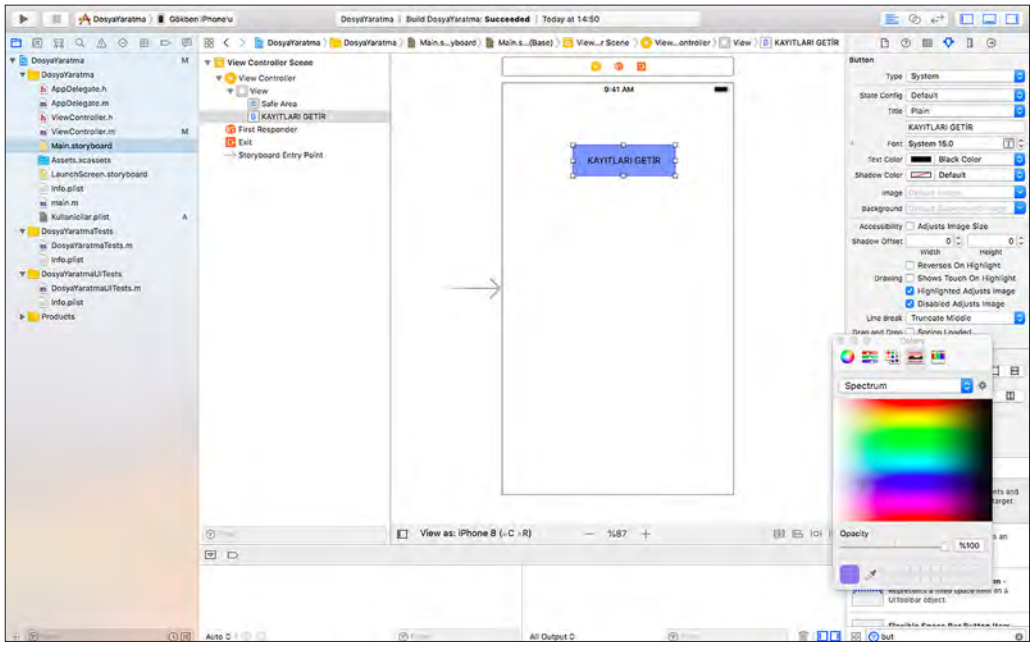
5.1.4. Dosyayı Ekrana Yazdırma

Bu uygulama için DosyaYaratma projesinden devam edilecektir. DosyaYaratma projesi açılarak Kullanicilar.plist dosyasında aşağıdaki kayıtlar girilir.



Ekran Görüntüsü 5.53

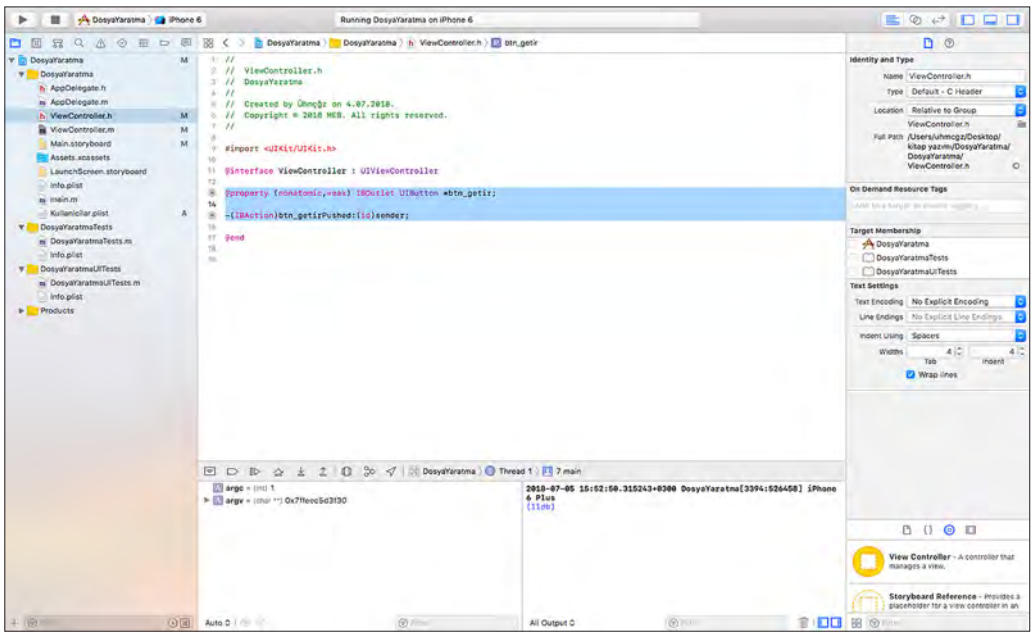
Main.storyboard ekranı açılarak “KAYITLARI GETİR” isimli bir Button oluşturulur.



Ekran Görüntüsü 5.54

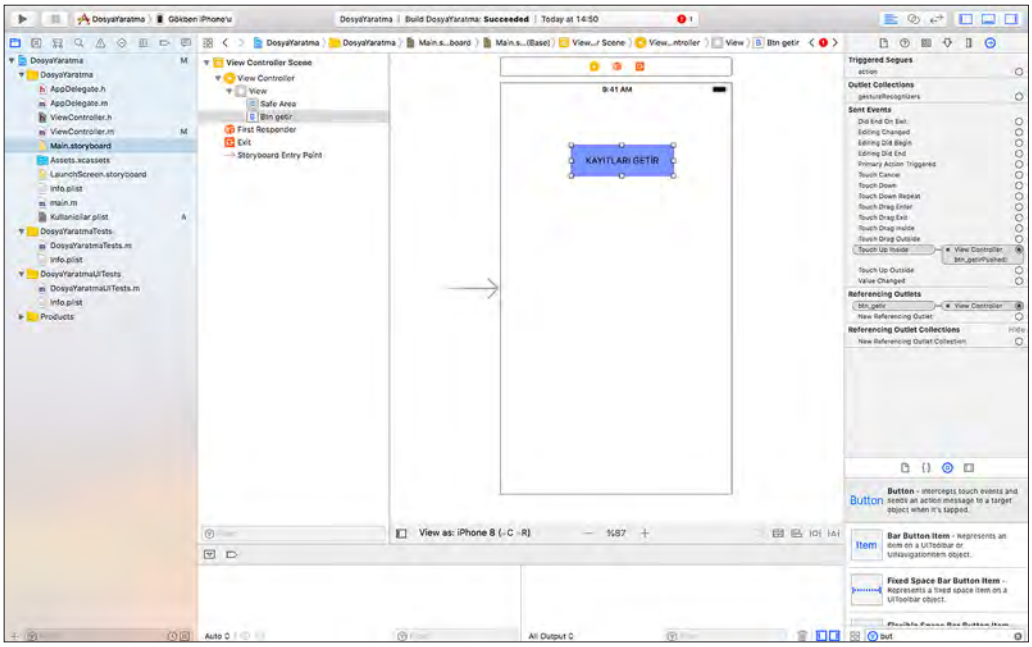
Düğmenin ViewController.h dosyasında isim ve action fonksiyonu tanımlaması yapılır.

```
@property (nonatomic,weak) IBOutlet UIButton *btn_getir;

-(IBAction)btn_getirPushed:(id)sender;
```



Ekran Görüntüsü 5.55

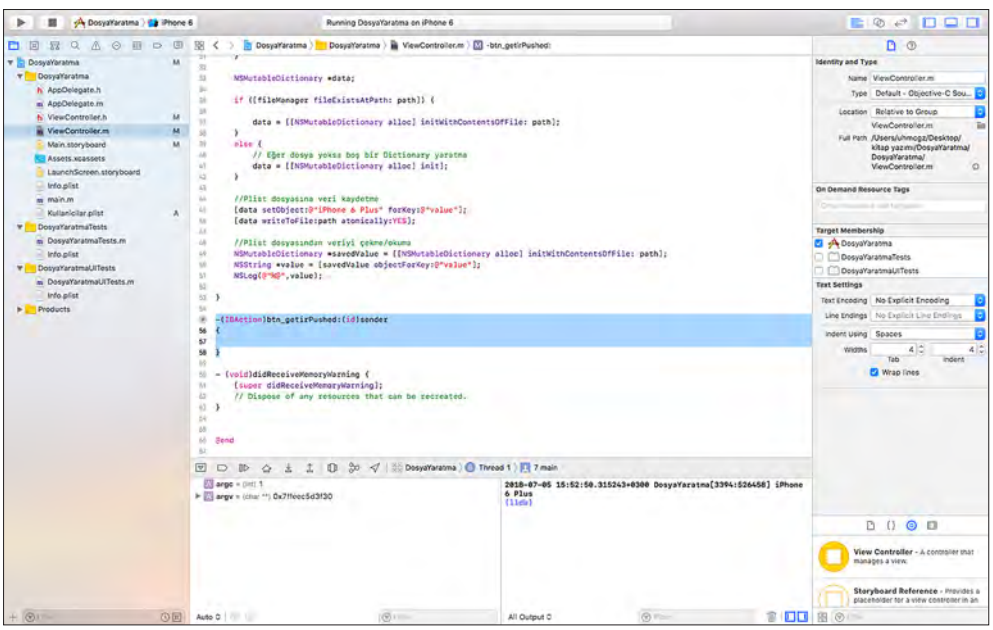
Düğmenin fiziksel olarak kodla ilişkilendirmesinin yapılması için Storyboard ekranı açılır.



Ekran Görüntüsü 5.56

ViewController.m dosyasında düğme tıklandığında yapılacak işlemlerin yazılacağı action fonksiyonu açılır.

```
-(IBAction)btn_getirPushed:(id)sender
{


}
```

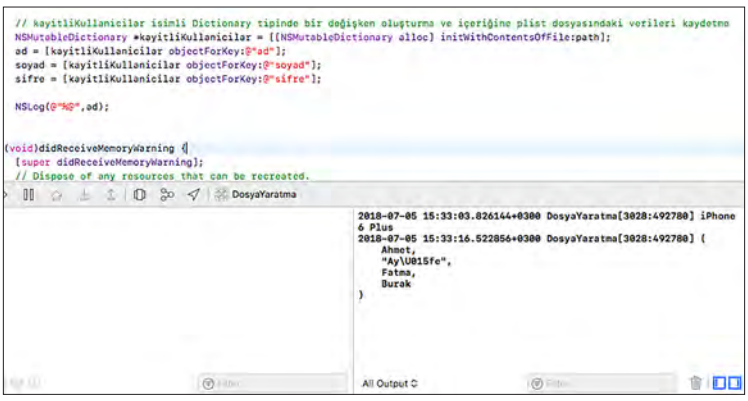


Ekran Görüntüsü 5.57

Action fonksiyonunun içerisine aşağıdaki kodlar sırasıyla eklenir.

```
-(IBAction)btn_getirPushed:(id)sender
{
NSString *ad;        //string tipinde dosyadan okunacak ad, soyad ve sifre bilgilerini tutacak değişkenler oluşturulur
NSString *soyad;
NSString *sifre;

// DOSYANIN AÇILMASI
NSError *error;
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
NSString *documentsDirectory = [paths objectAtIndex:0];
NSString *path = [documentsDirectory stringByAppendingPathComponent:@ "Kullanicilar.plist"];

NSFileManager *fileManager = [NSFileManager defaultManager];

if (![fileManager fileExistsAtPath: path]){
NSString *bundle = [[NSBundle mainBundle] pathForResource:@"Kullanicilar" ofType:@"plist"];
            [fileManager copyItemAtPath:bundle toPath: path error:&error];
}

// kayitliKullanicilar isimli Dictionary tipinde bir değişken oluşturma ve içeriğine plist dosyasındaki bilgilerin aktarımı
NSMutableDictionary *kayitliKullanicilar = [[NSMutableDictionary alloc] initWithContentsOfFile:path];
ad = [kayitliKullanicilar objectForKey:@"ad"];
soyad = [kayitliKullanicilar objectForKey:@"soyad"];
sifre = [kayitliKullanicilar objectForKey:@"sifre"];

NSLog(@"%@",ad);   // program arka panelinden kontrol amaçlı ad değişkeninin yazdırılması
}
```

Uygulama çalıştırılarak kontrol sağlanır. Kayıtları getir tıklandığında proje ekranının sağ alt köşesinde ad değişkeninin içeriğinin doğru bir biçimde ekrana yazıldığı görülür.



Ekran Görüntüsü 5.58

Ekran Görüntüsü 5.59

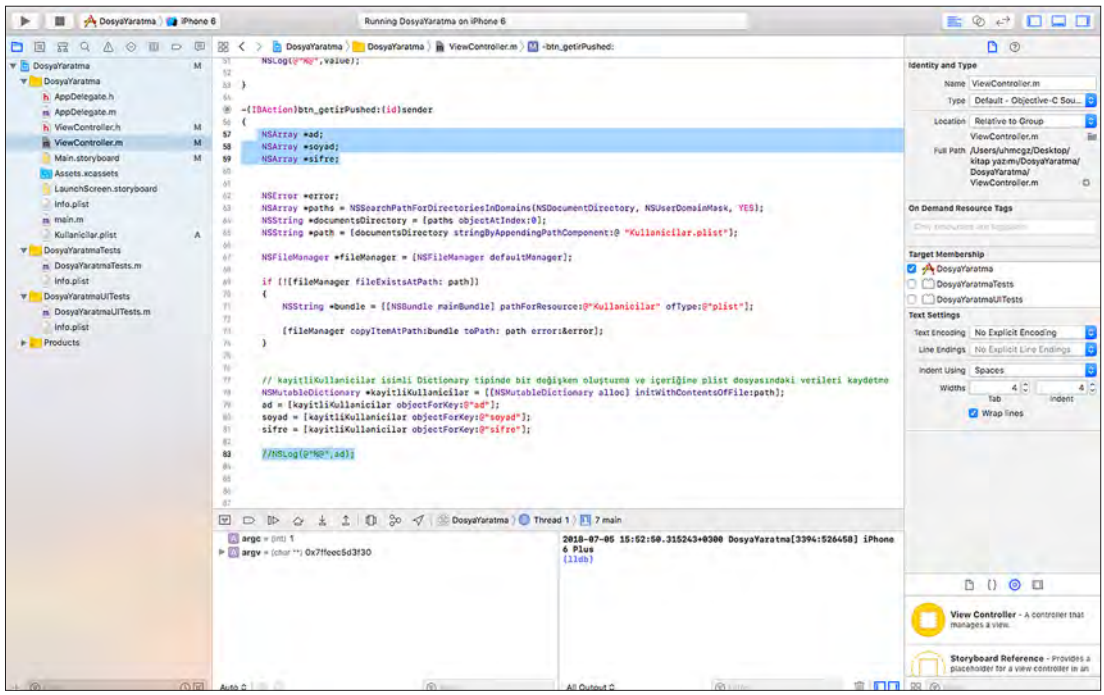
Kayıt içeriğinin uygulama ekranına yazdırılması için Label'lar kullanılacaktır. Bunun için for döngüsü kullanılarak kayıt sayısı kadar kodla Label'lar oluşturulur.

Öncelikle Label içerisine yazdırılacak ad, soyad ve şifre bilgilerinin dosyadan okurken kaydedildiği değişkenlerin tipi NSString'ten NSArray tipine çevrilir. Böylece her bir Label'a hangi kayıt yazılacağı belirlenebilecektir. Örneğin ad dizisinin 0., 1., 2. elemanı gibi.

```
NSArray *ad;
NSArray *soyad;
NSArray *sifre;
```

Ardından NSLog ile ad değişkenini yazdıran kodun ön kısmına // eklenerek açıklama satırı yapılır.

```
//NSLog(@"%@",ad);
```



Ekran Görüntüsü 5.60

Devamında aşağıdaki kodlar eklenir:

```
// Tüm dosya içeriğinin döngü kullanılarak Label içinde ekrana yazdırılması
for (int i=0; i<[ad count]; i++) {    // i 0'dan başlayarak, ad dizisi eleman sayısına kadar, i birer birer arttırılarak for döngüsü açılır.
UILabel *lbl_ad = [[UILabel alloc]initWithFrame:CGRectMake(15, 150+i*100, 80, 60)]; // ekranda bir label oluşturma
lbl_ad.text = [ad objectAtIndex:i];                // oluşturulan label'ın metnini ad dizisindeki i. elemen yapma
lbl_ad.backgroundColor = [UIColor yellowColor];    // label'ın arkaplan rengini sarı yapma
[self.view addSubview:lbl_ad];                     // Label'ı ana view'ın bir alt view'ı olarak ekrana ekleme

UILabel *lbl_soyad = [[UILabel alloc]initWithFrame:CGRectMake(100, 150+i*100, 80, 60)];
lbl_soyad.text = [soyad objectAtIndex:i];
lbl_soyad.backgroundColor = [UIColor greenColor];
[self.view addSubview:lbl_soyad];

UILabel *lbl_sifre = [[UILabel alloc]initWithFrame:CGRectMake(185, 150+i*100, 80, 60)];
lbl_sifre.text = [sifre objectAtIndex:i];
lbl_sifre.backgroundColor = [UIColor purpleColor];
[self.view addSubview:lbl_sifre];
}
```

Kodla bir Label oluşturmak için kullanılan;

```
UILabel *lbl_ad = [[UILabel alloc]initWithFrame:CGRectMake(15, 150+i*100, 80, 60)];
```

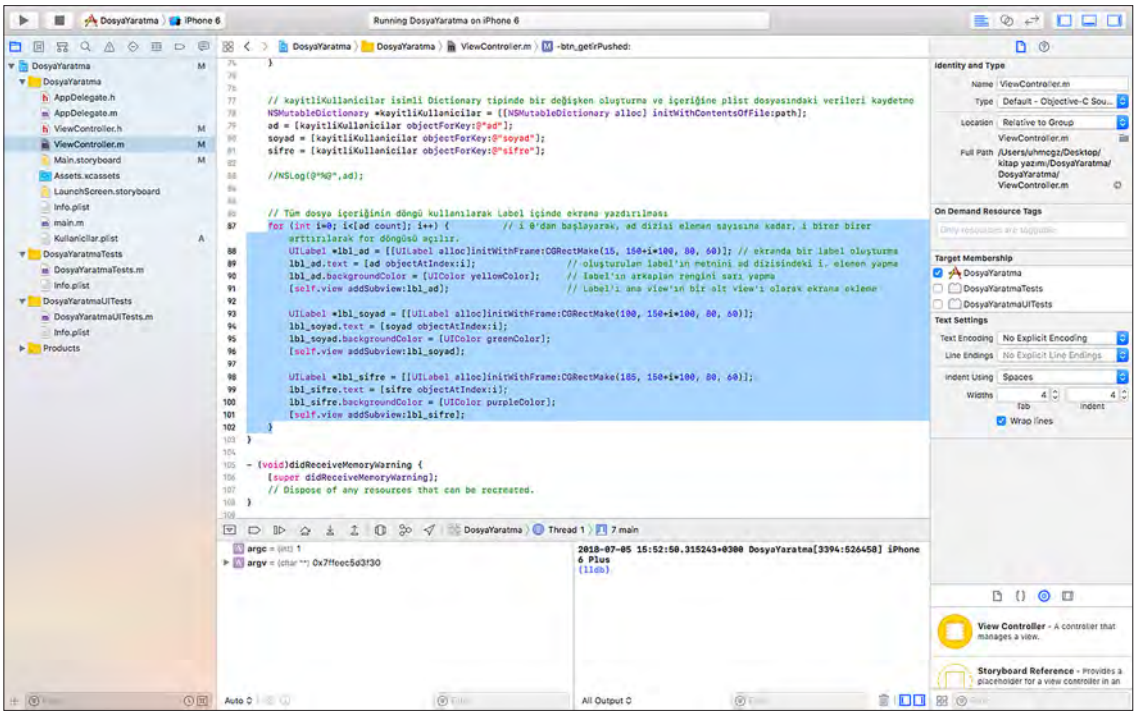
kod satırında

15: Label'ın koordinat düzlemindeki X konumunu

150+i*100 : Y konumunu

80: genişliğini

60: yüksekliğini pixel cinsinden vermektedir.



Ekran Görüntüsü 5.61

Uygulama simülatörde çalıştırıldığında ve KAYITLARI GETİR düğmesine tıklandığında ekran görüntüsü yandaki gibi olacaktır.

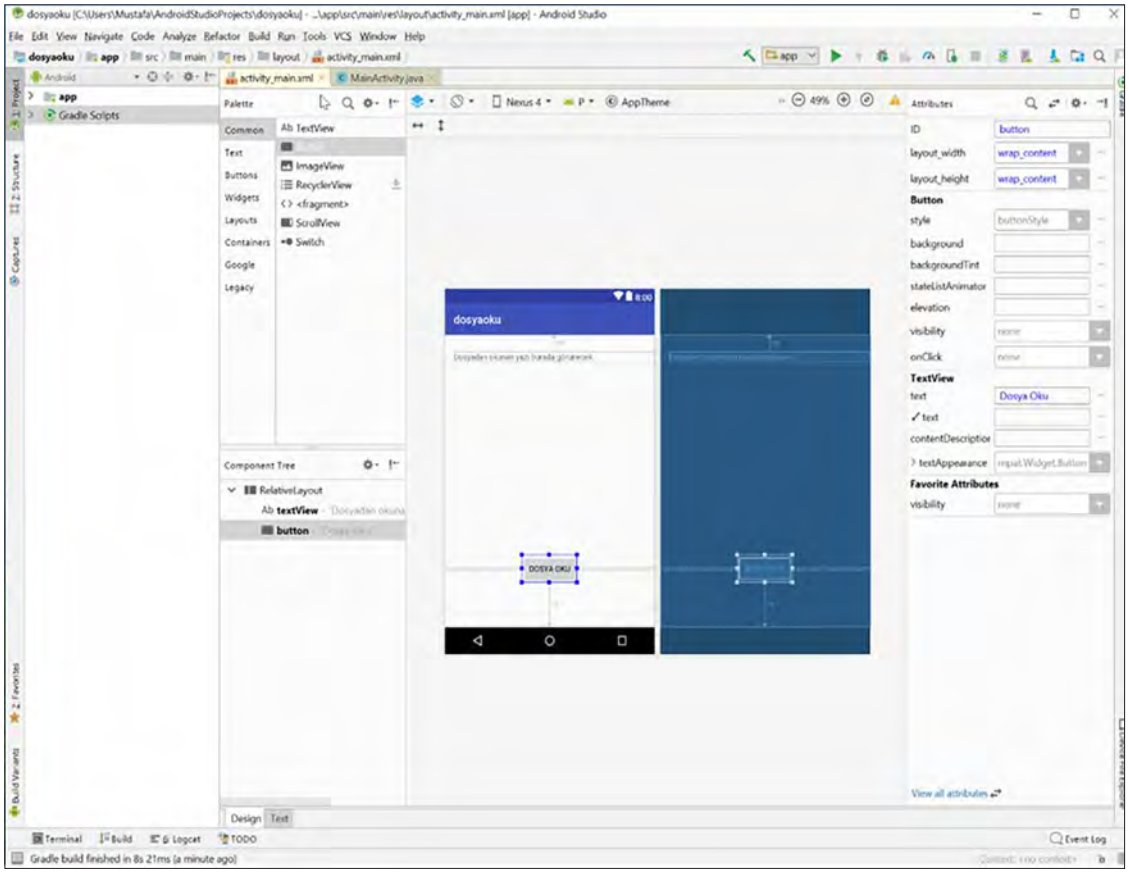
Ekran Görüntüsü 5.62

5.2. Android Studio Dosya İşlemleri

5.2.1 Dosyadan Metin Okuma ve Ekrana Yazma

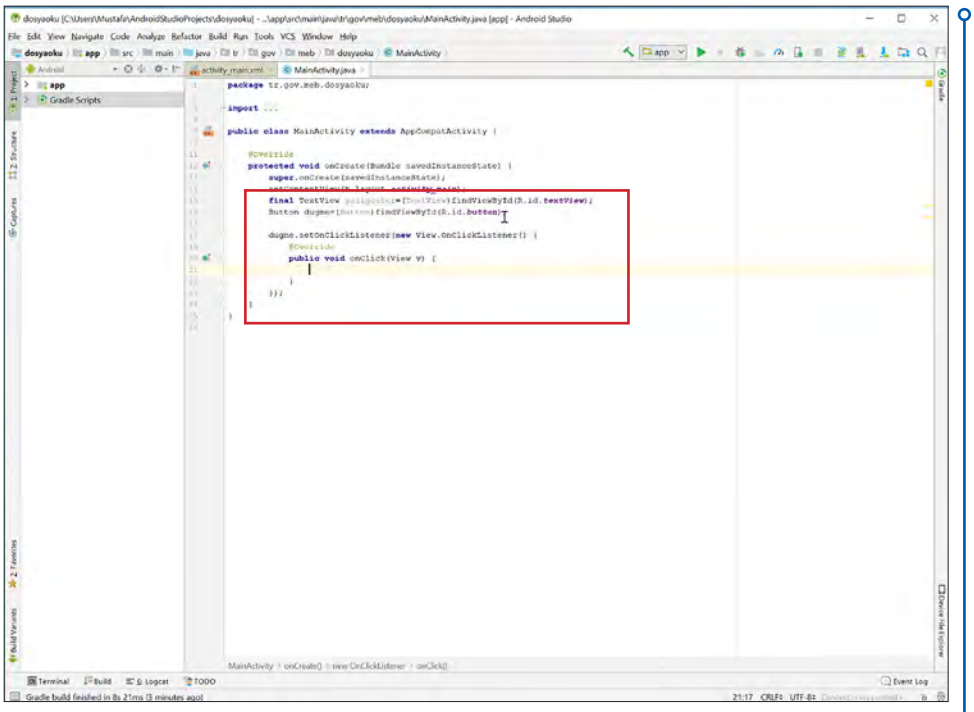
Oluşturduğumuz klasör içinde bulunan txt dosyasından veri okuyup TextView içerisinde gösterme işlemi yapan bir uygulama yapılacaktır. Yeni bir proje açınız ve adını "dosyaoku" yapınız. Empty Activity seçip işlemi tamamlayınız.

Oluşturduğunuz projeye bir TextView bir de Button ekleyiniz. TextView'de dosyadan okunan yazı burada görünecek, Button'da da Dosya Oku yazacaktır.



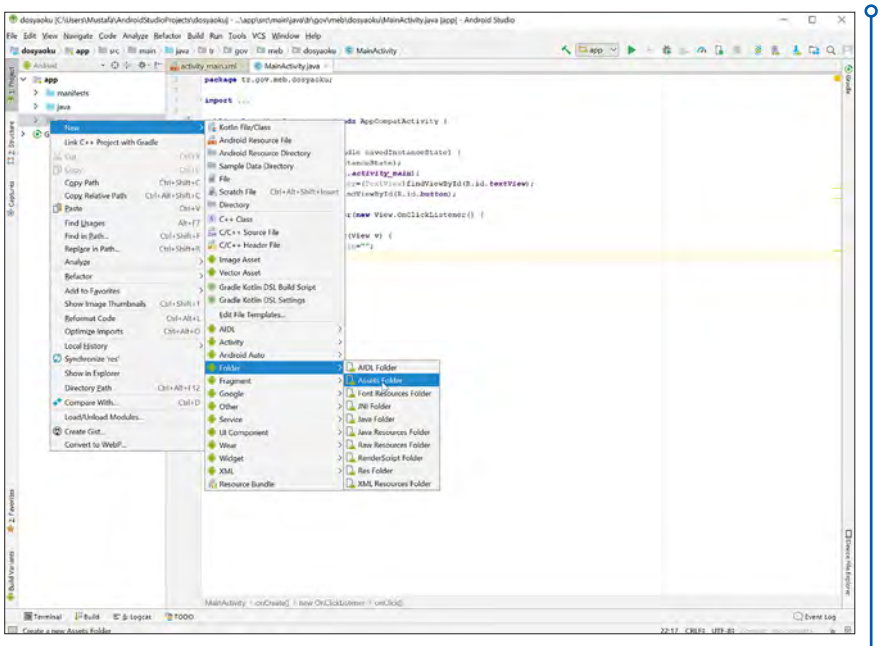
Ekran Görüntüsü 5.63

Java dosyanızda TextView ve button nesnelerini tanımlayınız. Daha sonra düğmesine basıldığında olayını gerçekleştirmek için kodlarınızı ekleyiniz.



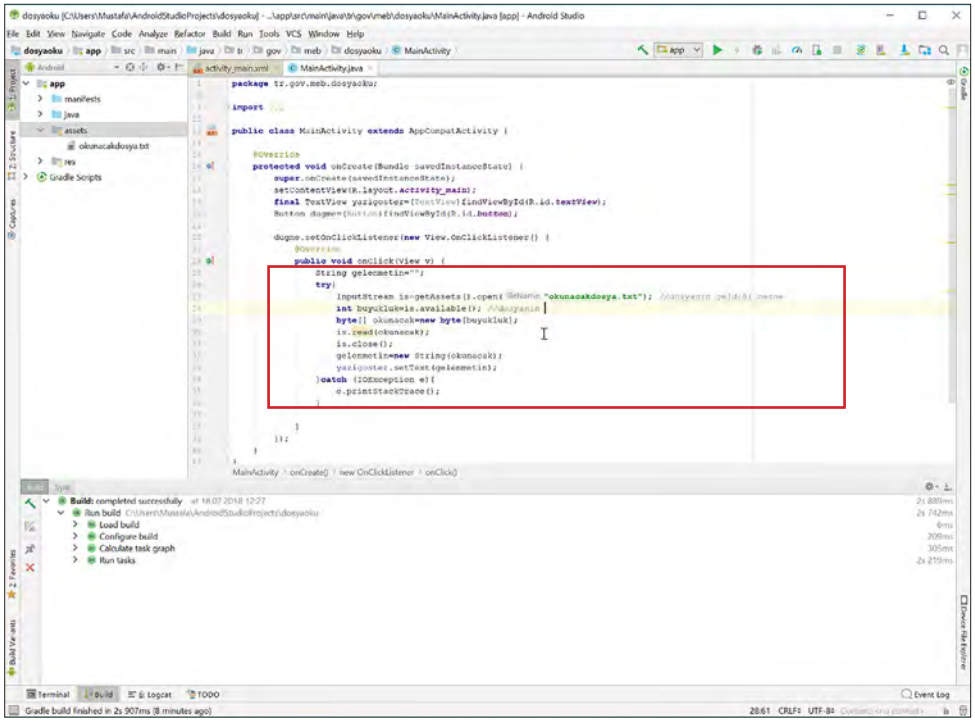
Ekran Görüntüsü 5.64

Dosyadan gelen metinlerin bir string değişkene atanması gerekmektedir Bu değişkeni oluşturunuz. Daha sonra res klasörü içerisine New -> Folder -> Assets Folder oluşturunuz. Bu klasör tipi daha çok sadece okunabilir dosyaların uygulamanın içine eklenmesi için kullanılmaktadır.



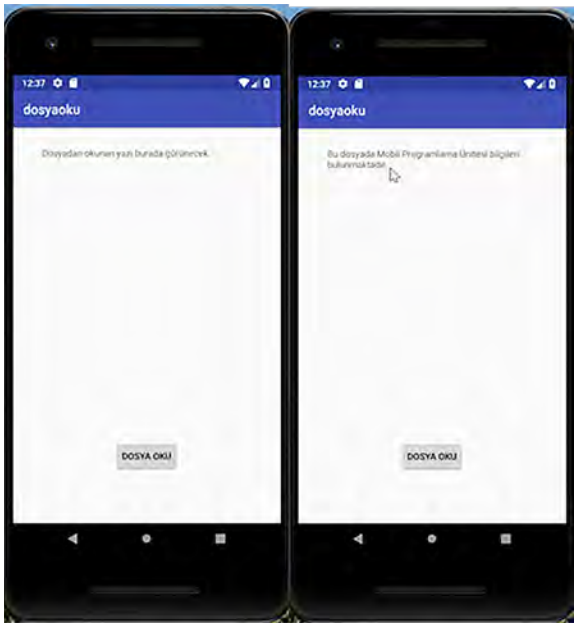
Ekran Görüntüsü 5.65

Assets Folder'ın içine okunacakdosya isimli bir txt dosyası oluşturunuz ve içine **"Bu dosyada mobil programlama ünitesi bilgileri bulunmaktadır"** yazınız. Daha sonra java dosyanıza okunacakdosya.txt içindeki tüm verilerin okunmasını sağlayan aşağıdaki kodları ekleyiniz. Hepsini okuyacağı için byte kodu kullanılmaktadır. Dosyanın boyutunun kaç byte olduğunu çeken değişken tanımlayınız ve bu değişken kadar verinin okunmasını sağlayınız. Gelen metin değişkenini TextView'de göstermesini sağlayan kodu ekleyiniz.



Ekran Görüntüsü 5.66

Tüm kodları ekleyip projeyi çalıştırdığınızda uygulama yandaki gibi görünecektir.



Ekran Görüntüsü 5.67

XML Kodları

```
<?XML version="1.0" encoding="utf-8"?>
<RelativeLayout XMLns:android="http://schemas.android.com/apk/res/android"
    XMLns:app="http://schemas.android.com/apk/res-auto"
    XMLns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/TextView"
        android:layout_width="367dp"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentTop="true"
        android:layout_marginTop="29dp"
        android:text="Dosyadan okunan yazı burada görünecek." />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="78dp"
        android:text="Dosya Oku" />
</RelativeLayout>
```

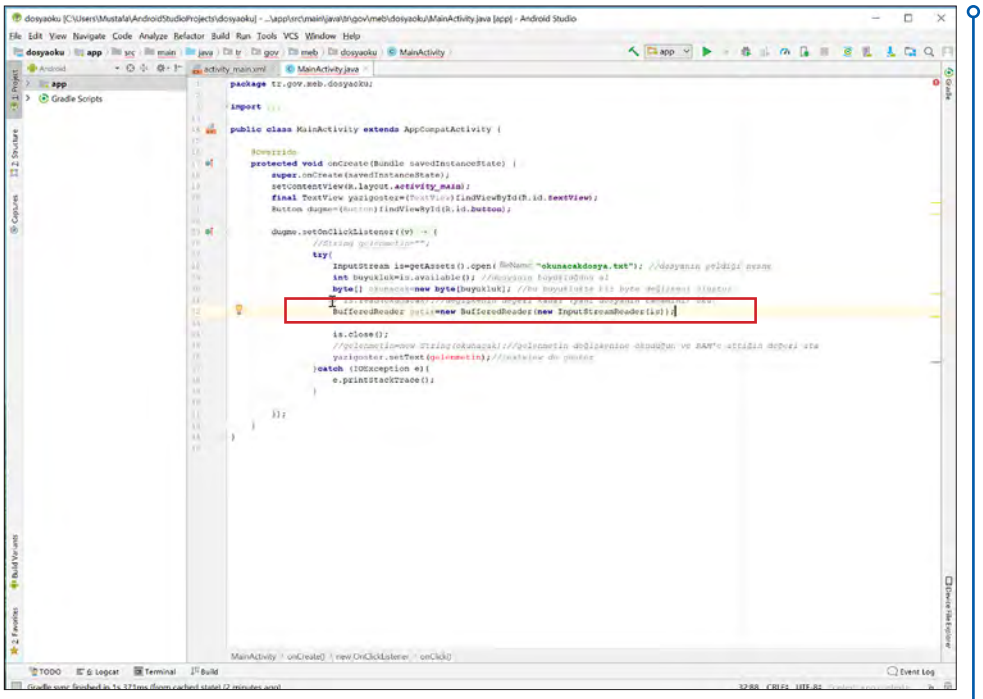
Java Kodları

```
package tr.gov.meb.dosyaoku;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;

public class MainActivity extends AppCompatActivity {
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    final TextView yazigoster=(TextView)findViewById(R.id.TextView);
    Button dugme=(Button)findViewById(R.id.button);

    dugme.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String gelenmetin="";
            try{
                InputStream is=getAssets().open("okunacakdosya.txt");  //dosyanın geldiği nesne
                int buyukluk=is.available();  //dosyanın büyüklüğünü al
                byte[] okunacak=new byte[buyukluk];  //bu büyüklükte bir byte değişkeni oluştur
                is.read(okunacak); //değişkenin değeri kadar (yani dosyanın tamamını) oku.
                is.close();
                gelenmetin=new String(okunacak);  //gelenmetin değişkenine okuduğun ve RAM'e attığın değeri ata
                yazigoster.setText(gelenmetin);  //TextView de göster
            }catch (IOException e){
                e.printStackTrace();
            }

        }
    });
}
}
```

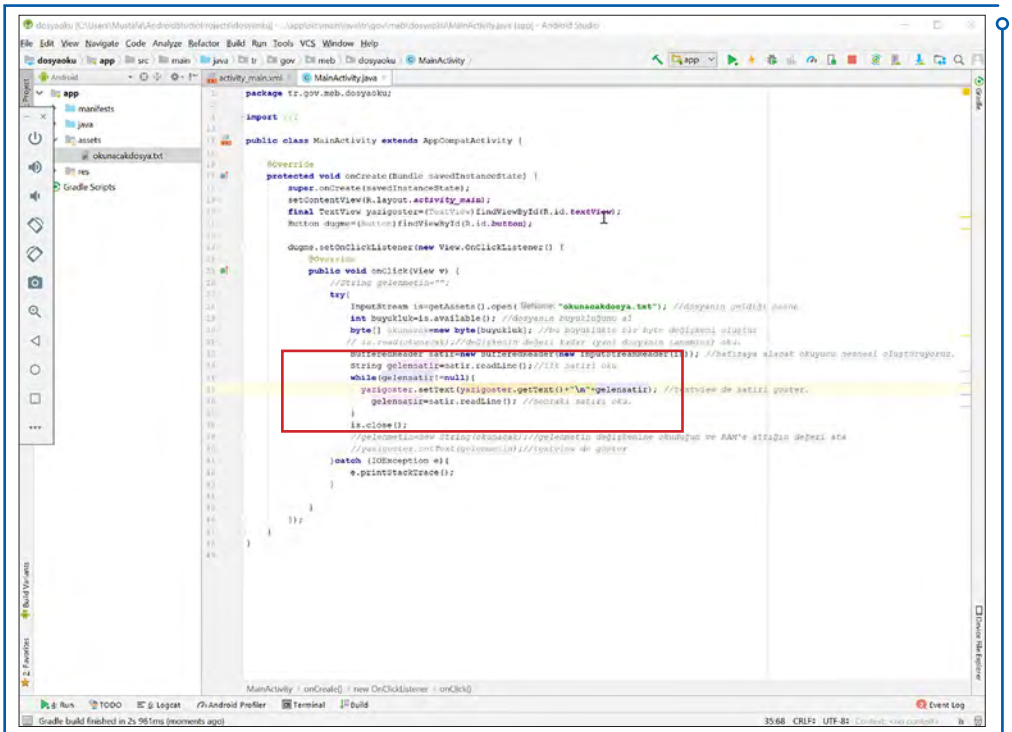
5.2.2. Dosyadan Satır Satır Veri Okuma ve Ekrana Yazma

Bu uygulama dosyaoku uygulaması üzerinde değişiklik yapılarak gerçekleştirilecektir. Daha önce yaptığınız dosyaoku projenizi açınız. Bu uygulama içindeki is.read okuma kodu (metodu) yerine BufferedReader sınıfı kullanılacaktır. BufferedReader daha yüksek performans bellek sağlar. BufferedReader kullanılmazsa read metodu her çağırıldığında dosyadan byte'lar çekilir, karakterlere dönüştürülür ve döndürülür. Bu da programın performansının düşmesine yol açar.



Ekran Görüntüsü 5.68

BufferedReader sınıfı dosyayı satır satır okumak için readLine() metodu kullanır. Gelen satırları bir değişkene aktarmak için gelensatir isimli string bir değişken oluşturunuz. Değişkene gelen satır verilerinin tek tek görüntülenebilmesi için bir döngüye ihtiyaç vardır. Döngü içinde TextView nesnenizde satırların yazdırılmasını sağlayan kodları ekleyiniz. /n java'da enter tuşunun görevini (bir alt satıra indirme) görür. Metinlerin bir alt satıra geçmesi istendiğinde /n kullanılmalıdır.



Ekran Görüntüsü 5.69

XML Kodları

```
<?XML version="1.0" encoding="utf-8"?>
<RelativeLayout XMLns:android="http://schemas.android.com/apk/res/android"
    XMLns:app="http://schemas.android.com/apk/res-auto"
    XMLns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/TextView"
        android:layout_width="367dp"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentTop="true"
        android:layout_marginTop="29dp"
        android:text="Dosyadan okunan yazı burada görünecek." />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="78dp"
        android:text="Dosya Oku" />
</RelativeLayout>
```

Java Kodları

```
package tr.gov.meb.dosyaoku;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class MainActivity extends AppCompatActivity {
```

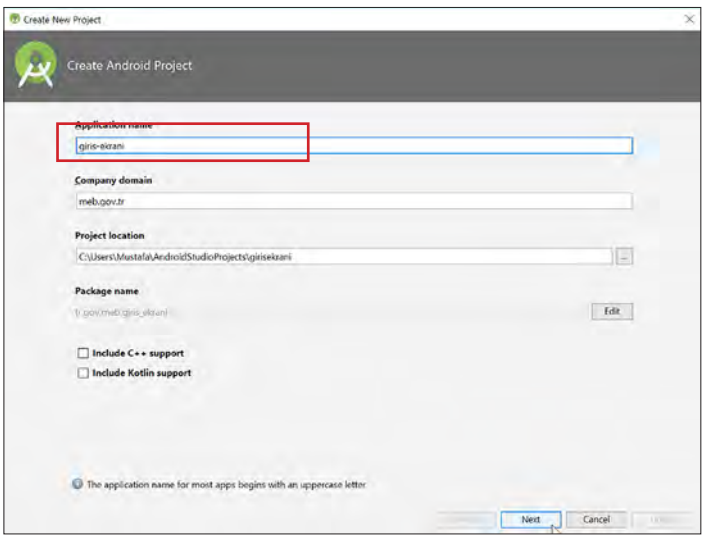
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    final TextView yazigoster=(TextView)findViewById(R.id.TextView);
    Button dugme=(Button)findViewById(R.id.button);

    dugme.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            try{
                InputStream is=getAssets().open("okunacakdosya.txt");  //dosyanın geldiği nesne
                int buyukluk=is.available();  //dosyanın büyüklüğünü al
                byte[] okunacak=new byte[buyukluk];   //bu büyüklükte bir byte değişkeni oluştur
                BufferedReader satir=new BufferedReader(new InputStreamReader(is));   //hafızaya alacak okuyucu
nesnesi oluştur.
                String gelensatir=satir.readLine();  //ilk satırı oku
                while (gelensatir!=null){
                    yazigoster.setText(yazigoster.getText()+"\n"+gelensatir);  //TextView de satırı göster
                    gelensatir=satir.readLine();   //sonraki satırı oku
                }
                is.close();  //dosya nesnesini kapat
            }catch (IOException e){
                e.printStackTrace();
            }
        }
    });
}
}
```

Tüm kodlarınızı eklediğinizde uygulama txt dosyası içindeki satırları yandaki gibi gösterecektir.

Ekran Görüntüsü 5.70

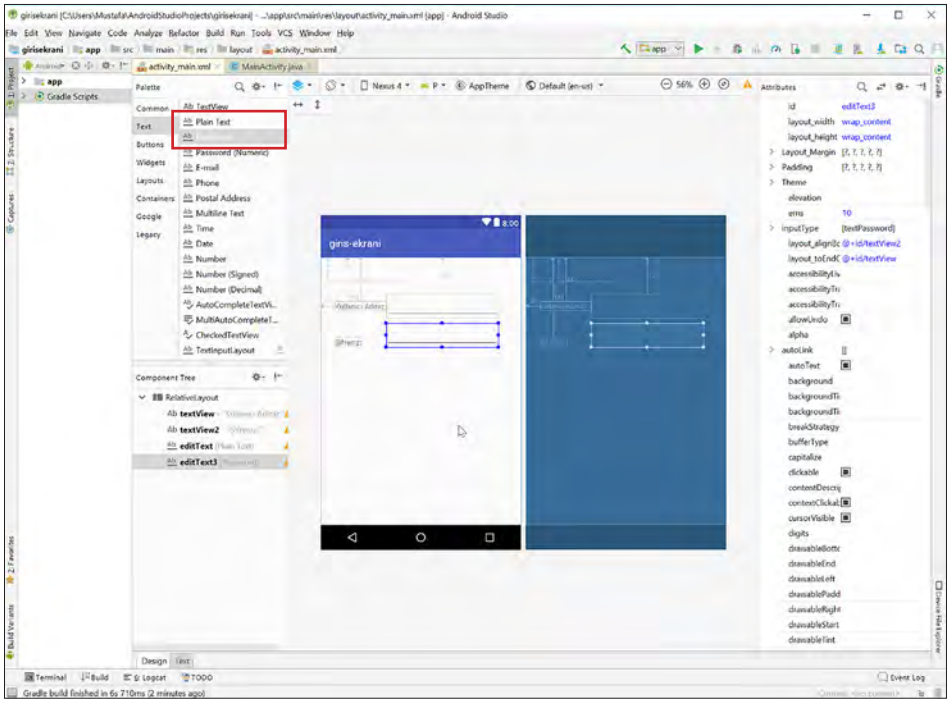
5.2.3. Kullanıcı Adı- Şifre Giriş Ekranı Uygulaması

Bu uygulamada txt uzantılı dosyadan kullanıcı adı-şifre çekip okuma ve kullanıcı giriş ekranı yapma gösterilmektedir. Android Studio'yu açınız. Yeni bir proje oluşturunuz ve adını 'giris-ekranı' veriniz.



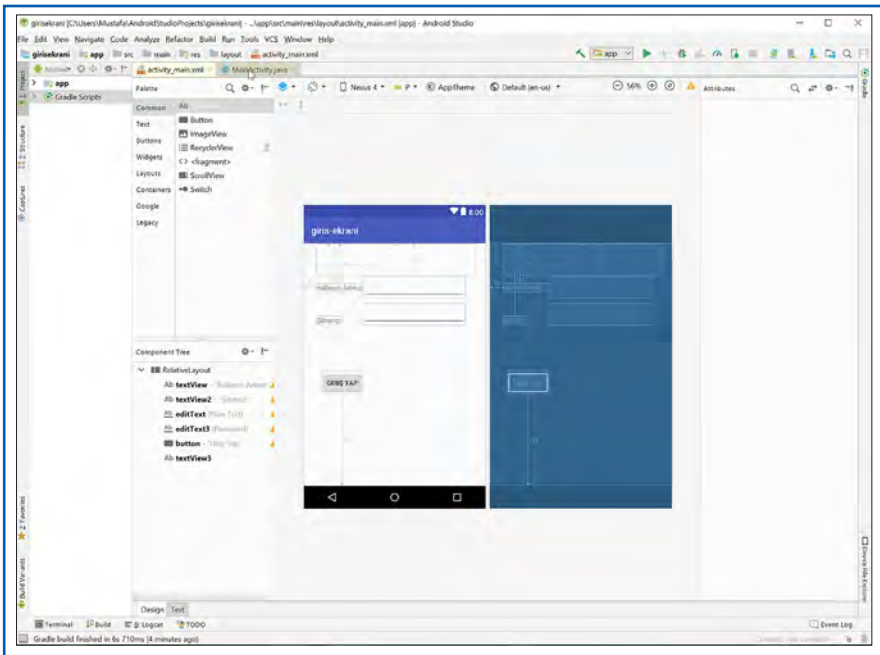
Ekran Görüntüsü 5.71

Empty Activity seçerek uygulamayı açınız. Kullanıcı Adınız ve Şifreniz isimli 2 tane TextView ekleyiniz. Daha sonra Text içinde yer alan Plain Text ve Password ekleyiniz. Her ikisi de aslında editText nesnesidir. Kullanıcının uygulamaya veri girişini sağlar. Plain Text normal veri girişi, Password şifre girişi için kullanılmaktadır.



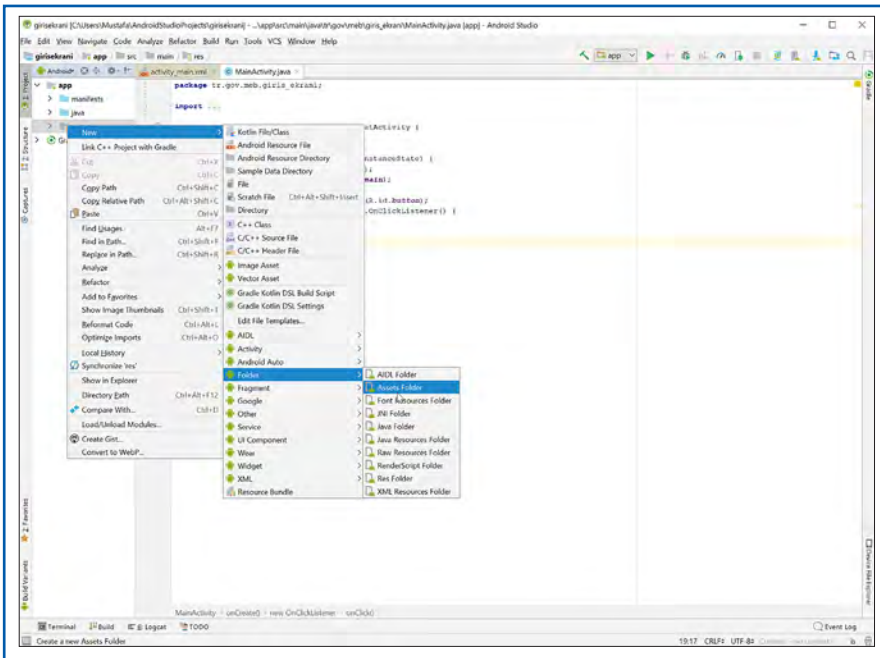
Ekran Görüntüsü 5.72

Giriş yap isimli bir button ve içi boş bir TextView ekleyerek tasarım ekranınızı sonlandırınız. İçi boş TextView'in size (yani genişlik) değerini 14 olarak ayarlayınız. Bu TextView giriş yapıldıktan sonra yazılmasını istediğimiz metni yazdırmak için kullanılacaktır.



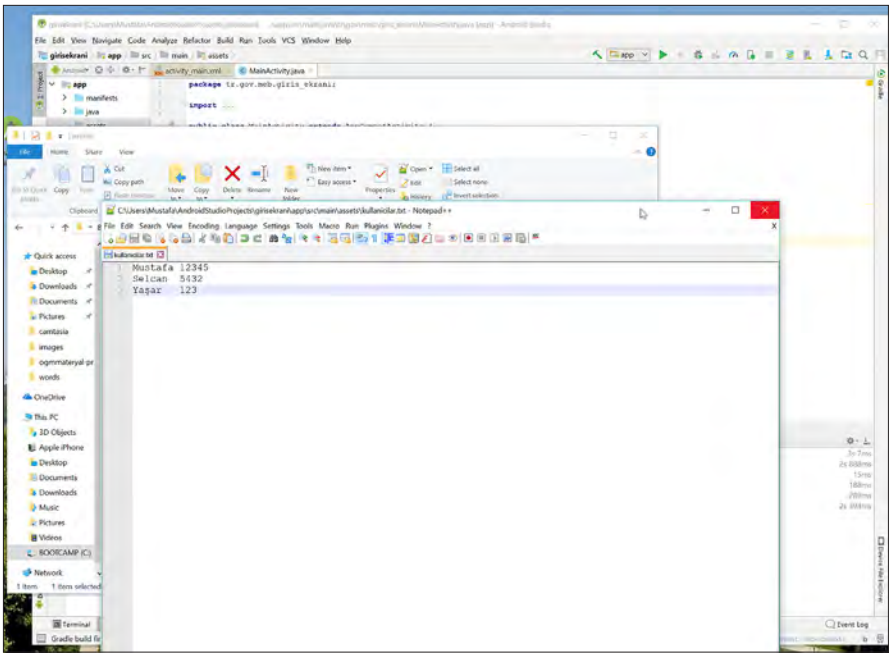
Ekran Görüntüsü 5.73

Tasarım ekranını bitirdikten sonra kodlara geçebilirsiniz. Öncelikle kullanacağınız nesneleri tanımlayan kodları ekleyiniz. Düğmeye tıklandığında olayını kodlara ekleyiniz. Kullanıcı adı ve şifreyi uygulama txt dosyasından satır satır okuyacaktır. Bu yüzden res üzerine sağ tıklayıp new-> Folder-> Assets Folder oluşturunuz.



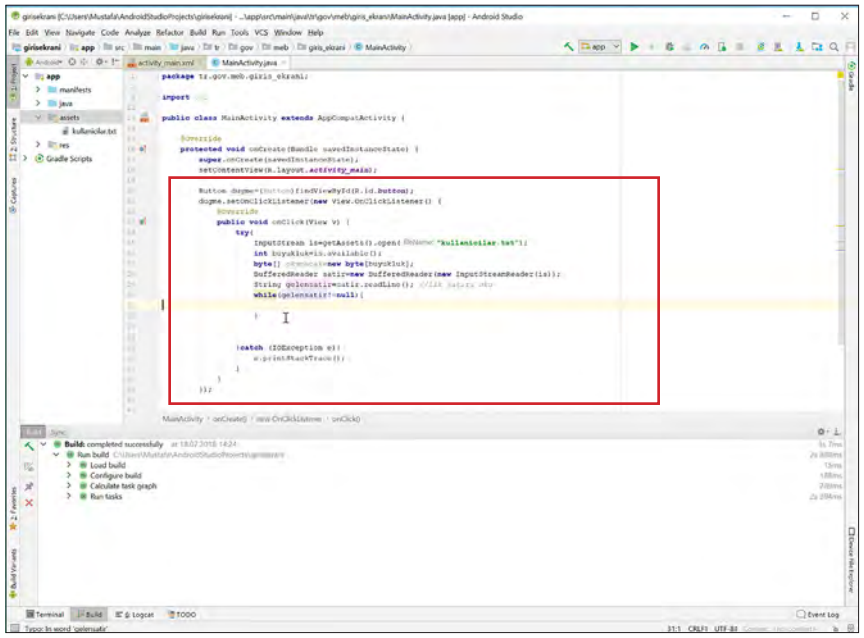
Ekran Görüntüsü 5.74

Assets Folder'ın içine txt dosyası eklemek için üzerine gelip sağ tıklayınız ve Show In Explorer seçip içine "kullanicilar" isimli bir txt dosyası oluşturunuz. txt dosyası içine 3 tane kullanıcı adı ve şifre ekleyiniz. Kullanıcı adı ve şifre arasındaki boşluğu Tab tuşu kullanarak oluşturunuz. Daha sonra kodlama kısmında bu iki metni ayırırken Tab tuşu kullanılacaktır.



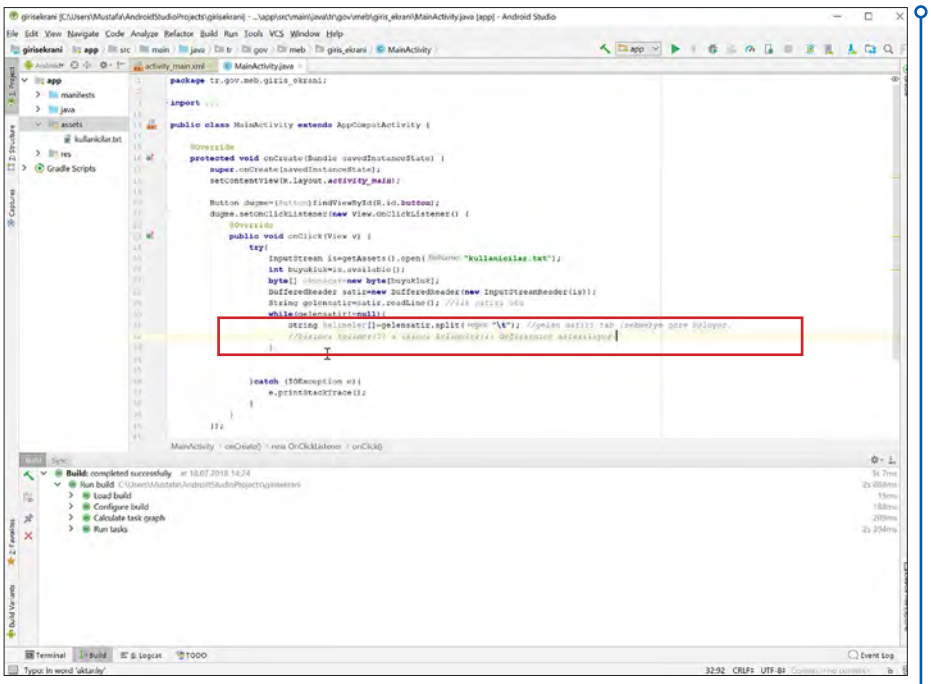
Ekran Görüntüsü 5.75

Daha önceki uygulamada yapıldığı gibi kullanicilar.txt dosyanızdaki metinleri dosyanızdaki metinler satır satır okuma kodlarını ekleyiniz.



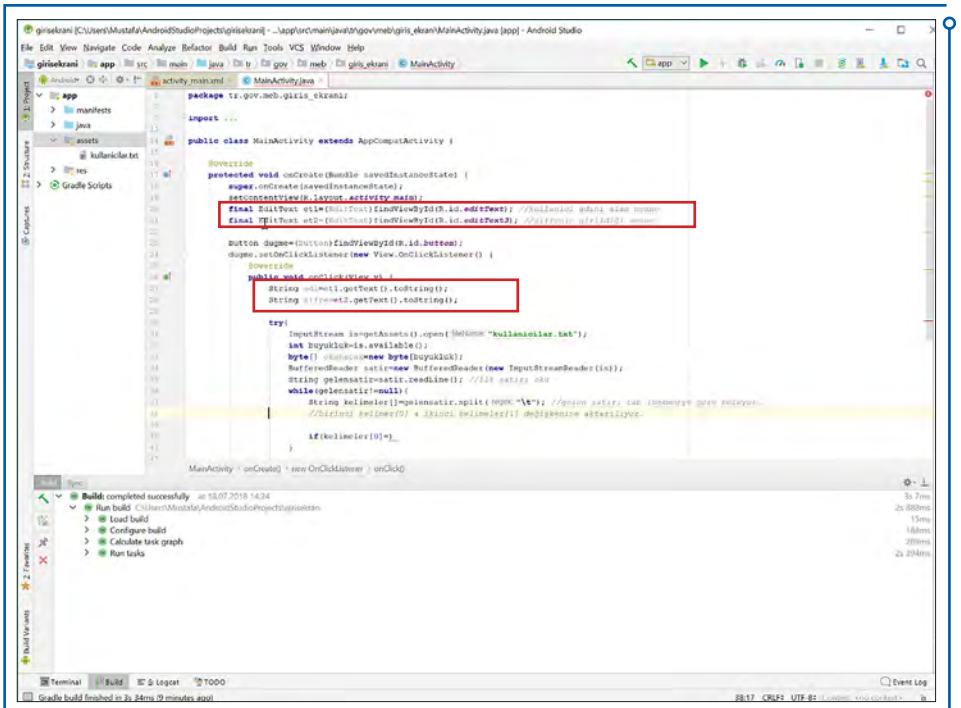
Ekran Görüntüsü 5.76

Her satırda iki metin olduğundan (kullanıcı adı ve şifre) bu metinlerin bölünmesi gerekmektedir. Bunun için "kelimeler" isimli bir string dizisi oluşturunuz. Bu diziyi "split" komutu ile bölmeniz gerekmektedir. Split komutunun metni bölerken neye göre böleceğini bilmesi gerekmektedir. Tab tuşu java dilinde \t olarak geçer. Bu şekilde kodlarınızı ekleyerek satırlardaki iki metni bölebilirsiniz. Birinci kelime 0 değişkenine, ikinci kelime 1 değişkenine aktarılır.



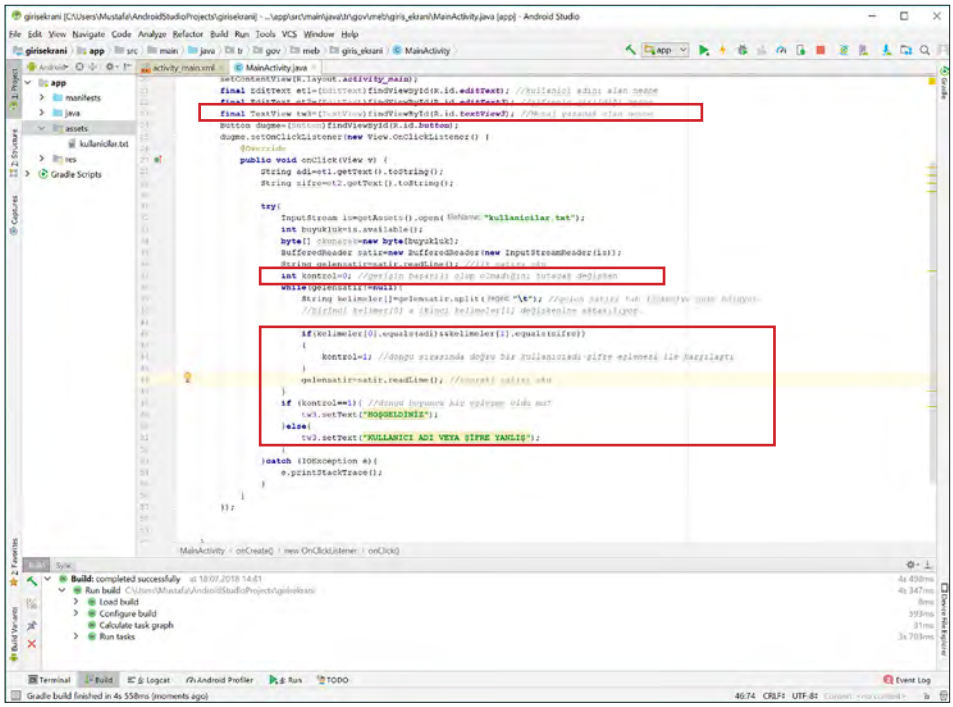
Ekran Görüntüsü 5.77

Daha sonra kullanıcının girdiği verilerin txt dosyası verileriyle aynı olup olmadığı kontrol eden kodlar eklenmelidir. Bunu editText nesnelerindeki veri ile dizinizin 0. ve 1. değişkenindeki verilerle eşitliğine bakarak kontrol edebilirsiniz. Bunun için öncelikle iki editText nesnesini tanımlayan kodları ve burdan gelen verileri string'e dönüştüren kodları ekleyiniz.



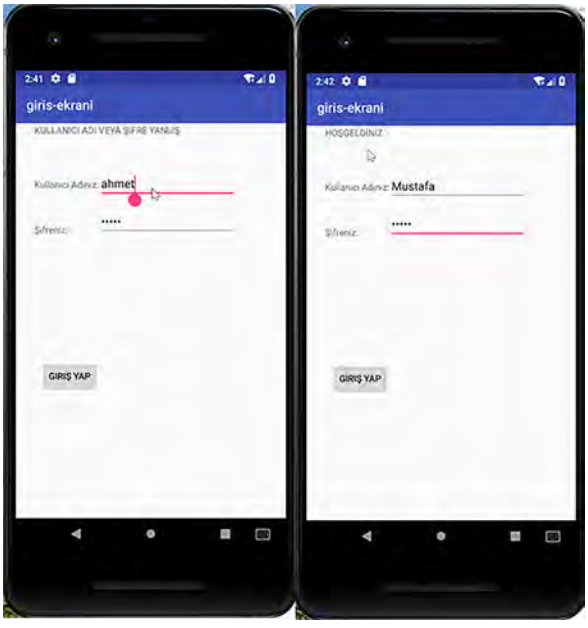
Ekran Görüntüsü 5.78

Artık editText'e girilen veri ile txt dosyanızdaki verileri karşılaştırabilirsiniz. Bunun için iki metnin aynı olup olmadığını kontrol eden kodları; aynı ise boş olarak eklediğiniz TextView'de "Hoşgeldiniz", değilse "Kullanıcı adı veya şifre yanlış" yazan kodları ekleyiniz. Bunun için önce TextView nesnesini tanımlayınız. Daha sonra bir döngü oluşturunuz. Döngü içinde textlerin aynı olup olmadığını tutan bir kontrol değişkeni tanımlayınız. Bu kontrol değişkeninin değeri 1 ise doğru giriş, değilse yanlış giriş yapıldığını belirten kodları ekleyiniz.



Ekran Görüntüsü 5.79

Tüm kodları eklediğinizde uygulamanızın ekran görüntüsü yandaki gibi olacaktır.



Ekran Görüntüsü 5.80

XML Kodları

```
<?XML version="1.0" encoding="utf-8"?>
<RelativeLayout XMLns:android="http://schemas.android.com/apk/res/android"
    XMLns:app="http://schemas.android.com/apk/res-auto"
    XMLns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/TextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_marginStart="28dp"
        android:layout_marginTop="84dp"
        android:text="Kullanıcı Adınız:" />

    <TextView
        android:id="@+id/TextView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignStart="@+id/TextView"
        android:layout_marginTop="153dp"
        android:text="Şifreniz:" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginTop="70dp"
        android:layout_toEndOf="@+id/TextView"
        android:ems="10"
        android:inputType="textPersonName" />

    <EditText
        android:id="@+id/editText3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/TextView2"
        android:layout_toEndOf="@+id/TextView"
```

```
    android:ems="10"
    android:inputType="textPassword" />

<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignEnd="@+id/TextView"
    android:layout_alignParentBottom="true"
    android:layout_marginBottom="193dp"
    android:text="Giriş Yap" />

<TextView
    android:id="@+id/TextView3"
    android:layout_width="336dp"
    android:layout_height="67dp"
    android:layout_alignParentTop="true"
    android:layout_alignStart="@+id/TextView"
    android:layout_marginTop="1dp"
    android:textSize="14sp"
    android:visibility="visible" />

</RelativeLayout>
```

Java Kodları

```
package tr.gov.meb.giris_ekrani;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

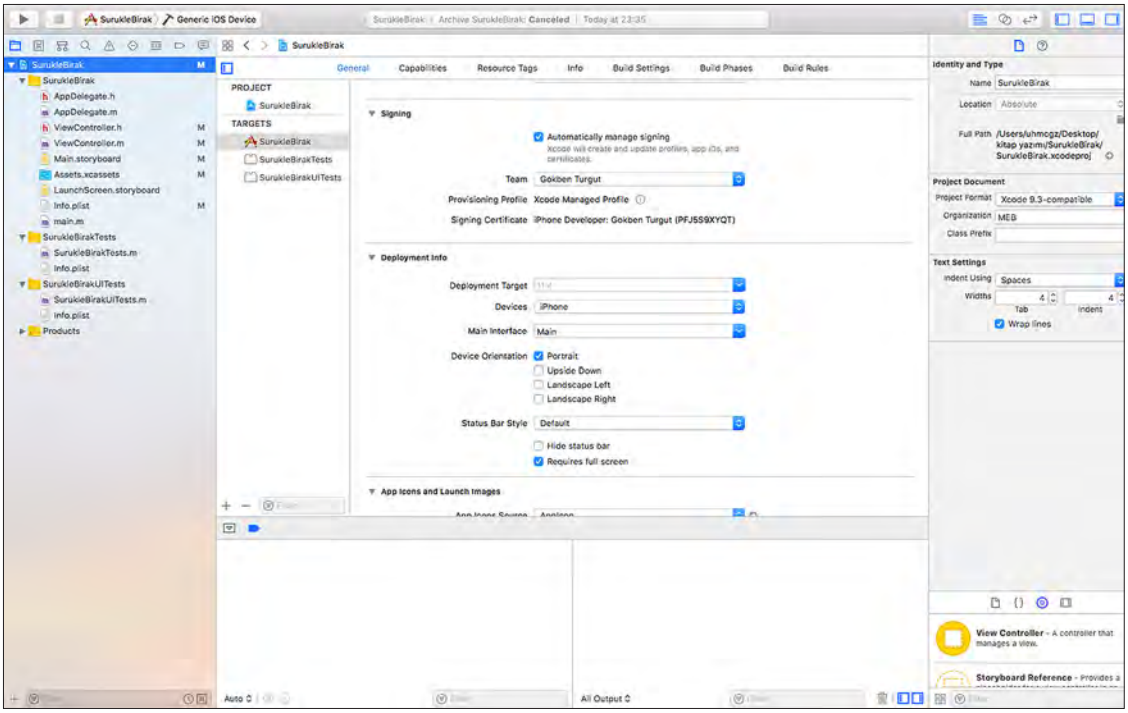
```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    final EditText et1=(EditText)findViewById(R.id.editText);  //kullanici adını alan nesne
    final EditText et2=(EditText)findViewById(R.id.editText3);  //şifrenin girildiği nesne
    final TextView tw3=(TextView)findViewById(R.id.TextView3);  //Mesaj yazacak olan nesne
    Button dugme=(Button)findViewById(R.id.button);
    dugme.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String adi=et1.getText().toString();
            String sifre=et2.getText().toString();

            try{
                InputStream is=getAssets().open("kullanicilar.txt");
                int buyukluk=is.available();
                byte[] okunacak=new byte[buyukluk];
                BufferedReader satir=new BufferedReader(new InputStreamReader(is));
                String gelensatir=satir.readLine();  //ilk satırı oku
                int kontrol=0;   //gerişin başarılı olup olmadığını tutacak değişken
                while(gelensatir!=null){
                    String kelimeler[]=gelensatir.split("\t");  //gelen satırı tab (sekme)ye göre bölüyor.
                    //birinci kelimer[0] a ikinci kelimeler[1] değişkenine aktarılıyor.

                    if(kelimeler[0].equals(adi)&&kelimeler[1].equals(sifre))
                    {
                        kontrol=1;   //döngü sırasında doğru bir kullanıcıadı-şifre eşlemesi ile karşılaştı
                    }
                    gelensatir=satir.readLine();   //sonraki satırı oku
                }
                if (kontrol==1){   //döngü boyunca hiç eşleşme oldu mu?
                    tw3.setText("HOŞGELDİNİZ");
                }else{
                    tw3.setText("KULLANICI ADI VEYA ŞİFRE YANLIŞ");
                }
            }catch (IOException e){
                e.printStackTrace();
            }
        }
    });

}
}
```

6. PROJE DERLEME VE YAYINA KOYMA

6.1. Proje Derleme

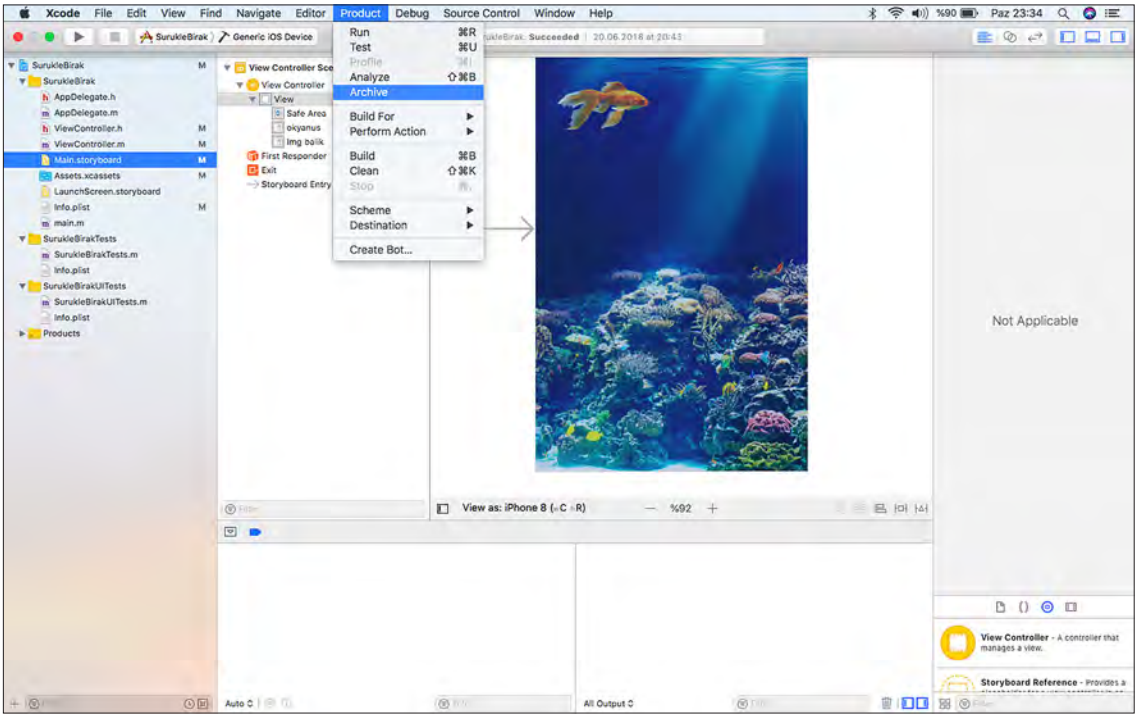
6.1.1. Xcode Proje Derleme

Geliştirilen uygulamanın AppStore'a yüklenmesi için öncelikle paketlenmesi gerekmektedir. Paketleme işlemi derleme ile başlayarak, yeni bir versiyon olarak AppStore'a yüklenmesi ile son bulur. Bu işlem için geliştiricinin bir "developer" hesabına sahip olması gerekmektedir. Derleme, Xcode ortamında menülerden yapılmaktadır. Öncelikle Xcode uygulama projesi açılır. Proje genel ayarlar bölümünden Developer hesap bilgileri seçilir.



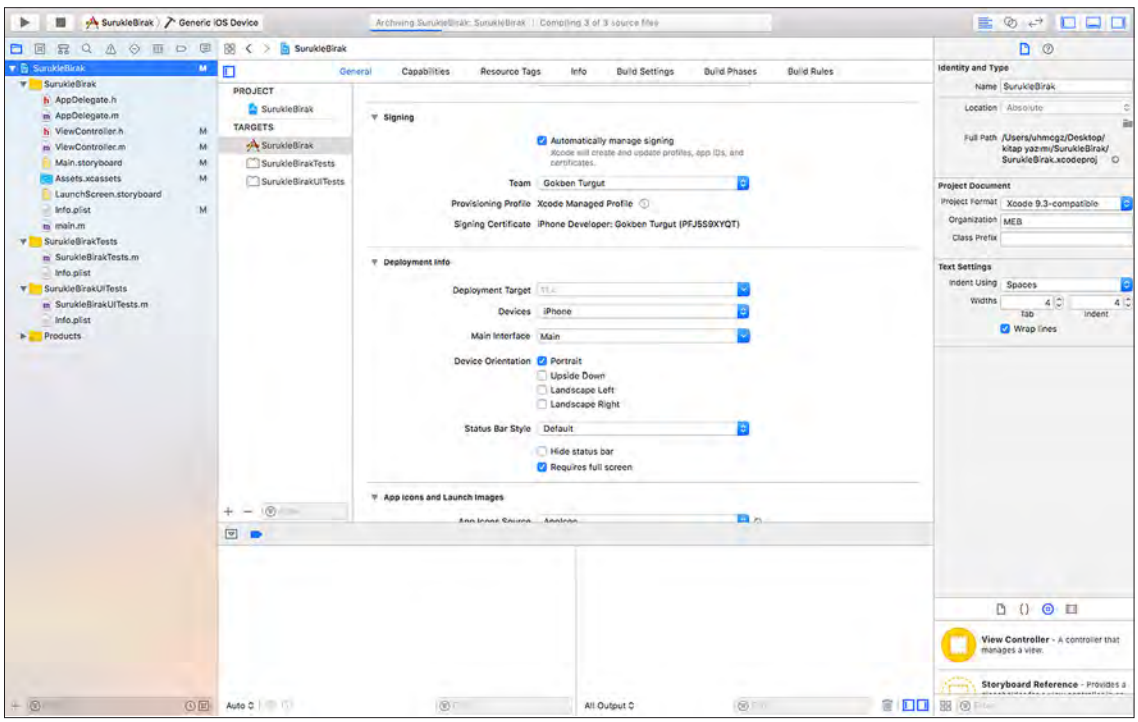
Ekran Görüntüsü 6.1

Ardından Product menüsünden Archive seçeneğine tıklanır.



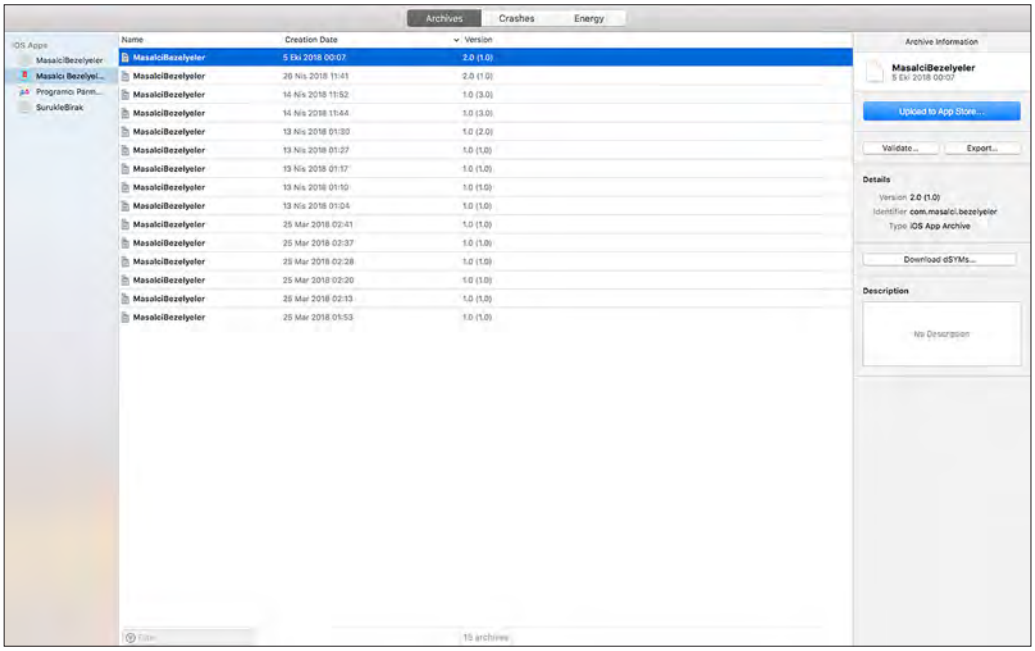
Ekran Görüntüsü 6.2

Proje compile edilmeye (derlenmeye) başlar. Bu işlem dosyanın büyüklüğüne göre biraz zaman alabilmektedir.



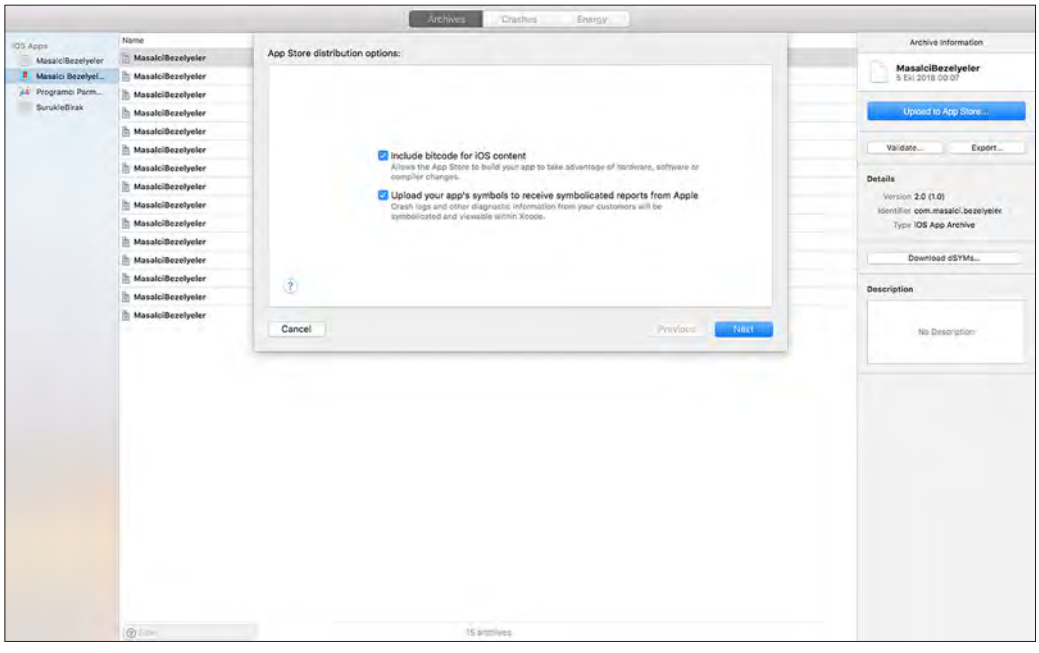
Ekran Görüntüsü 6.3

Derleme sırasında projede herhangi bir sorun yoksa bu işlem aşağıdaki pencerede olduğu gibi sonlanacaktır.

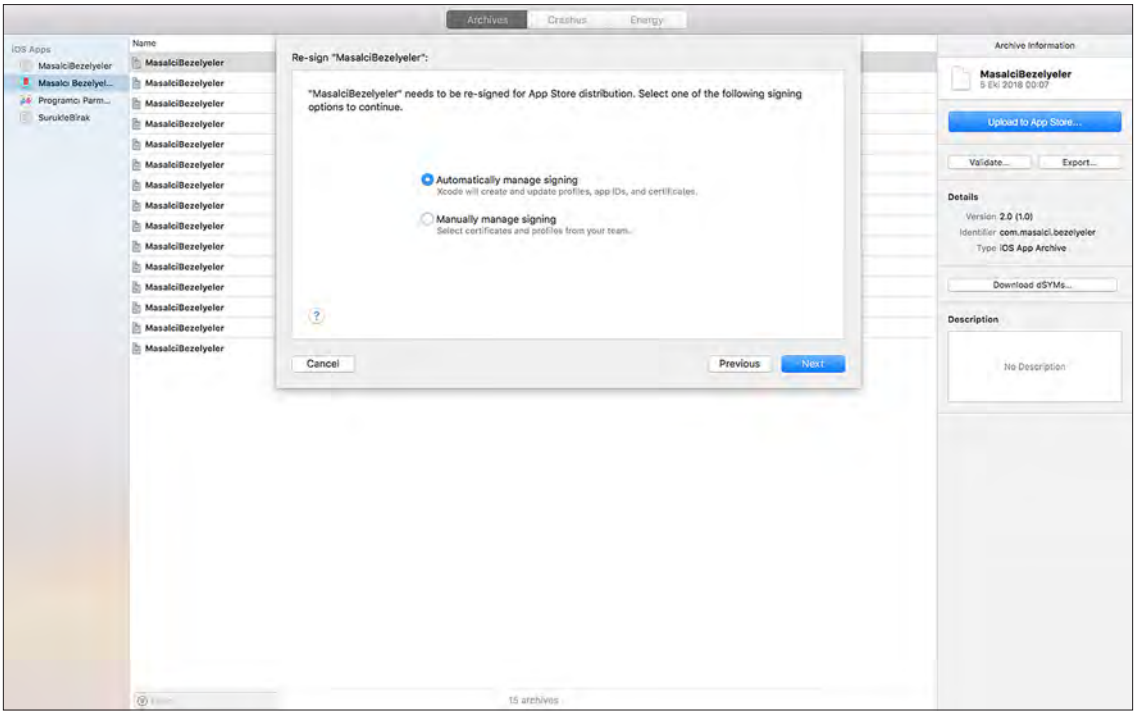


Ekran Görüntüsü 6.4

Paket haline getirilen bu dosyanın AppStore'a bir versiyon olarak yüklenmesi için sağ bölümdeki "AppStore'a Yükle" butonuna tıklanır. Karşılaşılan pencerelerde "Next" tıklanarak ilerlenir ve işlem bu şekilde tamamlanır.

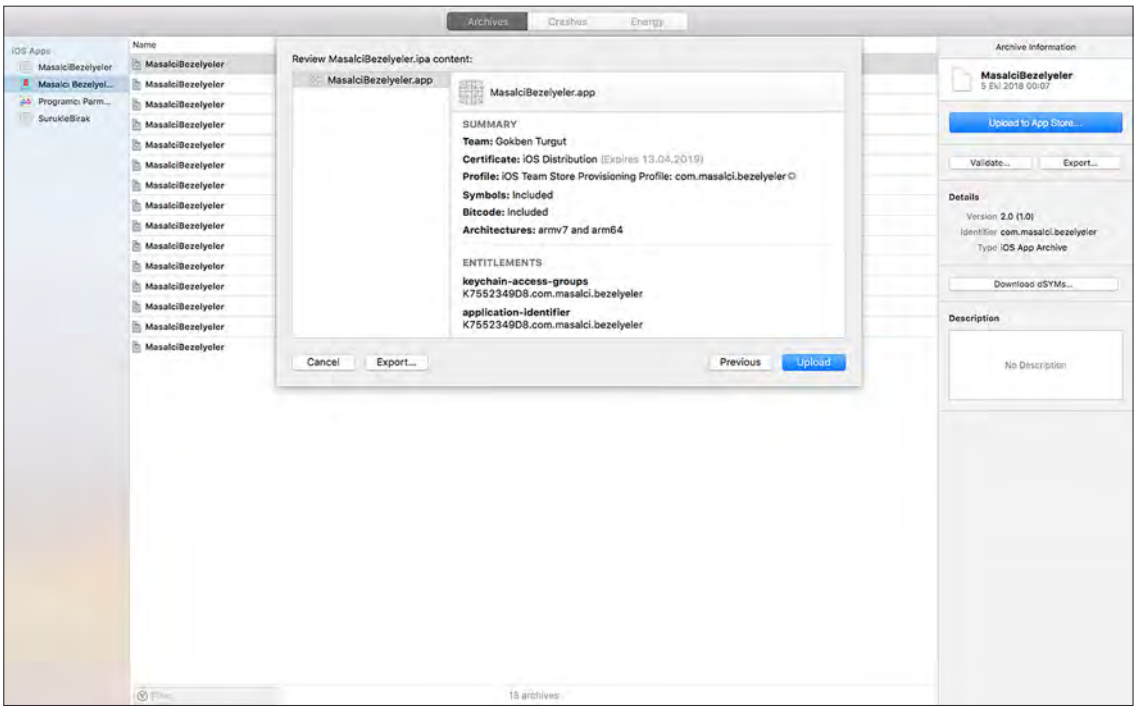


Ekran Görüntüsü 6.5

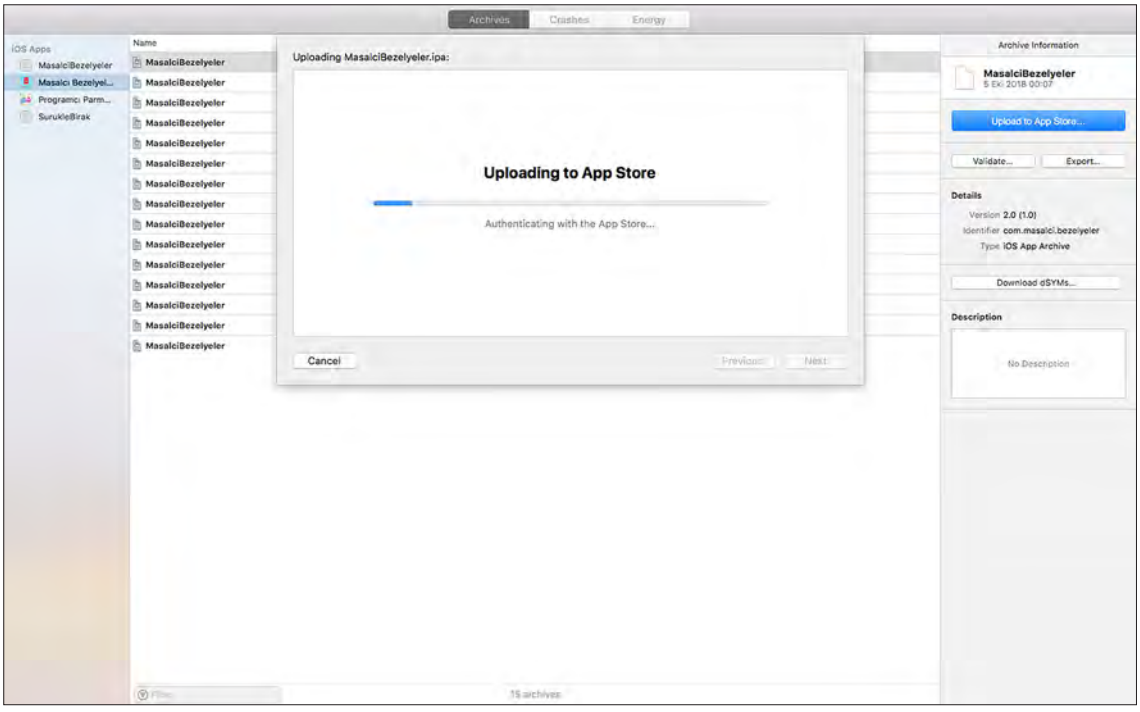


Ekran Görüntüsü 6.6

Ekran Görüntüsü 6.7



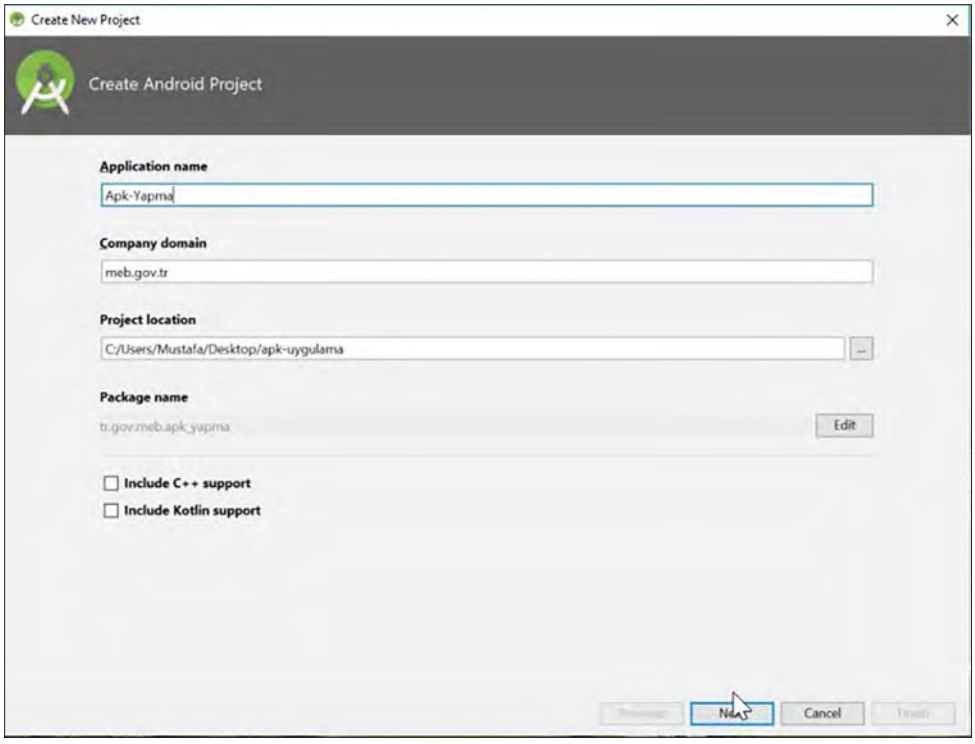
Ekran Görüntüsü 6.8



Ekran Görüntüsü 6.9

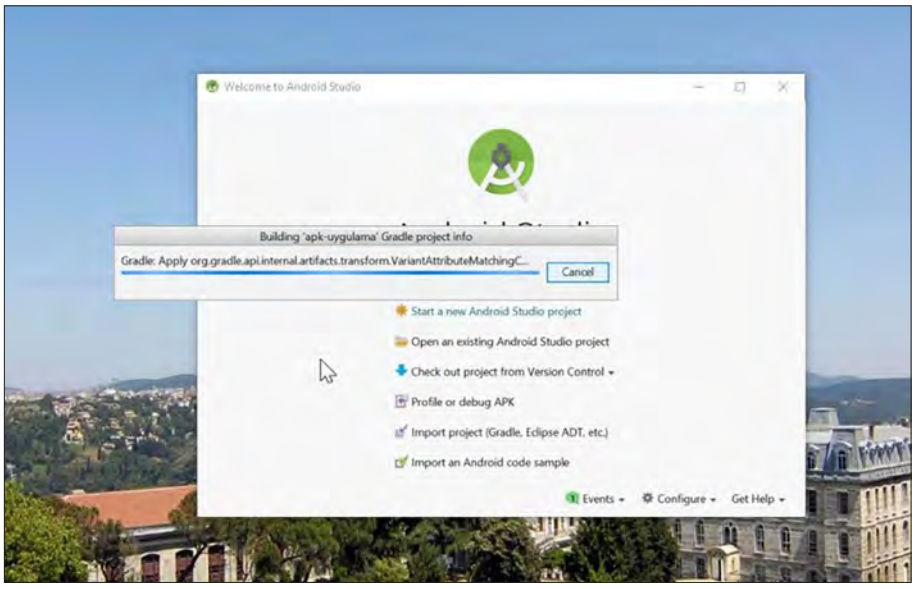
6.1.2. Android Studio Proje Derleme

Bu uygulamada, Android projesini paketleme (.apk dosyası haline getirme) işlemi yapılacaktır. Bunun için ilk olarak 'apk uygulama' isimli bir klasör oluşturunuz. Ardından Android Studio'yu açınız. Yeni bir proje oluşturunuz ve adını 'Apk Yapma' veriniz.



Ekran Görüntüsü 6.7

Projeyi, oluşturduğunuz apk uygulama klasörüne atınız. Basic bir activity oluşturunuz.

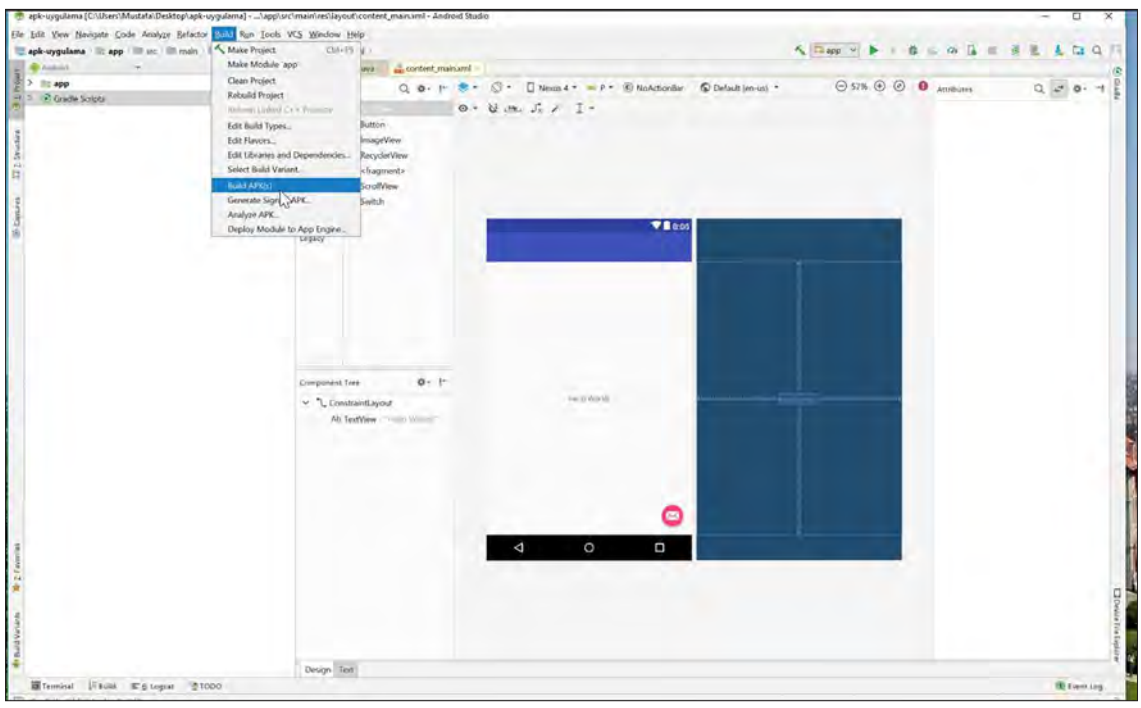


Ekran Görüntüsü 6.8

İki tür apk vardır. Birincisi doğrudan "build apk" yöntemi ile yapılmaktadır. Bu yöntemde apk'nızı herhangi bir Andoid cihaza atıp çalıştırabilirsiniz. İkinci yöntem, "imzalı (signed) apk" yöntemidir. İmzalı apk yöntemi, uygulamanızı Google PlayStore'da yayınlamak için daha doğru bir yöntemdir. Uygulamalarınız Google PlayStore'da yayınlamayacaksanız, build apk yöntemini tercih edebilirsiniz.

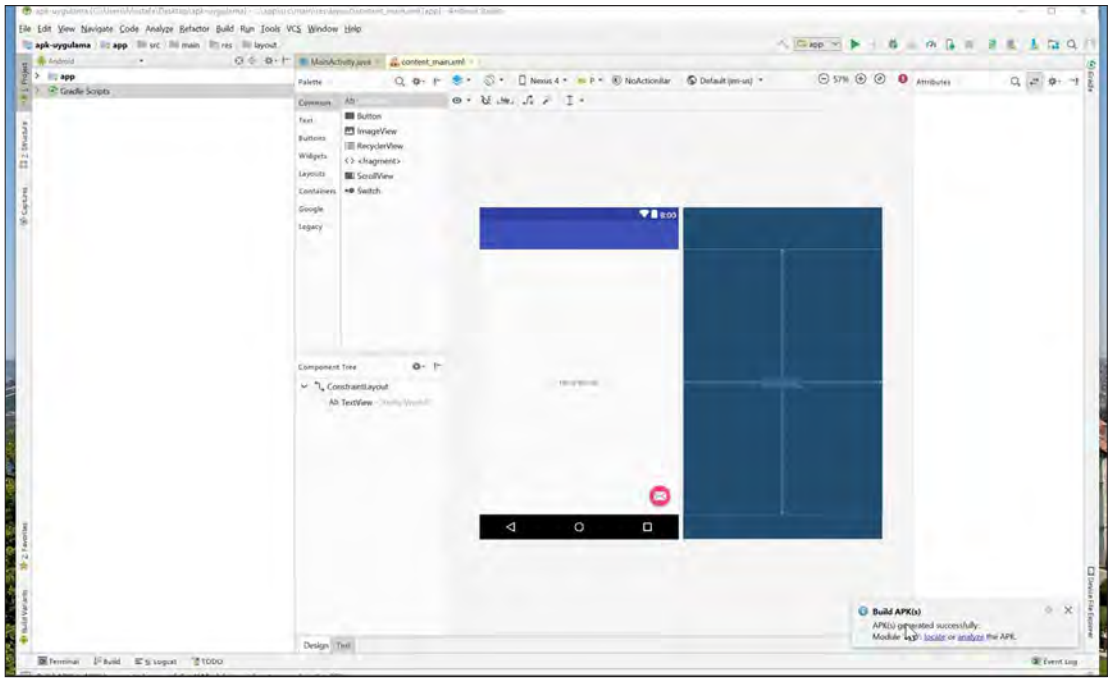
Apk yapma işlemi, Hello World uygulaması üzerinden yapılacaktır.

Build menüsünden Build Apk düğmesine bastığınızda dosyanızın apk'sı oluşturulacaktır.



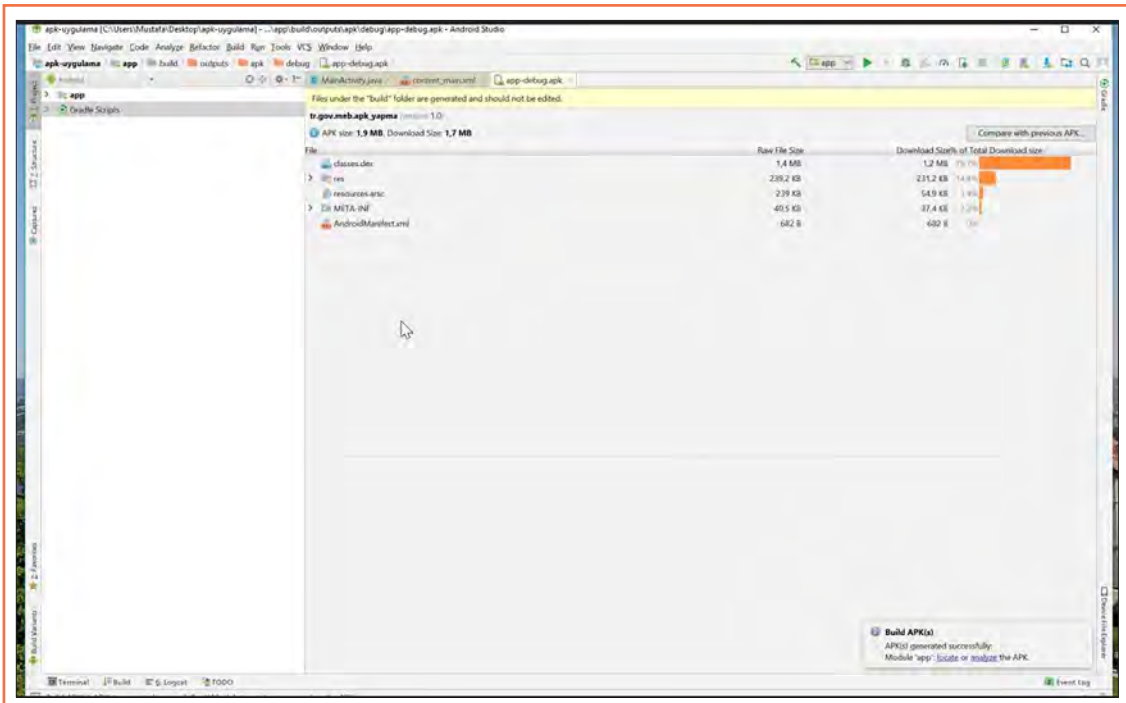
Ekran Görüntüsü 6.9

Build ettikten sonra apk oluşturulacak ve aşağıdaki görüntü ekrana gelecektir.



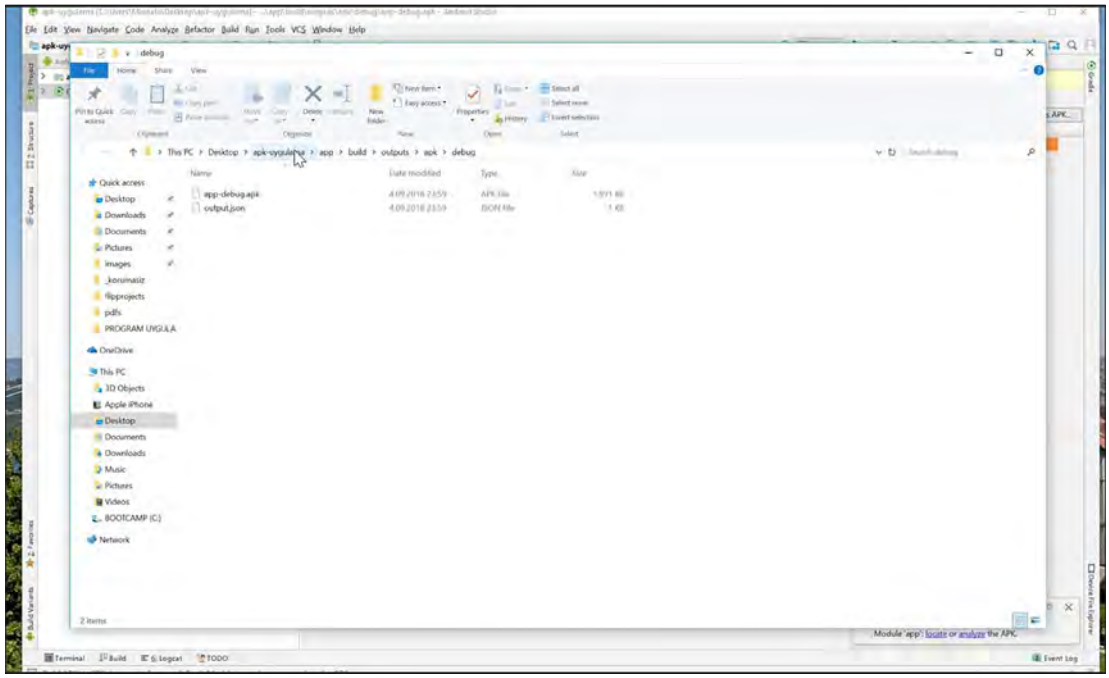
Ekran Görüntüsü 6.10

Analyse düğmesine bastığınızda apk dosyanızın bilgilerini görebilirsiniz.



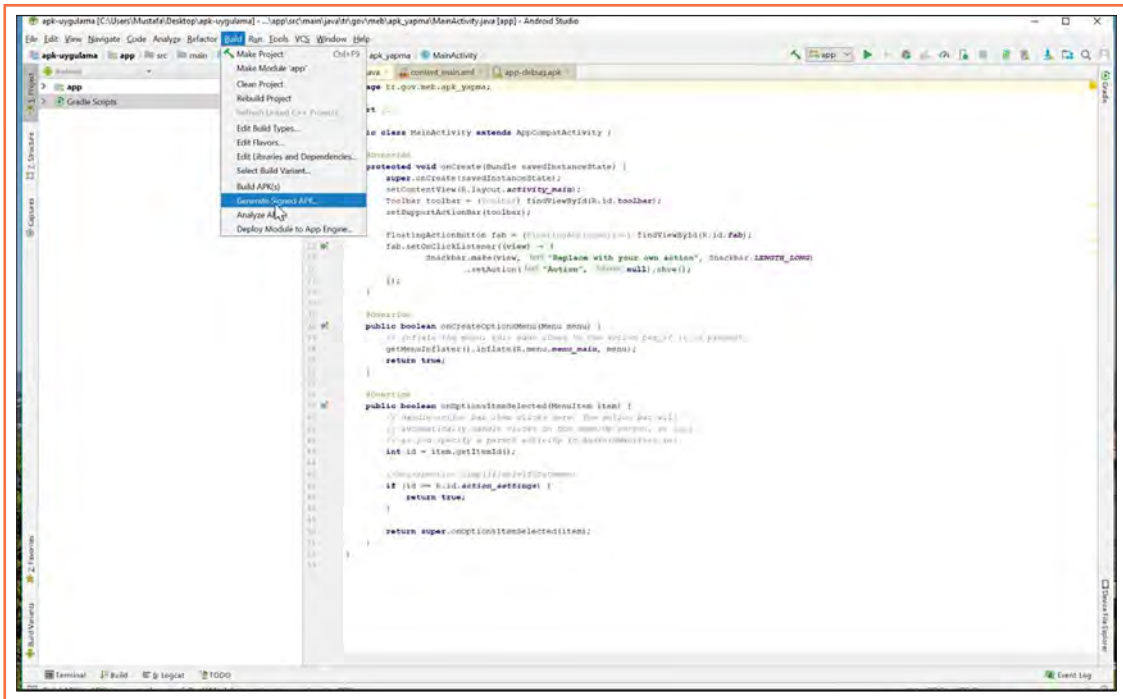
Ekran Görüntüsü 6.11

Locate düğmesine bastığınızda oluşturulan klasör açılmaktadır.



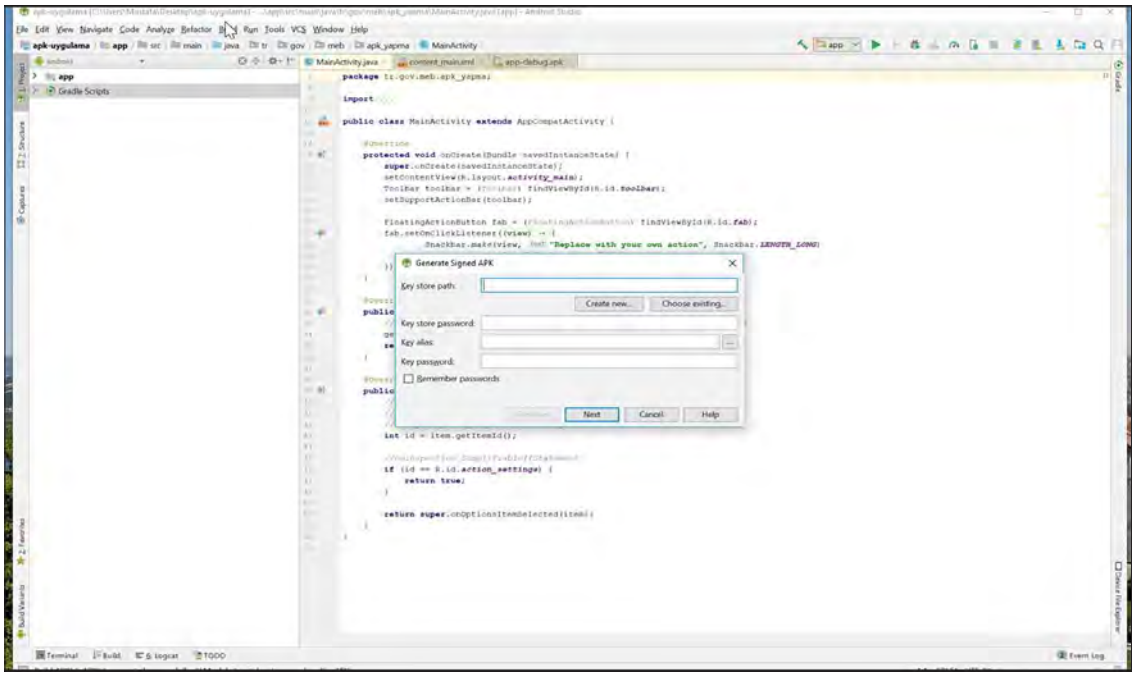
Ekran Görüntüsü 6.12

Bu apk dosyanızı, herhangi bir Android cihaza atıp çalıştırabilirsiniz. Fakat uygulamanızı Google PlayStore'da yayınlamak isterseniz, Build menüsünden "Generate Signed apk" seçiniz.



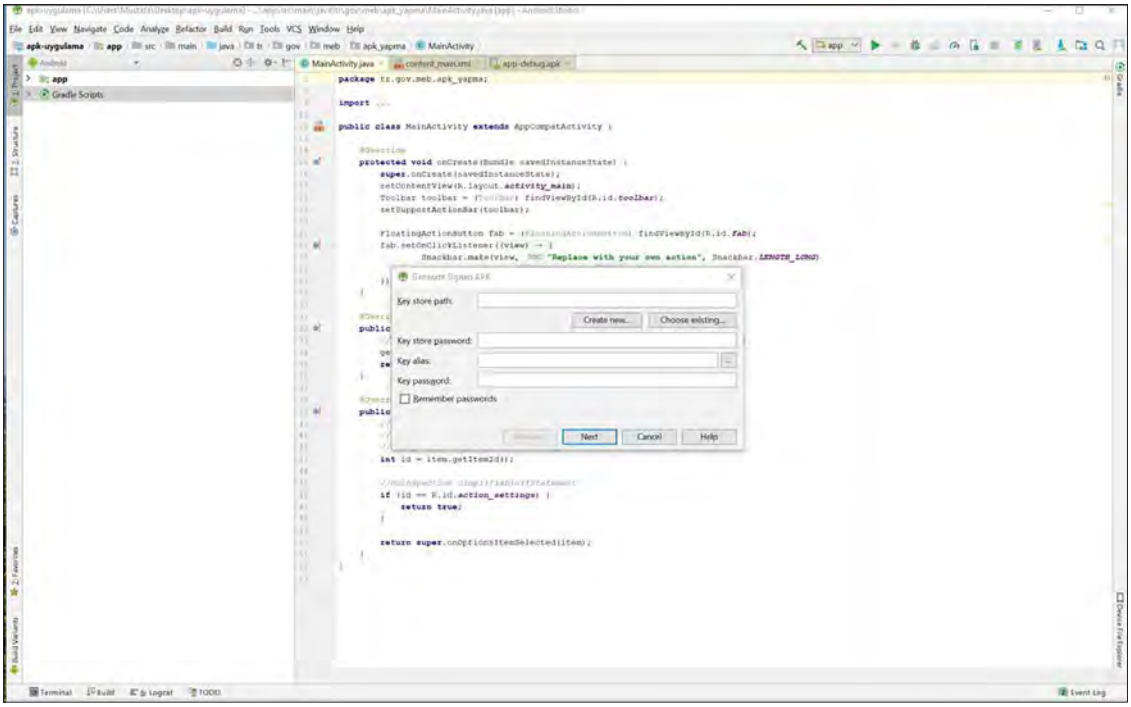
Ekran Görüntüsü 6.13

Build menüsünden Generate Signed apk seçtiğinizde, program sizden bir key (anahtar) oluşturmanızı isteyecektir. Eğer daha önceden kullandığınız bir key varsa "Choose Existing" düğmesine basarak bu key dosyasını çekebilirsiniz.



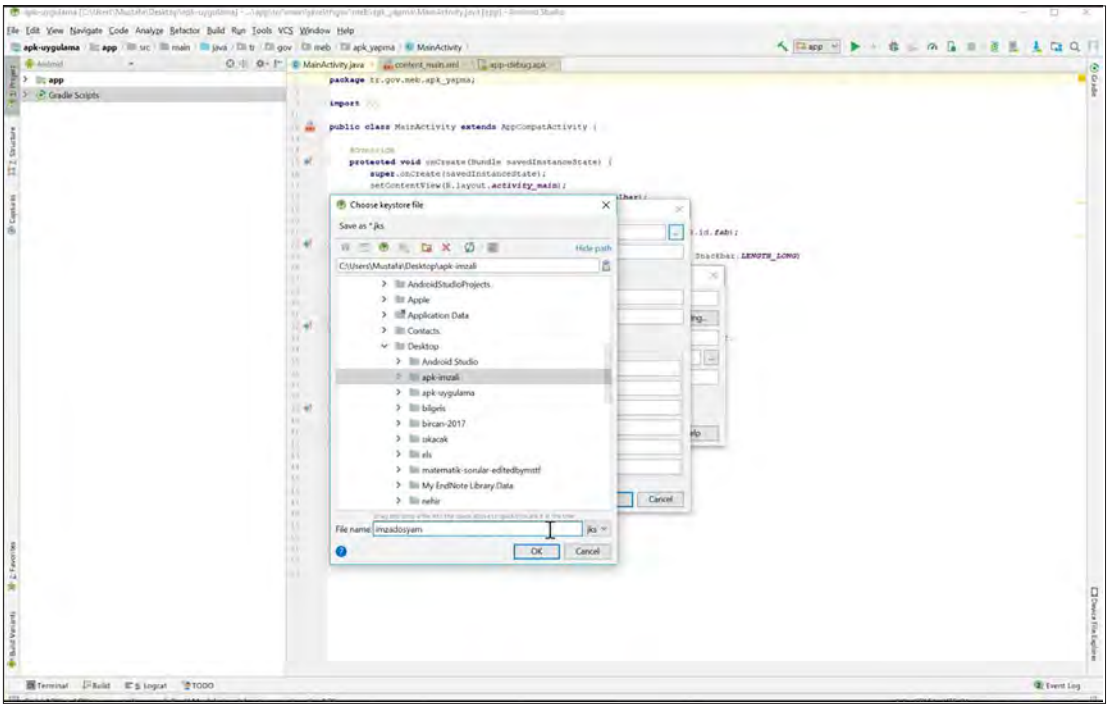
Ekran Görüntüsü 6.14

Şimdi "apkimzali" isminde bir klasör oluşturunuz. İmza oluşturmak için de "Create New" düğmesine basınız.

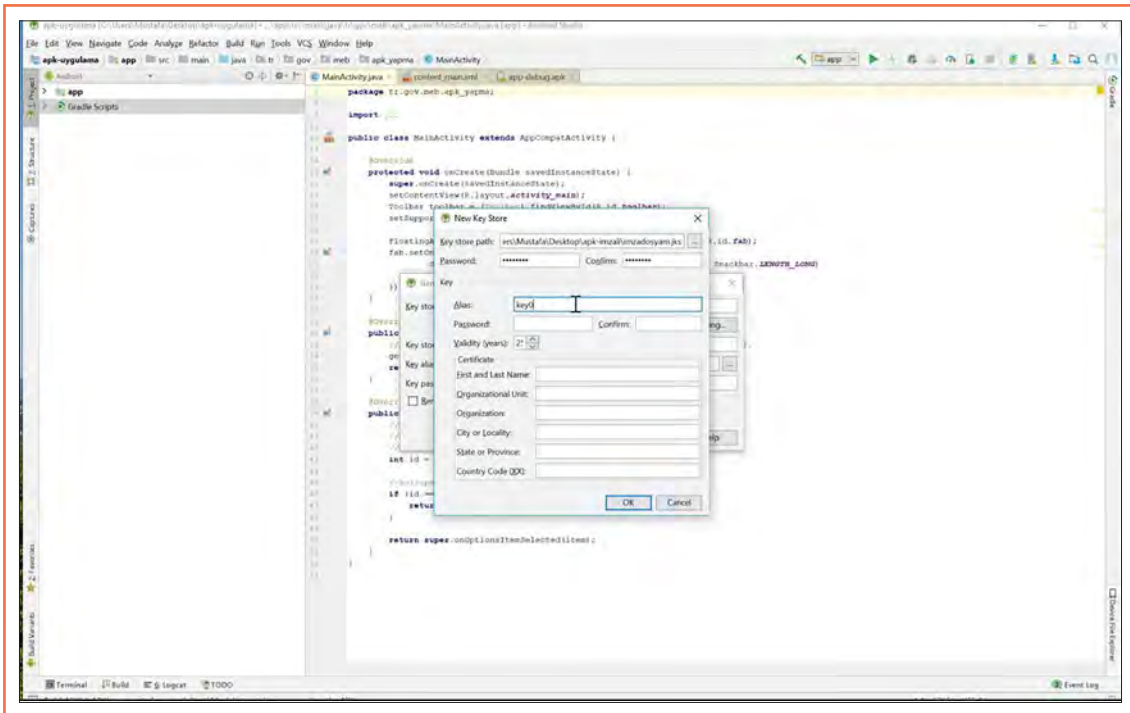


Ekran Görüntüsü 6.15

İmzanızı, oluşturduğunuz "apkimzali" klasörünün içine imzadosyam ismi ile kaydediniz ve şifresini oluşturunuz. Şifre en az altı karakterden oluşmalıdır.

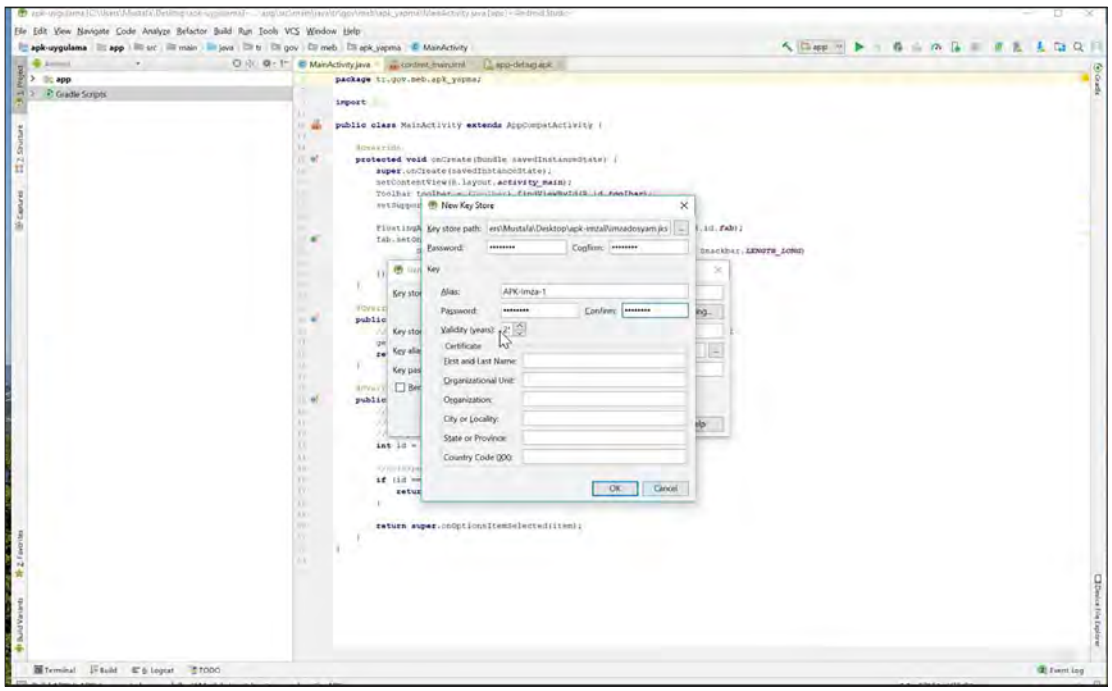


Ekran Görüntüsü 6.16



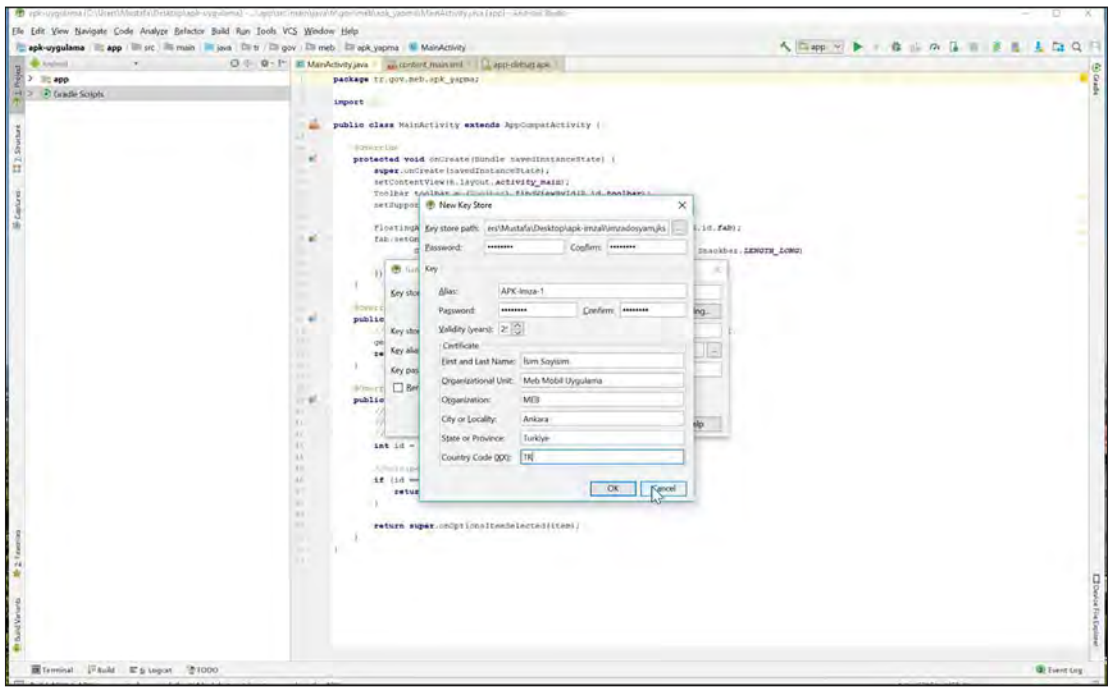
Ekran Görüntüsü 6.17

İmza dosyanızın içinde birden fazla imza olabilmekte, aynı dosyanın içinde birden fazla imza bulunabilmektedir. Aşağıda, bir imza dosyasının içinde APK İmza 1 isminde bir key bulunmaktadır.



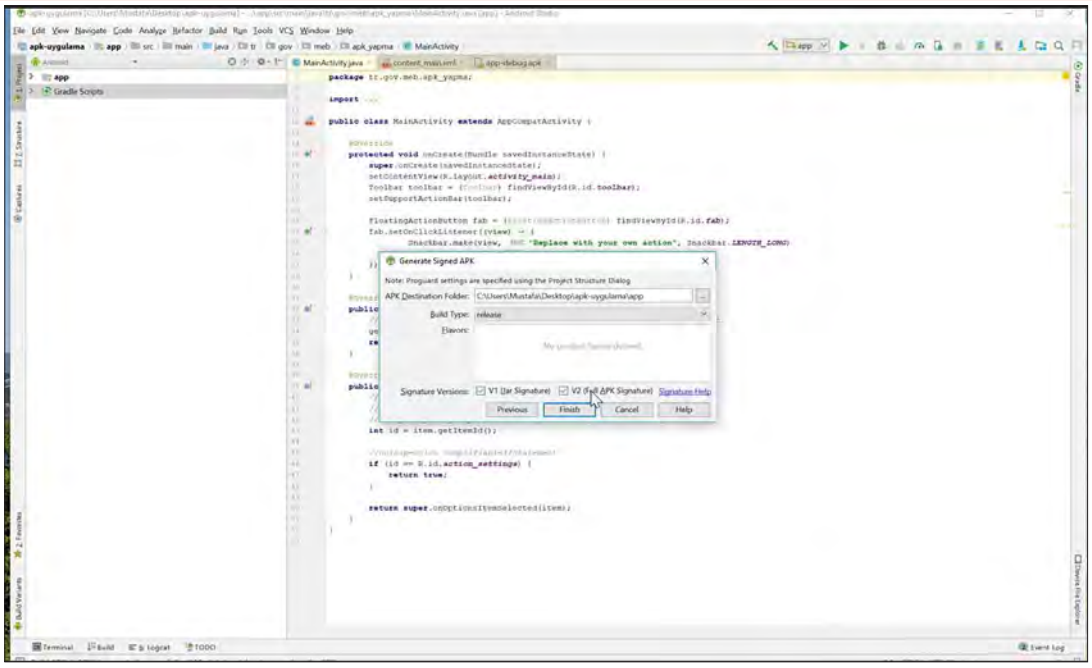
Ekran Görüntüsü 6.18

İmza dosyanızın içine, aşağıdaki gibi gereken bilgileri giriniz. İmza dosyanız, jks uzantılı olarak kaydedilecektir.



Ekran Görüntüsü 6.19

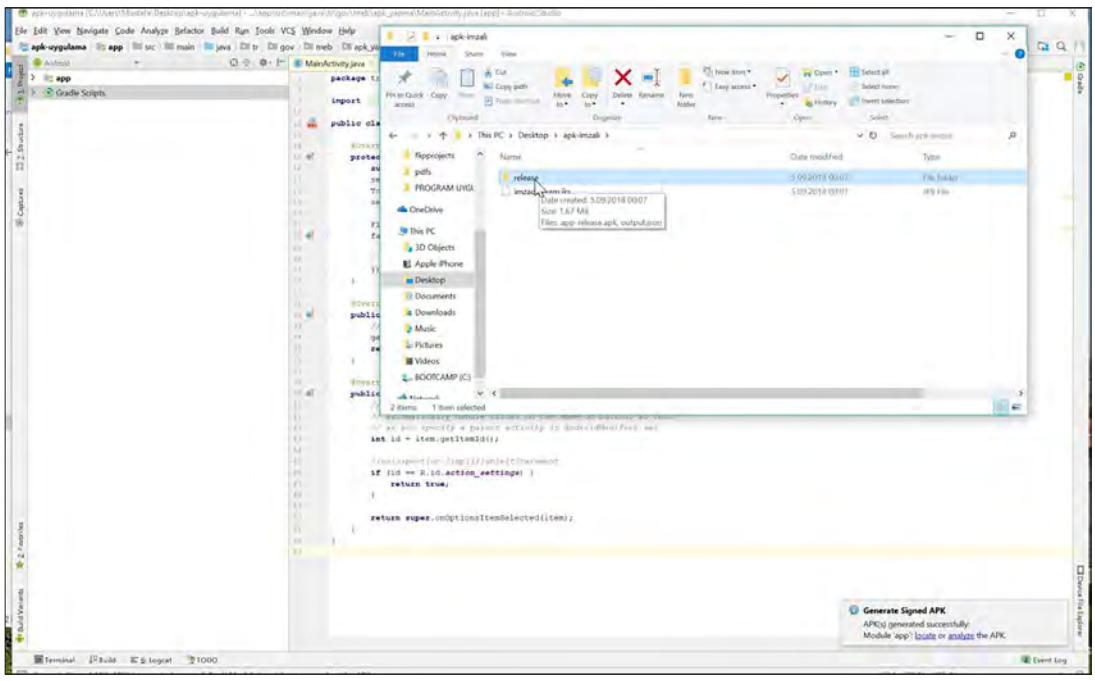
Versiyon 2 full apk imzası veya Versiyon 2 sadece jar imzası seçebilirsiniz. İkisini birden seçerseniz, iki imzayı da çalıştıran bir Android versiyonunda uygulamanızı çalıştırabilirsiniz.



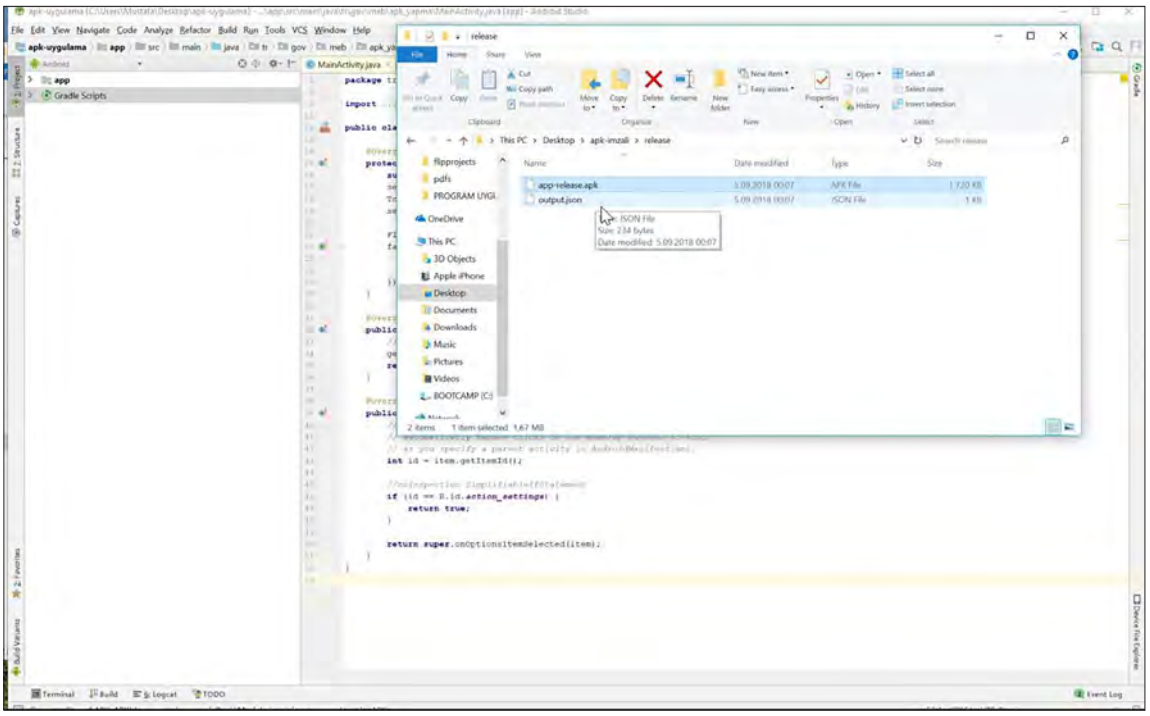
Ekran Görüntüsü 6.20

Bu işlemlerin ardından apk versiyonları oluşacaktır. Apk'ların her birinin bir versiyonu vardır ve yaklaşık olarak iki milyona kadar apk versiyonu yapılabilmektedir. Yani aynı uygulamanın bir kere daha apk'sını yaptığınızda, bu apk artık Versiyon 2 olarak kaydedilmektedir.

Şimdi apk imzali klasörünüzün içine release isimli bir klasör oluşturulmuştur. Burada apk'nın kendisi build edilmiş ve imza dosyası da oluşturulmuştur.



Ekran Görüntüsü 6.21



Ekran Görüntüsü 6.22

Paketleme işlemi tamamlanmıştır. Artık dilerseniz apk dosyanızı, belli bir ücret karşılığında Google PlayStore'a yükleyebilirsiniz yada ücretsiz olarak bir android cihaza kaydedip direk olarak çalıştırabilirsiniz.

6.2. Proje Yayınlama

6.2.1. App Store'da Proje Yayınlama

Xcode ortamında geliştirilen bir uygulamanın AppStore'da yayınlanabilmesi için Apple Developer hesabına sahip olmak gerekmektedir. Yıllık üyelik ücreti olan 99 $ ödenerek www.developer.apple.com sitesinden alınmaktadır. Geliştirici hesabının alınması ile üç çeşit sertifikaya sahip olunur: geliştirici sertifikası, yayınlayıcı sertifikası ve dağıtıcı sertifikası. Bu sertifikaların her biri ile projenin cihazlara (iPhone, iPad gibi) bilgisayara bağlantı kablosu ile yüklenmesi, projenin derlenerek paketlenmesi, AppStore'a yüklenmesi ve AppStore'da yayınlanması sağlanabilmektedir. Geliştirilen uygulamanın AppStore'a yüklenme işlemlerinin tümü Xcode ortamında çeşitli menü komutları kullanılarak yazılımın paketlenmesi ve developer.apple.com sitesi üzerinden çeşitli yönergelerle yayınlanması şeklinde yapılmaktadır. Yüklenen uygulama bir Apple geliştirici ekibi tarafından incelenerek en geç bir hafta içerisinde dönüş sağlanmaktadır. Eğer düzeltmelere ihtiyaç varsa, hataların düzeltilip uygulamanın tekrar yüklenmesi ile kısa bir zaman içerisinde AppStore'da uygulama yayınlanmaya başlayacaktır.

6.2.2. Google Play Store'da Proje Yayınlama

Derlenen uygulamaların Google Play Store'a yüklenebilmesi için bir Google hesabına ihtiyaç vardır. Bu hesapla Google Play Store'a belli bir ücret karşılığında üye olunması gerekmektedir. Uygulamanın sisteme yüklenmesi için Google Play Developer Console kullanılmaktadır. Hesabınızdan buraya giriş yaparak aşamaları takip edip uygulamalarınızı yayına koyabilirsiniz.

6.3. Proje Oluşturma Örnekleri

Mobil programlama modülü içinde anlatılan temel mobil programlama bilgilerinizi kullanarak aşağıdaki uygulamalardan birinde proje oluşturunuz. Projenizde Xcode ya da Android Studio'dan birini kullanabilirsiniz. Aşağıda özellikleri verilen uygulamalara sizler de eklemeler yapabilir, çalışmalarınızı daha profesyonel hale getirebilirsiniz. (Örneğin uyarıları sesli olarak verebilirsiniz.)

1. **Hesap Makinesi Mobil Uygulaması**

   a. Uygulamanızda 1'den 9'a kadar rakamlar ve Matematiksel işlem düğmeleri (+ - * /) tasarlayınız.

   b. Yazılan sayıları ve işlem sonuçlarını gösterebileceğiniz bir görüntüleme nesnesi koymayı unutmayınız.

   c. Her bir düğmeye tıklandığında yapılacak olanları ve ekrandaki değişimleri düşünün ve ilgili kod blokları içerisine kodlayınız.

2. **Resim Galerisi**

   a. En az 10 resimden oluşan bir galeri tasarlayınız.

   b. Resimler öncelikle küçük halleriyle ekranda sıralansın.

   c. Ardından kullanıcı üzerlerine tıkladığında resim ekranı kaplasın.

   d. Kullanıcı büyüttüğü resme tekrar tıkladığında resim tekrar küçülsün.

3. **Balon Patlatma Oyunu:**

   a. Resim galerisi uygulamasındaki gibi ekrana en az 10 tane balon belirsin.

   b. Ancak bu sefer balonlar belirli bir sürede ve sırayla gelsin.

   c. Üzerine tıklanılan balon resmi yok olsun ve puan tabelasındaki sayı (0'dan başlayacak) her seferinde 1 artsın.

   d. Eğer bir balon resmi belirdikten sonra 5 saniye içerisinde tıklanmazsa kaybolsun.

4. **Kelime Oyunu**

   a. Uygulama açıldığında 6 harfli kelimelerden oluşan bir kelime dizisinden herhangi bir kelime seçsin.

   b. Ekranda oluşturacağınız 6 düğmenin üzerine seçilen kelimenin harfleri karışık olarak yazılsın.

   c. Kullanıcı düğmelere tıkladığında tıklanılan harf yukarıdaki tabelada sıradaki boşluğa yazılsın.

   d. Düğmeye tıklandıktan sonra eğer tabeladaki kelime doğruysa kullanıcıya tebrik mesajı belirsin.

5. **Kavonazdaki Top Uygulaması**

   a. Hareket sensörleri ile yapacağınız bu uygulamada ekranda içi şeffaf bir kavanoz resmi olsun.

   b. Kavonozun içerisinde de küçük bir top resmi olsun.

   c. Telefon hareket ettirildiğinde top da sağa sola, yukarı aşağı kavanozun içerisinde harekete uygun olarak yer değiştirsin.

PROJE DEĞERLENDİRME ÇİZELGESİ

Uygulama Adı:

Öğrenci Adı Soyadı:

Sınıfı/ No::

KRİTER	PUAN	ALINAN PUAN
TASARIM		
Kullanılması gereken nesneler eklenmiş mi?	5	
Fazladan nesne var mı?	-5	
Nesne özellikleri doğru tanımlanmış mı?	5	
Modüler (sonradan kullanılabilir, dönüştürülebilir) bir tasarım yapılmış mı?	5	
Amaca uygun tasarım modeli (Layout) seçilmiş mi?	5	
TOPLAM	20	
KODLAMA		
Doğru kütüphaneler yüklenmiş mi (tanımlanmış mı)?	5	
Kodlar doğru yazılmış ve çalışıyor mu?	15	
Gereksiz kod satırı var mı?	-5	
Değişken isimleri anlaşılır mı?	5	
Değişkenler ve nesneler doğru formatta tanıtılmış mı?	5	
Metot, sınıf, fonksiyon adları doğru formatta oluşturulmuş mu?	5	
Değişken türleri tutacak oldukları değerle ilişkili mi?	5	
En az sayıda (minimum) kod satırı ile en kısa ve doğru çözüme ulaşılabilmiş mi? (Verimlilik)	10	
TOPLAM	50	
GENEL		
Program doğru çalışıyor mu?	10	
Mantık hatası var mı?	-10	
Yazım (syntax) hatası var mı?	-10	
Kodlar arasına açıklama notları eklenmiş mi?	5	
Dosya doğru derlenebilmiş (build) mi?	5	
APK paketi imzalı bir şekilde oluşturulabilmiş mi?	5	
İmza jpk dosyası oluşturulabilmiş mi?	5	
TOPLAM	30	
GENEL TOPLAM	100	

Öğretmen Adı Soyadı:

İmza:

Tarih:

III. BÖLÜM KAYNAKÇA

https://www.digit.in/technology-guides/fasttrack-to-mobile-and-laptop-hardware/unity-in-diversity.html

https://ademocut.com/cep-telefonu-hangi-donanimlardan-olusur/

https://www.notebookcheck.net/Smartphone-Processors-Benchmark-List.149513.0.html

https://www.androidpit.com/what-is-android

https://www.techopedia.com/definition/14873/android-os

https://developer.android.com/

http://www.iztim.com/Blog/YazilimTeknolojisi/ANDROID

https://www.telegraph.co.uk/technology/2017/10/09/microsoft-finally-killing-windows-phone/

https://developer.apple.com/

https://www.newgenapps.com/blog/bid/219838/10-steps-to-create-a-successful-mobile-application

https://gelecegiyazanlar.turkcell.com.tr/konu/android/egitim/android-401/androidde-sensor-kullanimi

https://www.gc.cuny.edu/CUNY_GC/media/Computer-Science/Student%20Presentations/Xing%20Su/Second_Exam_Slides_Xing_Su_6-12-2014.pdf

https://acikders.ankara.edu.tr/pluginfile.php/12514/mod_resource/content/1/11%20Androidde%20sensorler.pdf