

# İÇİNDEKİLER

## I. BÖLÜM

1. ROBOT VE ROBOT MİMARİSİ.....	16
1.1. Robot ve Robot Mimarisi .....	17
1.2. Robot Kontrol Yöntemleri .....	17
1.3. Robot Mimarisinde İlkeler .....	17
1.4. Düşünelim / Tartışalım.....	20
1.5. Değerlendirme Soruları.....	20
2. ROBOT TÜRLERİ VE EĞİTSEL AMAÇLI ROBOTLAR.....	23
2.1. Robot Türleri ve Eğitsel Amaçlı Robotlar.....	24
2.2. Kullanılan Uygulama Alanlarına Göre Robotlar.....	24
2.3. Hareket Mekaniğine Göre Robotlar.....	28
2.4. Eğitsel Amaçlı Robotlar .....	34
2.5. Düşünelim / Araştıralım .....	37
2.6. Değerlendirme Soruları.....	38
3. EĞİTSEL ROBOTTA MEKANİK BİLEŞENLER .....	40
3.1. Eğitsel Robotta Mekanik Bileşenler.....	41
3.2. Yapısal Bileşenler (Gövde, İskelet).....	41
3.2.1. Yapısal Bileşenlerin Görevleri.....	42
3.3. Montaj Bileşenleri (Bağlantı Parçaları) .....	42
3.3.1. Montaj Bileşenlerinin (Bağlantı Parçaları) Görevleri .....	43
3.4. Mekanik Hareket/Eylem Bileşenleri (Tekerler, Paletler, Ayaklar) .....	43
3.4.1. Mekanik Hareket/Eylem Bileşenlerinin (Tekerler, Paletler, Ayaklar) Görevleri .....	43
3.5. Düşünelim / Araştıralım .....	43
3.6. Değerlendirme Soruları.....	44
4. EĞİTSEL ROBOTTA MEKANİK BİLEŞENLER .....	46
4.1. Eğitsel Robotta Elektromekanik Bileşenler.....	47
4.2. Bağlantı Bileşenleri (Butonlar, Anahtarlar, Konektörler ve Klemensler).....	47
4.2.1. Bağlantı Bileşenlerinin (Butonlar, Anahtarlar ve Konektörler) Görevleri .....	48
4.3. Güç Bileşenleri (Pil, Akümülatör, Batarya).....	48
4.3.1. Güç Bileşenlerinin (Pil, Akümülatör, Batarya) Görevleri.....	49
4.4. Hareket Bileşenleri (Doğru Akım -DC-, Servo ve Adım Motorlar) .....	49
4.4.1. Hareket Bileşenlerinin (Doğru Akım -DC-, Servo ve Adım Motorlar) Görevleri ...	51
4.5. Düşünelim / Tartışalım.....	51
4.6. Değerlendirme Soruları.....	51
5. EĞİTSEL ROBOTTA ELEKTRONİK BİLEŞENLER .....	54
5.1 Eğitsel Robotta Elektronik Bileşenler .....	55
5.2. Motor Sürücü Kartları ve Görevleri .....	55

5.3. USB-UART Çeviriciler ve Görevleri .....	55
5.4. Kablosuz İletişim Bileşenleri ve Görevleri.....	56
5.5. Robotik Uygulamalarda Kullanılan Algılayıcılar (Sensörler) .....	57
5.5.1. Robotik Algılayıcı Türleri .....	57
5.5.2. Yaygın Kullanılan Robotik Algılayıcılar ve Görevleri.....	58
5.5.3. Aktif Algılayıcılar .....	58
5.5.4. Pasif Algılayıcılar .....	61
5.5.5. Algılayıcıların Mikrodenetleyici Kartlara Haberleşmesi / Bağlanması .....	64
5.6. Robotik Programlamada Kullanılan İşlemciler .....	65
5.6.1. Robotik Programlamada Kullanılan İşlemcilerinin Görevleri .....	66
5.7. Mikrodenetleyici Kartlar (Geliştirme Kartları) ve Görevleri.....	66
5.8. Mikrodenetleyici Kartlar (Geliştirme Kartları) İçin Kalkanlar (Shields) ve Görevleri .....	66
5.9. Robot Kontrol Kartları .....	67
5.9.1. Robot Kontrol Kartlarının Görevleri.....	67
5.10. Düşünelim / Araştırılma .....	68
5.11. Değerlendirme Soruları.....	68
6. BLOK TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI .....	71
6.1. Blok Tabanlı Robot Programlama Yazılımları ve Ortamları .....	72
6.2. mBlock Grafik Programlama Yazılımı ve Ortamı .....	72
6.3. mBlock Grafik Programlama Yazılımının Yüklenmesi.....	72
6.4. mBlock Programının Arayüzü, Yapısı ve Temel Özellikleri .....	74
6.5. mBlock Yüklü Bilgisayarla Robot Arasında Bağlantının Oluşturulması .....	78
6.6. mBlock Programlama Dilinin Temel Özellikleri ve Programlama Yapısı.....	79
6.6.1. Hareket Alt Başlığı Altında Verilen Komut Blokları.....	79
6.6.2. Görünüm Alt Başlığı Altında Verilen Komut Blokları .....	80
6.6.3. Ses Alt Başlığı Altında Verilen Komut Blokları .....	82
6.6.4. Kalem Alt Başlığı Altında Verilen Komut Blokları .....	84
6.6.5. Veri&Blok Alt Başlığı Altında Verilen Komut Blokları.....	85
6.6.6. Olaylar Alt Başlığı Altında Verilen Komut Blokları.....	92
6.6.7. Kontrol Alt Başlığı Altında Verilen Komut Blokları.....	93
6.6.7.1. Kontrol Örnekleri-Döngüler .....	94
6.6.7.2. Kontrol Örnekleri -Koşullar .....	95
6.6.8. Algılama Alt Başlığı Altında Verilen Komut Blokları .....	97
6.6.9. İşlemler Alt Başlığı Altında Verilen Komut Blokları.....	98
6.6.10. Robotlar Alt Başlığı Altında Verilen Komut Blokları .....	99
6.7. mBlock ile Arduino Kullanımı .....	100
6.8. Robotlar Alt Başlığı Altında Verilen mBot Komut Blokları.....	106
6.8.1. mBlock ile Arduino ve mBot Uyumlu Robot Kullanımı .....	108

6.9. Düşünelim / Araştırılma / Uygulayalım .....	117
6.10. Değerlendirme Soruları.....	118
7. METİN TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI .....	120
7.1. Metin Tabanlı Robot Programlama Yazılımları ve Ortamları.....	121
7.2. Arduino Geliştirme Kartları .....	121
7.3. Arduino UNO Geliştirme Kartı.....	122
7.4. Arduino IDE (Tümleşik Geliştirme Ortamı-Integrated Development Environment).....	124
7.5. Arduino IDE Yazılımının Yüklenmesi.....	124
7.6. Arduino Tümleşik Geliştirme Ortamının Temel Özellikleri.....	128
7.7. Arduino Tümleşik Geliştirme Ortamının Bölümleri .....	128
7.8. Arduino Tümleşik Geliştirme Ortamının “Program” Yapısı.....	129
7.8.1. Kontrol Yapıları.....	130
7.8.2. Söz Dizimi.....	137
7.8.3. Aritmetik Operatörler .....	138
7.8.4. Karşılaştırma Operatörleri.....	138
7.8.5. Boolean Operatörleri .....	139
7.8.6. İşaretçi Operatörler .....	140
7.8.7. Bitsel Operatörler .....	140
7.8.8. Birleşik Operatörler .....	141
7.9. Arduino Tümleşik Geliştirme Ortamının “Değişken” Yapısı.....	143
7.9.1. Sabitler .....	143
7.9.2. Veri Tipleri.....	145
7.9.3. Dönüşümler .....	149
7.9.4. Değişken Kapsamları .....	150
7.9.5. Yardımcılar .....	151
7.10. Arduino Tümleşik Geliştirme Ortamının “Fonksiyon” Yapısı.....	151
7.10.1. Dijital Giriş Çıkışlar .....	151
7.10.2. Analog Giriş Çıkışlar .....	152
7.10.3. Gelişmiş Giriş Çıkışlar .....	154
7.10.4. Gecikmeler .....	156
7.10.5. Matematiksel İşlevler .....	157
7.10.6. Trigonometri İşlevleri .....	159
7.10.7. Karakterler.....	159
7.10.8. Rastgele Sayılar .....	160
7.10.9. Bit ve Bayt’lar .....	161
7.10.10. İnterruptlar (Kesmeler) .....	161
7.10.11. Harici İnterruptlar (Kesmeler) .....	162
7.11. Arduino Tümleşik Geliştirme Ortamında Seri Haberleşme.....	163

7.12. Arduino Tümlşik Geliřtirme Ortamında Seri Haberleřme Protokolleri .....	165
7.12.1. I <sup>2</sup> C Veri Yolu.....	165
7.12.2. SPI Veri Yolu.....	167
7.13. Kütüphaneler.....	170
7.13.1. Arduino Standart Kütüphaneleri.....	171
7.13.2. Arduino Robot Kütüphanesi.....	171
7.14. Düşünelim / Arařtırılım / Uygulayalım .....	172
7.14.1. LED Yakma Uygulaması .....	172
7.14.2. PWM LED Uygulaması .....	173
7.14.3. RGB LED Uygulaması .....	173
7.14.4. Potansiyometre Uygulaması.....	174
7.14.5. Potansiyometre ile PWM LED Uygulaması.....	175
7.14.6. Buton ile LED Yakma Uygulaması .....	175
7.14.7. Iřık Algılayıcı Uygulaması.....	176
7.14.8. Engel Kaçınma Uygulaması .....	176
7.14.9. PIR Algılayıcı Uygulaması .....	177
7.14.10. Sesle LED Kontrol Uygulaması .....	177
7.14.11. Buzzer Uygulaması .....	178
7.14.12. DC Motor Kontrol Uygulaması.....	178
7.14.13. Adım (Step) Motor Kontrol Uygulaması .....	179
7.14.14. Servo Motor Kontrol Uygulaması.....	179
7.14.15. Isı Algılayıcısı Uygulaması .....	180
7.14.16. Nem-Isı Algılayıcısı Uygulaması .....	180
7.14.17. Ultrasonik Algılayıcı Uygulaması .....	181
7.14.18. İvmeölçer Uygulaması .....	181
7.14.19. Açılçer Uygulaması .....	182
7.14.20. Çizgi İzleyen Robot Uygulaması .....	182
7.15. DEĞERLENDİRME SORULARI.....	183
KAYNAKÇA.....	185

## II. BÖLÜM

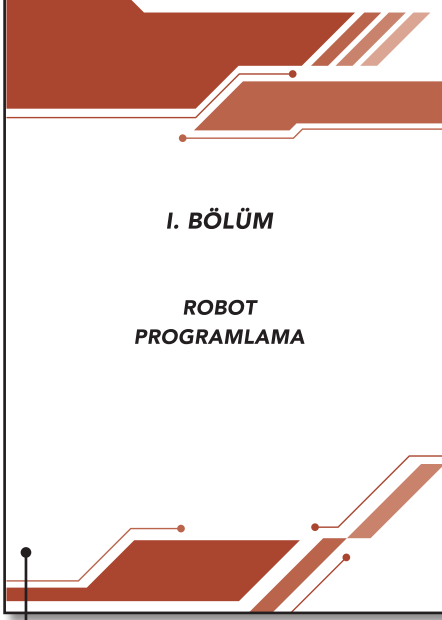
1. İNTERNET VE WEB SERVİSLERİ .....	190
1. İnternet ve Web Servisleri .....	191
Web Tarayıcıları .....	191
Web Teknolojileri .....	192
2. İřARETLEME DİLİNE GİRİř (HTML) .....	194
2. İřaretleme Diline Giriř (HTML).....	195
Ön Hazırlık .....	195

HTML .....	196
Resim Ekleme.....	196
HTML5 .....	198
1. UYGULAMA.....	199
2. UYGULAMA.....	201
ÇOKLU ORTAM İÇERİK EKLEME .....	202
1. UYGULAMA.....	204
2. UYGULAMA.....	199
PROJE ŞABLONUNUN OLUŞTURULMASI.....	199
Site Başlığı ve Dil Kodlaması .....	200
Site Banner'ı ve Gezinim Linklerinin Oluşturulması.....	201
Sitenin Slayt Resmi ve Ana İçerik Bölümünün Oluşturulması .....	202
3. STİL SAYFALARINA GİRİŞ .....	205
3. STİL SAYFALARINA GİRİŞ .....	206
CSS'İN YAZILACAĞI YERE GÖRE KODLAMA YÖNTEMLERİ.....	206
1. HTML Etiketleri İçinde CSS Kodlama.....	206
2. Bir Stil Bloğu (<style></style>) İçerisinde CSS Kodlama .....	207
3. Haricî Bir Stil Dosyası İçinde CSS Kodlama.....	208
ELEMENTİN ETİKETİNE GÖRE CSS KODLAMA.....	208
ELEMENTİN ÖZELLİKLERİNE GÖRE CSS KODLAMA.....	209
İç İçe Geçmiş CSS Kodlama .....	210
PROJE SİTESİNİN CSS İLE GÖRSELLEŞTİRİLMESİ .....	212
Varsayılan Ayarları.....	212
Site Şablonundaki Temel Çerçevelerin Biçimlendirilmesi .....	213
Banner Çerçevesinin Biçimlendirilmesi.....	213
Content Çerçevesinin Biçimlendirilmesi.....	214
Footer Çerçevesinin Biçimlendirilmesi.....	217
4. ETKİLEŞİM.....	219
4. ETKİLEŞİM.....	220
Javascript Kodlama Yöntemleri.....	220
İşlevsel Olmayan Kullanımı .....	220
Head Elementi Arasındaki Kullanımı .....	221
Body Elementi Arasındaki Kullanımı.....	221
Dış Bir Dosyaya Bağlantı Verilerek Kullanımı .....	222
HTML İÇERİĞİ DEĞİŞTİRME .....	222
HTML NİTELİKLERİNİ DEĞİŞTİRME.....	223
Kullanıcı Butona Tıklamadan Önce .....	223
Kullanıcı Butona Tıkladıktan Sonra .....	223

CSS DEĞİŞTİRME .....	224
MANTIKSAL SINAMA.....	225
SWİTCH KONTROL YAPISI .....	226
Tekrarlı (Döngü) Yapılar .....	227
FONKSİYON YAZIMI .....	230
DİZİLER .....	233
BASİT DİZİ İŞLEMLERİ .....	236
Ekleme .....	238
Değiştirme.....	238
NESNELER.....	240
5. VERİ TABANI YÖNETİMİ .....	245
5. VERİ TABANI YÖNETİMİ.....	246
VERİ TABANI .....	246
XAMPP KURULUMU.....	247
MARIADB VERİ TABANI YÖNETİMİ.....	247
Kayıt İşlemleri .....	255
SELECT .....	258
INSERT .....	259
UPDATE .....	259
DELETE .....	260
6. ETKİLEŞİM VE VERİ YÖNETİMİ.....	261
6. ETKİLEŞİM VE VERİ YÖNETİMİ.....	262
PHP (HYPERTEXT PREPROCESSOR) .....	262
DEĞİŞKENLER .....	263
FORM KULLANIMI.....	264
POST YÖNTEMİ.....	265
GET YÖNTEMİ.....	266
PHP İLE VERİ YÖNETİMİ.....	271
7. WEB TABANLI PROJE GELİŞTİRME .....	286
7. WEB TABANLI PROJE GELİŞTİRME .....	287
PLANLAMA .....	287
TASARIM .....	287
GELİŞTİRME.....	288
TEST ETME.....	288
WEB TABANLI PROGRAMLAMA İÇİN PROJE ÖRNEKLERİ.....	288



## ORGANİZASYON ŞEMASI



### BÖLÜM KAPAKLARI

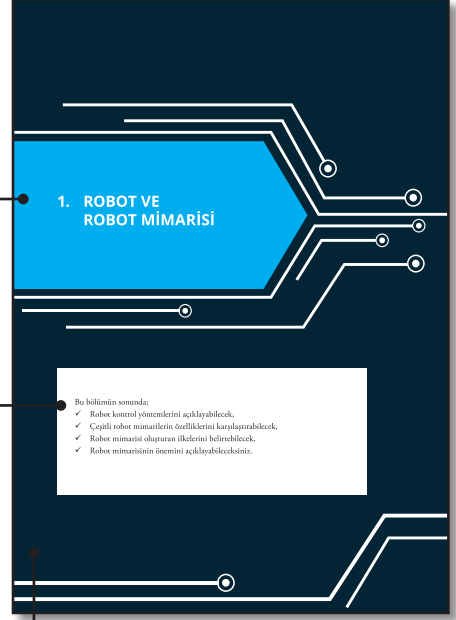
Kitap üç ana bölümden oluşmaktadır. Bu bölümler görselde yer alan kapaklar ile başlanmaktadır.

### ÜNİTE ADI VE NUMARASI

Ünitenin adı ve numarasını gösterir.

### EDİNİMLER

Ünite sonunda elde edilecek bilgi ve becerilerin listesi yer almaktadır.

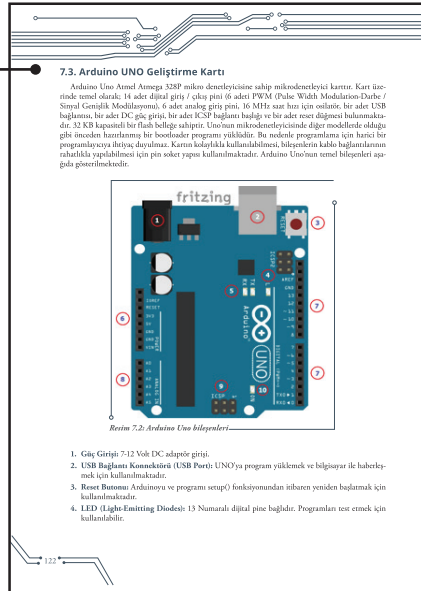


### ÜNİTE KAPAKLARI

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

### KONU BAŞLIKLARI

İçindekiler bölümündeki sırayla listelenen konuların başlıkları yer almaktadır.





# ***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 kullanılabilir. 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 duyuşsal 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 ilkeler **sense**, **plan** ve **act** arasındaki ilişkilere dayalı olarak açıklanmıştır. **Sense**, algılayıcılardan bilgi almayı ve diğer bileşen-

ler için “çıkıtı” 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 örneğinin 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

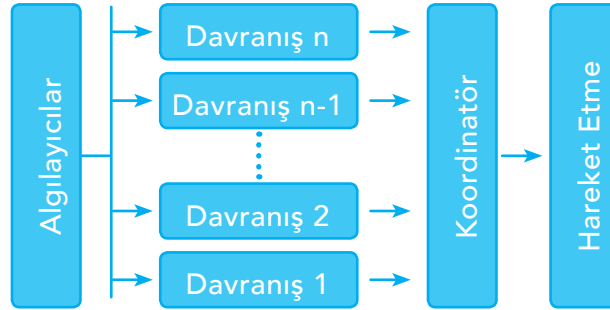
Şekil 1.2: Tepkisel mimari

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 karşılaşılabileceği her türlü durumla başa çıkabilecek bir tepkisel davranış sağlamaktadır. Aşağıdaki şekilde davranışsal mimaride ilkelerin ilişkileri gösterilmiştir.

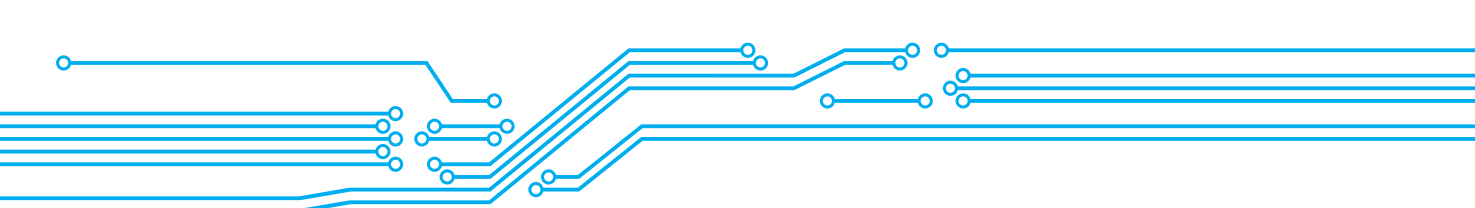


Şekil 1.4: Davranışsal mimari

Günümüzde ise Olasılıksal (Probabilistic) Robotik olarak da adlandırılan istatistiksel robotik alan; robotların öngörülemez, 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 araziye geçmek zorunda olan bir robotumuzun olduğunu düşünelim. Robotumuzun bu araziye bataklığa düşmeden, ağaçlara çarpmadan geçebilmesi istenmektedir. Buna göre:

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

## 1.5. Değerlendirme Soruları

- Bir robotun, kendi kendine (otonom) veya önceden programlanmış görevleri yerine getirebilmesi için aşağıda belirtilen özelliklerden hangisine sahip olması yeterlidir?**
  - Çevresini algılayabilme yeteneğinin bulunması yeterlidir.
  - Buldukları ortamdan bilgi alabilmeleri ve bu bilgileri işleyerek tepkide bulunabilmeleri yeterlidir.
  - İşlem yapma, işlemin sonucunu belirleme ve karar verme yeteneği bulunmalıdır.
  - Aldıkları bilgileri genellikle anlamlı bir amaç için kullanabilmeleri yeterlidir.
  - Bir operatörden bağımsız olarak işlem yapma yeteneklerinin bulunması yeterlidir.
- Robotları kontrol etmek için kullanılan farklı sistem ve yöntemler aşağıdakilerden hangisini oluşturmaktadır?**
  - Robot kontrol teknolojilerini
  - Robot kontrol yöntemlerini
  - Robot kontrol sistemlerini
  - Robot mimarisini
  - Robot paradigmalarını

3. Uyarıcı-cevap ikililerinden oluşan kurallar içeren robot kontrol yöntemi aşağıdakilerden hangisidir?
- Davranışsal (Behavioral) Kontrol
  - Tepkisel (Reactive) Kontrol
  - Karma (Hibrit) Kontrol
  - Bilinçli (Deliberative) Kontrol
  - Olasılıksal Kontrol
4. Aşağıdakilerden hangisi karma kontrole alternatif olarak sunulan robot kontrol yöntemidir?
- Davranışsal (Behavioral) Kontrol
  - Tepkisel (Reactive) Kontrol
  - Karma (Hibrit) Kontrol
  - Bilinçli (Deliberative) Kontrol
  - Olasılıksal Kontrol
5. Aşağıdakilerden hangisi önce ayrıntılı olarak düşünen, sonra bu düşünce sonucuna göre hareket eden kontrol yöntemidir?
- Davranışsal (Behavioral) Kontrol
  - Tepkisel (Reactive) Kontrol
  - Karma (Hibrit) Kontrol
  - Bilinçli (Deliberative) Kontrol
  - Olasılıksal Kontrol
6. Düşünme ve hareket işleminin paralel olarak yürütüldüğü kontrol yöntemi aşağıdakilerden hangisidir?
- Davranışsal (Behavioral) Kontrol
  - Tepkisel (Reactive) Kontrol
  - Karma (Hibrit) Kontrol
  - Bilinçli (Deliberative) Kontrol
  - Olasılıksal Kontrol
7. Robotun çalışmakta olduğu çevrenin değişmediği sabit durumlar (örneğin endüstriyel robotlar) için oldukça uygun olan robot mimarisi aşağıdakilerden hangisidir?
- Hiyerarşik Mimari
  - Tepkisel Mimari
  - Karma Mimari
  - Davranışsal Mimari
  - 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 farklılıklardan 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 sınıflandırılmaktadır (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 fiili olarak bulunmaması gereken nükleer, kimyasal felaketler gibi senaryolarda, sağlık alanında, askerî casusluk gibi birçok görevde kullanılması öngörülmuş 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.

## 2. ROBOT TÜRLERİ VE EĞİTSEL AMAÇLI ROBOTLAR



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: Tıbbi 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 kullanılabilirlerdir.



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

## 2. ROBOT TÜRLERİ VE EĞİTSEL AMAÇLI ROBOTLAR

**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.



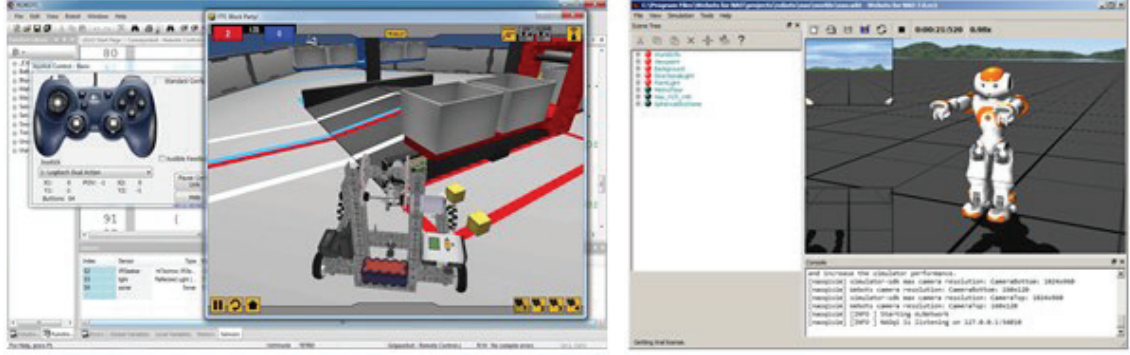
Resim 2.7: Uzay robotları

**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

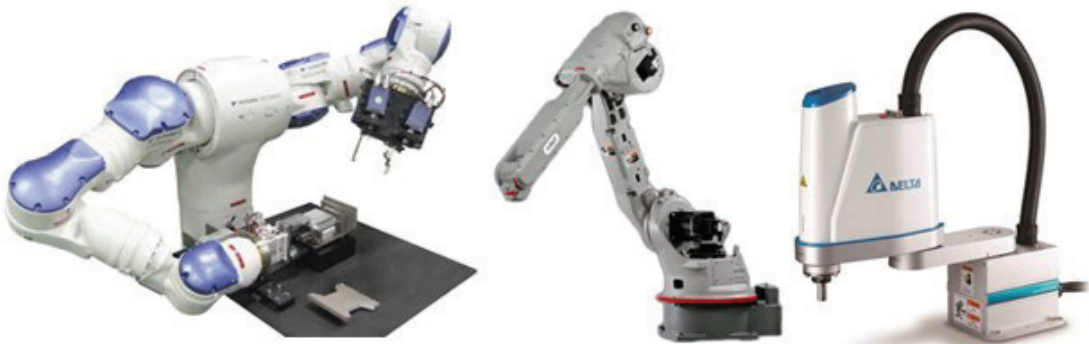
**Sanal Robotlar:** Sanal robotlar gerçek hayatta fiziksel olarak bulunmayan robotlardır. Sanal robotların yapı taşları bilgisayar programlarıdır. Sanal robotlar, gerçek bir robot simülasyonunu ya da sadece tekrarlanan bir görevi gerçekleştirebilirler. İnternet üzerinde kullanabileceğiniz sohbet robotları, 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 Mekanikğine Göre Robotlar

**Sabit Robotlar:** Sabit robotlar sürekli tekrarlayan görevlerini pozisyonlarını deęiřtirmeden yapan robotlardır. Robotun sabit olması ile anlatılmak istenen robotun temelini sabit olmasıdır. Yoksa robotun kolları hareket hâlinindedir. **Çoęu 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.



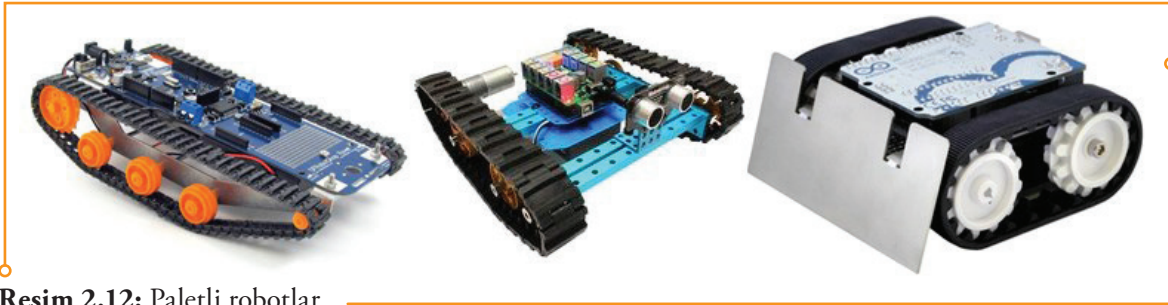
**Resim 2.10:** Sabit robotlar

**Tekerlekli Robotlar:** Tekerlekli robotlar pozisyonlarını tekerlekleri ile deęiřtirebilen mobil robotlardır. **Tekerlekli hareketi mekanik olarak saęlamak üretim açısından kolay ve düşük maliyetlidir.** Aynı zamanda tekerlekli hareketin kontrolü dięer mobil robotlara oranla daha kolaydır. Bu nedenle tekerlekli robotlar en sık karşılaşılan mobil robot tiplerindedir. Bu robot sınıfı kendi içerisinde çoęunlukla 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 kullanılabilir. 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.



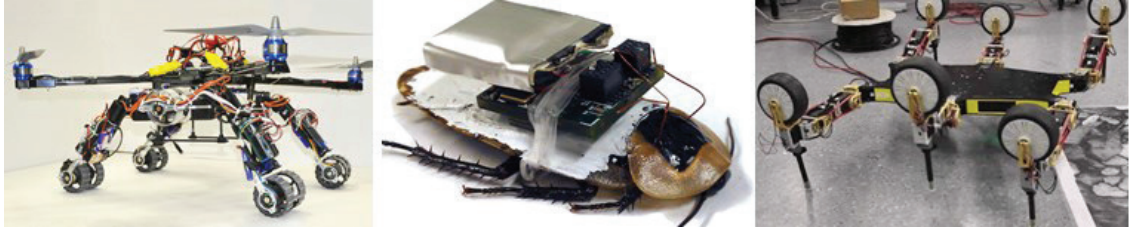
Resim 2.17: Yumuşak elastik robotlar

**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 gösterenler de sürü robotlarının elemanları çok daha fazla sayıda ve fonksiyonel açıdan çok daha basittir.



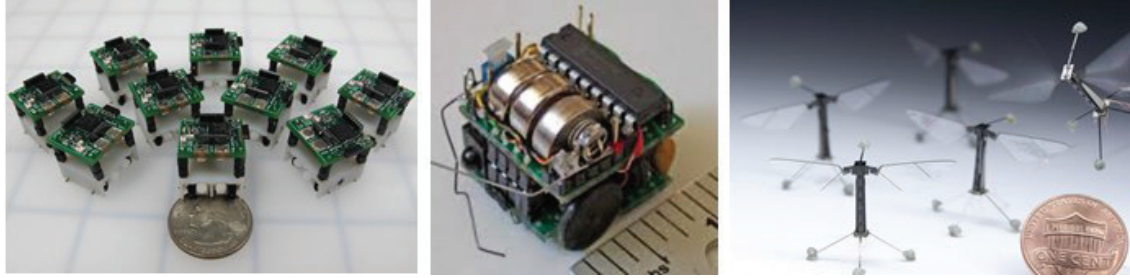
**Resim 2.20:** Sürü robotları

**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.



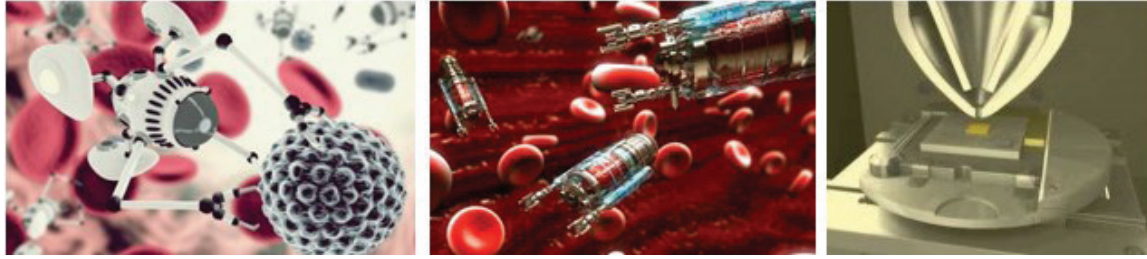
**Resim 2.21:** Modüler robotlar

**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, Aesthetics, 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 içgüdümlü hareket ederler. Genellikle güneş enerjisinden güçlerini alırlar. Bu tür robotlar doğadan esinlenerek yapılmaktadır.



Resim 2.24: Beam robotlar

## 2.4. Eğitsel Amaçlı Robotlar

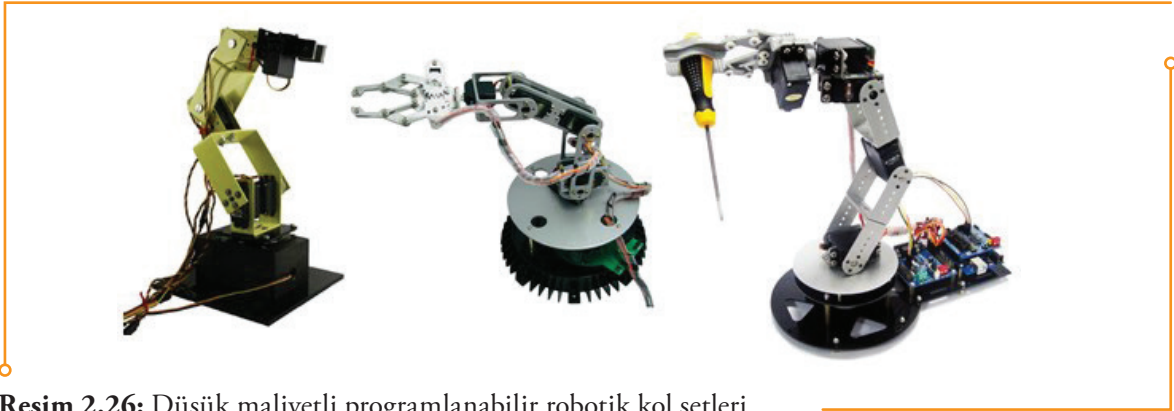
Yüzyılımızda robotların eğitsel amaçlarla kullanımı giderek artmaktadır. Eğitsel amaçlarla geliştirilen ve kullanılan çok fazla sayı ve türde eğitsel robot, robot kiti ve seti ortaya çıkmıştır. Robotlar eğitimde daha çok FTMM (Fen, Teknoloji, Mühendislik ve Matematik) eğitimini desteklemek amaçlı kullanılmaktadır. 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 kullanılacak 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 kullanılacak yaş gruplarına 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 robot 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 eklemelerin birbirlerine bağlanması ile oluşmaktadır. Eklemelerin 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

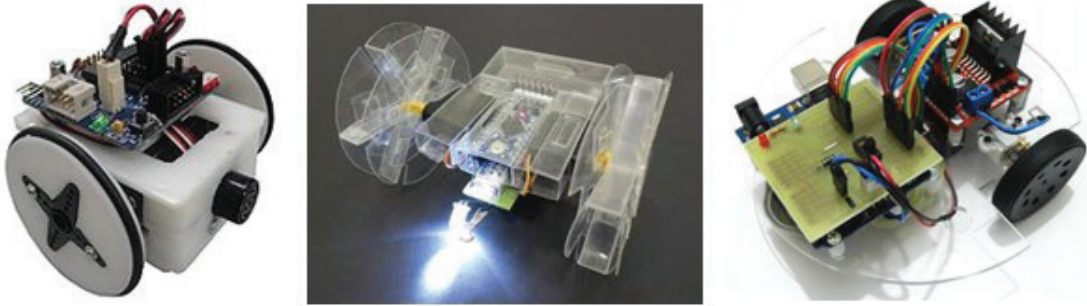
**Düşük Maliyetli Minimum Özelliklerde Mobil Robot Tasarım Kitleri:** Pek çok firmanın ürettiği bu tür eğitsel robotlar kullanıma hazır ama tamamen montajlanmamış şekilde satışa sunulmaktadır. Temel düzeyde özelliklere ve algılayıcılara sahip, ancak genişleme özellikleri ile sonradan herhangi bir bileşenin eklenmesine olanak veren kitlelerdir. 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 kitleleri

**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 paylaşımına 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 kullanılabilir. 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 yapılandırılabilir. 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 denemesi 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 / Araştırılım

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 senaryolar-da, sağlık alanında, askerî casusluk gibi birçok görevde kullanılması öngörölmüş 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 buldukları 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. EĐİTSEL ROBOTTA MEKANİK BİLEŐENLER

Bu bölümün sonunda,

- ✓ Eđitsel robotta kullanılan yapısal bileőenleri listeleyebilecek,
- ✓ Yapısal bileőenlerin görevlerini açıklayabilecek,
- ✓ Eđitsel robotta kullanılan montaj bileőenlerini tanımlayabilecek,
- ✓ Montaj bileőenlerinin görevlerini örneklendirebilecek,
- ✓ Eđitsel robotta kullanılan hareket/eylem bileőenlerini sıralayabilecek,
- ✓ Hareket/eylem bileőenlerinin görevlerini açıklayabileceksiniz.

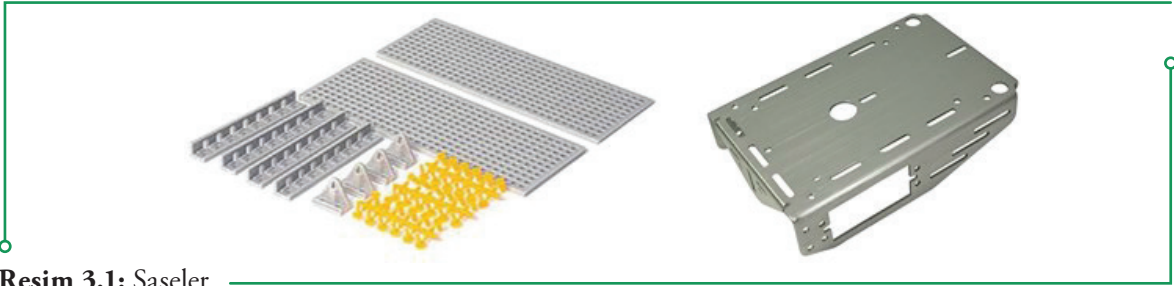
### 3.1. Eğitsel Robotta Mekanik Bileşenler

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)

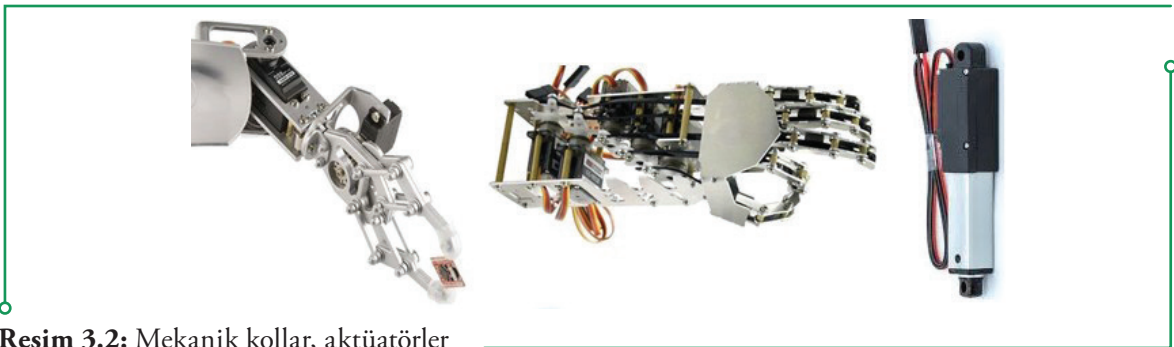
Yapısal bileşenler; robotun gövdesini, ana yapıyı oluşturan, diğer bileşenleri üstünde taşıyan gövde, iskelet gibi yapılardır. Plastikten, metalden veya her ikisinden de yapılabilir. Üreticiler tarafından satılan hazır gövdeler, gövde elemanları ve kitleri bulunmaktadır. Standart olarak üretilen pek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

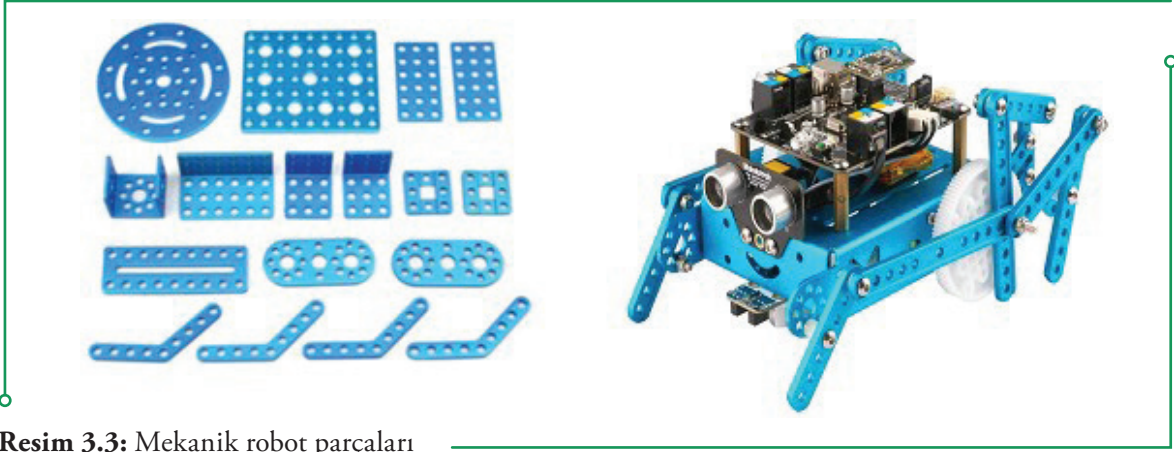
**2. Mekanik Kollar, Aktüatörler:** Robotun bir nesneyi tutması, kaldırması, sürüklemesi sağ sol, yukarı aşağı (pan/tilt) hareketi yapması için kullanılan mekanik bileşenlerdir. Genellikle iki kısığa bölünmüş kol şeklindedir. Fakat daha fazla uzvu bulunan el şeklinde kollar da bulunmaktadır. Elektronik 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) eklem 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. Gerekli 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.



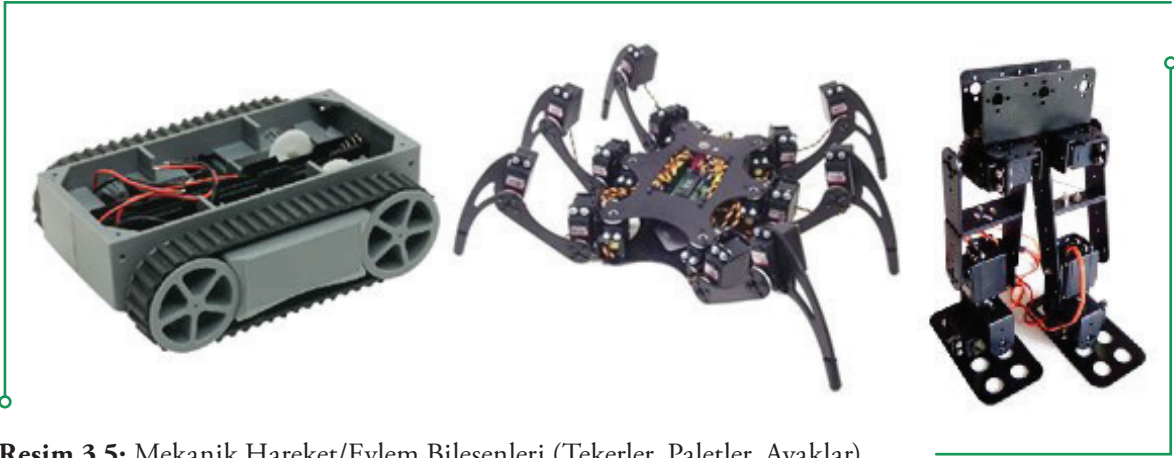
**Resim 3.4:** Montaj Bileşenleri (Bağlantı Parçaları)

### 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 / Araştırılmalı

Robot programlama dersinde kullanmak üzere bir eğitsel robot yapacağınızı düşünerek gerekli olacak 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) Gerekli 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, yüksekteç, küçük delikli levha gibi elemanlara ne ad verilir?
- Elektromekanik bileşenler
  - Mekanik hareket/eylem bileşenleri
  - Yapısal bileşenler
  - Montaj bileşenleri
  - Elektronik bileşenler
7. Aşağıdakilerden hangisi montaj bileşenlerin görevlerinden biri değildir?
- Robotu meydana getiren bileşenleri gövdeye veya birbirine bağlamaktır.
  - Robotun mekanik tasarımını kolaylaştırmaktır.
  - Robotu meydana getiren bileşenlerin bir bütün oluşturmalarını sağlamaktır.
  - Hareket esnasında robotun zarar görmesini önlemektir.
  - Bileşenlerin istenilen şekilde bağlanmasını sağlayarak daha esnek kullanım olanağı sunmaktadır.
8. Düzgün olmayan yüzeylerde hızlıca hareket etmesi için geliştirilen bir robot için uygun hareket/eylem bileşeni aşağıdakilerden hangisidir?
- Tekerlek
  - İki ayak
  - İkiden fazla ayak
  - Palet
  - Kanat
9. Eğitsel robotta kullanılan mekanik bileşenler için aşağıdaki ifadelerden hangisi yanlıştır?
- Mekanik bileşenleri olmayan robot yapmak imkânsızdır.
  - Mekanik bileşenler robotun bir bütün olmasını sağlar.
  - Mekanik bileşenler sağlam robotlar yapmak için gereklidir.
  - Mekanik bileşenler sayesinde modüler robotlar geliştirilebilmektedir.
  - Mekanik bileşenler metal, plastik veya ağaç gibi materyallerden meydana gelebilir.
10. Aşağıdakilerden hangisi hareket/eylem bileşenlerinden biri değildir?
- Tekerlekler
  - Ayaklar
  - Paletler
  - Kanatlar
  - Kollar

## 4. EĐİTSEL ROBOTTA ELEKTROMEKANİK BİLEŐENLER

Bu bölümün sonunda,

- ✓ Buton, anahtarlar ve konektör bileőenlerinin görevlerini açıklayabilecek,
- ✓ Güç bileőenlerini listeleyebilecek,
- ✓ Güç bileőenlerinin 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 mekanizmasıdır. 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

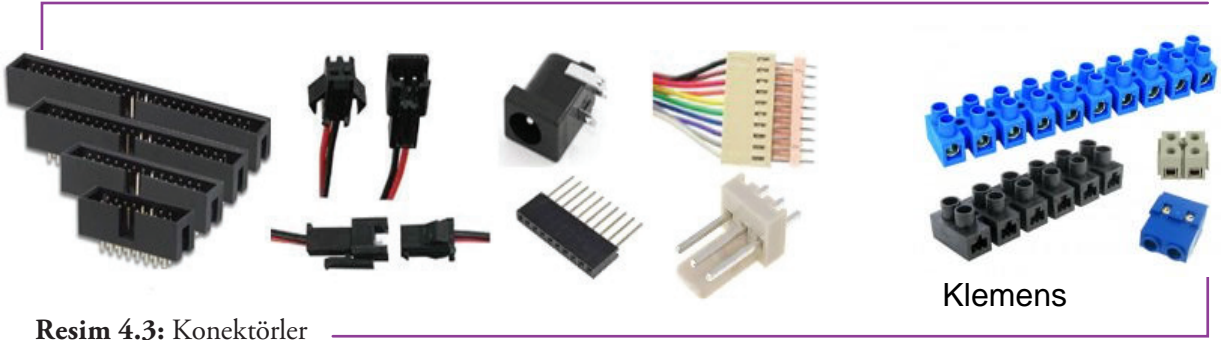
2. **Anahtarlar:** Elektrikle çalışan bütün sistem ve devrelerde, devreyi açıp kapatmaya yarayan elemanlardır. Basmalı veya çevirmeli tiplerde pek çok çeşidi bulunmaktadır. Örneğin robotun veya kontrol devrelerinin açılması ve kapatılması için kullanılan iki yollu anahtarların iki konumu ve normalde 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

Anahtar => Switch

**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. Eğitimsel 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. fırçalı, fırçasız, reduktörlü, enkoderli, enkoderli ve reduktörlü çeşitleri bulunmaktadır. Fırçalı motor, motorun hareketli olan bölümüne elektrik akımını aktarılabilmesi için fırça ve kolektör kullanılan motor türüdür. Fırçasız motor ise motorun hareketli olan bölümüne elektrik akımı aktarılabilmesi 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 uygulamalar için fırçasız motorlar kullanılmaktadır. Motorun devir hızını azaltarak daha yüksek tork (motordan tekerleğe iletilen itme -dönme momenti- kuvveti) elde etmeyi gerektiren uygulamalar için ise reduktörlü 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 / Araştırılım

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

- Her türlü elektrik ve elektronik bileşenin kablolarla birbirine bağlanması için geliştirilmiş kablo bağlantı yapılarına ne ad verilir?**
  - Buton
  - Anahtar
  - Konektörler
  - Klemens
  - Duy
- Aşağıdakilerden hangisi bağlantı bileşenlerinin görevi değildir?**
  - Önceden belirlenmiş bir sürecin başlamasını, sonlanmasını veya kontrol edilmesini sağlamak
  - Bütün elektrik ve elektronik sistem ve devrelerde, devreyi açıp kapatmak
  - Her türlü donanımın kablolarla birbirine bağlanmasını sağlamak
  - Her türlü kablonun birbirine bağlanmasını sağlamak
  - 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?
- Fırçasız motor
  - Step motor
  - Enkoderli motor
  - Servo motor
  - Redüktörlü motor
9. Dönme hareketini istenildiği kadar açığa bölerek, açısal konumu adımlar hâlinde değiştirebilen, hassas konum ve pozisyon düzenlemeleri yapabilen motor çeşidi aşağıdakilerden hangisidir?
- Fırçasız motor
  - Step motor
  - Enkoderli motor
  - Servo motor
  - Redüktörlü motor
10. Dönme hızı ve dönme sayısını kontrol etmeyi gerektiren uygulamalar için hangi tür motorlar kullanılmalıdır?
- Redüktörlü motor
  - Enkoderli motor
  - Fırçasız motor
  - Fırçalı motor
  - Adım motor

## 5. EĐİTSEL ROBOTTA ELEKTRONİK BİLEŐENLER

Bu bölümün sonunda,

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



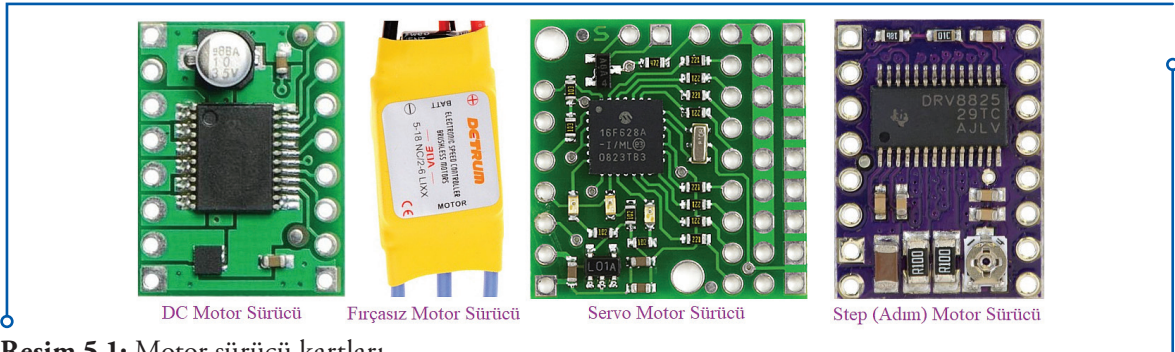
## 5.1 Eğitsel Robotta Elektronik Bileşenler

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. Aynı 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. Motorların aşırı akım çekip kontrol kartının bozulmasını engellemek.✓

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.



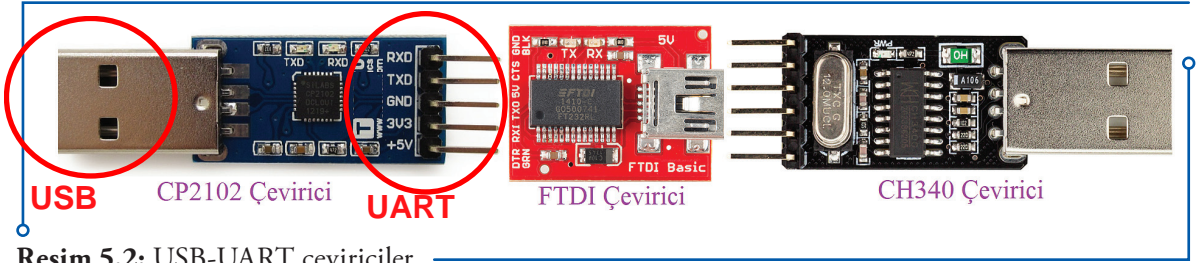
Resim 5.1: Motor sürücü kartları

## 5.3. USB-UART Çeviriciler ve Görevleri

Bilgisayar ve ona bağlanabilen her türlü çevresel aygıt seri haberleşme tekniğini (seri iletişim) kullanılmaktadır. 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 gerçekleştirilebilmektedir. 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.

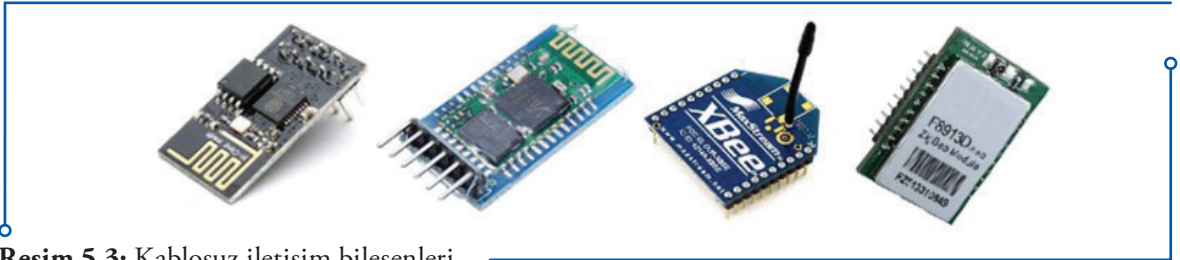


## 5.4. Kablosuz İletişim Bileşenleri ve Görevleri

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 kullanılabilir.

**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 yapılabilmektedir.** **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 yapılabilmektedir.** **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 algırlarlar. 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, kütesel 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 algılayıcıdır. Aktif algılayıcılar sinyallerini kendileri yaydıklarından daha fazla enerjiye ihtiyaç duyarlar. Bu nedenle fiziksel ya da kimyasal değerleri ölçmek için dışarıdan harici bir güç kaynağı kullanılmaktadır. Bu algılayıcıların en önemli özelliklerinden biri, zayıf sinyalleri oldukça hassas biçimde ölçmek için kullanılabilmesidir. 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 çevrilmesi 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 algılayıcıdır.



Resim 5.4: Çizgi takip algılayıcı

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

**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.

**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 algırlar. Algılayıcının ön yüzünde ısı ışınlarını IR algılayıcı üzerinde çeşitli noktalara odaklayan çok sayıda fresnel mercekler bulunmaktadır.

**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 tanımmasını sağlar.

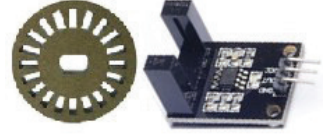
**Işık Kesici Algılayıcılar (Photo Interrupter Sensors):** Algılayıcının kolları arasında bulunan kızılötesi ışık demeti arasından bir nesne geçtiğinde ışının kırılması sonucu robotun o nesneyi algılamasını sağlayan algılayıcılardır.

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

**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 5.5: Engel kaçınma algılayıcı



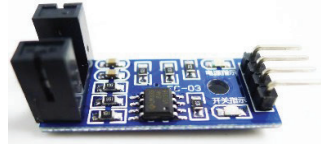
Resim 5.6: Enkoder algılayıcı



Resim 5.7: Hareket algılayıcı



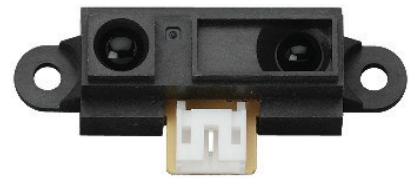
Resim 5.8: Hareket kontrol algılayıcı



Resim 5.9: Işık kesici algılayıcı



Resim 5.10: Kızılötesi termometre algılayıcı



Resim 5.11: Kızılötesi yakınlık algılayıcı

### Lazer Tarama Algılayıcılar (Laser Scanner Sensors):

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 nesnelere algılaması, hız ölçmesi için kullanılan algılayıcıdır. Sistemin çalışma mantığı Doppler Efketine dayanır.



Resim 5.13: Mikrodalga hareket dedektörü 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 nesnelere tespit etme ve bir nesne (bir kişi gibi) algılama bölgesine girdiğinde rapor vermek için de kullanılan algılayıcıdır. **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 algılayıcıdır. 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 algılayıcıdır. 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 algılayıcıdır. 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.

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

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

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

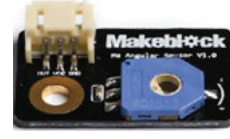
**Basınç / Yükseklik Algılayıcılar (Barometric Pressure / Altitude Sensors):** Robotun barometrik basınç ölçmesi için kullanılan algılayıcıdır. Basınç yükseklik ile değiştiği için aynı zamanda bir altimetre (yükseklikölçer) olarak da kullanılabilir.

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

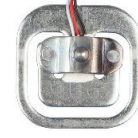
**Çarpma Algılayıcılar (Crash Sensors):** Robotun herhangi bir nesneye veya yapıya çarptığını algılaması için kullanılan algılayıcıdır. Algılama çarptıktan sonra gerçekleşmektedir.

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

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



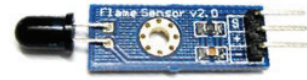
Resim 5.19: Açısal algılayıcı



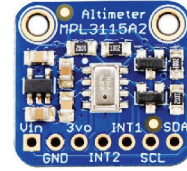
Resim 5.20: Ağırlık algılayıcı



Resim 5.21: Akım algılayıcı



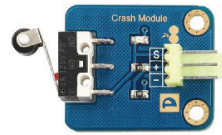
Resim 5.22: Alev algılayıcı



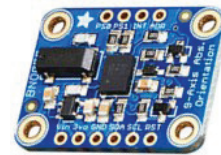
Resim 5.23: Basınç algılayıcı



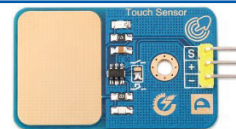
Resim 5.24: Buhar algılayıcı



Resim 5.25: Çarpma algılayıcı



Resim 5.26: Çoklu algılayıcı



Resim 5.27: Dokunma algılayıcı

dır. İnsan derisine duyarlıdır. 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 (NO<sub>2</sub>), Doğalgaz (CNG), Hidrojen (H<sub>2</sub>), sıvılaştırılmış petrol gazı (LPG), Bütan, Propan, Metan (CH<sub>4</sub>), Alkol, Amonyak (NH<sub>3</sub>) ve duman gibi gazlarla, toksik gazları algılamak için kullanılan algılayıcılardır. Hava kalitesini ölçmek için kullanılan çeşitleri de bulunmaktadır.



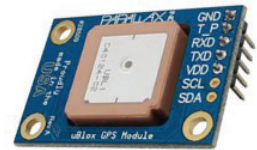
Resim 5.30: Gaz algılayıcı

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



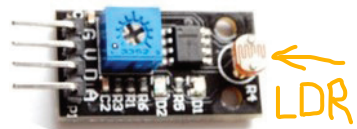
Resim 5.31: Görüntü algılayıcı

**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.



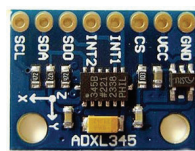
Resim 5.32: GPS algılayıcı

**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 sağlar.



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 hareketsetel 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 istenilene 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.



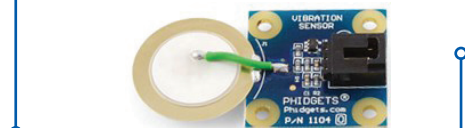
Resim 5.38: Nem algılayıcı

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



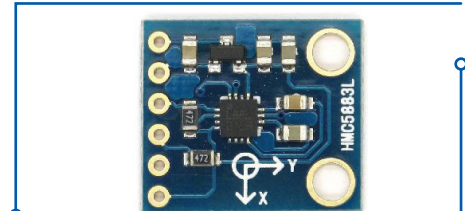
Resim 5.39: Parlaklık algılayıcı

**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.

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

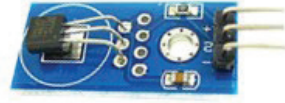
**Renk Algılayıcılar (Color Sensors):** Robotun renkleri algılaması, tanımlaması ve renk ölçümlerini doğru yapabilmesi için kullanılan algılayıcılardır.

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

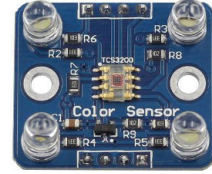
**Titreşim Algılayıcılar (Vibration Sensors):** Robotun meydana gelen titreşimleri ve hızlanmayı algılaması için kullanılan algılayıcılardır. Titreşim miktarının veya hızlanmanın ölçümü için kullanılmaz.



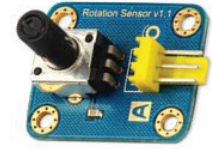
Resim 5.42: Ses algılayıcı



Resim 5.43: Sıcaklık algılayıcı



Resim 5.44: Renk algılayıcı



Resim 5.45: Rotasyon algılayıcı



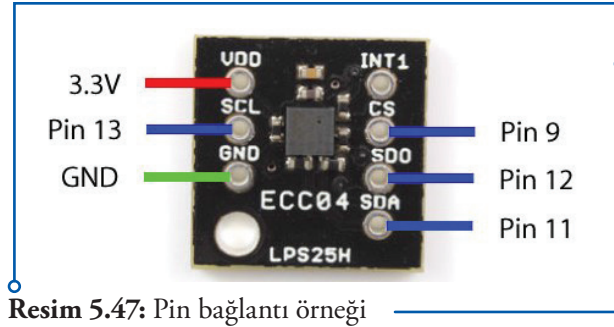
Resim 5.46: Titreşim algılayıcı

### 5.5.5. Algılayıcıların Mikrodenetleyici Kartlarla Haberleşmesi/Kartlara Bağlanması

Algılayıcıların mikrodenetleyici kartlarla haberleşmesi ile ilgili birçok standart protokol bulunmaktadır. SPI (Seri Çevresel Arayüz - Serial Peripheral Interface) ve I<sup>2</sup>C (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 Arduinonun 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 kartlarla haberleşmesi ise daha fazla pin kullanımını gerektirebilir. Eğer algılayıcıda I<sup>2</sup>C 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 aşağıdaki örnekte olduğu gibi gösterilmekte veya açıklanmaktadır. Konu ile ilgili ayrıntılar Arduino IDE konusunda verilmektedir.



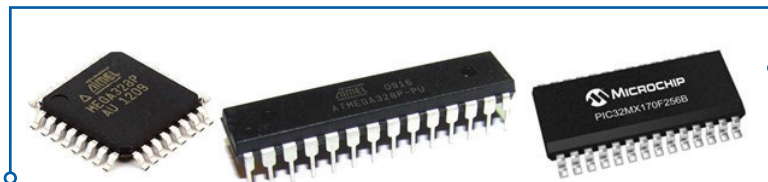
Resim 5.47: Pin bağlantı örneği

## 5.6. Robotik Programlamada Kullanılan İşlemciler

Robotik programlamada kullanılan işlemciler 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 Analoga (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, Microchip, National Semiconductor, 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++, Python gibi yüksek seviyeli dillerle de programlanabilmektedir. Hemen hemen bütün işletim sistemleri ile kullanılabilir. 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. Arduino UNO, Raspberry PI, Beagle Bone robotik uygulamalar için yaygın olarak kullanılan kartlardan bazılarıdır.

Arduino açık kaynaklı donanımdır. ✓



Resim 5.49: Mikrodenetleyici kartlar (Geliştirme kartları)

## 5.8. Mikrodenetleyici Kartlar (Geliştirme Kartları) için 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 anlaşılacağı gibi mikrodenetleyici kartla bluetooth kullanarak haberleşmeyi, veri alış-verişi 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

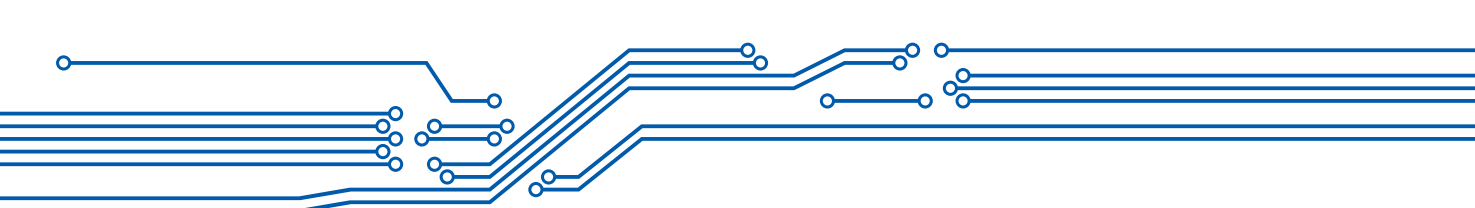
Özellikle robotik uygulamalar için geliştirilmiş olup üzerinde bir mikrodenetleyici, motor sürücü, Wi-Fi veya Bluetooth gibi kablosuz iletişim parçası bulunan kartlardır. Bazılarında her üç bileşen bulunabildiği gibi, daha az veya daha çok bileşen bir arada bulunabilir. Bazı çeşitlerde bir robotu programlayarak 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 / Araştırılma

Robot programlama dersinde kullanmak üzere bir eğitsel robot yapacağınızı düşünerek gerekli olacak 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?**

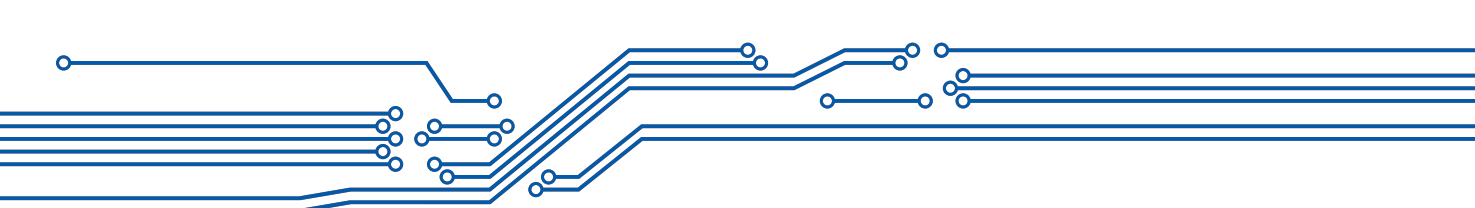
- a) USB
- b) UART
- c) Seri
- d) Paralel
- e) I<sup>2</sup>C

**7. Mikrodenetleyicilerde aşağıdaki bileşenlerden hangisi yer almaz?**

- a) I/O (İnput – Output-Seri Giriş ve Çıkış Birimleri)
- b) GUI (Grafiksel Kullanıcı 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 geliřtirmek, yeni fonksiyon ve özellikler kazandırmak veya kolayca diđer kart yapıdaki bileřenleri eklemek için kullanılan ve dođrudan 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 bileřenlerin tamamını veya önemli bir kısmını karşılayarak robot yapımını kolaylařtıran kart türü ařađıdakilerden 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

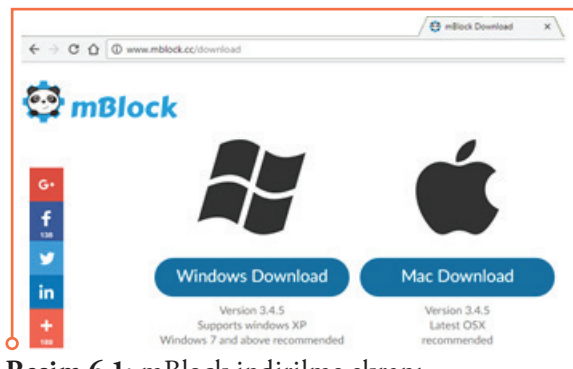
mBlock Makeblock tarafından geliştirilen Arduino projelerinde programlamayı ve etkileşimli uygulamalar oluşturmayı kolaylaştıran grafik ara yüzü görsel programlama yazılımı ve ortamıdır. Fiziksel dünya ile etkileşim içinde interaktif uygulamalar (oyun, hikâye, animasyon) ve kablosuz olarak programlanabilen 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 kullanılabilir. Bu amaçla Arduino Board Standartlarını desteklemektedir. Arduino UNO, Leonardo, Nano, Mega128, Mega 2560, PicoBoard, Makeblock mCore ve Arduino uyumlu diğer kartlarla kullanılabilir.

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 kullanılabilir. Kablosuz haberleşme protokolleri de desteklediği 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

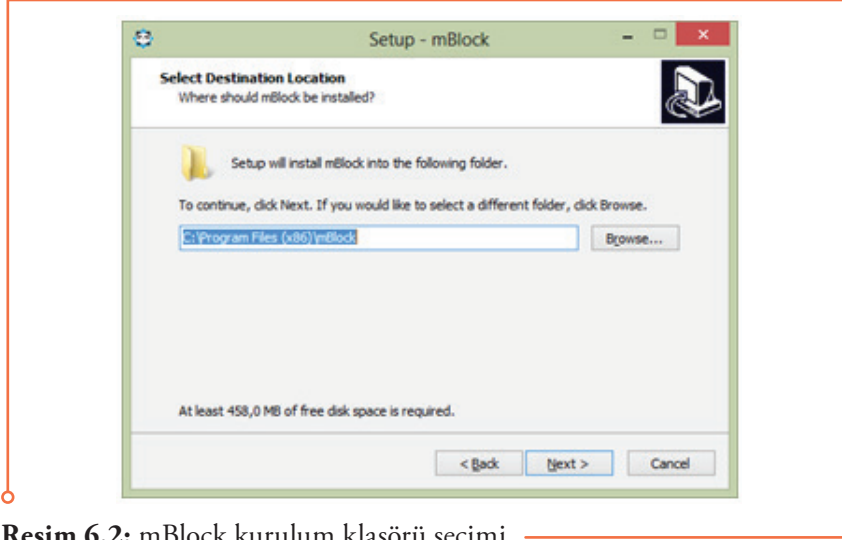
mBlock yazılımının son sürümü <http://www.mblock.cc/download> sitesinden, kullanılacak işletim sistemi (Windows veya Mac) seçilerek ücretsiz olarak indirilebilir. Hem Windows hem de Mac işletim sistemlerinin eski ve güncel sürümlerinde çalışabilir. Seçimden sonra doğrudan exe dosyası olarak indirilmektedir. 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

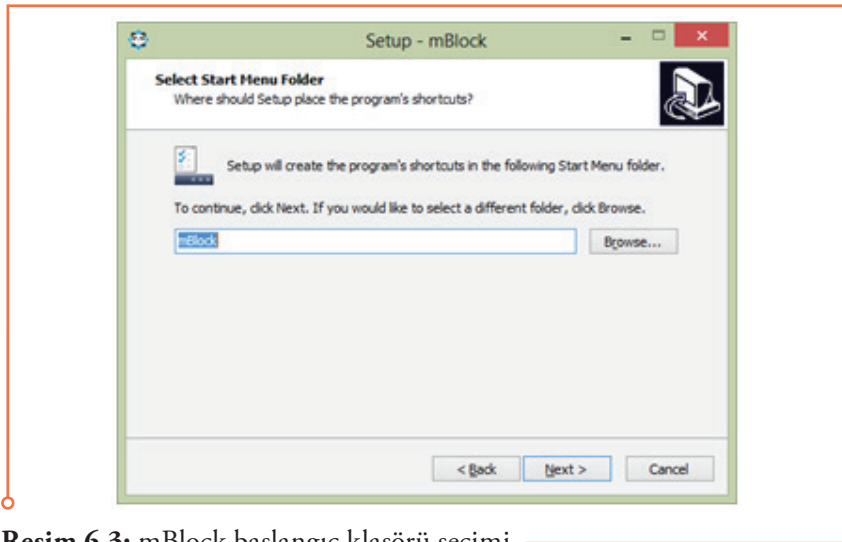
## 6. BLOK TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI

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



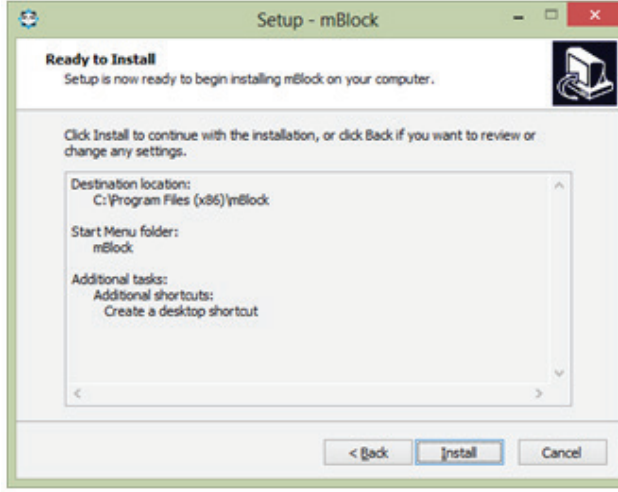
**Resim 6.2:** mBlock kurulum klasörü seçimi

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



**Resim 6.3:** mBlock başlangıç klasörü seçimi

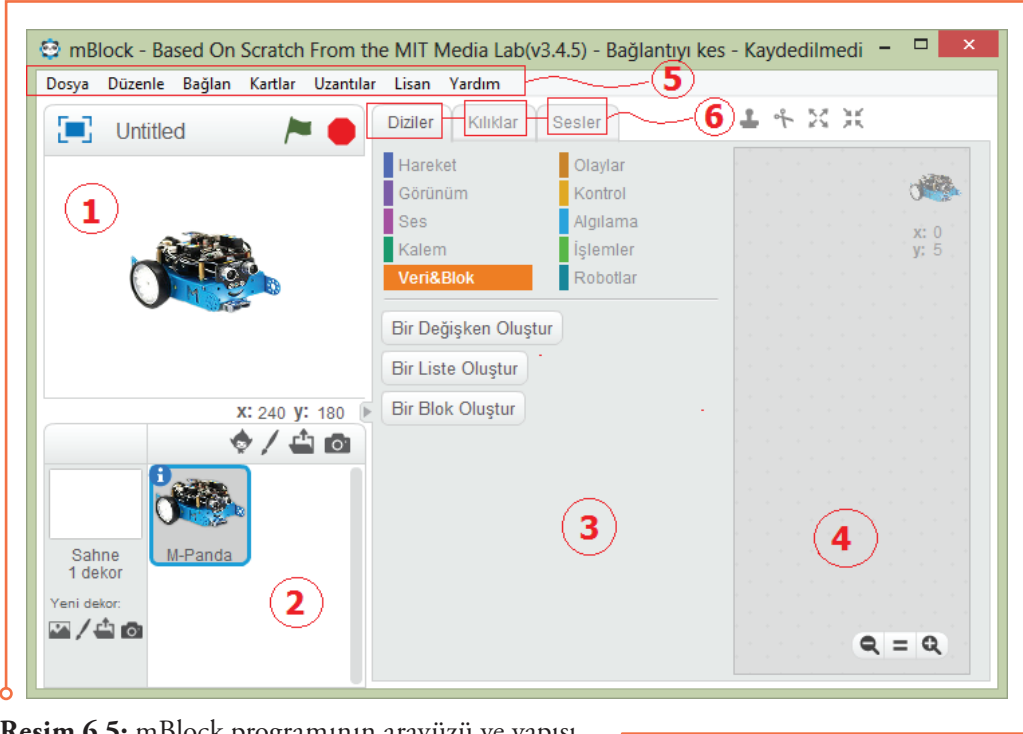
Kurulum yapılacak başlangıç menüsü klasörü seçildikten sonra programın kurulumu install seçeneğinin tıklanmasıyla başlar. Bittiği zaman programın masaüstündeki kısa yoluna tıklanarak program çalıştırılır.



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.



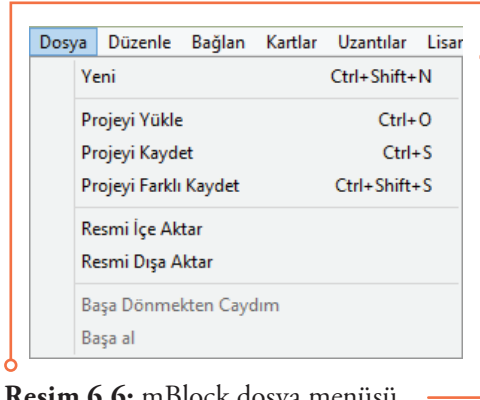
Resim 6.5: mBlock programının arayüzü ve yapısı

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

## 6. BLOK TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI

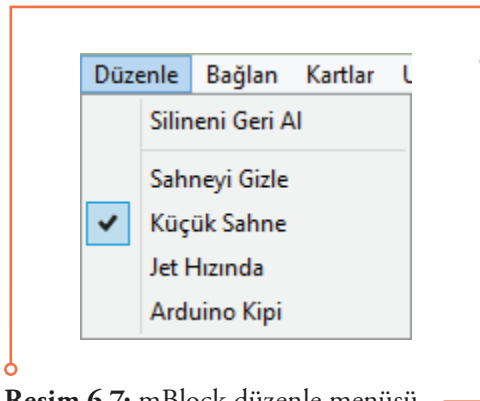
numaralı bölümde sahne (dekor) ve figür (kılık, kukla) seçimi yapılmaktadır. 3 numaralı bölüm programlama ortamını oluşturan blok yapıları ayrılmıştır. Programın yazıldığı alan ise 4 numara ile belirtilmiştir. mBlock programlama ortamı (6 numaralı alan) diziler (blok kategorileri), sahneler (dekor), 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 kullanılabilir. 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 aktarılabilir.



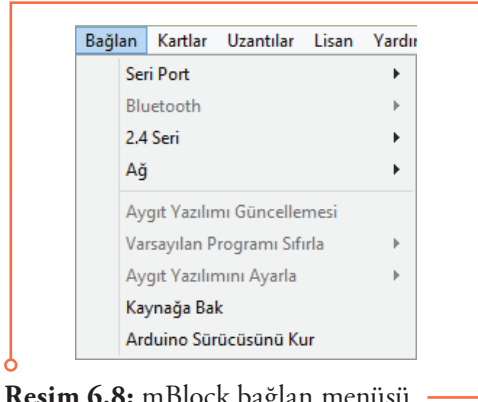
Resim 6.6: mBlock dosya menüsü

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



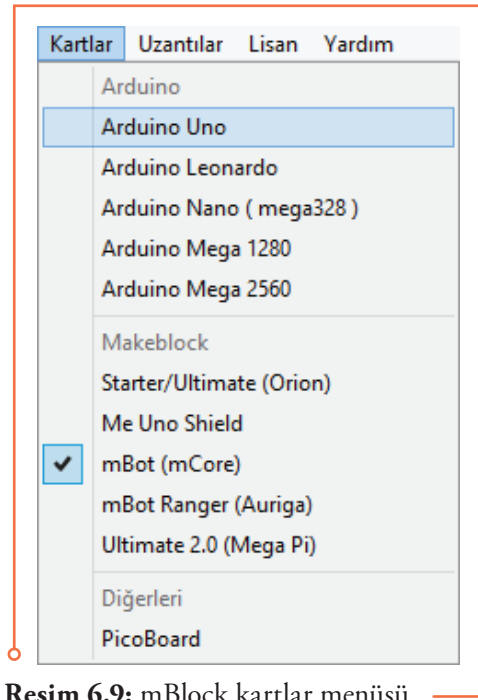
Resim 6.7: mBlock düzenle menüsü

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 sürücülerini standart olarak kurulumaktadır.



**Resim 6.8:** mBlock bağlan menüsü

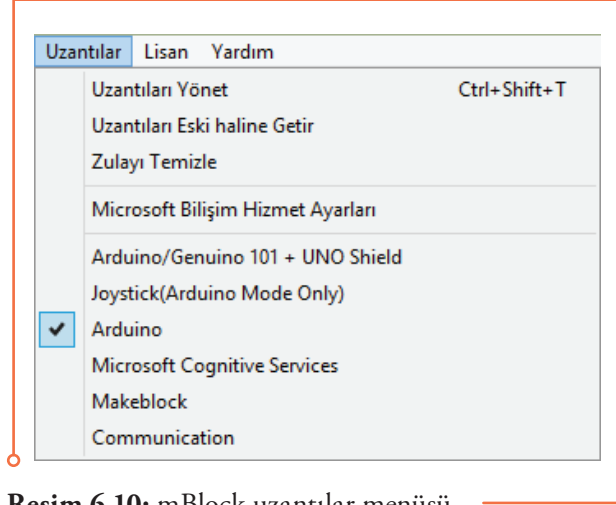
Kartlar menüsü ile kullanılan Arduino kart modelleri, Arduino uyumlu kartlar ve Makeblock tarafından üretilen robot kontrol kartları içerisinde seçim yapılmaktadır. Robotik kontrol kartı hangi model Arduino içeriyorsa o kart modeli seçilmelidir. Arduino uyumlu kartlar için Arduino, diğer kartlar için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



**Resim 6.9:** mBlock kartlar menüsü

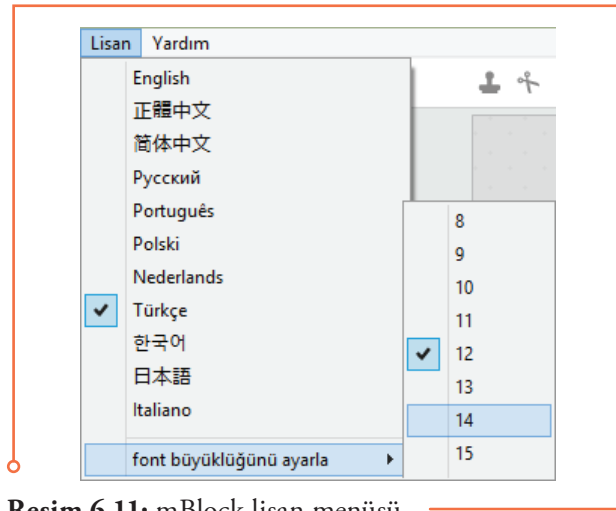
## 6. BLOK TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI

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.



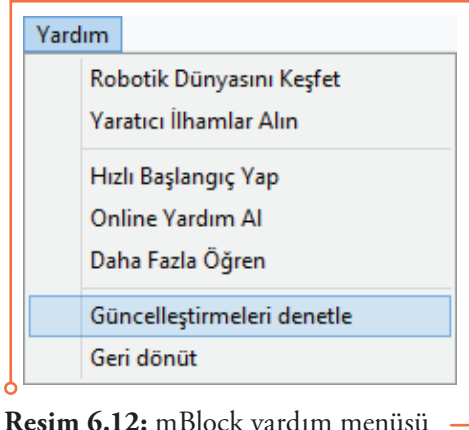
Resim 6.10: mBlock uzantılar menüsü

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



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.



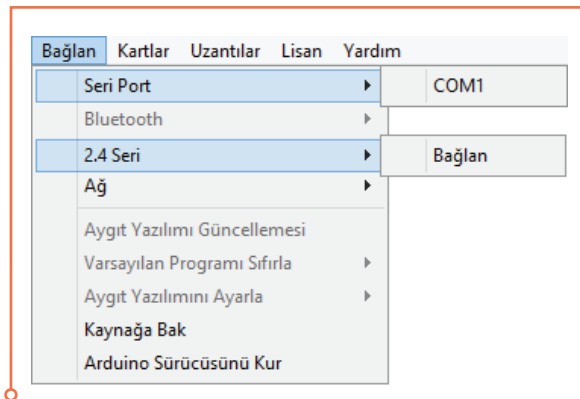
Resim 6.12: mBlock yardım menüsü

## 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 kullanılacak 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 sürücülerini 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 sürükleyip bırakarak 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 kullanılacak 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.

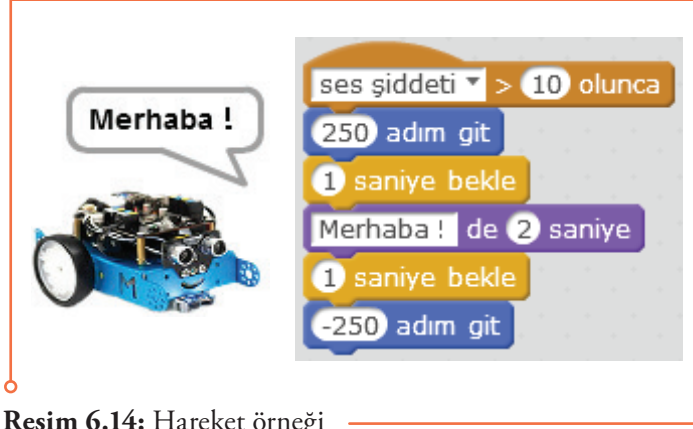
### 6.6.1. Hareket Alt Başlığı Altında Verilen Komut Blokları

"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 dönmeye izin verilip verilmeyeceği de belirlenebilir. "Hareket" alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

	Belirtilen adım kadar hareket etmesi için kullanılır.
	Belirtilen açıda soldan sağa dönmesi için kullanılır.
	Belirtilen açıda sağdan sola dönmesi için kullanılır.
	89 Derece sağa, -90 derece sola, 0 derece yukarı ve 180 derece aşağı dönmesi için kullanılır.
	Kuklanın fare okuna, Kuklaya, rasgele yatay, dikey ve sahne noktasına dönmesi için kullanılır.
	Belirtilen x ve y noktasına gitmesi için kullanılır.
	Kuklanın fare okuna, Kuklaya, rasgele yatay, dikey ve sahne noktasına gitmesi için kullanılır.
	Belirtilen süre içerisinde, belirtilen x ve y noktalarına süzülmesi için kullanılır.
	x'i belirtilen değer kadar arttırmak için kullanılır.
	x'i belirtilen nokta yapmak için kullanılır.
	y'i belirtilen değer kadar arttırmak için kullanılır.
	y'i belirtilen nokta yapmak için kullanılır.
	Kenara geldiysen sekmesi için kullanılır.
	Sağa sola dönebilmesi, hiç dönmemesi veya her yöne dönebilmesi için kullanılır.
	x konumunun ekranda gösterimi için kullanılır.
	y konumunun ekranda gösterimi için kullanılır.
	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 bekleddikten sonra ekrana “Merhaba!” yazısı 2 sn boyunca çıkmaktadır. 1 sn daha bekleddikten sonra geldiği noktaya geri dönmektedir.

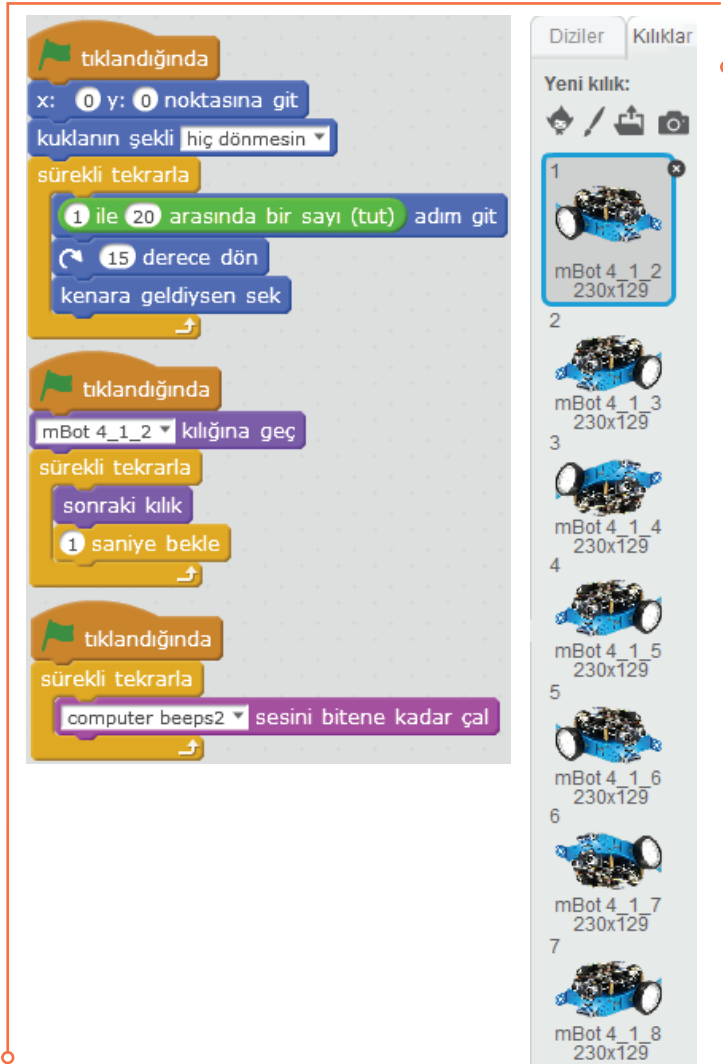


Resim 6.14: Hareket örneği

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 geldiye 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.

### 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



Resim 6.15: mBlock hareket örneği

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 nesnelere) 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 nesnelere 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 görünmemesi) 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 arttırmak 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 büyütme için kullanılır.
büyüklüğü % 100 yap	Kuklanın büyüklüğünü 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 beklemeden sonra büyüklüğü %50 oranında değiştirilmektedir.



Resim 6.16: Görünüm örneği

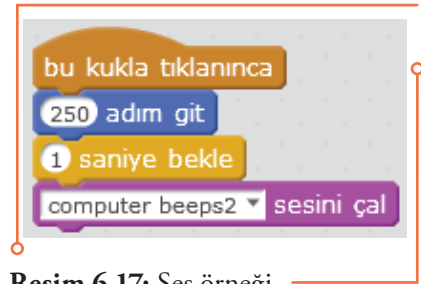
### 6.6.3. Ses Alt Başlığı Altında Verilen Komut Blokları

“Ses” alt başlığı altında verilen bloklar sanal bir robotun (kütüphaneden kukla, figür kullanılarak veya oluşturularak) sesle kullanımını, 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.

	Belirtilen sesi çalmak için kullanılır.
	Belirtilen ses bitene kadar çalmak için kullanılır.
	Tüm sesleri durdurmak için kullanılır.
	Belirtilen ses kaynağını (18 adet) belirtilen vuruş oranı kadar çalmak için kullanılır.
	Belirtilen vuruş oranı kadar susması için kullanılır.
	Belirtilen notayı belirtilen süre boyunca çalmak için kullanılır.
	Müzik enstrümanını (21 adet) değiştirmek için kullanılır.
	Sesi düzeyini belirtilen birim kadar değiştirmek için kullanılır.
	Ses şiddetini belirtilen oran kadar değiştirmek için kullanılır.
	Ses şiddetin ekranda göstermek için kullanılır.
	Tempoyu belirtilen oranda değiştirmek için kullanılır.
	Tempoyu belirtilen vuruş/dakika oranına ayarlamak için kullanılır.
	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 beklemeden sonra “computer beeps2” sesini çalmaktadır. Ses kaynağı ses kütüphanesinden örnek olarak seçilmiştir.



Resim 6.17: Ses örneği

## 6. BLOK TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI

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 bloğun nota numarasına tıklanarak belirlenebilir.



Resim 6.18: 2. Ses örneği



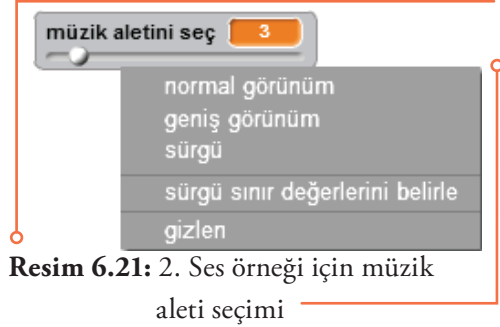
Resim 6.19: 2. Ses örneği için nota

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 oluşturularak) 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	Kalemin kullanılacak yüzeyde çizmeye başlaması için kullanılır.
kalemi kaldır	Kalemi kaldırarak çizimi durdurmak için kullanılır.
kalem rengini <input type="checkbox"/> yap	Kalemin rengini değiştirmek için kullanılır.
kalem rengini 10 değiştir	Kalemin rengini belirtilen değer kadar değiştirmek için kullanılır.
kalem rengini 0 yap	Kalemin rengini belirtilen renk yapmak için kullanılır.
kalem tonunu 10 arttır	Kalemin tonunu belirtilen oranda arttırmak için kullanılır.
kalem tonunu 50 yap	kalemin tonunu belirtilen değer yapmak için kullanılır.
kalem kalınlığını 1 arttır	Kalemin kalınlığını belirtilen değerde arttırmak 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 (Kordinat 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 kullanılabilir. 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

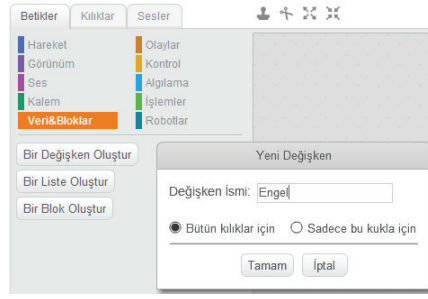
“Veri&Blok” alt başlığı altında verilen bloklar sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robotun programlanması ile diğer uygulama programlarında gerekli olabilecek “değişken”, değişken listesi “liste” ve “blok” oluşturulması ve bunların düzenlenmesi için kullanılmaktadır.

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

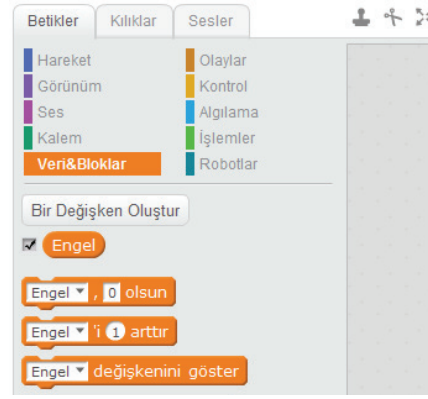
Tablo 6.5: Veri&Blok alt başlığı altında verilen komut blokları

**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.

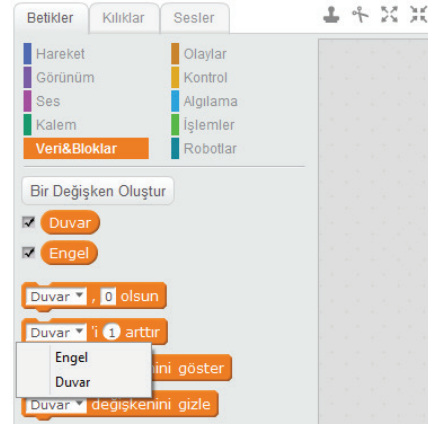
mBlock'ta değişkenler “Veri&Bloklar” kategorisinde “Bir Değişken Oluştur” seçeneğinde bulunmaktadır. “Bir Değişken Oluştur” butonuna tıklandıktan sonra, değişken için bir ad girilmelidir.



Değişkenler programın sol tarafında listelenmektedir. Eklenen her bir değişken için değer atama blokları otomatik olarak eklenmektedir.

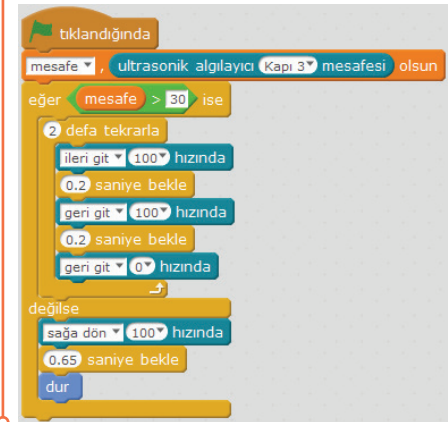


Değer atama bloklarından hangi değişkene değer atanacağı aşağı okuyla seçilmektedir.




**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 hareket 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 bulunan  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. Diziyeye değerler bu tablonun sol alt köşesindeki + işaretine tıklayarak oluşan kutucuğa yazıyla girilebilir. İstenirse program akışı içerisinde de listeye değer eklenip çıkarılabilmektedir.

<p>mBlock’ta listeler “Veri&amp;Bloklar” kategorisinde “Bir Liste Oluştur” seçeneğinde bulunmaktadır. “Bir Liste Oluştur” butonuna tıklandıktan sonra, liste için bir ad girilmelidir.</p>	
<p>Liste değişkenleri ekranın sol tarafında listelenmektedir. Listeyle yapılabilecek işlemlere ait bloklar otomatik olarak eklenmektedir.</p>	

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.

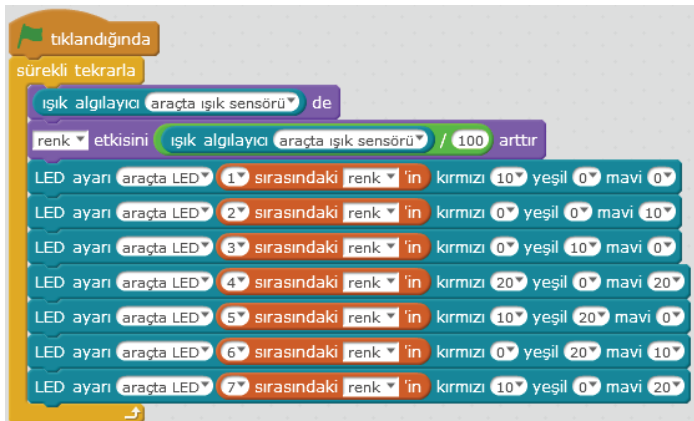


**Tablo 6.7:** Liste ve dizi oluşturulması aşamaları

**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 artır** bloğu seçilerek üzerine "İşlemler" kategorisinde bulunan **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



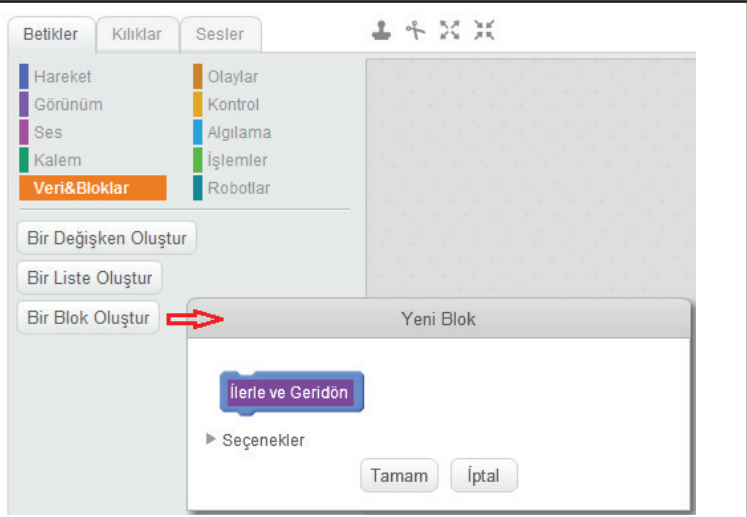
**Resim 6.26:** Liste oluşturma örneği

## 6. BLOK TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI

kullanılarak “renk” adını taşıyan bir liste yapılmıştır. “renk” listesini oluşturan değişkenler yazılım tarafından programın sol tarafında otomatik olarak listelenmiştir. “renk” adlı listeye 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 LED ayarı araçta LED hepsi kırmızı 0 yeşil 0 mavi 0 bloğu seçilerek “hepsi” yerine, oluşturulan listeden 1 sırasındaki renk'in 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 yapılarıdır. Program akışı içinde tekrarlayan ifadelerin her seferinde tekrar tekrar yazılması yerine, bir kere ayrı bir yerde yazılıp tekrarlanan her yerde kullanmak için uygundur. Aşağıda “İlerle ve Geri dön” adında bir blok oluşturulması işlemi gösterilmiştir. Burada bu bloğun 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.

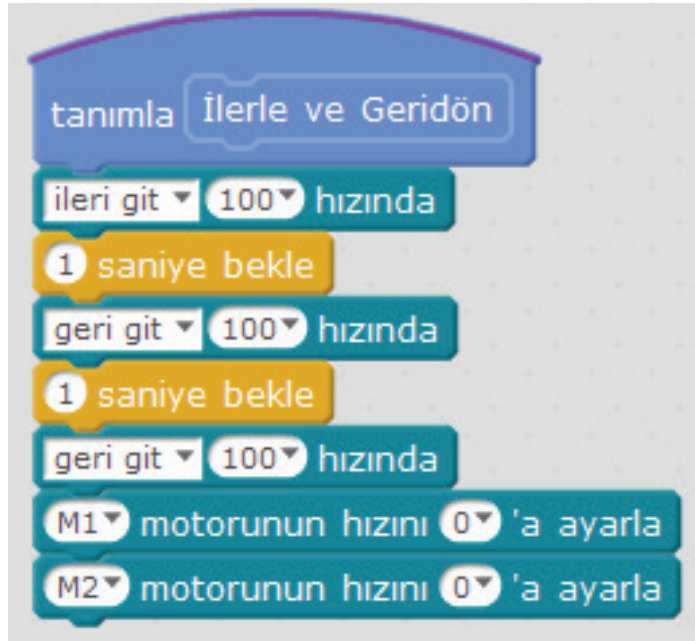
mBlock'ta blok oluşturulması “Veri&Bloklar” kategorisinin altında “Bir Blok Oluştur” seçeneğinde bulunmaktadır. “Bir Blok Oluştur” butonuna tıklandıktan sonra, oluşan “Yeni Blok” penceresindeki isimlendirme alanına blok için bir ad girilmeli ve onaylanmalıdır.



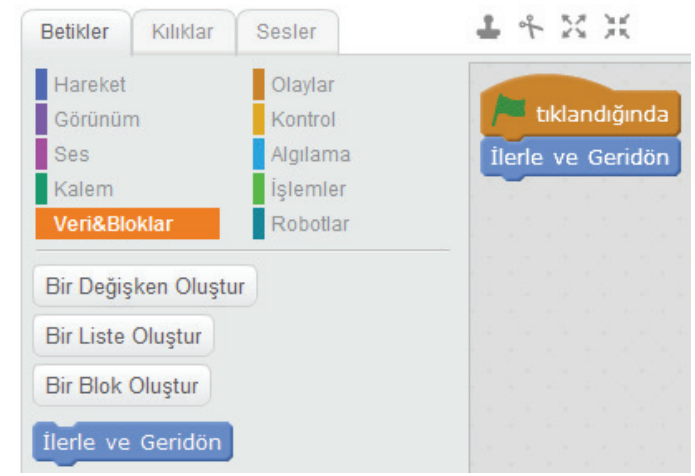
Programda kullanılacak ve çağrılacak blok otomatik olarak eklenmektedir.



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.

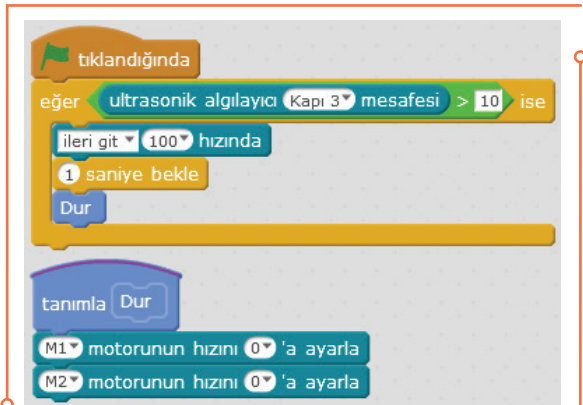


Blok kullanılmak istendiğinde ana programdan çağrılmaktadır.




**Tablo 6.8:** Blok oluşturma aşamaları

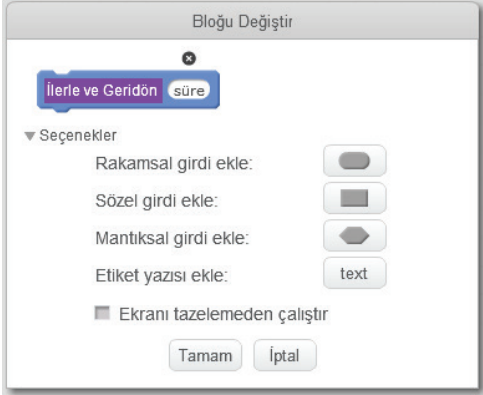
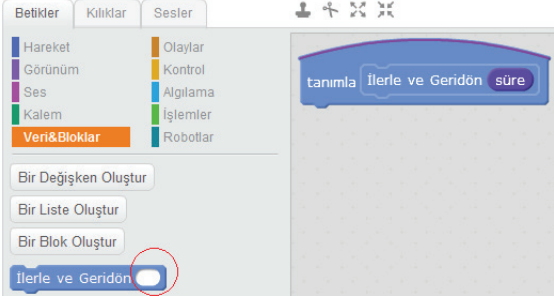

**Blok (Prosedür) Oluşturma Örneği:** Bu örnekte robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 10 cm'den büyük ise program çalışmaktadır. Engele olan uzaklık 10 cm'den büyük ise her tıklamada 100 rpm hıza göre 1 saniye ileri doğru 10 cm kalıncaya 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 bulunan  işlem bloğu kullanılarak yazılmıştır.

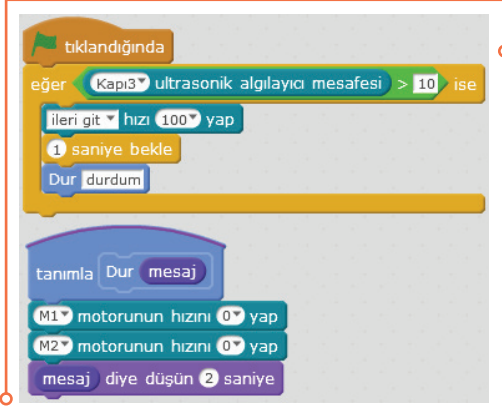
**Oluşturulan Blok (Prosedür) İçin Parametre Tanımlanması:** Prosedürlere değer taşıyan değişkenlere parametre adı verilir. Bir prosedür çağrıldığı zaman aynı zamanda parametre değerlerini de vermek gerekmektedir. Oluşturulan "İlerle ve Geridön" bloğu bu anlamda bir prosedürdür. Bu prosedüre parametre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

<p>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.</p>	
<p>Yeni tanımlanan bu prosedürde yapılacak işler parametre değeri ile birlikte tanımlanır.</p>	
<p>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.</p>	

**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 bulunan  işlem bloğu kullanılarak yazılmıştır.



Resim 6.28: Parametre örneği





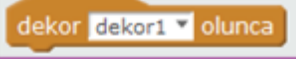


Resim 6.29: Parametre örneği ekran görüntüsü

Bu prosedüre parametre tanımlamak için “Dur” değişkenine sağ tıklanmış ve açılan “Düzenle” düğmesinde bulunan “Seçenekler” içerisinde “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

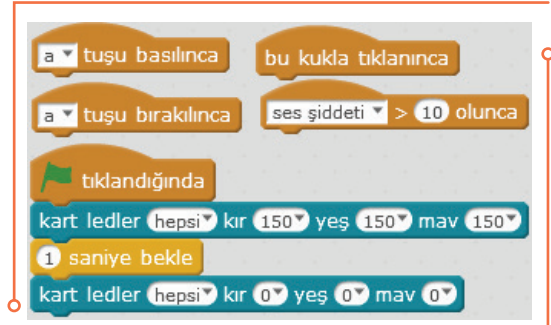
Olaylar alt başlığı altında verilen bloklar sanal (kütüphaneden kukla, figür kullanarak veya oluşturularak) veya fiziksel bir robot için oluşturulan programların ve diğer uygulamaların çalıştırılmasında kullanılmaktadır. Buradaki blokların kullanılmasıyla bir tuşun basılı olup olmaması veya seçilen nesnenin 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.

	Herhangi bir uygulamanın tıkladığında çalışması için kullanılır.
	Herhangi bir uygulamanın belirlenen tuşa basılmasıyla çalışması için kullanılır.
	Herhangi bir uygulamanın belirlenen tuşun bırakılmasıyla çalışması için kullanılır.
	Herhangi bir uygulamanın kuklaya tıklanmasıyla çalışması için kullanılır.
	Herhangi bir uygulamanın belirlenen dekorun kullanılmasıyla çalışması için kullanılır.

	Herhangi bir uygulamanın ses şiddeti, süre ölçer veya video hareketi belirlenen değerden büyük olunca çalışması için kullanılır.
	Herhangi bir uygulamanın belirtilen ileti veya belirtilen yeni ileti gelince çalışması için kullanılır.
	Herhangi bir uygulamada belirtilen ileti veya belirtilen yeni ileti gelince açıklanması için kullanılır.
	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 oluşturularak) 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.

	Programın belirtilen saniye kadar beklemesi için kullanılır.
	Döngüler tekrarlanan işleri yapmak için kullanılan temel yapıdır. Bu döngü bloğu belirtilen sayı kadar işlemi tekrarlamak için kullanılır.
	Verilen işlemi sürekli tekrarlamak için kullanılan döngü bloğudur.
	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 yapılarıdır.
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 ultrasonik algılayıcıya yaklaşıtırsanız) koşul gerçekleşince 100 rpm hıza göre 1 saniye ilerleyip M1 ve M2 motorlarının hızlarını sıfıra düşürmektedir. Uzaklık şartı olan 20 cm "İşlemler" kategorisinde bulunan  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 ilerleyip koşul gerçekleşince M1 ve M2 motorlarının hızlarını sıfıra düşürmektedir. Uzaklık şartı olan 20 cm "İşlemler" kategorisinde bulunan  işlem bloğu kullanılarak yazılmıştır.

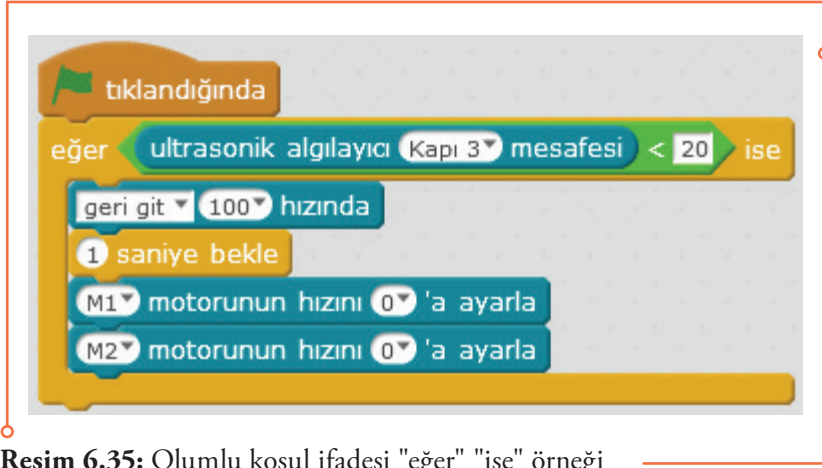


Resim 6.34: Belirli bir şart gerçekleşene kadar döngü örneği


### 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 bulunan  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 bulunan  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 oluşturularak) 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.

<input type="checkbox"/> a değdi (mi?)	Kuklanın fare okuna, sağ, sol, üst veya alt kenara değdiğini belirlemek için kullanılır.
<input type="checkbox"/> rengine değdi (mi?)	Kuklanın belirlenen herhangi bir renge değdiğini belirlemek için kullanılır.
<input type="checkbox"/> rengi <input type="checkbox"/> rengine değdi (mi?)	Kuklanın belirlenen herhangi bir renginin belirlenen alanda herhangi bir renge değdiğini belirlemek için kullanılır.
<input type="checkbox"/> 'a mesafe	Fare okuna, kuklaya, rasgele yatay, dikey ve sahne noktasına olan mesafeyi belirlemek için kullanılır.
<input type="checkbox"/> İsmi ne? diye sor ve bekle	Belirtilen ifadeyi (ad, isim, yer vs.) sormak ve beklemek için kullanılır.
<input type="checkbox"/> yanıt	Yantını ekranda almak için kullanılır.
<input type="checkbox"/> boşluk tuşu basılı (mi?)	Belirtilen tuşun basılı olup olmadığını belirlemek için kullanılır.
<input type="checkbox"/> fareye basılı mı?	Farenin tuşunun (aktif tuş) basılı olup olmadığını belirlemek için kullanılır.
<input type="checkbox"/> farenin x'i	Farenin x koordinatını belirlemek için kullanılır.
<input type="checkbox"/> farenin y'si	Farenin y koordinatını belirlemek için kullanılır.
<input type="checkbox"/> ses şiddeti	Ses şiddetini ekranda görmek için kullanılır.
<input type="checkbox"/> video hareket on bu kukla	Kullanılan kukla veya sahne üzerinde video hareketi ve yönü belirtmek için kullanılır.
<input type="checkbox"/> videoyu aç	Videoyu açmak, kapatmak ve açıp solu sağ yapmak için kullanılır.
<input type="checkbox"/> video saydamlığı % 50 olsun	Video saydamlığını belirtilen orana ayarlamak için kullanılır.
<input type="checkbox"/> süre ölçer	Süre ölçeri ekran üzerinde açmak için kullanılır.
<input type="checkbox"/> süre ölçeri sıfırla	Süre ölçeri sıfırlamak için kullanılır.
<input type="checkbox"/> x konumu değeri mBot 4_1_1'in	x, y konumu, yönü, klık no, kılığın ismi, büyüklük ve ses şiddeti değerleri mevcut kukla veya sahne için kullanılır.
<input type="checkbox"/> şimdiki dakika	Ekranda saniye, dakika, saat, haftanın günü, tarih, ay ve yıl bilgilerini göstermek için kullanılır.
<input type="checkbox"/> 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.
	Belirtilen değerler arasında rasgele sayı üretmek için kullanılır.
	"<" karşılaştırma işlemleri için kullanılır.
	"=" karşılaştırma işlemleri için kullanılır.
	">" karşılaştırma işlemleri için kullanılır.
	Mantıksal işlemler için kullanılır. Her iki koşul da doğru olduğunda "ve" yapı taşı doğru olacaktır. Aksi takdirde yanlış olur.

	Mantıksal işlemler için kullanılır. Her iki durumdan biri doğru olduğunda, "veya" yapı taşı doğrudur. Aksi takdirde yanlış olur.
	Mantıksal işlemler için kullanılır. Mantıksal işlem "değil" ise yapı taşı doğrudur.
	Belirtilen iki kelimeyi birleştirmek için kullanılır.
	Belirtilen kelimenin belirtilen harfiyle işlem yapmak için kullanılır.
	Belirtilen kelimenin uzunluğuyla işlem yapmak için kullanılır.
	Belirtilen kelime içinden alınan kelimenin indeksi ile işlem yapmak için kullanılır.
	Belirtilen oyuncuyu dizgeye taşımak için kullanılır.
	Mod işlemleri yapmak için kullanılır.
	Belirtilen değeri yuvarlamak için kullanılır.
	Belirtilen sayının karakökünü, mutlak değerini almak, aşağı veya yukarı yuvarlamak ve açısız 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.

### 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



Resim 6.38: İşlem örneği

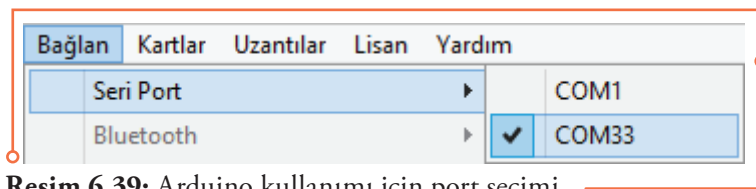
Arduino uyumlu kartlar için kullanılacak bloklar verilmiştir. Robotlar alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

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.

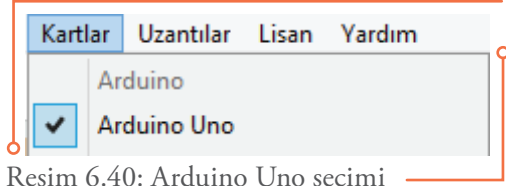


**Resim 6.39:** Arduino kullanımı için port seçimi

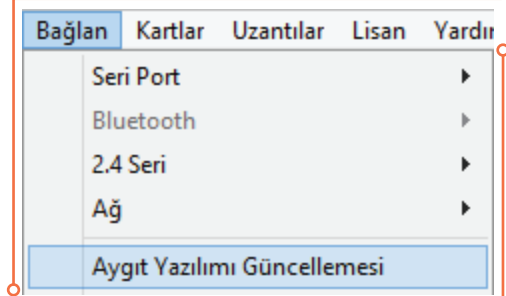
Arduino ile kullanılacak elektronik bileşenlerin önceden hazırlanması, bağlantılarının yapılması sonra da uygulamanın hazırlanarak Arduino'ya yüklenmesi gerekmektedir. Yükleme için “Arduino Prog-

ramı” **Arduino Programı** blokunun üzerine tıklanarak açılan ekrandan “Arduinoya Yükle” seçeneğinin tıklanması gerekmektedir. Bu durumda uygulama ekranının 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 kullanılacaktır.

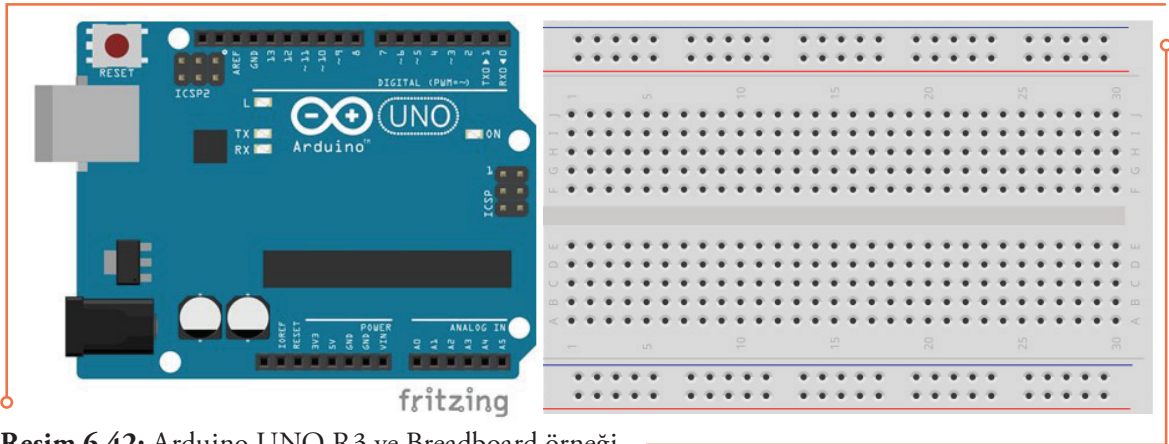
Arduino uygulamalarında kullanılacak elektronik bileşenler için Breadboard kullanılması uygulamaların hızlı, kolay ve en önemlisi lehim yapmadan yapılmasına olanak tanıyacaktır. Breadboardların kenarında bulunan alanlar voltaj bağlantıları için enine bağlı, diğer alanlar ise dikine bağlıdır. Bu bağlantı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



Resim 6.40: Arduino Uno seçimi



Resim 6.41: Aygıt yazılımının güncellenmesi



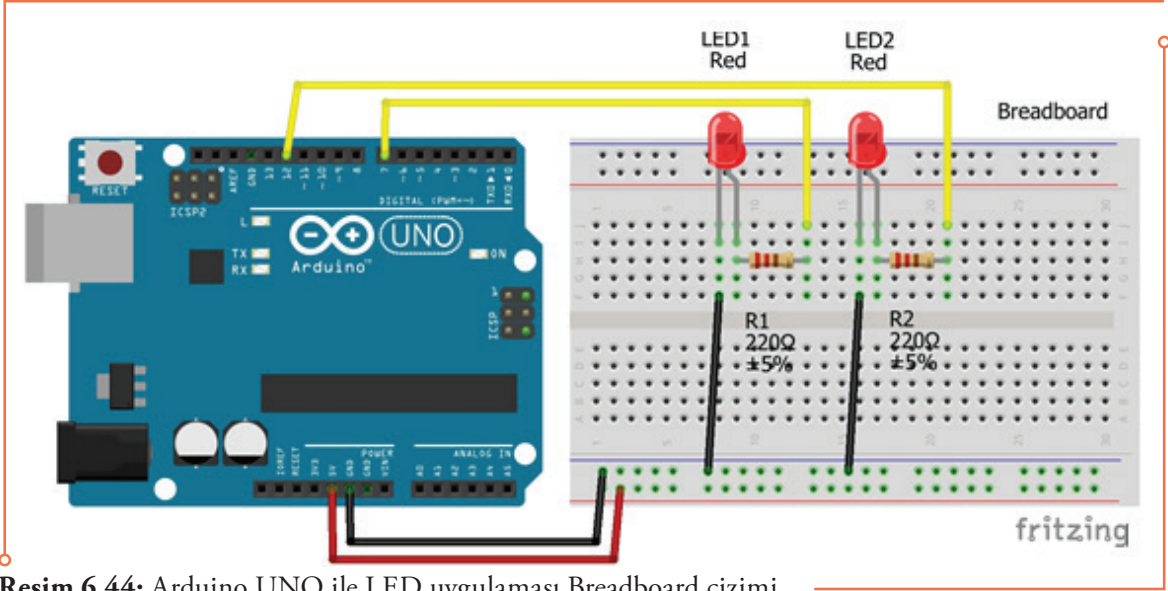
Resim 6.42: Arduino UNO R3 ve Breadboard örneği

**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.

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 6.43: Dijital kontrol pimlerin ayarlanması ve okunması



**Resim 6.44:** Arduino UNO ile LED uygulaması Breadboard çizimi

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

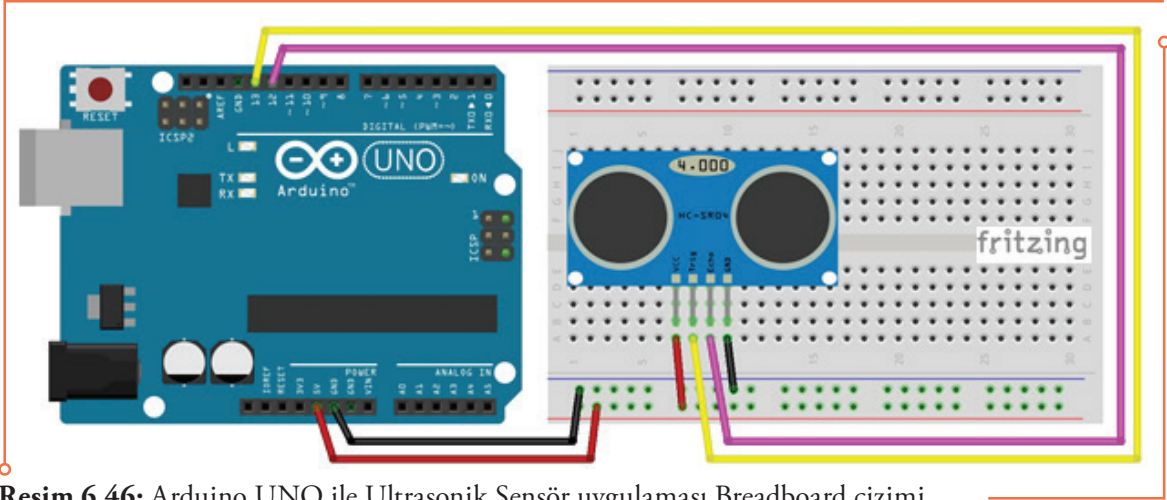


**Resim 6.45:** Arduino UNO ile LED uygulaması örneği

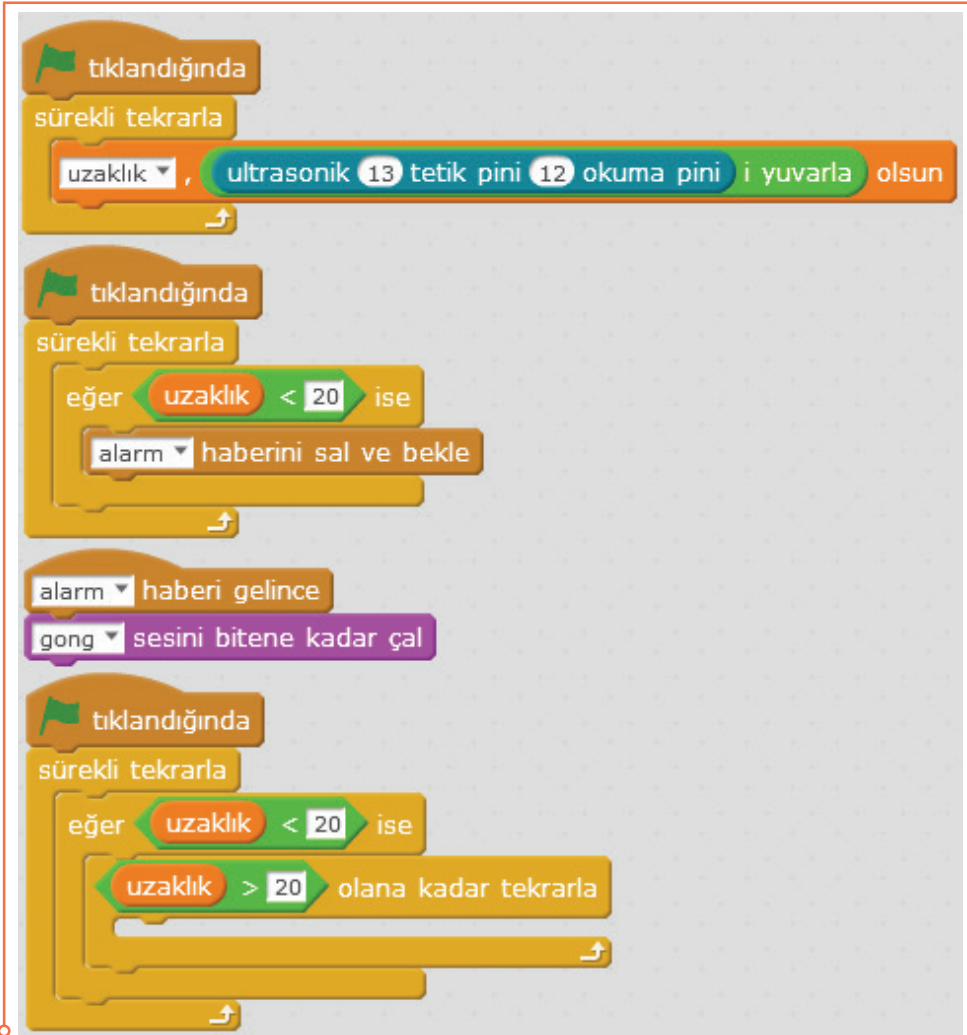
Aşağıda verilen örnekte Arduino'nun dijital pinlerine bağlı HY-SRF 05 ultrasonik sensör kullanılarak bir alarm uygulaması hazırlanmıştır. Ultrasonik sensörün tetik (Trig) pini 13 numaralı, okuma (Echo) pini 12 numaralı Arduino UNO pinine bağlanmıştır. Bağlantılar aşağıdaki breadboard çizimi üzerinde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



## 6. BLOK TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI



Resim 6.46: Arduino UNO ile Ultrasonik Sensör uygulaması Breadboard çizimi

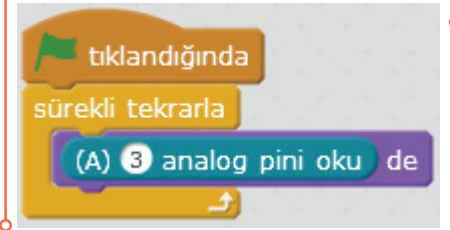


Resim 6.47: Arduino UNO ile Ultrasonik Sensör uygulaması örneği

### 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.

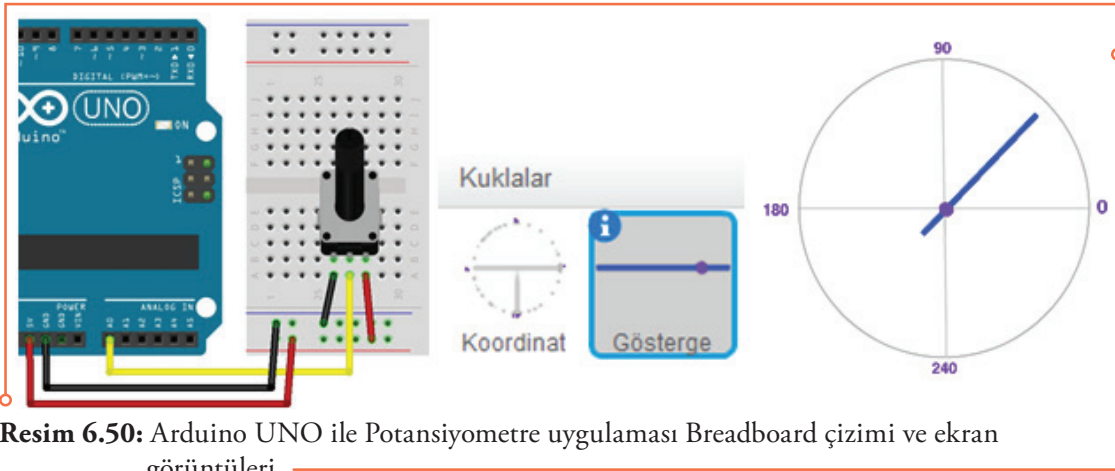
Aşağıda verilen örnekte A1 pinine bağlı bir potansiyometre kullanılarak 3600'lik bir açıölçer uygulaması yer almaktadır. Potansiyometrenin hareketi ile çizilen koordinat sistemi üzerinde gösterge istenen açıda ayarlanabilmektedir. Arduino ile okunan analog değerler 0 ile 1023 arasında olduğu, koordinat 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 6.48: Analog pimlerin ayarlanması ve okunması



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.

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 çalıştırdığı 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.

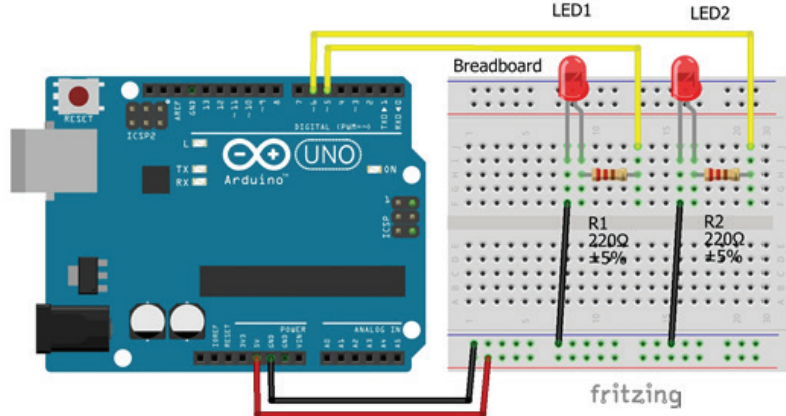


**Resim 6.51:** PWM Dijital kontrol pimlerin ayarlanması ve okunması işlemi



**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 kullanılacak 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 kapağıya bağlı, seçilen kanaldaki servo motorun açısını (0, 45, 90, 135 veya 180) derece yapmak için kullanılır.
kart ledler hepsi kırmızı yeşil mav	Robot kontrol kartı üzerinde bulunan RGB ledlerin renklerini ayarlamak için kullanılır.
Kapı1 led hepsi kırmızı yeşil mav	Genişleme kaplarına bağlı bulunan RGB ledlerin (4 adet) renklerini ayarlamak için kullanılır.
Kapı1 şerit led Kanal2 hepsi kırmızı yeşil mav	Genişleme kaplarına bağlı bulunan şerit RGB ledlerin (4 adet), seçilen kanaldaki renklerini ayarlamak için kullanılır.
ses tonunu C4 notasında Yanmı 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 displayde 100 yaz	Belirtilen kapağıya bağlı 7 parçalı displaye, belirtilen sayıyı yazdırmak için kullanılır.

## 6. BLOK TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI

Kapı3 ışık algılayıcı Aç	Kapı 3 veya 4'te bulunan ışık algılayıcıyı açmak veya kapatmak için kullanılır.
Kapı1 kamera perdesini Basıldı yap	Belirtilen kapıya bağlı kamera perdesini basıldı, bırak, odaklan veya kaydır yapmak için kullanılır.
Kapı1 mini fan saat yönünde döndür	Belirtilen kapıya bağlı mini fanı saat yönünde, saat yönünün tersinde döndürmek veya durdurmak için kullanılır.
ışık algılayıcı kartta ışık sensörü değeri	Robot kontrol kartı üzerinde bulunan veya kapılara bağlı olan ışık sensorunun değeri ile ilgili işlem yapmak için kullanılır.
basıldı düğmesi basıldığında	Basıldı düğmesine basıldığında veya serbest bıraktığında programı çalıştırmak için kullanılır.
basıldı düğmesi	Basıldı düğmesine basıldığında veya serbest bıraktığında işlem yapmak için kullanılır.

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

Kapı3 ultrasonik algılayıcı mesafesi	Belirtilen kapıya bağlı ultrasonik algılayıcı mesafesi ile ilgili işlemler yapmak için kullanılır.
Kapı2 çizgi izleyen	Belirtilen kapıya bağlı çizgi izleme algılayıcılara ilgili işlemler yapmak için kullanılır.
Kapı2 çizgi izleyen solTarafta siyah ise	Belirtilen kapıya bağlı sağ veya sol çizgi izleme algılayıcılara ilgili işlemler yapmak için kullanılır.
Kapı3 joystick X-ekseni	Belirtilen kapıya bağlı joystick'in x ve y eksenini ile ilgili işlemler yapmak için kullanılır.
Kapı3 potansiyometre	Belirtilen kapıya bağlı potansiyometre ile ilgili işlemler yapmak için kullanılır.
Kapı3 ses algılayıcı	Belirtilen kapıya bağlı ses algılayıcı ile ilgili işlemler yapmak için kullanılır.
Kapı1 limit anahtar Kanal1	Belirtilen kapıya bağlı limit anahtar ile ilgili işlemler yapmak için kullanılır.
Kapı3 sıcaklık Kanal1 °C	Belirtilen kapıya bağlı sıcaklık algılayıcı ile ilgili işlemler yapmak için kullanılır.
Kapı2 pır hareket algılayıcı	Belirtilen kapıya bağlı pır hareket algılayıcı ile ilgili işlemler yapmak için kullanılır.
3-eksenli jiroskop X-ekseni açısı	Belirtilen kapıya bağlı 3 eksenli jiroskopun x, y ve z eksenleriyle ilgili işlemler için kullanılır.
Kapı1 nem algılayıcı nem	Belirtilen kapıya bağlı nem ve sıcaklık algılayıcıyla ilgili işlemler için kullanılır.
Kapı3 alev algılayıcı	Belirtilen kapıya bağlı alev algılayıcıyla ilgili işlemler için kullanılır.
Kapı3 gaz algılayıcı	Belirtilen kapıya bağlı gaz algılayıcıyla ilgili işlemler için kullanılır.
Kapı1 pusula	Belirtilen kapıya bağlı pusula algılayıcıyla ilgili işlemler için kullanılır.
Kapı1 dokunma algılayıcı	Belirtilen kapıya bağlı dokunma algılayıcıyla ilgili işlemler için kullanılır.
Kapı3 buton key1 basıldı	Belirtilen kapıya bağlı butonlarla ilgili işlemler 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 kullanılacak 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 kullanılabilir. 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 metnin (yanıt)

uzunluđu tespit edilmekte ve bu deđer 6 ile arpılarak buna “yer” bilgisi eklenmektedir. Her bir karakter yatayda 6 sıra LED kullandıđı için 6 ile arpılmaktadır. Yer bilgisi için “yer” adında bir deđişken oluşturulmuş olup, deđişkeninin sağdan başlanarak (“yer” 16 olsun blođu) her harf için -1 azaltılarak yeni konumuna yerleřtirilmesi sağlanmıştır. Yanıtın kayarak gösterilmesi için kapı 4’de x:yer y:0 konumunda yanıt harflerini göster blođu kullanılmıştır.

```

when clicked
  Lütfen metni giriniz! diye sor ve bekle
  yer , 16 olsun
  sürekli tekrarla
    yanıt in uzunluđu * 6 + yer defa tekrarla
    Kapı4 de x: yer y: 0 konumunda yanıt harf(ler)ini göster
    yer 'i -1 arttır
  yer , 16 olsun
  1 saniye bekle
  
```

**Resim 6.57:** 8\*16 LED Matris kullanım örneđi

**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.

```

when clicked
  sürekli tekrarla
    Kapı1 Kanal1 servo açısını 90 yap
    4 saniye bekle
  
```

**Resim 6.58:** Servo motor kullanım örneđi

**7 Segment Ekran Kullanımı:** 7 segmentli ekranlar, genellikle hızı, zamanı, algılayıcıların deđerini 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.

```

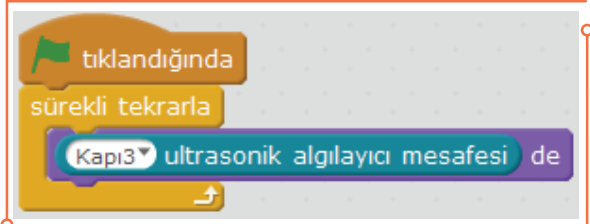
when clicked
  sürekli tekrarla
    Kapı1 7 paralı displayde 100 yaz
    2 saniye bekle
  
```

**Resim 6.59:** Servo motor kullanım örneđi

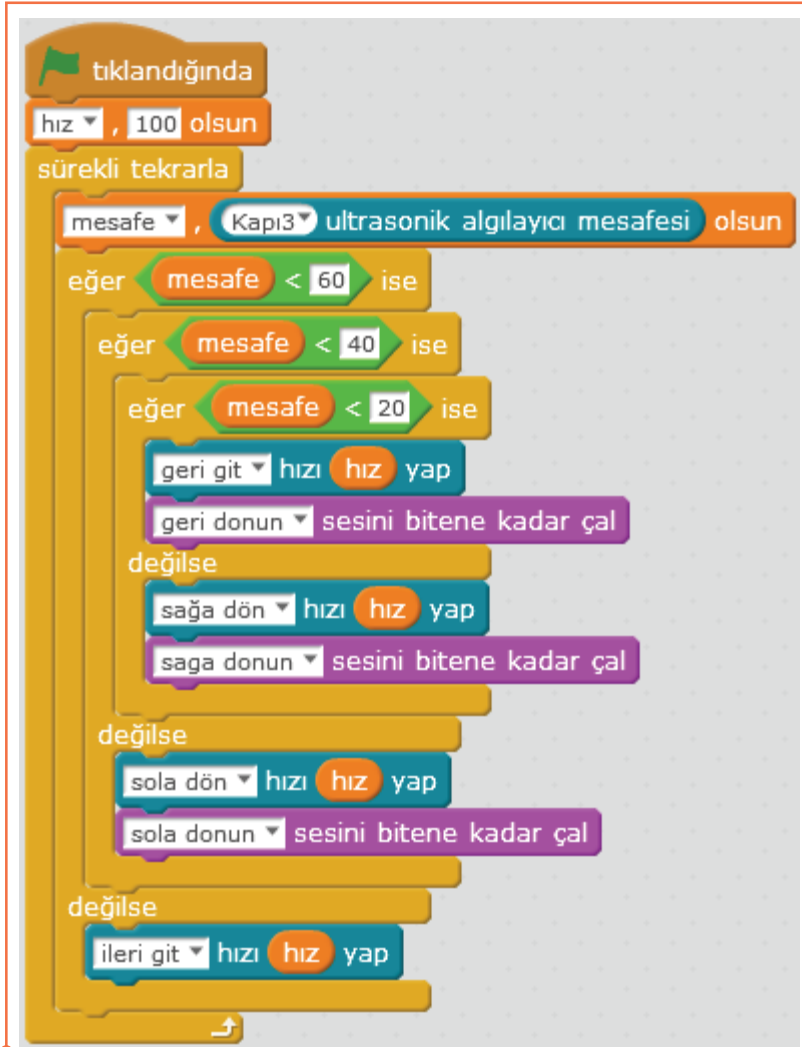


**Ultrasonik Algılayıcı Kullanımı:** Ultrasonik algılayıcılar genellikle algılayıcı ile engeller arasındaki mesafeyi ölçmek için kullanılırlar. Direkt olarak (Kapı 1, 2, 3 veya 4) bağlanabilirler. Yandaki örnekte engele olan uzaklığın ölçülmesi gösterilmiştir.

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 6.60:** Ultrasonik algılayıcı ile mesafe ölçme örneği



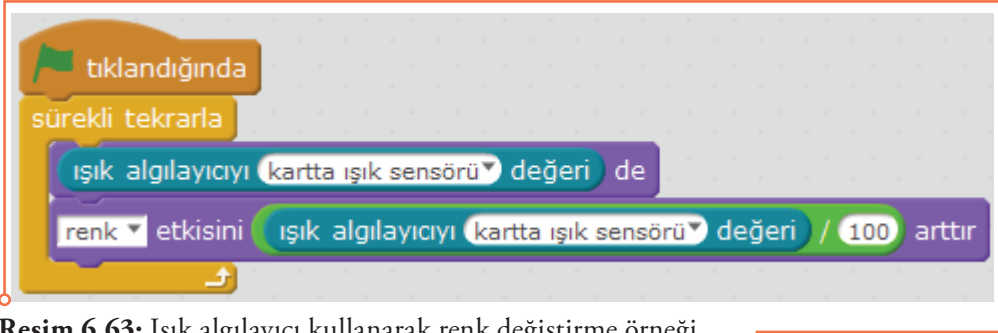
**Resim 6.61:** Ultrasonik algılayıcı kullanarak engele çarpmadan dolaşma uygulaması

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

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 6.62:** Işık algılayıcı kullanım örneği

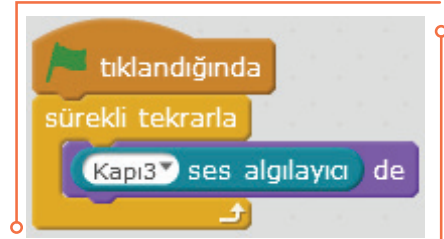


**Resim 6.63:** Işık algılayıcı kullanarak renk değiştirme örneği

**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.

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 6.64:** Ses algılayıcı kullanım örneği



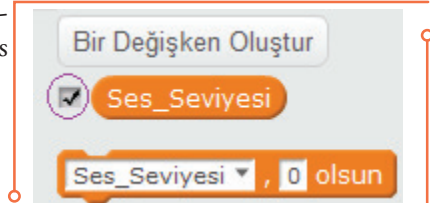
**Resim 6.65:** Ses seviyesine göre ekrandaki kuklayı zıplatan uygulama



**Resim 6.66:** Ses seviyesine göre ekrandaki kuklayı zıplatan uygulama

## 6. BLOK TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI

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 nesnelere 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.



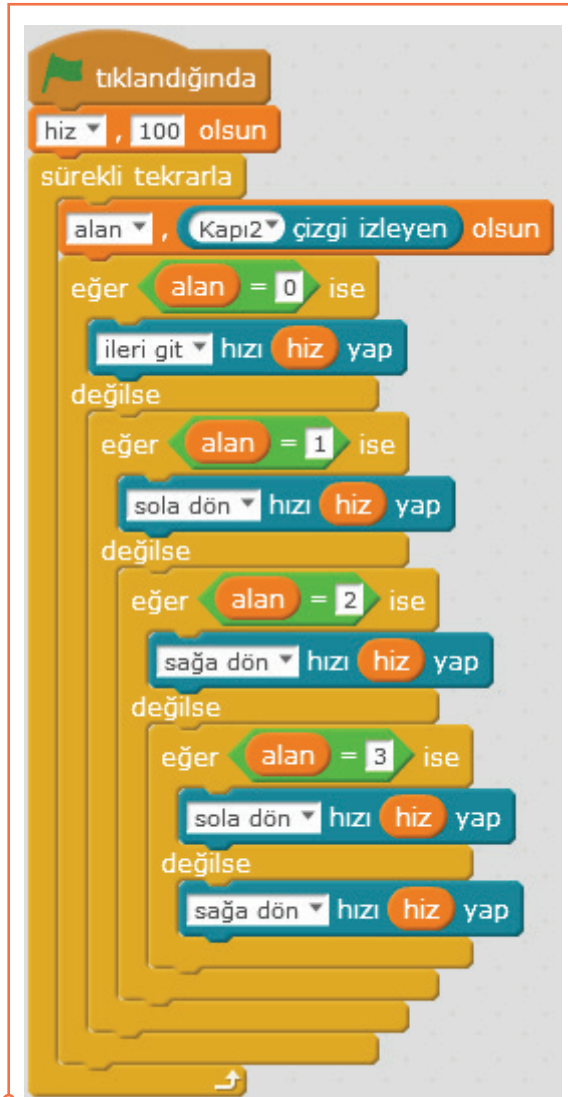
Resim 6.69: Robotun çizgi algılayıcıları ile çizginin neresinde olduğunu belirleyen uygulama örneği

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 “hız” ve “alan” adında iki değişken oluşturulmuştur. “hız” 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 üretil-  
diği 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.



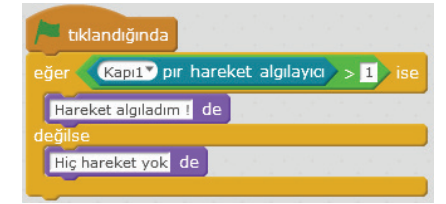
Resim 6.72: Potansiyometre kullanım örneği

**Sıcaklık Algılayıcı Kullanımı:** Sıcaklık algılayıcı uç noktadaki 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.



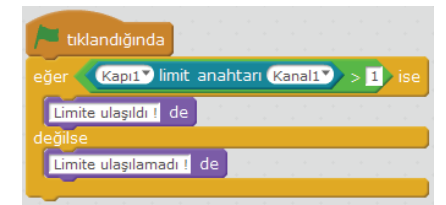
Resim 6.74: PIR Hareket algılayıcı kullanım örneği

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



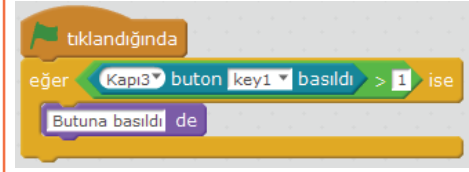
Resim 6.75: Dokunma algılayıcı kullanım örneği

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



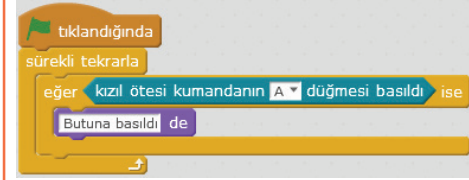
Resim 6.76: Limit anahtarı kullanım örneği

**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.



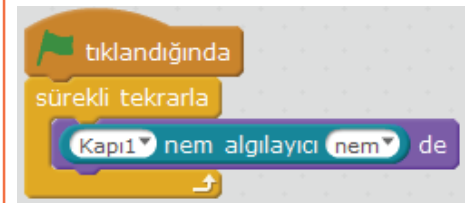
Resim 6.77: Buton kullanım örneği

**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.



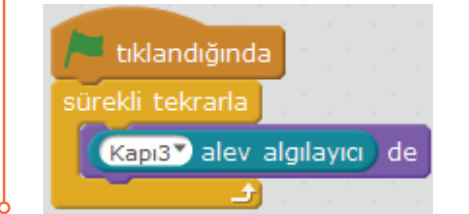
Resim 6.78: Kızılötesi kumanda kullanım örneği

**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.



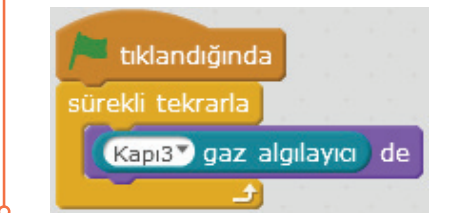
Resim 6.79: Nem ve sıcaklık algılayıcı kullanım örneği

**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 eksen 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 / Araştırılmalı / 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. Deęerlendirme Soruları

- 1. mBlock ykl bilgisayarla Arduino destekli robot arasında baęlantının oluřturulması iin ařaęıdaki seeneklerden hangisi kullanılabilir?**
  - a) USB port
  - b) Aę zerinden baęlantı
  - c) Bluetooth modl ile
  - d) 2.4 Ghz Dongle modl
  - e) Hepsi
- 2. mBlock ykl bilgisayarla Arduino destekli robotun gncellenmesi ve programlanması iin ařaęıdaki seeneklerden hangisi kullanılabilir?**
  - a) USB port
  - b) Aę zerinden baęlantı
  - c) Bluetooth modl
  - d) 2.4 Ghz Dongle modl
  - e) Hepsi
- 3. Girilen deęerleri alan veya programın alıřmasıyla bazı deęerlerin atandıęı veri tutucularından oluřan temel programlama yapılarına ne ad verilir?**
  - a) Fonksiyon
  - b) Liste
  - c) Deęiřken
  - d) Dizi
  - e) Prosedr
- 4. ok sayıda deęiřkenle alıřmak iin oluřturulmuř temel programlama yapılarına ne ad verilir?**
  - a) Fonksiyon
  - b) Liste
  - c) Deęiřken
  - d) Dizi
  - e) Prosedr
- 5. Program akıřı iinde tekrarlayan ifadelerin her seferinde tekrar tekrar yazılması yerine, bir kere ayrı bir yerde yazılıp tekrarlanan her yerde kullanmak iin ařaęıdakilerden hangisinin yapılması uygundur?**
  - a) Prosedr oluřturmak
  - b) Dizi oluřturmak
  - c) Liste oluřturmak
  - d) Blok oluřturmak
  - e) Fonksiyon oluřturmak



6. Prosedürlere değer taşıyan değişkenlere ne adı verilir?
- Blok
  - Fonksiyon
  - Dizi
  - Liste
  - 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?
- Hareket alt başlığı altında verilen blok gurupları
  - Görünüm alt başlığı altında verilen blok gurupları
  - Olaylar alt başlığı altında verilen blok gurupları
  - Kontrol alt başlığı altında verilen blok gurupları
  - İşlemler alt başlığı altında verilen blok gurupları
8. Arduino blokları ile diğer blokların Arduino uygulamalarında birlikte kullanılabilmesi için mBlock ortamında aşağıdakilerden hangisinin yapılması gereklidir?
- Aygıt Yazılımı Güncellemesi yapılmalıdır
  - Arduino Programı blokunun üzerine tıklanmalıdır
  - Arduino'ya Yükle seçeneğini üzerine tıklanmalıdır
  - Varsayılan Program sıfırlanmalıdır
  - Arduino sürücüsü yüklenmelidir
9. Arduino uygulamalarında elektronik bileşenlerin lehim yapmadan kullanılabilmesi için aşağıdakilerden hangisi kullanılmalıdır?
- Soket
  - Konnektör
  - Delikli kart
  - Breadboard
  - Pertinaks
10. Aşağıdakilerden hangisi blok tabanlı robot programlama yazılımlarından ve ortamlarından biri değildir?
- miniBlog
  - ModKit
  - Ardublock
  - Snap4Arduino
  - Srach for Arduino (S4A)

## 7. METİN TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI

Bu bölümün sonunda,

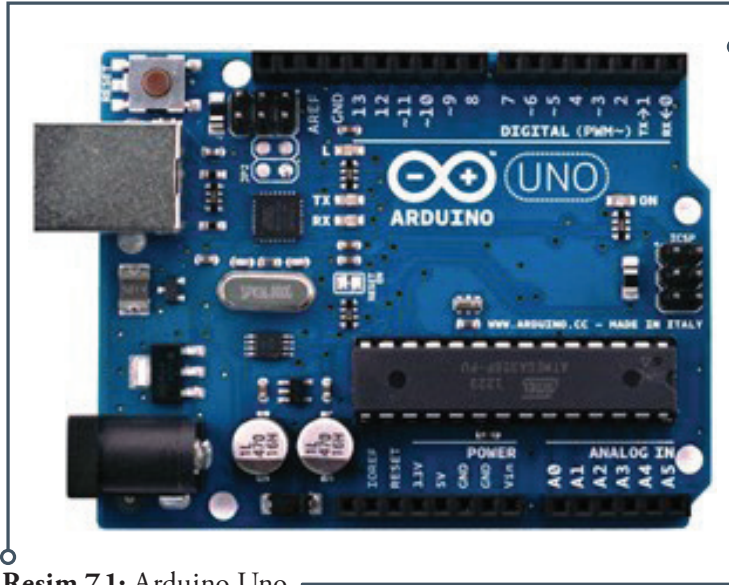
- ✓ Arduino tümleşik geliştirme ortamının temel yapısını tanımlayabilecek,
- ✓ Arduino tümleşik geliştirme ortamının kullanım özelliklerini açıklayabilecek,
- ✓ Arduino UNO geliştirme kartının kullanım özelliklerini ve yapısını açıklayabilecek,
- ✓ Geliştirme yapılan bilgisayarla Arduino kartları arasında 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 düzenleyebilecek,
- ✓ 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.

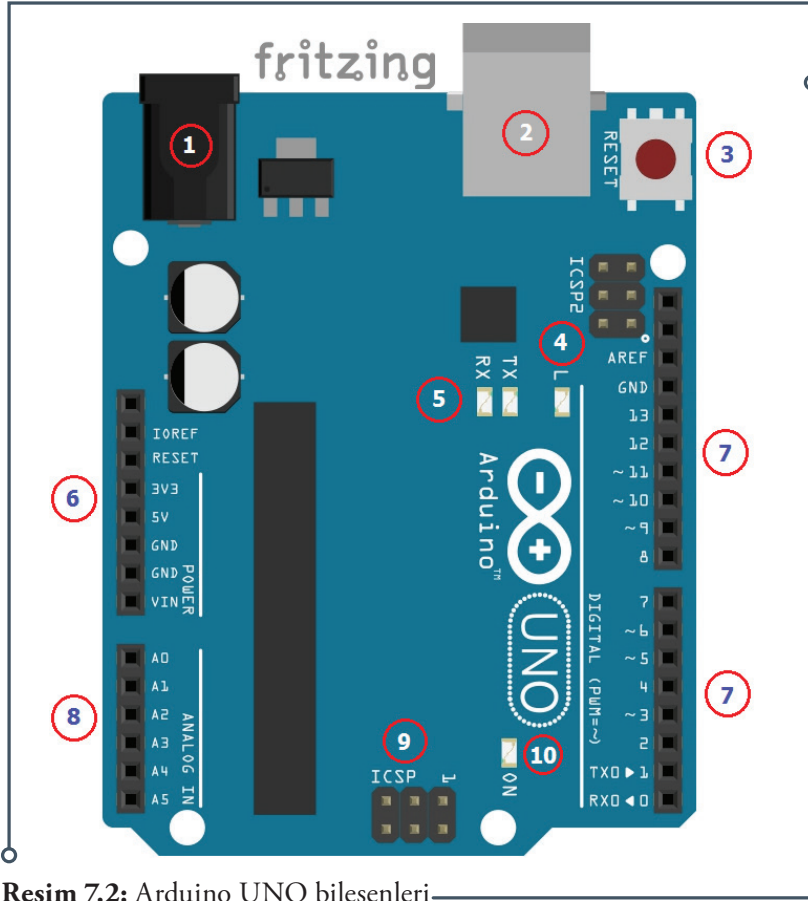


Resim 7.1: Arduino Uno

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 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 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.



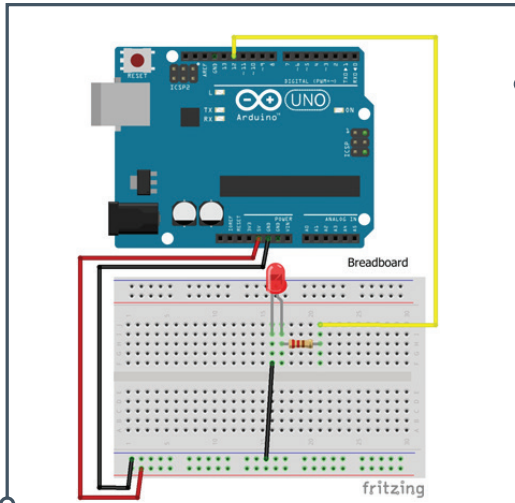
Resim 7.2: Arduino UNO bileşenleri

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.

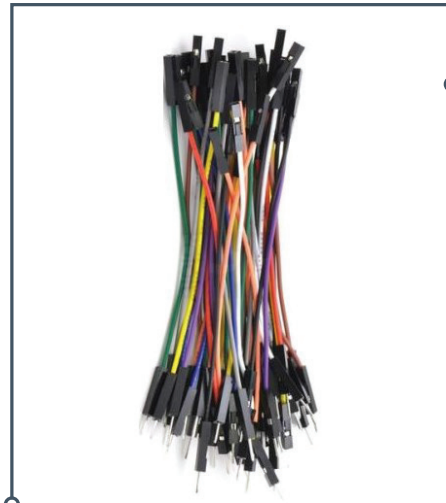
5. **RX (Receiving) ve TX (Transmitting) LED'leri:** Seri haberleşme için kullanılan RX ve TX pininin durumunu gösteren LED'lerdir. Veri alışverişi olduğunda bu LED'ler yanar.
6. **Güç Bağlantıları:** Bu kısımda güç çıkışları yer almaktadır.
7. **Dijital Giriş /Çıkış Pinleri:** Yanında ~ işareti bulunan 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. **Analog 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)

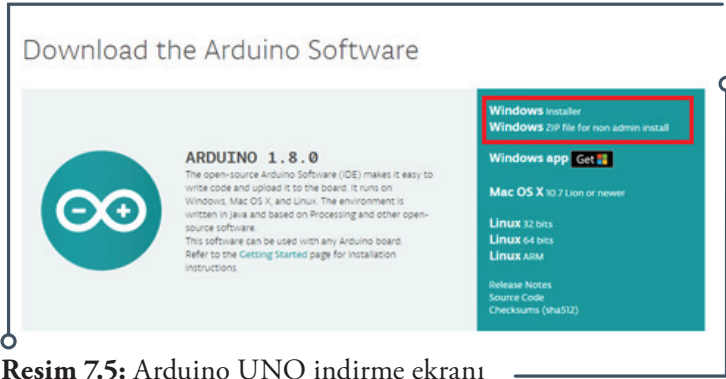
Arduino IDE; kod yazım editörü, tümleşik bir derleyici, yorumlayıcı ve hata ayıklayıcı olarak görev yapan, aynı zamanda derlenen programı karta yükleme işlemini de yapabilen, her platformda çalışabilen Java programlama dilinde yazılmış bir uygulamadır. Arduino tümleşik geliştirme ortamı (IDE); Arduino bootloader (Optiboot), Arduino kütüphaneleri, AVR Dude (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 yazılımını 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.

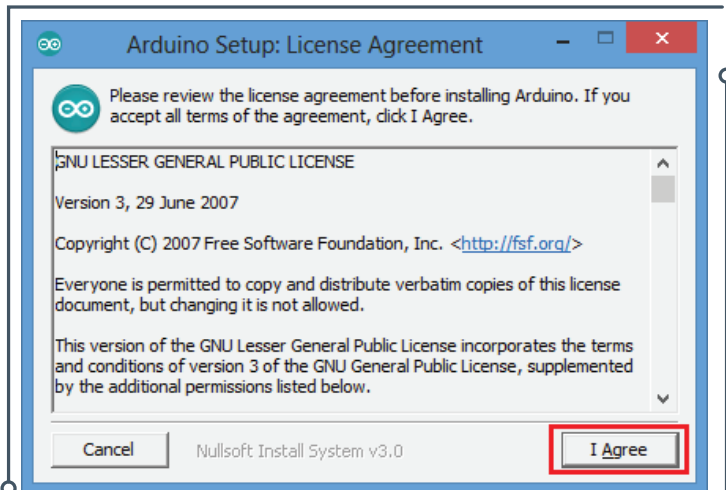
## 7.5. Arduino IDE Yazılımının Yüklenmesi

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 zip 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.

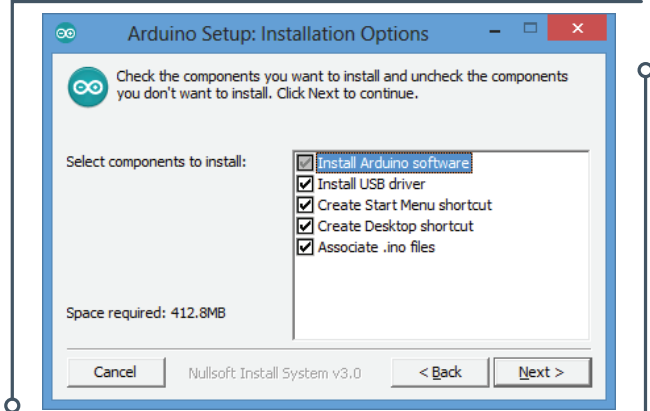


Resim 7.5: Arduino UNO indirme ekranı



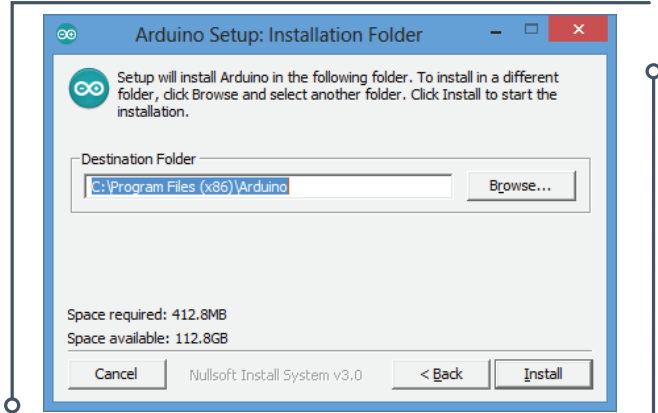
Resim 7.6: Arduino IDE kurulum aşaması

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 sürücülerini de kurulduğu için Arduino destekli bütün kartlar otomatik olarak tanınacaktır. Ayrıca tanıtılmasına gerek yoktur.

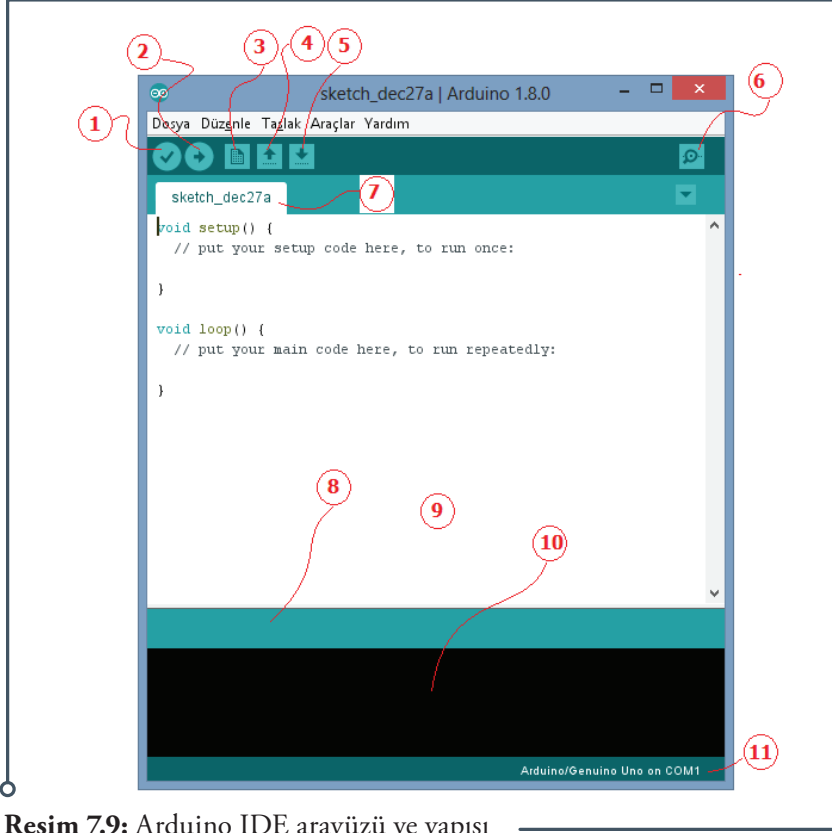


Resim 7.8: Arduino IDE kurulum klasörü

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

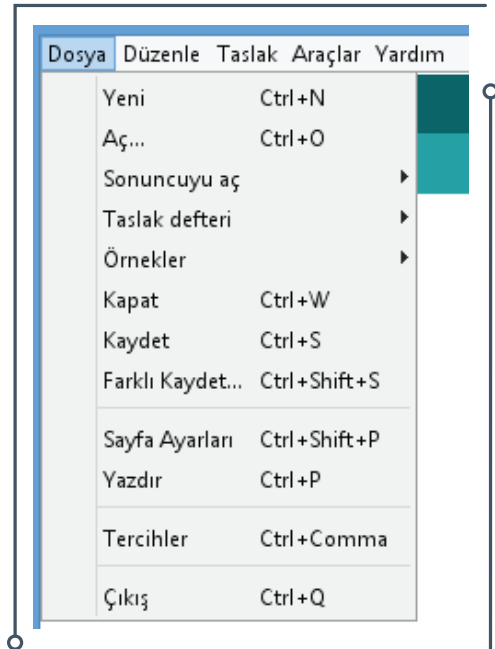
1. **Kontrol Et:** Yazılan kodları derler ve hataları bulur.
2. **Yükle:** Yazılan programı Arduino kartına yükler.
3. **Yeni:** Yeni çalışma sayfası açar.
4. **Aç:** Kayıtlı bir programı açar.
5. **Kaydet:** Yazılan programı kaydeder.
6. **Seri Port Ekranı:** Arduino ile seri iletişim yaparak ekran 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

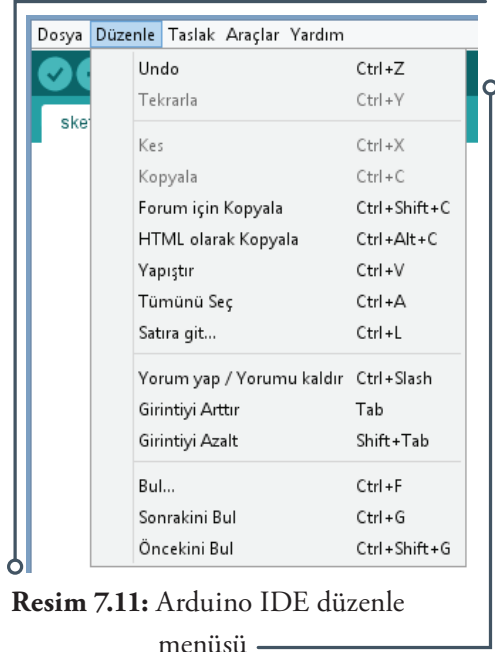
Dosya menüsünde temel komutlar yer almaktadır. Ayrıca “Örnekler” komutu ile program kütüphanesinde yer alan kodlar incelenebilmektedir. Bu bölümde çok sayıda örnek bulunmakta, gerekli bağlantılar yapılarak görsel olarak incelenebilmekte, küçük değişiklikler yaparak etkisi görülebilmektedir. Bu bölüm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



**Resim 7.10:** Arduino IDE dosya menüsü

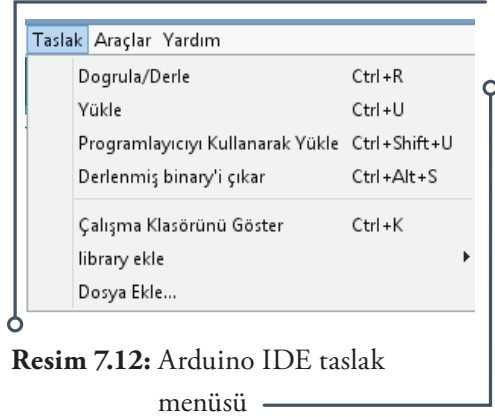


Düzenle menüsünde standart işlemlerin dışında “Forum için Kopyala” ve “HTML olarak Kopyala” seçenekleri kullanılarak internet ortamında, Arduino arayüzünde yazılan biçimde (renk- yazı tipi) paylaşabilmektedir.



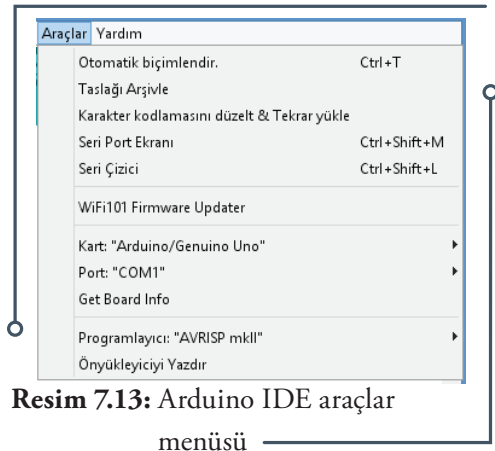
**Resim 7.11:** Arduino IDE düzenle menüsü

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



**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 Tümlleşik Geliştirme Ortamının Temel Özellikleri

- ✓ Arduino IDE tümlleşik 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 Tümlleşik 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()	<b>3. Aritmetik Operatörler</b>	<b>6. İşaretçi Operatörler</b>
<b>1. Kontrol Yapıları</b>	= 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	<b>4. Karşılaştırma Operatörleri</b>	<b>7. Bitsel Operatörler</b>
<b>2. Söz Dizimi</b>	== (eşit eşit) != (eşit değil) < (küçük) > (büyük) <= (küçük eşit) >= (büyük eşit)	& (Bitsel Ve)   (Bitsel Veya) ^ (Bitsel Xor) ~ (Bitsel Değil) << (Bitshift Sol) >> (Bitshift Sağ)
; {} // /**/ #define #include	<b>5. Boolean Operatörleri</b>	<b>8. Birleşik Operatörler</b>
	&& (ve)    (veya) ! (değil)	++ (arttırma) -- (azaltma) +- (Birleşik arttırma) -= (Birleşik çıkarma) *= (Birleşik çarpma) /= (Birleşik bölme) %= (Birleşik mod) &= (Bitsel Lojik Ve)  = (Bitsel Lojik Veya)

Tablo 7.1: Arduino IDE'nin program yapısı

B. DEĞİŞKENLER	C. FONKSİYONLAR	
<b>1. Sabitler</b>	<b>1. Dijital Giriş Çıkışlar</b>	<b>8. Rastgele Sayılar</b>
HIGH   LOW INPUT   OUTPUT   INPUT_PULLUP LED BUILTIN True   false integer constants floating point constants	pinMode(pin,mod) digitalWrite(pin,değer) digitalRead(pin)	randomSeed() random()
<b>2. Veri Tipleri</b>	<b>2. Analog Giriş Çıkışlar</b>	<b>9. Bit ve Bayt'lar</b>
void boolean char unsigned char byte int unsigned int word long unsigned long short double string – char array substring – object array	analogRead(pin,mod) analogWrite(pin,değer) analogReference(tip) analogReadResolution () analogWriteResolution ()	lowByte() high(Byte) bitRead() bitWrite() bitSet() bitClear() bit()
<b>3. Dönüşümler</b>	<b>3. Gelişmiş Giriş Çıkışlar</b>	<b>10. Harici İnterruptlar (Kesmeler)</b>
char() byte() int() word() long() float()	tone() noTone() shiftOut() shiftIn() pulseIn()	attachInterrupt(interrupt, function, mode) detachInterrupt(interrupt)
<b>4. Değişken Kapsamları</b>	<b>4. Gecikmeler</b>	<b>11. İnterruptlar (Kesmeler)</b>
static volatile const	delay(milisaniye) unsigned long millis() delay(Microse conds (mikrosaniye)	interrupts() noInterrupts()
<b>5. Yardımcılar</b>	<b>5. Matematiksel İşlevler</b>	<b>12. Seri Haberleşme</b>
PROGMEM Sizeof()	min(x,y) max(x,y) abs(x) constrain(x, a, b) map(value, fromLow fromHigh, toLow, toHigh) pow(base, esponent) sqrt(x)	Serial.begin(hız) int Serial.available() int Serial.read() Serial.flush() Serial.print(data) Serial.println(data)
	<b>6. Trigonometri İşlevleri</b>	<b>13. Haberleşme Protokolleri</b>
	sin(rad) cos(rad) tan(rad)	I2C Veri Yolu SPI Veri Yolu
	<b>7. Karakterler</b>	
	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 Tümlleşik 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 bacağıının) 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 arasında 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 butonPin = 4; // butonu 4. pine tanımlandı

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

void loop()
{
  if (digitalRead(butonPin) == 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 ulaşıp ulaşılamadığını test eder. Parantez içindeki ifadeler karşılanıyorsa, parantez içindeki ifadeler çalıştırılır. Değilse, program kod üzerinde atlanır. Aşağıda verilen örnekte Arduino

```
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

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 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 ulaşıyorsa bir eylem, ulaşılmıyorsa başka bir eylem yapılır. Aşağıda verilen örnekte Arduino UNO'nun 4 numaralı dijital pinine bağlı bir LED'in yanık kalma sü-

```

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

resini 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

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

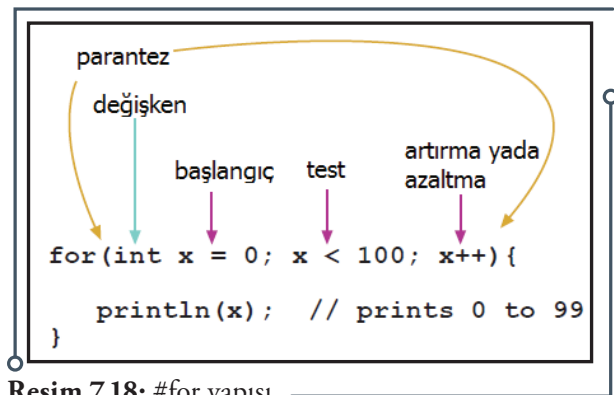
void setup()
{
  pinMode(Led, OUTPUT); // LED çıkış olarak tanımlanıyor
}

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 çıkışı 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 çıkacaktır.

```
for (x = 0; x < 255; x ++)  
{  
    analogWrite(PWPin, 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(PWPin, 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 istenen 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 Virgöl

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

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

**Resim 7.28:** ;noktalı virgöl  
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 slashdan 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 virgöl 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 (<...>) şeklinde 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 değerlendirilmesini ve eşit işaretin solundaki değeri saklamasını söyler.

```
int algıDeg; // algıDeg adını taşıyan bir tamsayı değişkeni
algıDeg = analogRead(1); // analog pin 1 in giriş gerilimi algıVal 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 kullanımını aşağıda verilmiştir. Genellikle if içerisinde karşılaştırma yaparken kullanılan operatörlerdir. "if" karşılaştırma operatörüyle birlikte kullanıldığında, belli bir değer 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: 111111110011000 = -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: 000000000101000 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****++ (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 arttırır 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.

<pre> 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. </pre>	<pre> 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 </pre> <p>X: herhangi bir değişken türü Y: herhangi bir değişken türü veya sabit</p>
--	---

**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 Tümeleşik 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 kullanılabilir. Dijital pinlerin elektriksel davranışı pinMode() ile değiştirilebilir.

```
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 yanlış 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 yanlış
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) kullanılabilir.

Decimal kullandığımız 10luk sayı sistemidir.

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

Binary ikili sayı sistemidir. 0 ve 1 kullanılmaktadır.

$$B101 == ((1 * 2^2) + (0 * 2^1) + 1) \text{ decimal}$$

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

$$0101 == ((1 * 8^2) + (0 * 8^1) + 1) \text{ decimal}$$

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	"O"	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

$$2.34E5=2.34 * 10^5 \text{ 234000; } 67e-12= 67.0 * 10^{-12} .000000000067$$

## 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 ikisinde 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 ile 4,294,967,295 arasında değişen 4 baytlık (32 bit) bir pozitif değeri 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 = 0 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 değışkене 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 değışken adı
// val -o değışkене 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 dizilerini 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 kullanılacak 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 diziyeye 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.
- ✓ Tanımlanmış 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] yazılmaktadır. Örnekte "mySensVals" kümemize 1. eleman olarak 10 değeri atanmıştır. Yine aynı 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 sönmelerini 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 söndürülüyor
    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 oluşturulmuş 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 kontrolörünün 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 Tümlleşik Geliştirme Ortamının “Fonksiyon” Yapısı

### 7.10.1. Dijital Giriş Çıkışlar

#### pinMode()

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

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

Resim 7.86: pinMode() kullanım örneği

## digitalWrite()

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

```
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 Çıktılar

### 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.
- ✓ **INTERNALV1:** 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 pimdendiğeri 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 değerler 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 Çıktılar

#### 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 çalışıyorsa, tone() çağrısının etkisi olmayacaktır. Yandaki örneği uygulayınız.

#### 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 bittten 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.

```
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

```
// 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()

millis, 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 sönen 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 mikrosaniye duraklatıldı
  digitalWrite(outPin, LOW);  // pin kapalı
  delayMicroseconds(50);      // 50 mikrosaniye 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()

Sensör 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:** Sensörden gelen değer
- ✓ **fromLow:** Değerin geçerli aralığının alt sınırı
- ✓ **fromHigh:** Değerin geçerli aralığının üst sınırı
- ✓ **toLow:** Değerin hedef aralığının alt sınırı
- ✓ **toHigh:** Değerin 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 arasındadır 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 255'dir
```

**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 kosinüsünü 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 heksadesimal (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 üretici, 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 üretici, 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 Interrupt) 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 şekildedir:

**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.

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

```
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

**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 işlemi. 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( kontrol );
}
void loop() {
  void kontrol()
  {
    digitalWrite(13);
    digitalWrite(13) ^ 1 ;
  }
}
```

**Resim 7.110:** Timer Interrupt() kullanım örneği

## 7.11. Arduino Tümeleşik 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 normal 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.println() 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():** İlk 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 Tümlleşik Geliştirme Ortamında Seri Haberleşme Protokolleri

Dijital sistemlerde kablolu seri haberleşme ile ilgili birçok standart protokol bulunmaktadır. SPI ve I<sup>2</sup>C 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. I<sup>2</sup>C Veri Yolu

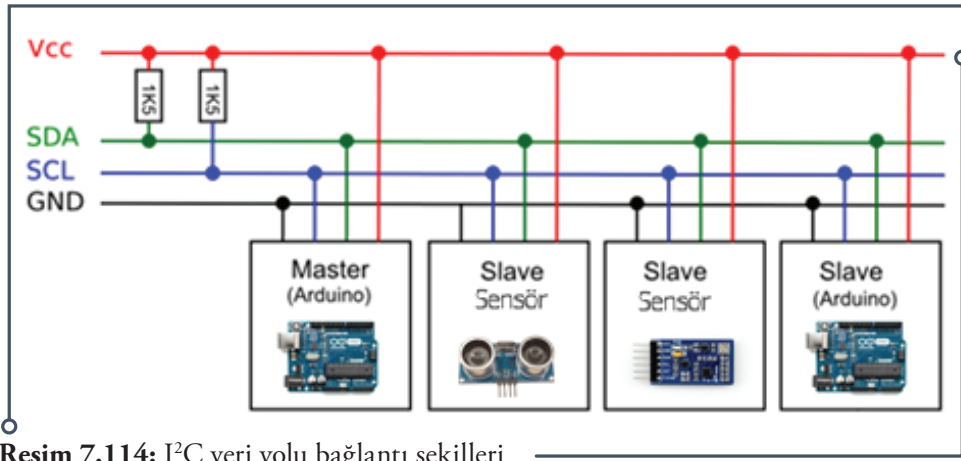
I<sup>2</sup>C (Inter-Integrated Circuit), oldukça hızlı veri aktarımına olanak tanıyan seri haberleşme türlerindedir. Bir arada çalışan, belirli aralıklarla birbiriyle haberleşen, çeşitli çevresel cihazların çok az harici donanım gereksinimiyle haberleşmelerini sağlar. Uzun mesafeli haberleşmelerde 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.

I<sup>2</sup>C 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

I<sup>2</sup>C 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:** I<sup>2</sup>C veri yolu bağlantı şekilleri

Aşağıdaki örnekte, robotik uygulamalarda yaygın olarak kullanılan MPU6050 3 eksenli gyro ve 3 eksen açısal ivme ölçer sensörünün I<sup>2</sup>C bağlantısı yapılarak test edilmesi sağlanmaktadır. Arduino Uno'da sda ve scl I<sup>2</sup>C 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

  // Açışal 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
}

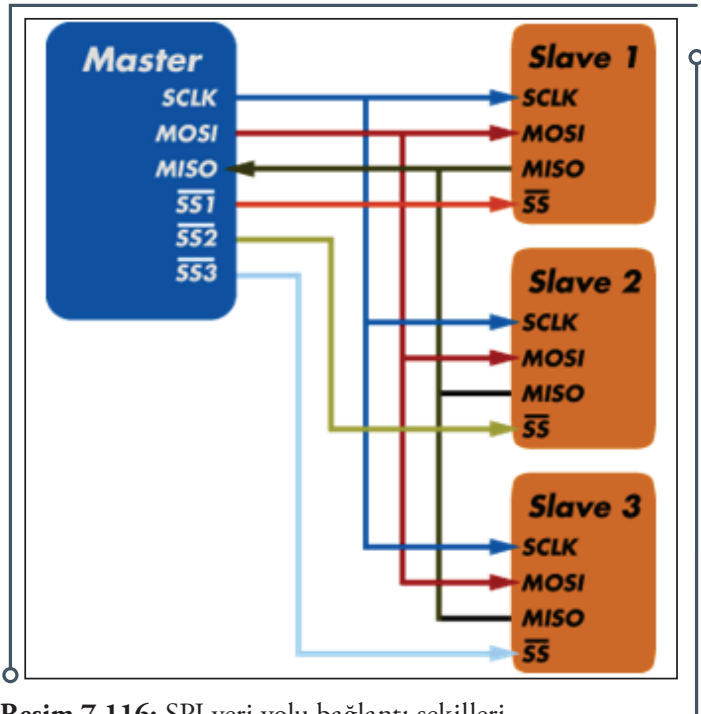
```

Resim 7.115: I<sup>2</sup>C 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 I<sup>2</sup>C ile oldukça benzerlik göstermektedir. Bir Arduino'nun diğer Arduino veya sensörlerle kısa mesafede haberleşmesini sağlamak için kullanılmaktadır. SPI protokolünde de I<sup>2</sup>C'de olduğu gibi bir adet Master cihaz bulunmaktadır. Bu cihaz hatta bağlı diğer çevresel cihazları kontrol etmektedir. SPI bağlantısı için 4 adet pin kullanılmaktadır.

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



Resim 7.116: SPI veri yolu bağlantı şekilleri

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 I<sup>2</sup>C'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 kullanılacak 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 kullanılır.

**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 yer almaktadır. 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 dizisi 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

```

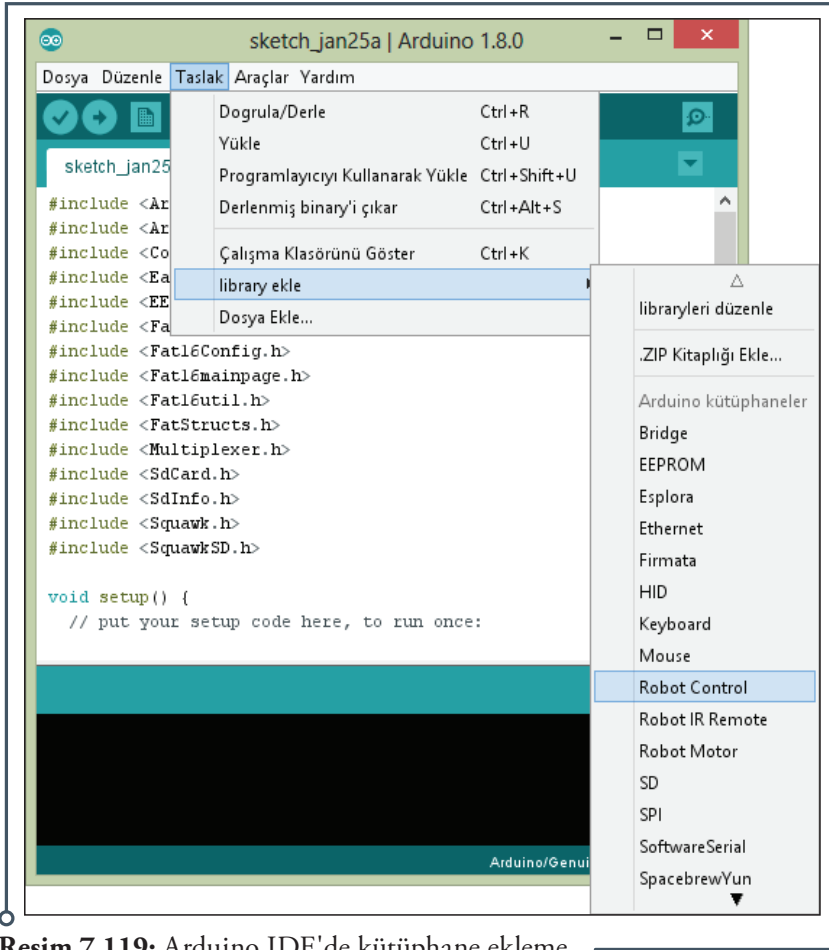
#include <SPI.h>
char buff [50];
volatile byte indx;
volatile boolean process;
void setup (void) {
  Serial.begin (115200); // Seri iletim Master ile aynı olmalı
  pinMode(MISO, OUTPUT); // Çıktı olarak ayarlanacak şekilde master'ı gönderiliyor
  SPCR |= _BV(SPE); // Köle modunda SPI'yi açılıyor
  indx = 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 elemanların Arduino ile kullanımı, o eleman tarafından yapılacak iş için gerekli kodlamanın da programcı tarafından yapılmasını gerektirmektedir. Bu gerekliliğin sağlanmasını, bu elemanlar ile Arduino programlamanın birleştirilmesi ve aralarında köprü görevi görecekteki 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<sup>2</sup>C 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<sup>2</sup>C 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 belenimini (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 bir araya 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 düşünerek 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 ulaşılabilir. Dökümanları inceleyerek bileşenlerin çalışma yapısı, elektriksel özellikleri ve mikrodenetleyici kartlarla kullanım şekillerini, bağlantı pinleri ve yapılarını öğrenebilirsiniz.

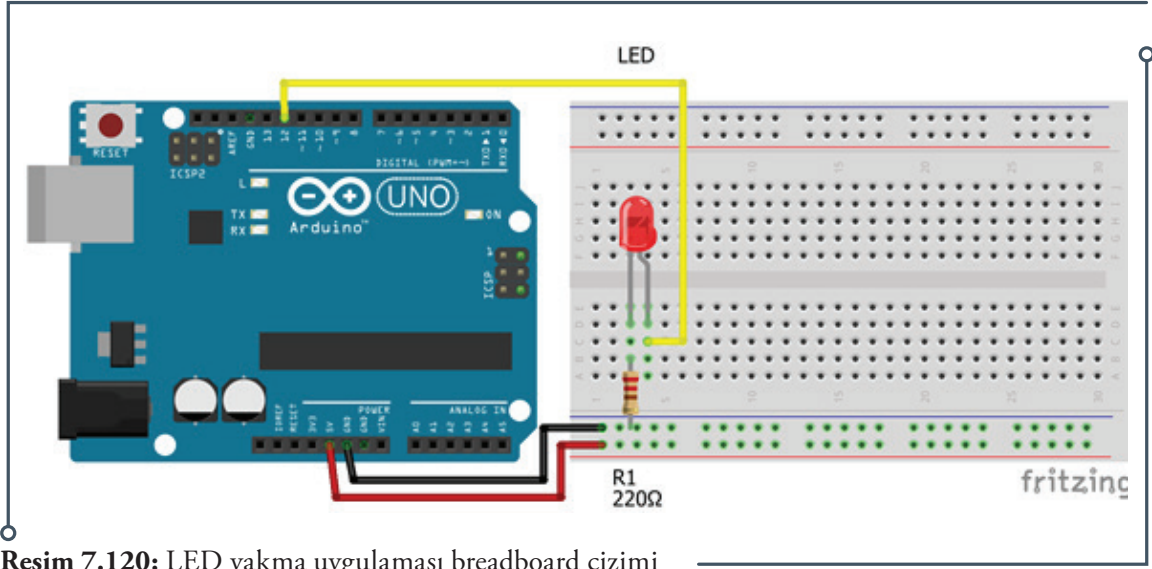
Eğer uygulamalarda kullanılan algılayıcıların arduino ile haberleşmesinde kütüphane kullanılacaksa veya bu zorunluysa uygun kütüphanenin çalışmaya eklenmesi gerektiği unutulmamalıdır. Zira eklenen kütüphane dosyaları algılayıcıdan gelen farklı yapıdaki bilginin okunması görevini üstlenerek kullanımı kolaylaştırmaktadır. Bu kütüphaneler uzmanlar tarafından hazırlanmakta ve algılayıcılardan gelen bilgiyi anlamlı hale getiren kod parçacıklarından meydana gelmektedir. Algılayıcılara ait Arduino kütüphane dosyalarına da İnternette yapılan kısa bir arama ile ulaşılabilir. 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 sınırlandıran ve gerilimi bölen, iki uçlu bir elemandır. R harfi ile sembolik olarak gösterilir ve birimi Ohm ( $\Omega$ )'dur. Örnek uygulamalarda LED'lerin ve gerekli 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

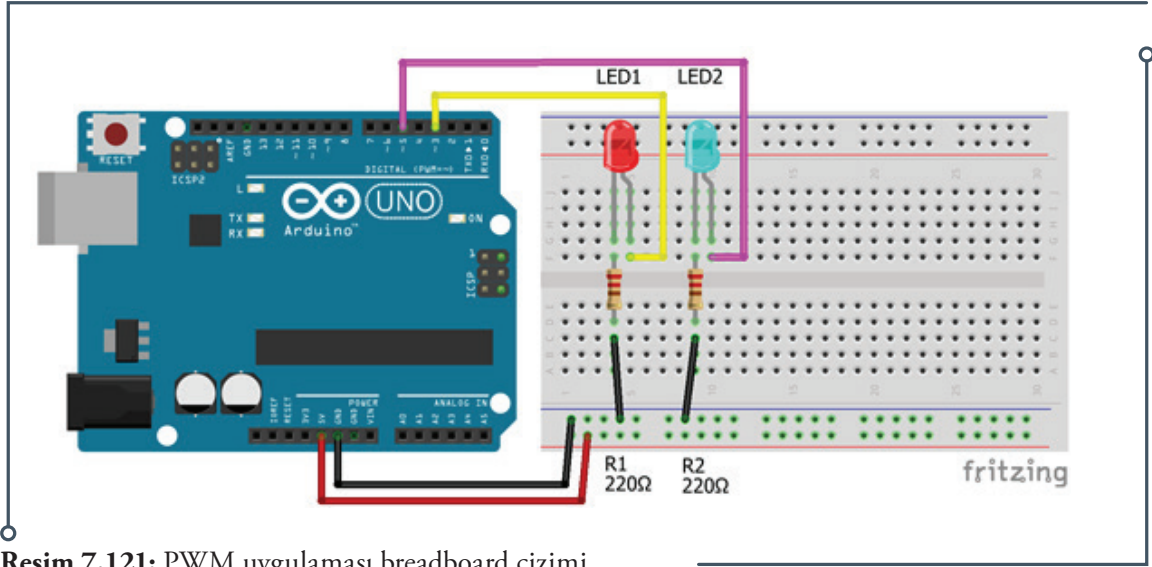
Arduino'nun 12 numaralı dijital pinlerine bağlı bir LED'in 1 sn aralıklarla yanıp sönmesini sağlayacak bir uygulama hazırlayınız. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir. LED'ler düşük güçte çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 220 ohm ( $\Omega$ ) 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  $\Omega$  direnç kullanılmıştır.

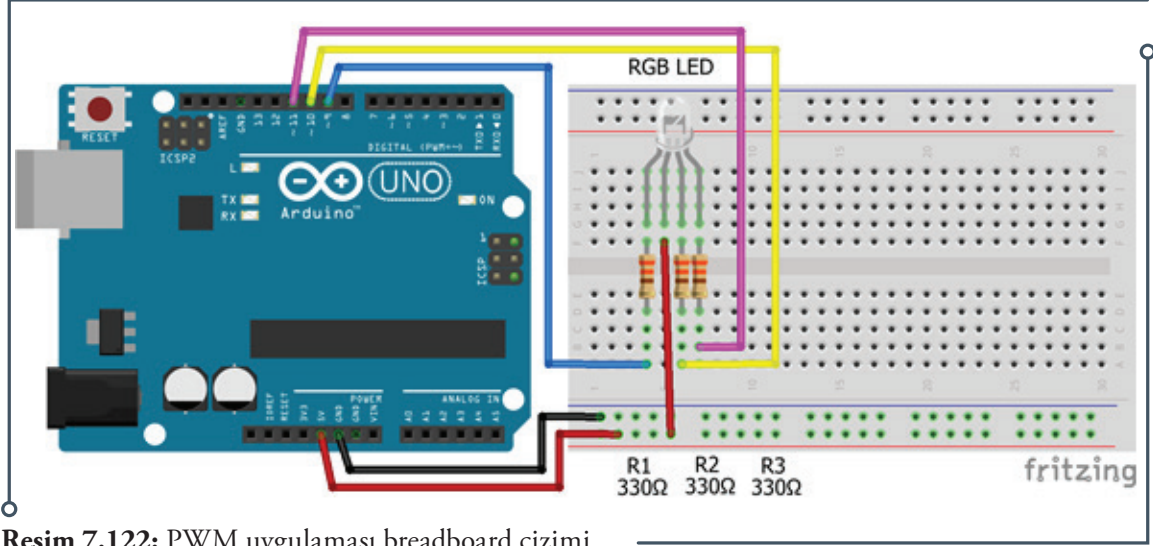


Resim 7.121: PWM uygulaması breadboard çizimi

### 7.14.3. RGB LED Uygulaması

RGB LED'in kırmızı Led pinini Arduino'nun 9, yeşil Led pinininin 10 ve mavi Led pinini 11 numaralı pwm pinlerine bağlayarak RGB LED'in kırmızı, yeşil, mavi, sarı, camgöbeği ve beyaz renklerde veriler

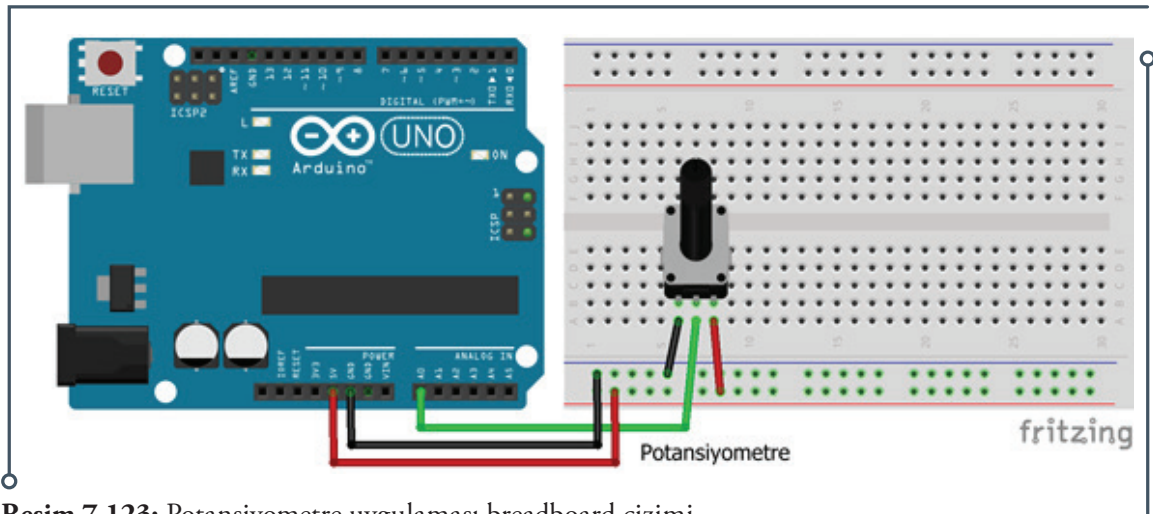
sırayla yanması sağlayacak bir uygulama hazırlayınız. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir RGB LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 3 adet  $330 \Omega$  direnç kullanılmıştır.



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 ( $k\Omega$ ) potansiyometre kullanılmıştır.

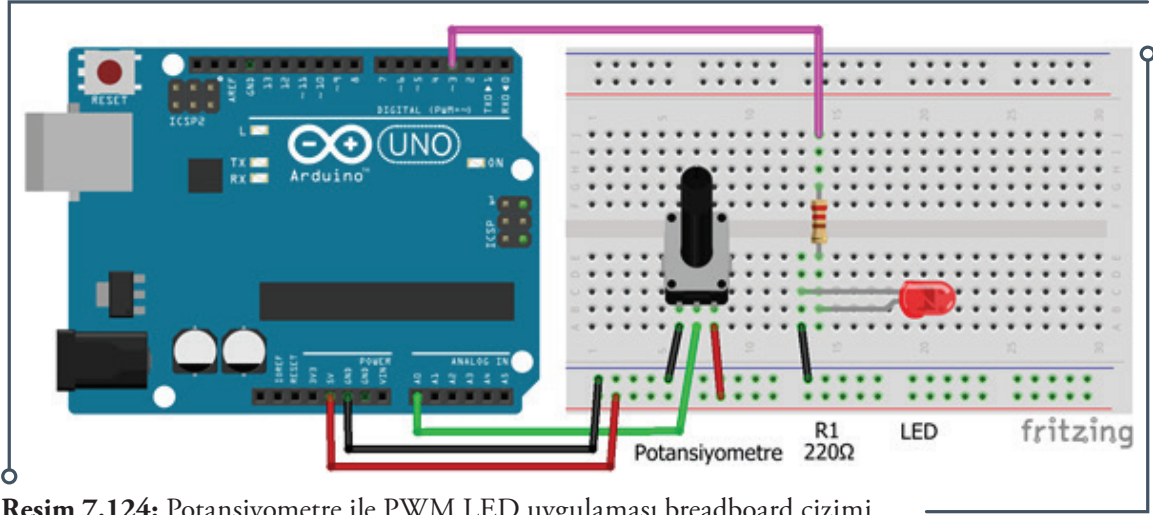


Resim 7.123: Potansiyometre uygulaması breadboard çizimi



### 7.14.5. Potansiyometre ile PWM LED Uygulaması

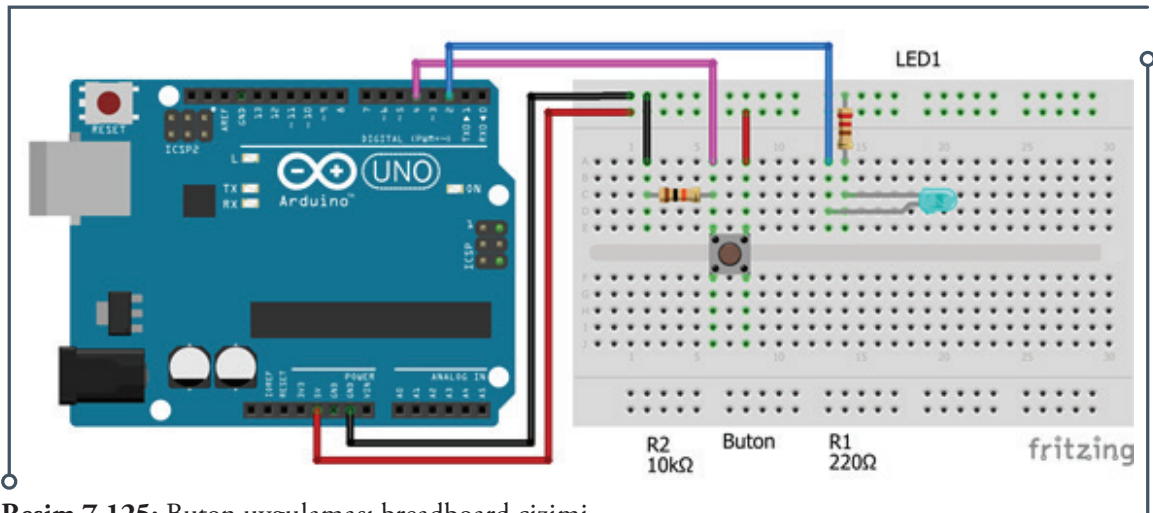
Arduino'nun A0 numaralı analog pinine bağlı bir potansiyometre kullanılarak, Arduino'nun 3 numaralı PWM pinine bağlı bir LED'in ışık seviyesinin kontrolünü sağlayacak bir uygulama hazırlayınız. Potansiyometrenin hareketi ile ışığın parlaklığı ayarlanabilmelidir. Potansiyometre ve LED'in Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada  $220\ \Omega$  direnç ve  $100\ k\Omega$  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 kullanarak 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  $220\ \Omega$ , buton için  $10\ k\Omega$  direnç kullanılmıştır.

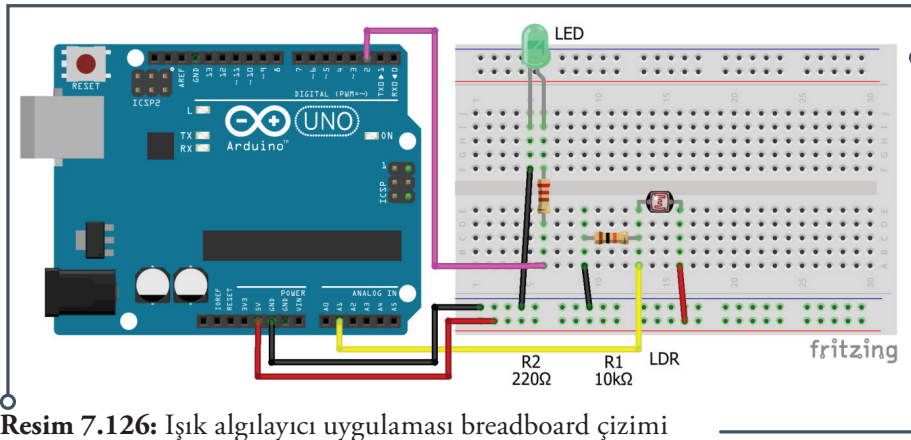


Resim 7.125: Buton uygulaması breadboard çizimi

### 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.

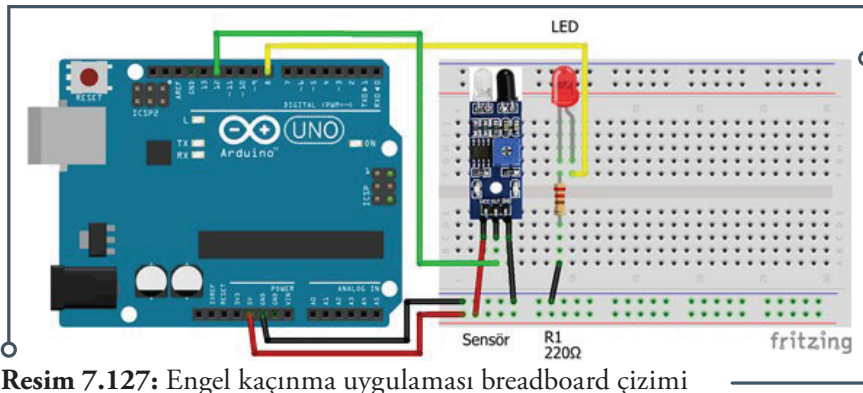
Aynı devre üzerinde yapacağınız ikinci uygulamada ise LED'in karanlıkta yanmasını aydınlıkta sönmesini sağlayınız. LDR algılayıcısının algıladığı ışık şiddetini belirleyerek, okunan değer sizin belirleyeceğiniz değerden küçük olunca LED'in yanmasını değilse (büyükse) sönmesini sağlayınız. Işık algılayıcısının 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  $\Omega$ , LDR için 10 k $\Omega$  direnç kullanılmıştır.



Resim 7.126: Işık algılayıcı uygulaması breadboard çizimi

### 7.14.8. Engel Kaçınma Uygulaması

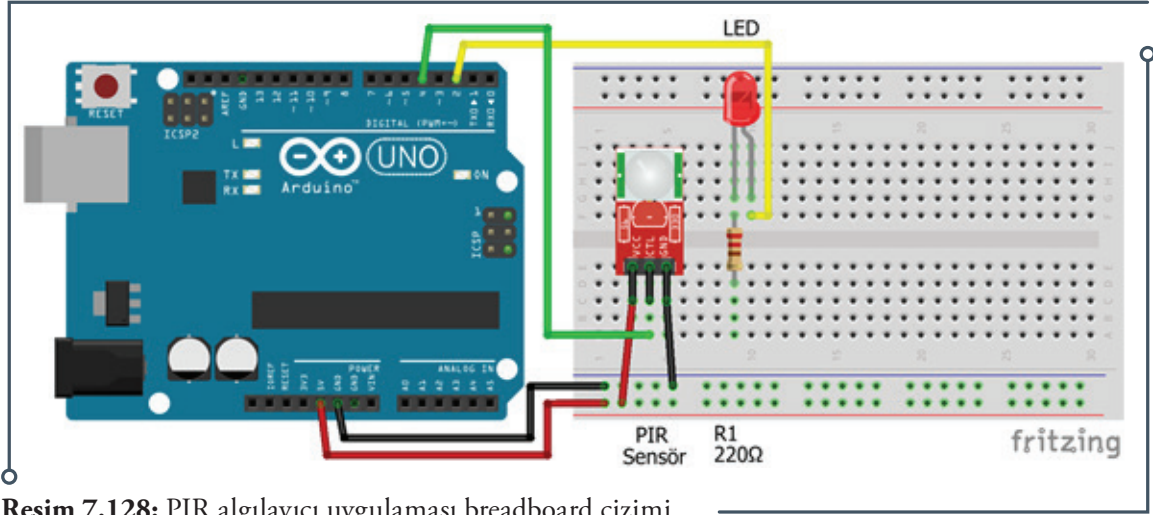
Sinyal çıkışı Arduino'nun 12 numaralı pinine bağlı bir engel kaçınma algılayıcısı kullanılarak engeli algıladığı zaman 4 numaralı dijital pine bağlı LED'i yakan bir uygulama hazırlayınız. Engel kaçınma algılayıcısının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada LED için 220  $\Omega$  direnç kullanılmıştır.



Resim 7.127: Engel kaçınma uygulaması breadboard çizimi

### 7.14.9. PIR Algılayıcı Uygulaması

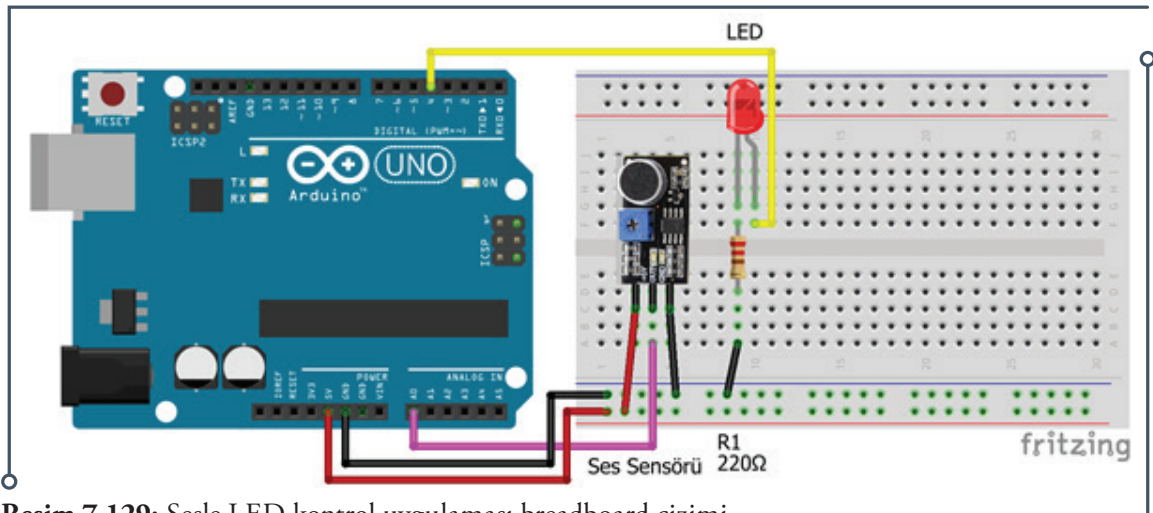
Sinyal çıkış pini Arduino'nun 2 numaralı dijital pinine bağlı bir PIR algılayıcısı kullanarak canlı algıladığı zaman 4 numaralı dijital pine bağlı LED'i yakan bir uygulama hazırlayınız. PIR algılayıcısının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçte çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada  $220\ \Omega$  direnç kullanılmıştır.



Resim 7.128: PIR algılayıcı uygulaması breadboard çizimi

### 7.14.10. Sesle LED Kontrol Uygulaması

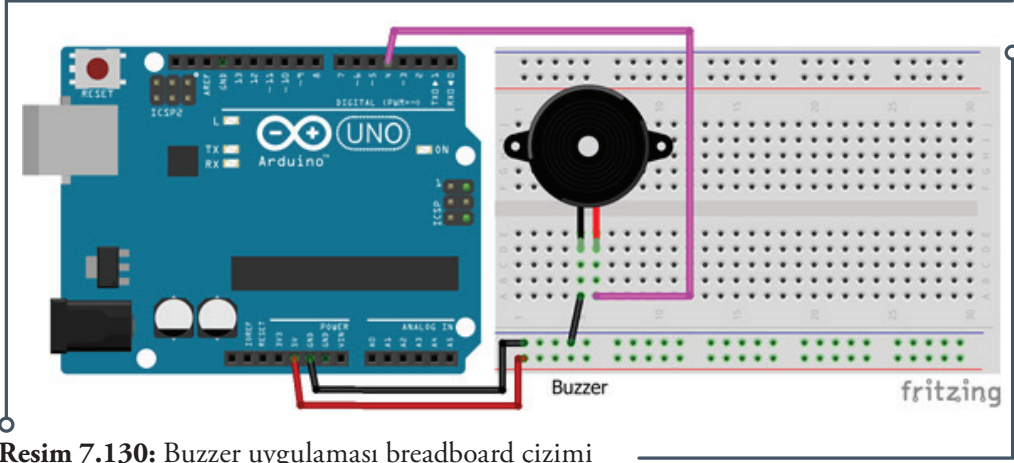
Sinyal çıkışı Arduino'nun A0 numaralı analog pinine bağlı bir ses algılayıcısı kullanarak ses seviyesine göre 4 numaralı dijital pine bağlı LED'i yakıp söndüren ses ile LED kontrol uygulaması hazırlayınız. Ses algılayıcısının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçte çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada  $220\ \Omega$  direnç kullanılmıştır.



Resim 7.129: Sesle LED kontrol uygulaması breadboard çizimi

### 7.14.11. Buzzer Uygulaması

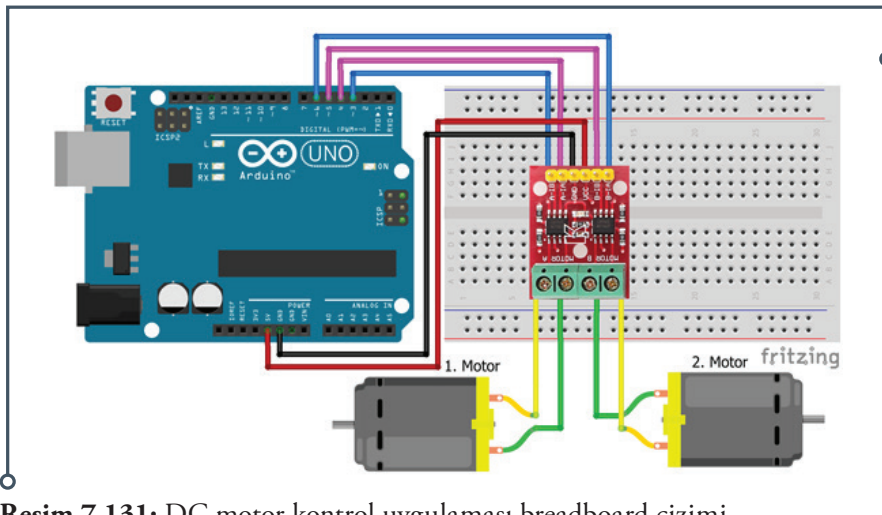
Arduino'nun 4 numaralı dijital pinine bağlı bir buzzer kullanılarak 2 farklı tonla alarm sesi üretmeyi sağlayacak bir uygulama hazırlayınız. Buzzer'ın Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmiştir. Örnek uygulamada standart bir buzzer kullanılmıştır.



Resim 7.130: Buzzer uygulaması breadboard çizimi

### 7.14.12. DC Motor Kontrol Uygulaması

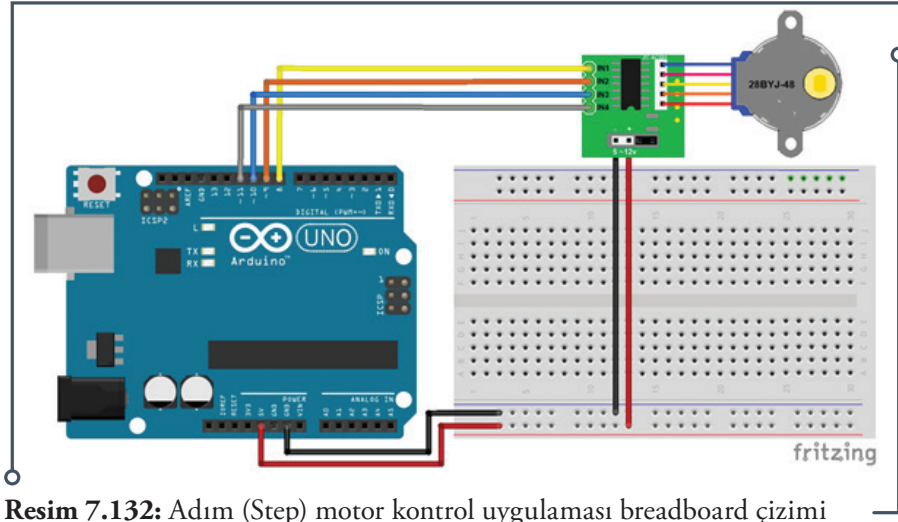
Motor sürücüyeye bağlı bir çift motoru ileri, geri, sağa ve sola hareket ettirip durduran bir motor kontrol uygulaması hazırlayınız. Bu amaçla farklı motor sürücü kartları kullanılabilir. Uygun voltajda (en fazla 6 volt) herhangi bir dc motor kullanılabilir. Daha güçlü motor kullanılacaksa Arduino UNO'nun 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

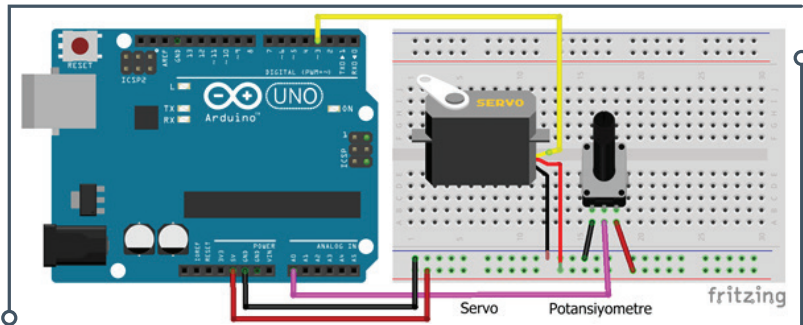
Adım motor sürücüsüne bağlı bir adım motorun; saat yönünde ve saat yönünün tersinde, verilecek olan adım sayısı kadar adım atmasını sağlayacak adım motor kontrol uygulaması hazırlayınız. Bu amaç için hazırlanan örnekte 28 BYJ-48 redüktörlü adım motor ve ULN2003A adım motor sürücü kartı kullanılmıştır. Daha güçlü adım motor kullanılacaksa Arduino UNO'nun zarar görmemesi için harici bir güç kaynağı ile adım motorun beslenmesini sağlayınız. Motor sürücüsünün 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 sürücüsünün 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

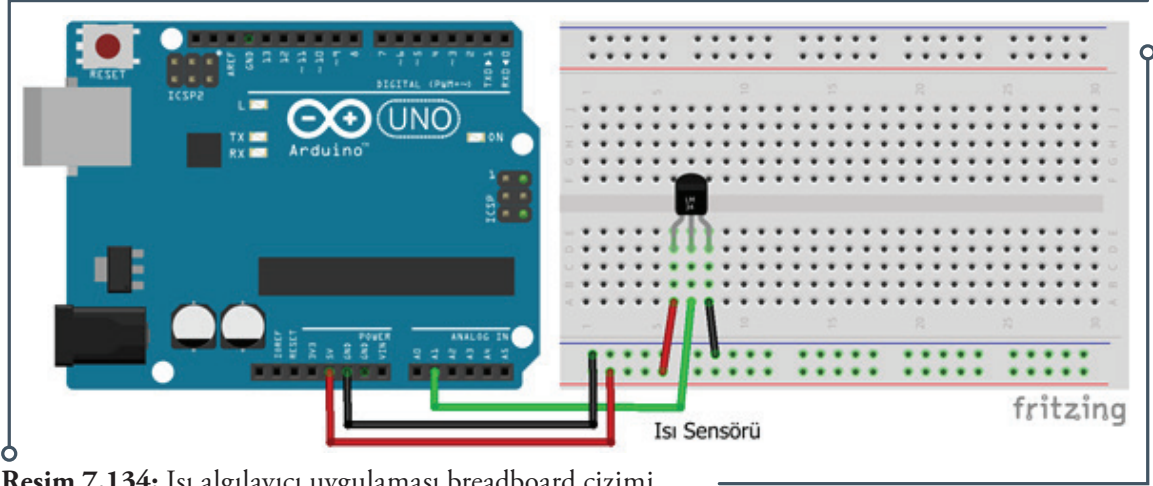
Arduino'nun A0 numaralı analog pinine bağlı bir potansiyometre kullanılarak 0 ile 180 derece arasında hareket eden bir servo motor kontrol uygulaması hazırlayınız. Potansiyometrenin değerini değiştirerek servo motorun 0 ile 180 derece arasında pozisyon alması gerekmektedir. Potansiyometreden okunan analog 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 k $\Omega$  potansiyometre kullanılmıştır.



Resim 7.133: Servo motor kontrol uygulaması breadboard çizimi

### 7.14.15. Isı Algılayıcısı Uygulaması

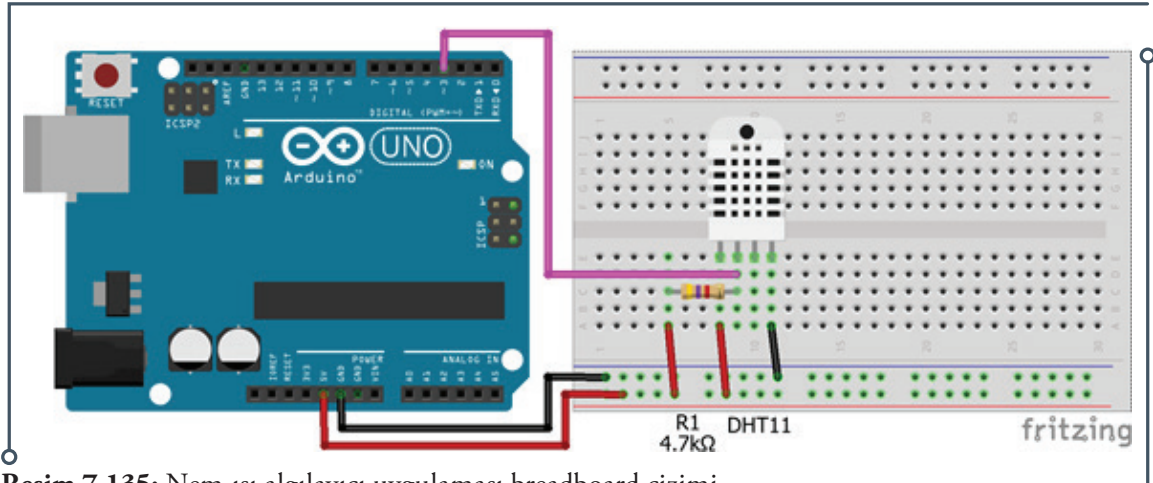
Sensör ucu Arduino'nun A1 numaralı analog pinine bağlı bir ısı algılayıcısı kullanılarak ortam ısını ölçmeyi sağlayacak bir uygulama hazırlayınız. Ortam ısı bilgisi Arduino IDE seri port ekranından okunmalıdır. Bu amaç için hazırlanan örnekte LM34 ısı algılayıcısı kullanılmıştır. Isı algılayıcısının 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 ısını 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.

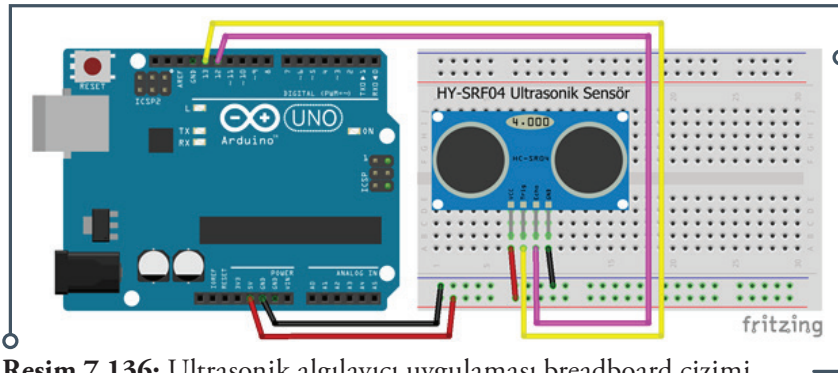


Resim 7.135: Nem-ısı algılayıcı uygulaması breadboard çizimi

### 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.

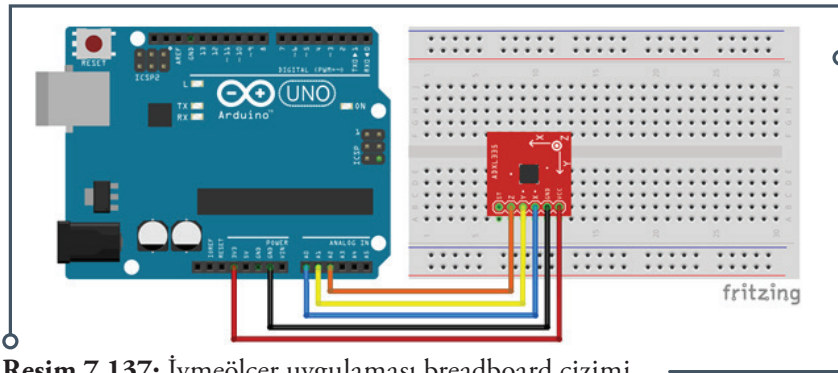
Aynı devre üzerinde yapacağınız ikinci mesafe ölçme uygulamasında ise aynı ultrasonik algılayıcının tetik (trig) ve okuma (echo) pini birleştirilerek, Arduino'nun 12 numaralı dijital pinine bağlanmalıdır. Bunun dışındaki bağlantılar aynı şekilde kalmalıdır. Uygulamada mesafe bilgisi Arduino IDE seri 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

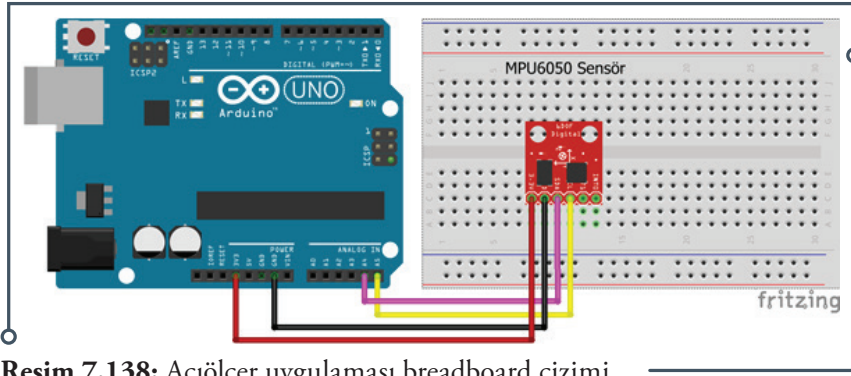
İvme ölçerin X eksenini sinyal çıkışı Arduino'nun A0, Y eksenini sinyal çıkışı Arduino'nun A1 ve Z eksenini sinyal çıkışı Arduino'nun A2 numaralı analog pinlerine bağlı ivmeölçer algılayıcısı kullanarak 3 eksenli ivme ölçme uygulaması hazırlayınız. Bu amaçla farklı ivmeölçer algılayıcıları kullanılabilir. 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. Açölçer Uygulaması

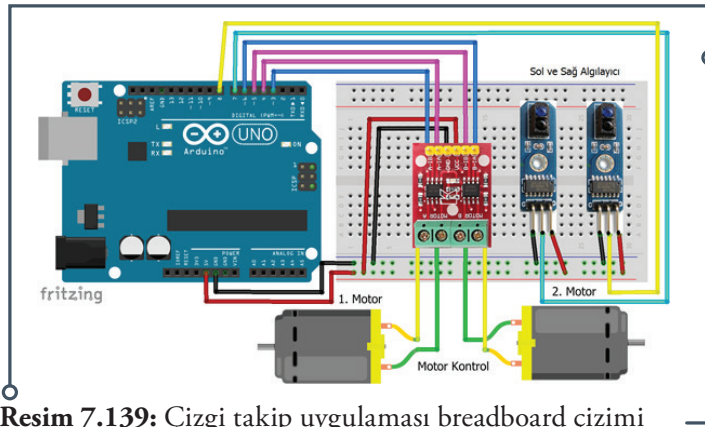
Sinyal çıkışları Arduino'nun A4 (SDA) ve A5 (SCL) numaralı analog pinlerine bağlı gyro ve ivmeölçer algılayıcısı kullanarak bir açölçer uygulaması hazırlayınız. Bu amaçla farklı gyro algılayıcılar kullanılabilir. Bu amaç için hazırlanan örnekte MPU-6050 algılayıcı kartı kullanılmıştır. Bu üzerinde 3 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: Açölçer 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 grup 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 algılayıcısının sinyal pini 7, sağ çizgi algılayıcısının 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 süreçte 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<sup>2</sup>C 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.robots-hop.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](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](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 Action.
- 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.

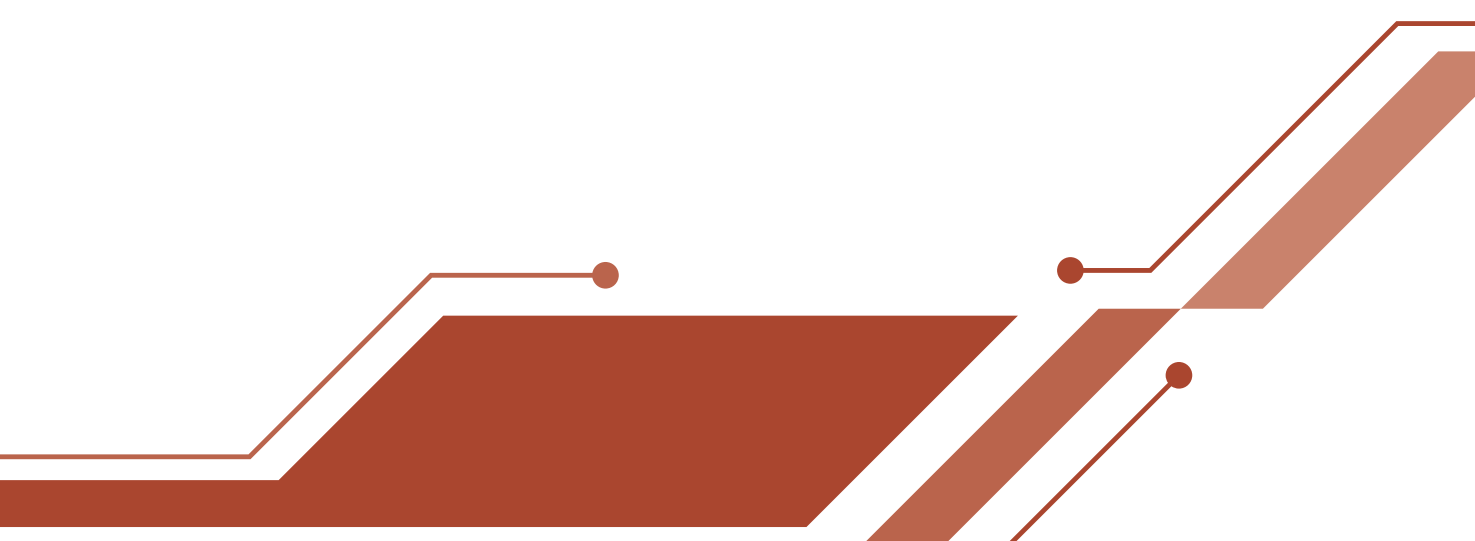
- Pololu (2017). Elektronik. [Çevrim-içi: <https://www.pololu.com/category/6/electronics>, Erişim tarihi: 13.02.2017].
- Premeaux, E., & Evans, B. (2012). *Arduino Projects to Save the World*. Apress.
- Qureshi, F., Terzopoulos, D., & Gillett, R. (2004). The cognitive controller: a hybrid, deliberative/reactive 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](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](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](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](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](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

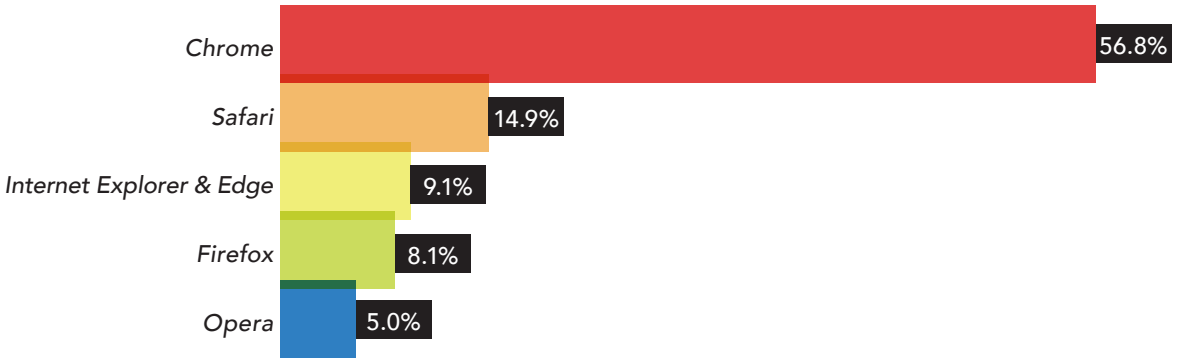


## 1. İnternet ve Web Servisleri

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 kullandığımız web tarayıcılar farklılık gösterebiliyor. Ağustos 2017 verilerine göre dünya genelinde en çok kullanılan tarayıcılar Chrome, Safari, İnternet Explorer, Firefox, Opera olarak sıralanmaktadır<sup>1</sup> (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 sunuculardır. 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

<sup>1</sup> <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, Wikipedia ... 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 okunabilir.	Web sayfası hem okunabilir hem de yazılabilir.	Web sayfası yazılabilir okunabilir ve programlanabilir.
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 duyu- larla şekillenebilen arayüz- ler ve kullanıcıya göre de-ği- şen arayüz tasarımı vardır.

**Tablo 1:** Web 1.0, Web 2.0 ve Web.3.0 Karşılaştırma

## 2. İŞARETLEME DİLİNE GİRİŞ (HTML)

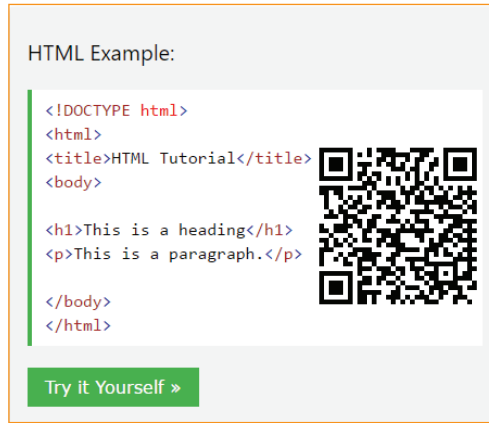
## 2. İşaretleme Diline Giriş (HTML)

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

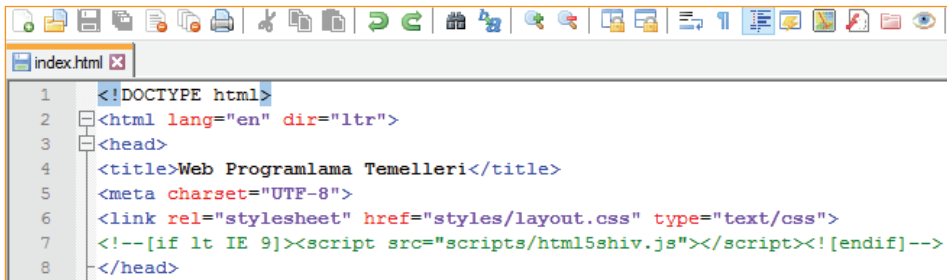
Kodlama yaparken yararlanabileceğimiz birçok editör var. Biz bu bölümde iki editörü tanıtacağız. Bu editörler dışında öğretmeninizin önerdiği editörleri de kullanabilirsiniz.

**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.

```
<!DOCTYPE html> <!-- Kodlamada kullanılacak HTML standartını belirtir. Bu örnek için HTML5 kullanılacaktır. -->
<html> <!-- Kodun başlangıcını belirtir.-->
  <body> <!-- Web sayfasındaki metinler, sesler, resimler, animasyonların kodlandığı bölümün başlangıcı-->
    <h1> h1 ve p kodlarının işlevi </h1>
    <p> h1 kodu bir paragrafın başlığını yazarken kullanılabilir.</p>
    <p> p kodu ise; yeni bir paragrafa başlarken kullanılabilir. </p>
  </body> <!-- Web sayfasındaki metinler, sesler, resimler, animasyonların kodlandığı bölümün bitişi -->
</html> <!-- Kodun bitişini gösterir. -->
```

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>
    
    <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?

```

```

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.

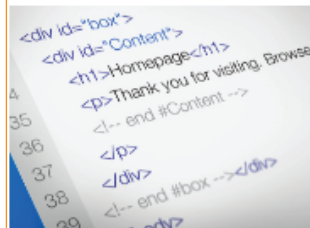
- Resim eklerken resmin genişliği (width) ve yüksekliğini (height) belirtmek için style kodunu kullanmıştık.
- Resmi sağa ya da sola yaslamak için de yine bu kodun içerisine float:left; kodunu ekliyoruz ve kodu test ediyoruz.

### Dikkat!

- style kodu içerisinde birden fazla kod yazarken kodlar arasına ";" işareti eklendiğini fark ettiniz mi?
- style="width: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.

### Resim Ekleme



Geleceğin kodlayıcıları olarak hepimiz çok çalışmak zorundayız.

### 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> </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("yenesne")</script> <!--ie9 ve öncesi için gerekli-->
  <style>
    yenesne {
      display: block;
      background-color: #dddddd;
      padding: 50px;
      font-size: 30px;
    }
  </style>
</head>
<body>
  <h1>Bilinmeyen Nesnenin Öğretilmesi?</h1>
  <yenesne>Daha önce tagı olmayan nesnemi yarattım</yenesne>
</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 <div>, <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.

<p>&lt;header&gt; &lt;/header&gt;</p> <p>Bir belge veya bölüm için bir başlık belirtir.</p>	<p>&lt;section&gt; &lt;/section&gt;</p> <p>Genellikle başlıklı tematik bir içerik gruplandırmasıdır. Bir anasayfa giriş, içerik ve iletişim bilgileri gibi bölümlere ayrılabilir.</p>	<p>&lt;article&gt; &lt;/article&gt;</p> <p>Forum yazısı, blog yazısı, gazete makalesi gibi. Bağımsız, kendine yeten içeriği belirtir.</p>	<p>&lt;footer&gt; &lt;/footer&gt;</p> <p>Bir belge veya bölüm için altbilgiyi belirtir.</p>
<p>&lt;footer&gt; &lt;/footer&gt;</p> <p>Bir belge veya bölüm için altbilgiyi belirtir.</p>	<p>&lt;nav&gt; &lt;/nav&gt;</p> <p>Gezinti bağlantılarını belirtir.</p>	<p>&lt;aside&gt; &lt;/aside&gt;</p> <p>İçeriği (yan bar gibi) bir kenara koyar.</p>	<p>&lt;figure&gt; &lt;/figure&gt; &lt;figcaption&gt; &lt;/figcaption&gt;</p> <p>Bir resim ve resim yazısı, bir &lt;figure&gt; ögesinde gruplandırılabilir.</p>
<p>&lt;mark&gt; &lt;/mark&gt;</p> <p>İşaretili/vurgulanmış metinleri tanımlar.</p>	<p>&lt;details&gt; &lt;/details&gt;</p> <p>Kullanıcının görüntüleyebileceği veya gizleyebileceği ek ayrıntıları tanımlar.</p>	<p>&lt;progress&gt; &lt;/progress&gt;</p> <p>Görevin ilerlemesini temsil eder.</p>	<p>&lt;dialog&gt; &lt;/dialog&gt;</p> <p>Bir iletişim kutusu veya pencere tanımlar.</p>

### 1. UYGULAMA

Notpad++ editöründe aşağıdaki basamakları takip ederek kodlama yapalım.

- 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.

```
5 <section> <!-- HTML5 elementlerini tanıtmak için bir bölüm açıyoruz. -->
6
7 <header> <!-- Bölümümüzün başlık etiketini açıyoruz. -->
8   <figure> <!-- Bölümümüzün başlığı içerisinde bir resim elementi görünsün istiyoruz. -->
9     
11     <figcaption>Fig.1 - HTML5</figcaption> <!-- Resmin açıklaması -->
12   </figure> <!-- Resim etiketini kapatıyoruz. -->
13 </header> <!-- Bölümümüzün başlık etiketini kapatıyoruz. -->
14
15 <article> <!-- Bölüm içerisindeki birinci makale içeriği -->
16   <header> <!-- Birinci makale başlığı -->
17     <h1>Anlamsal Elementler</h1> <!-- h1,h2,h3 Metnin büyüklüğünü değiştirir. -->
18   </header> <!-- Birinci makale başlığını kapatıyoruz. -->
19   <p>form, table, article, audio, video vb.</p> <!-- p bir paragrafa başlarken kullanılır. -->
20 </article> <!-- Birinci makale içeriğini kapatıyoruz -->
21
22 <article> <!-- Bölüm içerisindeki ikinci makale içeriği -->
23   <header> <!-- İkinci makale başlığı -->
24     <h2>Anlamsal Olmayan Elementler</h2> <!-- h1,h2,h3 Metnin büyüklüğünü değiştirir. -->
25   </header> <!-- İkinci makale başlığını kapatıyoruz. -->
26   <p>div, p, span vb. </p> <!-- p bir paragrafa başlarken kullanılır. -->
27 </article> <!-- Birinci makale içeriğini kapatıyoruz -->
28
29 </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.

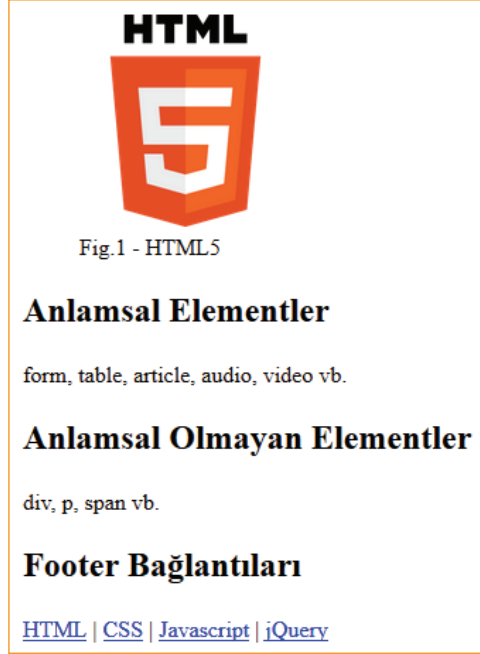
### Kodlayıcı Görevi

- <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.

```
31 <footer> <!-- Sayfanın altında görünecek altbilgi -->
32   <header>
33     <h2>Footer Bağlantıları</h2>
34   </header>
35   <nav> <!-- Sayfanın altında görünecek bağlantılar (linkler) -->
36     <a href="/html/">HTML</a> |
37     <a href="/css/">CSS</a> |
38     <a href="/js/">JavaScript</a> |
39     <a href="/jquery/">jQuery</a>
40   </nav> <!-- Gezinim etiketini kapatıyoruz. -->
41 </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

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <!-- <mark> ile etiketlenen yazı sarı ile vurgulanır. -->
6 <p>Bu metinde bir vurgulanmış <mark>yazı</mark> var.</p>
7
8 <br/>
9
10 <!-- <details> ile açılır kapanır içerikler oluşturabiliriz. -->
11 <details>
12 <summary> <!-- Üzerine tıklandığında açılır metin -->
13   Denizer YILDIRIM
14 </summary>
15 <p>Ankara Üniversitesi, Uzaktan Eğitim Merkezi</p>
16 <p>Öğretim Tasarımı & Web Programlama</p>
17 </details>
18
19 <br/><br/>
20
21 <!-- Herhangi bir görevin ilerlemesini gösteren etiket
22   value: görevin hangi aşamada olduğunu belirten değer
23   max: görevin tamamlanma aşamasını gösteren değer
24 -->
25 <progress value="22" max="100"></progress>
26
27 <br/><br/>
28
29 <!--Kullanıcıya bildirim vermek için kullanılan etiket -->
30 <dialog open> Dialog etiketi ile kullanıcıya uyarı verebiliyoruz. </dialog>
31
32 </body>
33 </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

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <header> <p>Video Ekleme</p> </header>
6 <video id="video1" width="420" controls>
7 <!-- Video etiketini açıyoruz -->
8 <source src="mov_bbb.mp4" type="video/mp4">
9 <!-- Video kaynağını ve türünü belirtiyoruz. Siz indirmiş olduğunuz videonun ismini yazınız.-->
10 Tarayıcınız mp4 videoyu desteklemiyor.
11 <!-- Eğer tarayıcı eski ise videonun (mp4) çalıştırılmayacağı uyarısı veriyoruz. -->
12 </video>
13 <!-- Video etiketini kapatıyoruz -->
14
15 <br/><br/>
16 <!-- Video ve ses arasına iki satır boşluk bırakıyoruz. -->
17
18 <header> <p>Ses Ekleme</p> </header>
19 <audio controls>
20 <!-- Ses etiketini açıyoruz -->
21 <source src="horse.mp3" type="audio/mpeg">
22 <!-- Ses kaynağını ve türünü belirtiyoruz. Siz indirmiş olduğunuz ses dosyasının ismini yazınız.-->
23 Tarayıcınız mp3 sesi desteklemiyor.
24 <!-- Eğer tarayıcı eski ise sesin (mp3) çalıştırılmayacağı uyarısı veriyoruz. -->
25 </audio>
26 <!-- 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 oynatıcısının genişliğini ayarlar.

<source> kodu içerisinde 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/K6S40-VtZBI">
7 </iframe>
8
9 <!--Yöntem 2-->
10 <object width="420" height="315" data="https://www.youtube.com/embed/K6S40-VtZBI">
11 </object>
12
13 <!--Yöntem 3-->
14 <embed width="420" height="315" src="https://www.youtube.com/embed/K6S40-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/K6S40-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 ünite de 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>
13      <!-- İçerik -->
14    </div>
15    <div>
16      <!-- Footer -->
17    </div>
18  </body>
19 </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

```

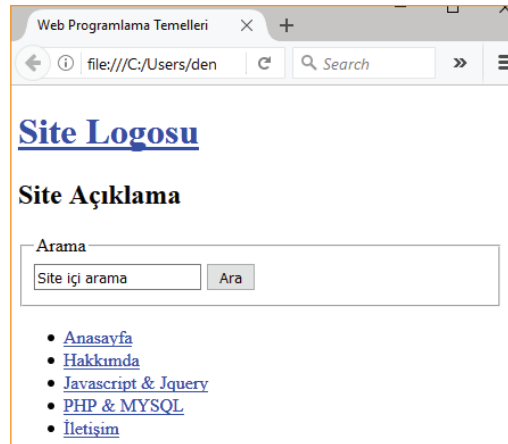
11 <!-- banner -->
12 <div>
13 <header>
14
15 <div id="hgroup">
16 <h1><a href="#">Site Logosu</a></h1>
17 <h2>Site Açıklama</h2>
18 </div>
19
20 <!-- Site içi arama için bir metin girişi-->
21 <form action="#" method="post">
22 <fieldset>
23 <legend>Arama</legend>
24 <input type="text" value="Site içi arama">
25 <input type="submit" id="sf_submit" value="Ara">
26 </fieldset>
27 </form>
28
29 <nav>
30 <ul>
31 <li><a href="#">Anasayfa</a></li>
32 <!-- Şuan tasarlamaya çalıştığımız sitenin anasayfa bağlantısı -->
33 <li><a href="#">Hakkımda</a></li>
34 <!-- Siteyi kodlayanla ilgili kişisel bilgilerin yer alacağı hakkımızda sayfası -->
35 <li><a href="#">Javascript & Jquery</a></li>
36 <!-- İlerleyen bölümlerde Javascript ya da Jquery ile yapacağımız kodlamaların yer alacağı site sayfası -->
37 <li><a href="#">PHP & MYSQL</a></li>
38 <!-- İlerleyen bölümlerde PHP & MYSQL ile yapacağımız kodlamaların yer alacağı site sayfası -->
39 <li><a href="#">İletişim</a></li>
40 <!-- İletişim sayfası -->
41 </ul>
42 </nav>
43
44 </header>
45 </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; <ul> 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">
54
55 <!-- Sitenin Ana İçeriğini id' si intro olan bir div elementi içerisinde topluyoruz. -->
56 <div id="intro">
57
58
59
60
61
62
63
64
65 </div>
66 </div>
67 </div>
```

```
<!-- Sitenin slayt gösterisini id' si slider olan div elementi içerisinde topluyoruz.-->
<section id="slider">
<figure>
<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>
```



### Slogan

Bilgisayarlar öğrenebilen varlıklardır. Fakat öğretirken algoritmalara gereksinim duyar.

Şimdi söz geleceğin kodlayıcılarında: "Algoritma bizim işimiz :)"

[Devamını Oku »](#)



## 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"></a></li>
<!-- Madde işaretli, a1.html sayfasına bağlantılı genişliği:130px, yüksekliği:130px bir resim ekliyoruz. -->
<li><a href="a2.html"></a></li>
<!-- Madde işaretli, a2.html sayfasına bağlantılı genişliği:130px, yüksekliği:130px bir resim ekliyoruz. -->
<li><a href="a3.html"></a></li>
<!-- Madde işaretli, a3.html sayfasına bağlantılı genişliği:130px, yüksekliği:130px bir resim ekliyoruz. -->
</ul>
<figcaption><a href="#">Resim Galerisi sraquo</a></figcaption> <!-- Buraya dikkat yeni bir kod! -->
</figure>
</article>
</section>
</div>

```

**Biz Kodlayıcılar,**

Web programlama bölümünü bitirdiğimizde örnek bir site tasarlamış olacağız.

Siteyi tasarlarken hem kodların ne işe yaradığını öğrenecek hem de kodları örnek tasarımımız için uygulayacağız.

Programlama yapmaktan keyif aldıkça kendi kendinize daha ilgi çekici, beğeni oranı yüksek siteler tasarlayabileceksiniz.



[Resim Galerisi »](#)

## 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 &raquo; kodu ile elde edilir. Bu türden kodlar varlık isimleri olarak adlandırılır. Benzer şekilde birçok simgenin varlık ismi vardır. Yandaki QR kodlu bağlantıda<sup>2</sup> 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>
```

## Hızlı Bağlantılar

- [Anasayfa](#)
- [Hakkımda](#)
- [Javascript & JQuery](#)
- [PHP & MYSQL](#)
- [İletişim](#)

Copyright © 2017 - [Siteyi Kodlayan kişi İsmi](#)

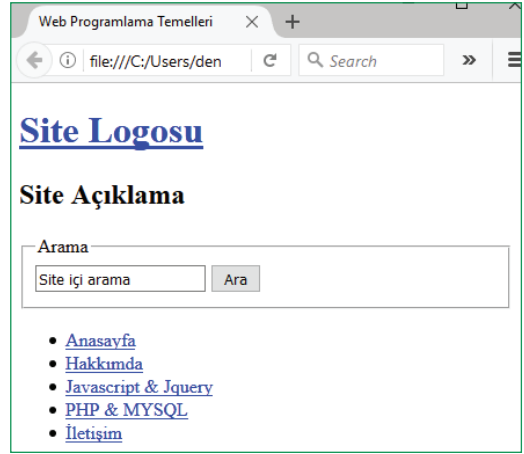
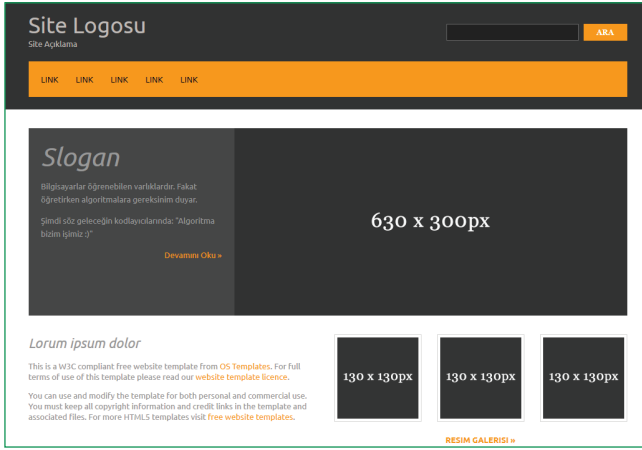
Bu ünite sonunda HTML5 ile site şablonunun ana çerçeveleri oluşturuldu. Böylece üniteadaki kazanımlara yönelik kodlama örneklerini uygulama fırsatı oldu. Şimdi yapılan kodlamaları, index.html şeklinde kendi isminizle oluşturduğunuz dizinin (klasörün) içerisine kaydediniz.

<sup>2</sup> [https://www.w3schools.com/charsets/ref\\_html\\_entities\\_4.asp](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.

## İlk CSS Örneğimiz

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.

### 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 gördük.

```
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 incelediğinde 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>

```

## İkinci CSS Örneğimiz

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. 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. Bu bir paragraf yazısıdır. Bu bir paragraf yazısıdır.

### İç İç Geçmiş CSS Kodlama

Son olarak, iç içe 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>

```

## Üçüncü CSS Örneğimiz

### Parağraf Başlığı

Bu bir parağraf yazısıdır. Bu bir parağraf yazısıdır.

### Parağraf Başlığı

Bu bir parağraf yazısıdır. Bu bir parağraf yazısıdır.

[Ankara Üniversitesi](#) [Google Web Sitesi](#)

Bu bir parağraf yazısıdır. Bu bir parağraf yazısıdır.

### Dikkat!

- Class özelliği “renkli” olan iki p elementinin birbirinden farklı biçime sahip olduğunu farkettiler mi?
- İki h1 elementinin birbirinden farklı biçime sahip olduğunu farkettiler mi?
- Aynı footer elementi içerisinde olup farklı biçime sahip iki a elementini farkettiler mi?

## PROJE SİTESİNİN CSS İLE GÖRSELLEŞTİRİLMESİ

### Varsayılan Ayarları

Bir web sitesinin CSS ile görselleştirilirken ilk olarak yapılması gereken varsayılan olarak kullanılacak genel CSS ayarlarını oluşturmaktadır.

```
<style>
/*-----Genel Ayarlar-----*/
* {
  /* Web sayfasındaki tüm etiketlerin seçimi */
  margin:0; /* Elementlerin kendinden önce ve sonraki elementlerle olan mesafesi 0 pixel (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 üstteki div elementinin class özelliği “çerçeve1”, sitenin logosunun ve açıklamasının bulunacağı div etiketinin id özelliği “logogrup” olarak değiştirilir.

```

<!-- Banner -->
<div class="cercevel">
  <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*/
.cercevel {
  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;}

```

```

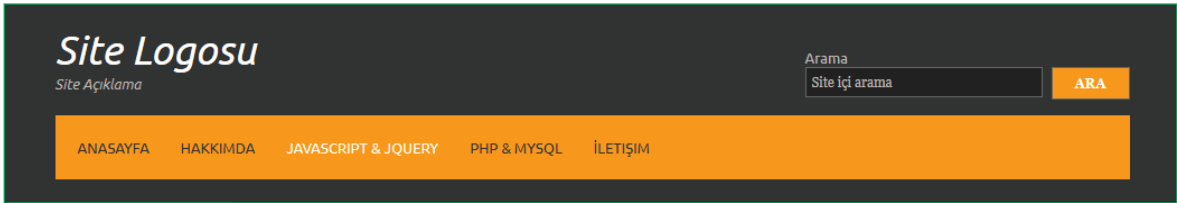
/*---Header içindeki form elementi---*/
header form{
  width:290px;
  float:right; /*sağa hizala*/
  margin-top:20px;
  padding:0;
}
header form input{
  float:left; /*sola hizala*/
  width:200px;
  margin:0;
  padding:5px;
  color:#C0BAB6;
  background-color:#232323;
  border:1px solid #666666;
}
header 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>
        <figcaption>
          <h2>Slogan</h2>
          <p> Bilgisayarlar öğrenebilen varlıklardır.
          Fakat öğrenirken algoritmalara gereksinim duyar. </p>
          <p> Şimdi söz geleceğın kodlayıcılarında:
          "Algoritma bizim işimiz :)"</p>
          <footer class="devam"><a href="#">Devamını Oku &raquo;</a></footer>
        </figcaption>
      </figure>
    </section>
    <!-- Acıklama -->
    <div id="intro">
      <section class="clear">
        <!-- article 1 -->
        <article class="acıklama">
          <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="acıklama ek">
          <figure>
            <ul class="clear">
              <li>
                <a href="#">
                  
                </a>
              </li>
              <li>
                <a href="#">
                  
                </a>
              </li>
              <li class="last">
                <a href="#">
                  
                </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;}

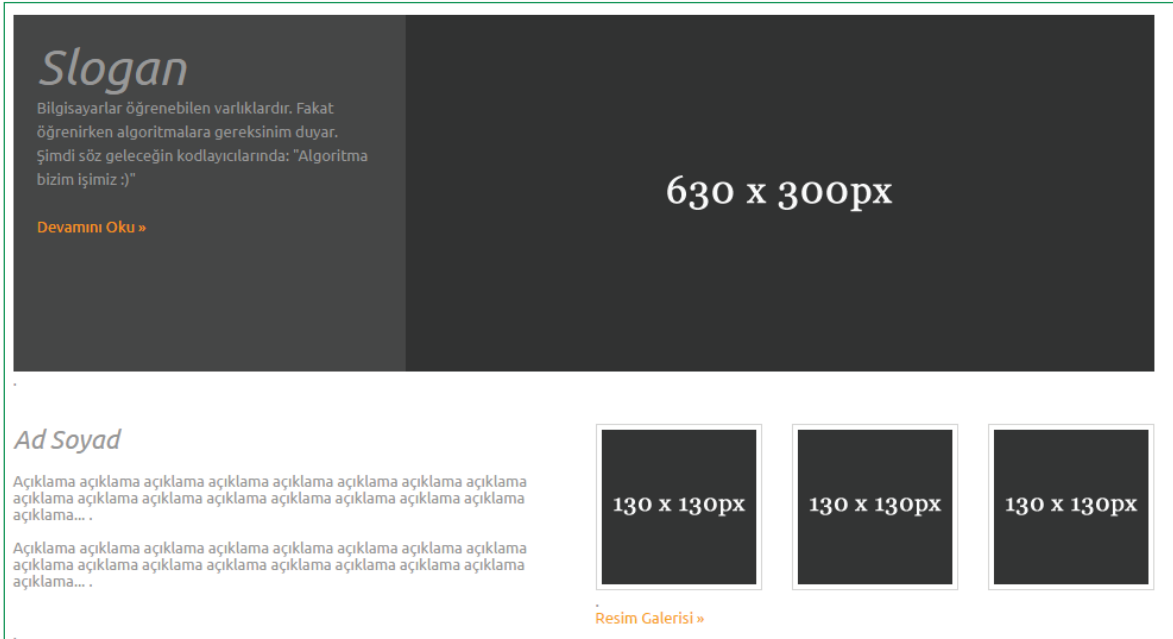
/* -----Intro----- */

#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 üstteki 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>

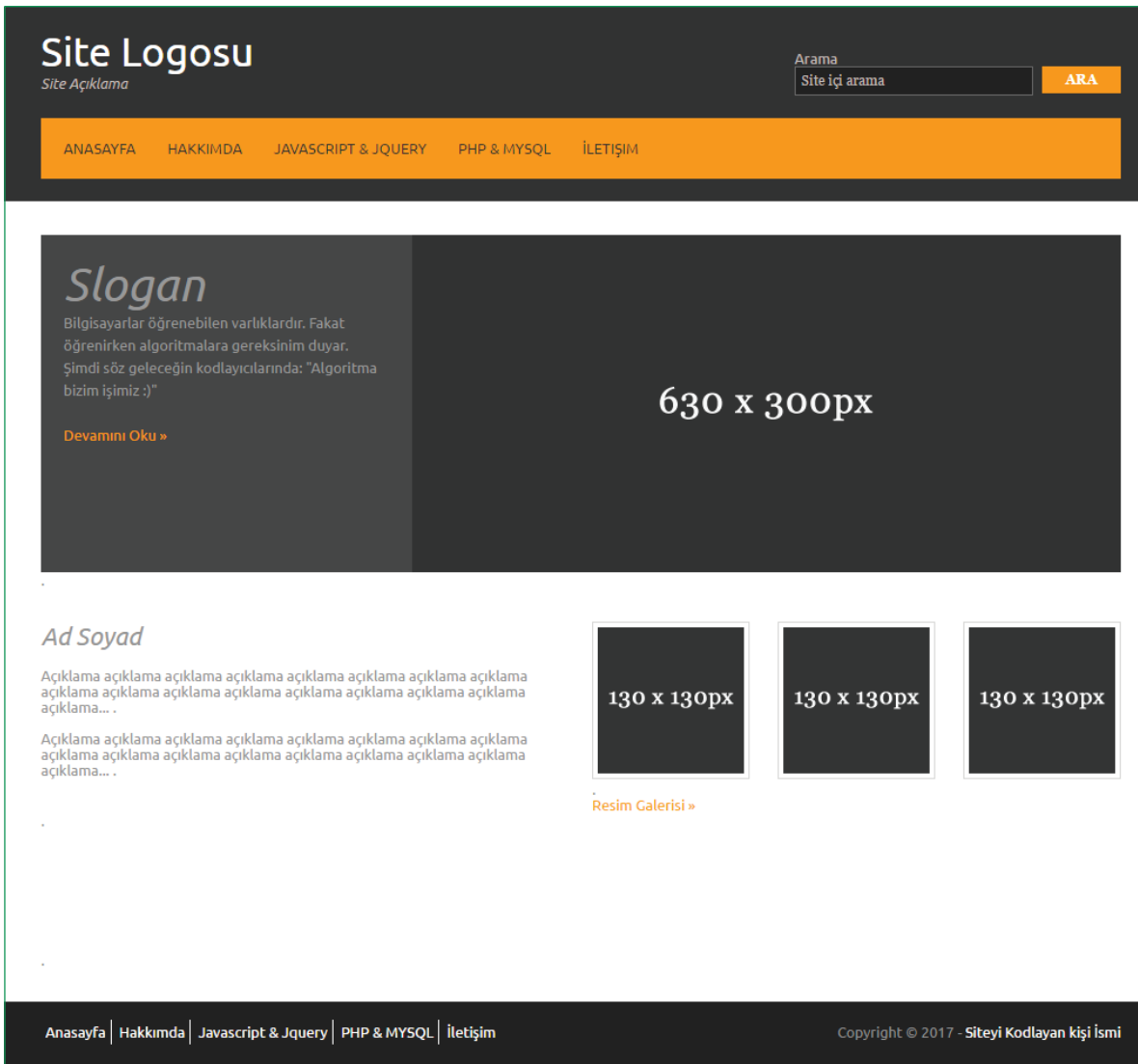
```

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;
/* menüyu 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 sınımalara 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">Parağraf</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 deęiřti.";
      }
    </script>
  </head>
  <body>
    <h2>head elementi arasındaki kullanımı </h2>
    <p id="demo">Paraęraf</p>
    <button type="button" onclick="degistir()">Deęiřtir</button>
  </body>
</html>

```

### head elementi arasındaki kullanımı

Butona tıklayınca paraęraf deęiřti.

## Body Elementi Arasındaki Kullanımı

```

<!DOCTYPE html>
<html>
  <body>
    <script>
      function degistir() {
        document.getElementById("demo").innerHTML = "Butona tıklayınca paraęraf deęiřti.";
      }
    </script>
    <h2>body elementi arasındaki kullanımı </h2>
    <p id="demo">Paraęraf</p>
    <button type="button" onclick="degistir()">Deęiřtir</button>
  </body>
</html>

```

### body elementi arasındaki kullanımı

Butona tıklayınca paraęraf deęiřti.

## 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 değışti.";
      }
    -->

    <!-- ya da
      <script src="https://www.w3schools.com/js/myScript.js"></script>
      <script src="js/myScript.js"></script>
      şekilde 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 değışti.

Değiştir

Şimdi bu yöntemlerden bazılarını kullanarak senaryo temelinde kodlama yapılacaktır.

## HTML İÇERİĞİ DEĞİŞTİRME

Javascriptle HTML içeriğinin değıştirilebileceğinden bahsedilmiştir. Web sayfamızda bir buton aracılığıyla güncel tarih ve saatin gösterilmesi bu türden uygulamalara bir örnek olarak verilebilir.

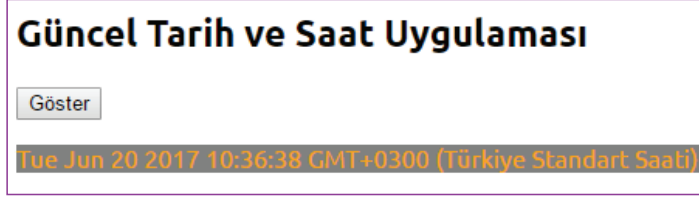
```
<!DOCTYPE html>
<html>
<body style="font-family:Ubuntu;">

<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 tarihi ve saat ile değıştir.
-->

<p id="demo" style="background-color:grey;color:orange;"></p>
</body>
</html>
```

Aşağıdaki ekran görüntüsünde “Göster” butonuna her tıkladığında 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

### Elementin Niteliğini Değiştirme



Kullanıcı Butona Tıkladıktan Sonra

### Elementin Niteliğini Değiştirme

## 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 tıkladığında, id'si p1 olan elementin stilleri içerisinde
  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 tıkladığında, id'si p1 olan elementin stilleri içerisinde
  display özelliğine block yap.
  Not: ...style.visibility='visible' da kullanılabilir.
-->
</body>
</html>
```

### Elementleri Gizleme ya da Görünür Yapma

Gizle Göster

### Elementleri Gizleme ya da Görünür Yapma

Bu bir parağraf. Bu bir parağraf. Bu bir parağraf.

Gizle Göster

Şimdiye kadar Javascript kullanarak basit etkileşimli kodlamalar yapıldı. Artık hedefi daha yükseklere taşımanın sırası gelmedi mi sizce? Hatırlarsanız, javascript’ de mantıksal sınımalara göre işlem yapılabileceğinden bahsedilmişti. Örneğin

- Eğer ... olduysa, .... yap.
- Eğer ... ise ... değiştir.
- Eğer ... ise ... göster.

Bu şekilde sayısız senaryolar oluşturulabilir. Peki bu sınıama işlemini javascript ile nasıl yapabiliriz?

### MANTIKSAL SINAMA

Bu konu başlığı altında javascript’ de mantıksal sınıama 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:
      // deęişkenin x olma durumunda yapılacak kodlama
      break;
    case y:
      // deęişkenin y olma durumunda yapılacak kodlama
      break;
    default:
      // deęişkenin 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 deęişkenleri oluşturuldu.
  gun = new Date().getDay(); // getDay() ile o andaki tarihin gün numarasını gun deęişkenine eşitliyoruz.
  switch (gun) {          //

    case 0:                // gün deęişkeni 0 ise;
      mesaj = "Pazar";    // mesaj deęişkenini "Pazar" a eşitle
      break;              // dięer durumları kontrol etmene gerek kalmadı.

    case 1:                // gün deęişkeni 1 ise;
      mesaj = "Pazartesi"; // mesaj deęişkenini "Pazartesi" ye eşitle
      break;              // dięer durumları kontrol etmene gerek kalmadı.

    case 2:                // gün deęişkeni 2 ise;
      mesaj = "Salı";     // mesaj deęişkenini "Salı" ya eşitle
      break;              // dięer durumları kontrol etmene gerek kalmadı.

    case 3:                // gün deęişkeni 3 ise;
      mesaj = "Çarşamba"; // mesaj deęişkenini "Çarşamba" ya eşitle
      break;              // dięer durumları kontrol etmene gerek kalmadı.

    case 4:                // gün deęişkeni 4 ise;
      mesaj = "Perşembe"; // mesaj deęişkenini "Perşembe" ye eşitle
      break;              // dięer durumları kontrol etmene gerek kalmadı.

    case 5:                // gün deęişkeni 5 ise;
      mesaj = "Cuma";     // mesaj deęişkenini "Cuma" ya eşitle
      break;              // dięer durumları kontrol etmene gerek kalmadı.

    case 6:                // gün deęişkeni 6 ise;
      mesaj = "Cumartesi"; // mesaj deęişkenini "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 çalıştırdıktan 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:#ffc000;}
</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 bloğunu 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 değer üzerine yeni değer 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 giriniz.");

        // 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>
```

Bir sayı giriniz.

Tamam İptal

### While Döngüsü Örneği

1 den 5 e kadar olan çift sayıların toplamı: 6

1 den 5 e kadar olan tek sayıların toplamı: 9

**Dikkat!**

```
// ----String Birleştirme İşlemi-----
document.getElementById('demo').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 javascript 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 çalıştırdıktan 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

### Fonksiyon Kullanım Örneği

- 2- Kullanıcının doğum tarihine göre yaşını hesaplayan fonksiyonun yazılması

```
// Yöntem 1 ile kullanıcının yaşını hesaplama
function Hesapla(){

    // id'si yıl 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- Javascript 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 değişken-  
de toplanabilir ve daha kolay yönetilebilir. Örneğin:

```
var otolar= ["Volvo", "Opel", "Ford"];  
var takimler= ["Çorumspor", "Muğlaspor", "Gençlerbirliği"];
```

ya da

```
var otolar = new Array("Volvo", "Opel", "Ford");  
var takimler= 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 iç-  
risindeki öğe sayısından 1 eksik değer arasında olabilir. Örneğin takimler dizisi 3 öğe içermektedir.  
Burada "Muğlaspor" değerini ekrana yazdırmak için;

```
document.getElementById("demo").innerHTML = takimler[1];
```

kodlaması kullanılabilir. Dizinin tüm öğelerini ekrana yazdırmak için ise,

```
document.getElementById('mesaj').innerHTML = takimler;
```

## 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 kaçınıcı 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 takimler= ["Çorumspor", "Muğlaspor", "Gençlerbirliği"];  
    document.getElementById('mesaj').innerHTML = takimler[a];  
  }  
</script>
```



Ekran Çıktısı

### Dizi Kullanım Örneği

Gençlerbirliği

**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 takimler= ["Çorumspor", "Muğlaspor", "Gençlerbirliği"];
  if (a==0 || a==1 || a==2)
  {
    document.getElementById('mesaj').innerHTML = takimler[a];
  } else {
    document.getElementById('mesaj').innerHTML = takimler;
  }
}
</script>

```

Ekran Çıktısı

### Dizi Kullanım Örneği

Çorumspor,Muğlaspor,Gençlerbirliği

## BASİT DİZİ İŞLEMLERİ

Önceki örneklerde sabit bir dizi tanımlayarak, dizi içerisindeki öğelere erişilmeye çalışıldı. Şimdi ise, diziyeye 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. Öğesinden 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="Kaçınıncıdan 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>
```

## Ekleme

**Dizinin herhangi bir yerine yeni bir öge ekleme ya da herhangi bir ögesini değiştirme**

Volvo,Mercedes,Opel,Ford

Mercedes ▾ 1 0 İşlem

## Değiştirme

**Dizinin herhangi bir yerine yeni bir öge ekleme ya da herhangi bir ögesini değiştirme**

Volvo,Mercedes,Ford

Mercedes ▾ 1 1 İşlem

## 4. Dizinin Öğelerini 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 Sıralama</button>
<button onclick="NumerikSiralama()">Numerik Sıralama</button>

<p id="demo" class="demo"></p>
```

## Javascrypt 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 sıralanmış hallerini göstermek için fonksiyonu çağırıyoruz.
    Goster();
  }

  function NumerikSiralama() {
    // 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 ikili olarak karşılaştırır.
    // eğer küçükten büyüğe sıralamak istiyorsak a-b
    // büyükten küçüğe sıralamak istiyorsak b-a döndürülür.

    Goster();
  }
</script>

```

## Ekran Çıktıları

**JavaScript Dizi Sıralama**

Alfabetik Sıralama
Numerik Sıralama

1,10,100,25,40,5

ankara,istanbul,muğla

34,48,6,ankara,istanbul,muğla

**JavaScript Dizi Sıralama**

Alfabetik Sıralama
Numerik Sıralama

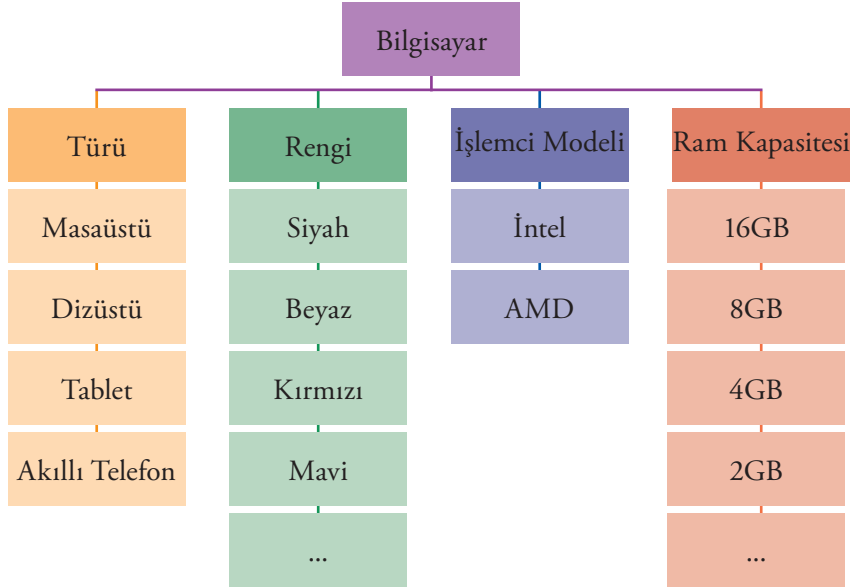
1,5,10,25,40,100

ankara,muğla,istanbul

6,34,48,ankara,muğla,istanbul

## 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" . "özellik" 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 nesnelere bir dizi içerisinde tutulup 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 Nesnelere 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 diziyeye ekliyoruz.
    dizi.push(bilgisayar);
}
```

### 3. Dizi İçerisindeki Nesnelere 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 ögesi 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:" ", işlemci:" ", 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 nesnelere
    // ekrana yazdıracak fonksiyonu çağırıyoruz.
    diziGoster();
}

function Sil() {
    dizi.pop();
    diziGoster();
}

```

## Ekran Çıktısı

### Dizi ve Nesne Kullanımı

Türü:

Markası:

İşlemci Modeli:

RAM Kapasitesi:

Fiyatı (KDV' siz):

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

## 5. VERİ TABANI YÖNETİMİ

## 5. VERİ TABANI YÖNETİMİ

Veri tabanı yönetimine geçmeden önce, bir önceki ünite de 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 sahibiyse 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 ünite de, ücretsiz bir veri tabanı yönetim sistemi olan MariaDB yönetimi ele alınacaktır.

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.

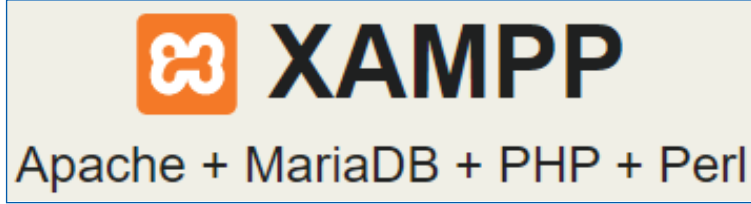
Tablolar arasındaki bir ilişki bir tabloya, başka bir tablodaki kaydı bağlamayı sağlar. Bu şekilde veriler daha az yer kaplar ve işlemleri kolaylaştırır.

#### Veritabanı Sistemleri

- MySQL
- MariaDB
- Microsoft SQL
- Microsoft Access
- PostgreSQL
- Oracle
- Sybase
- Berkeley

## 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.

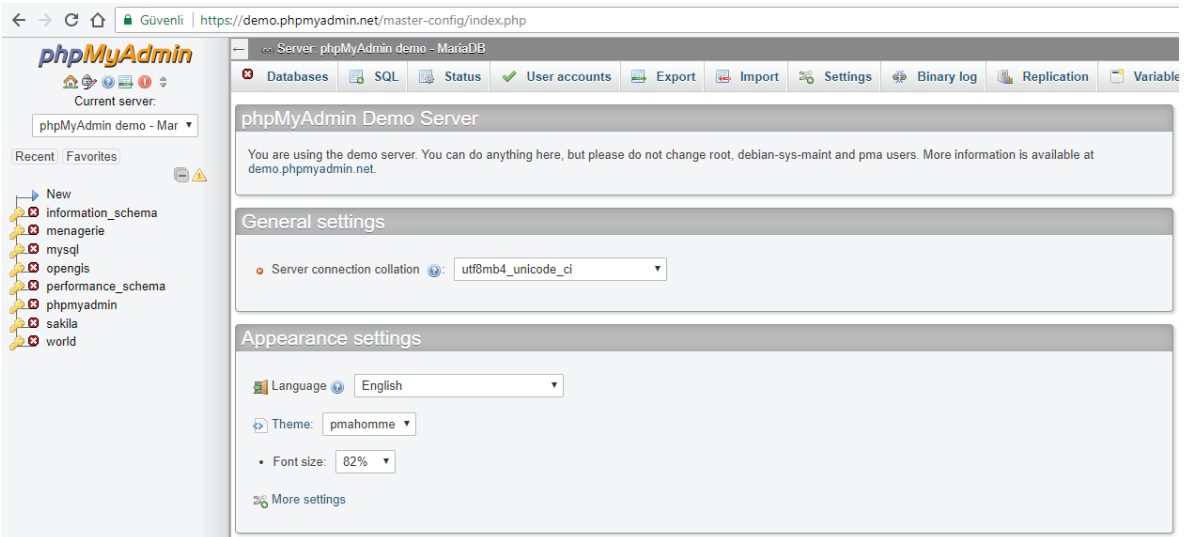


Bilgisayarınıza XAMPP kurduğunuzda, kurulum yapmış olduğunuz bilgisayarı basit bir web sunucusuna dönüştürmüş olursunuz. Daha önceden bilgisayarınız bir web sitesine bağlanırken ve istemci olarak çalışırken artık sunucu olarak da çalışabilir. Daha açık bir ifadeyle kendi bilgisayarınızda oluşturmuş olduğunuz bir web sitesinin kaynaklarına yine kendi bilgisayarınızın tarayıcısını kullanarak erişebilirsiniz. Bir sonraki ünite de bu konu daha detaylı incelenecektir.

## MARIADB VERİ TABANI YÖNETİMİ

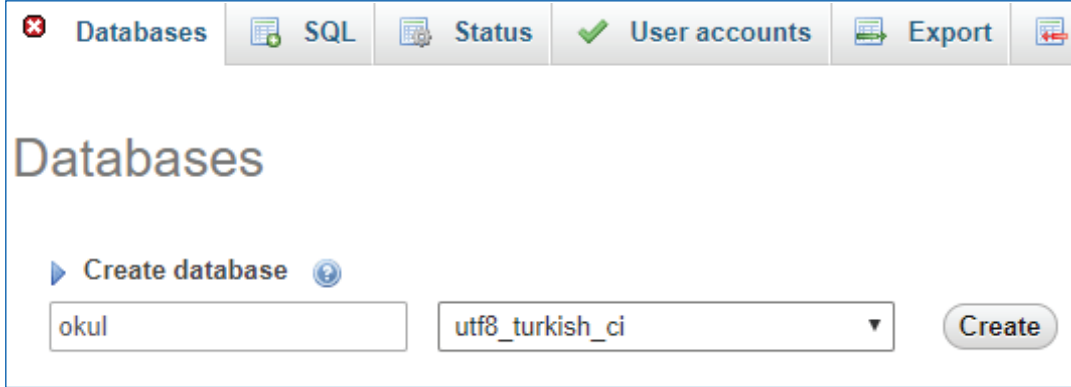
Bilgisayarınızda herhangi bir tarayıcı açılıp ve <http://localhost/phpmyadmin/> adresine bağlanılır. Burada localhost, web sitesi kaynaklarının tarayıcının açık olduğu bilgisayarda olduğu belirtilmektedir. Eğer XAMPP kurulu olmayan bir bilgisayarda çalışıyorsanız dipnottaki adresini kullanarak, phpmyadmin 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 içerisinde 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 sonradan erişilmek istendiğinde tekrar Databases sekmesine tıklanarak ulaşılabilir. Eğer veri tabanını silmek isterseniz veri tabanının en solundaki kutucuğu seçerek Drop (Çıkar) butonuna tıklayabilirsiniz.

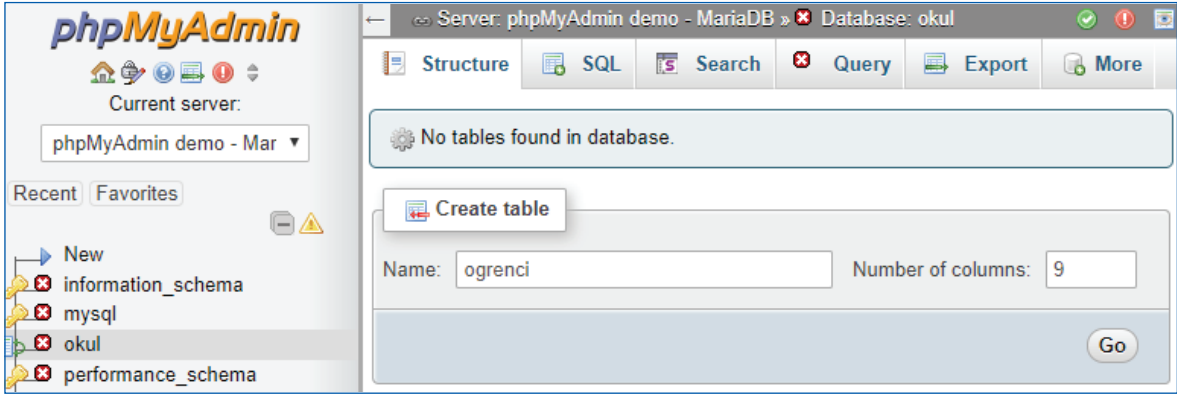
Database	Collation	Master replication	Action
<input type="checkbox"/> information_schema	utf8_general_ci	Replicated	Check privileges
<input type="checkbox"/> mysql	latin1_swedish_ci	Replicated	Check privileges
<input checked="" type="checkbox"/> okul	utf8_turkish_ci	Replicated	Check privileges
<input type="checkbox"/> performance_schema	utf8_general_ci	Replicated	Check privileges
<input type="checkbox"/> phpmyadmin	utf8_bin	Replicated	Check privileges
Total: 5		latin1_swedish_ci	

Check all With selected: Drop

## Tablo Oluşturma

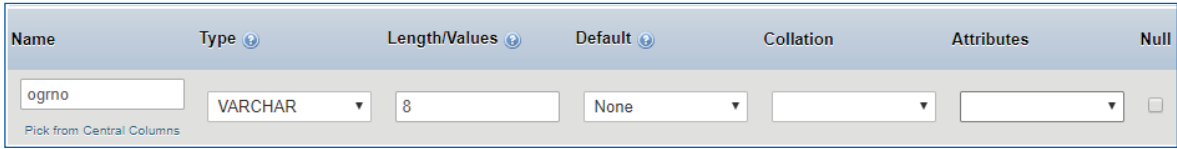
Bir veri tabanı tablolardan, tablolar ise satır ve sütunlardan oluşur. Şimdi oluşturduğunuz veri tabanına tablo ekleyiniz. Bunun için öncelikle tablo içeriğimiz üzerinde tartışalım. Bir okulu ele aldığımızda, okul içerisinde; öğretmen, öğrenci, alt yapı, mali işler vb. alt varlıklar vardır. İlişkisel 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

- Databases 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 karşılaşacaksınız. Şu an bizim için önemli olan 7 özellik üzerinde duracağız.



### Name (Sütun İsmi)

Öğrenci varlığı 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 ünite de 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 sütun isimleri anlamlı şekilde kısaltılmalıdır (Ör: YazNot1, YazNot2).

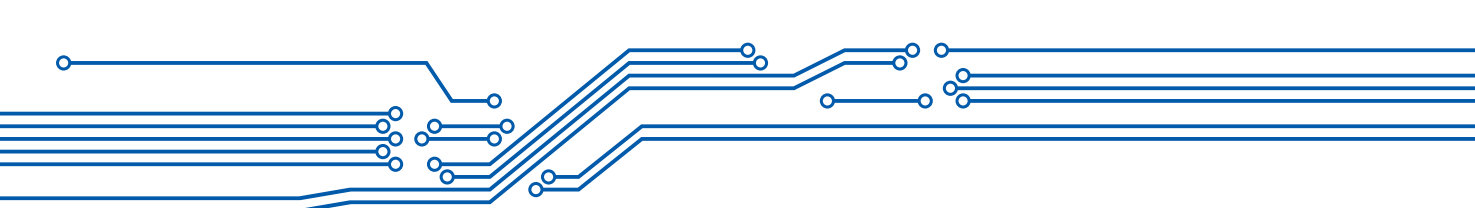
### Type (Veri Türü)

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 tutulacağını 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 en yüksek 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 atmasına 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

Name	Type	Length/Values
ogrnno	VARCHAR	8
yas	TINYINT	2
ortalama	DECIMAL	4,2

### Default

Veri tanımlaması yapıldıktan sonra bazı sütunlara herhangi bir değer girilmemesi durumunda varsayılan olarak bu sütunlarda hangi değerın görüntüleneceğinin ayarlanmasını sağlar. None, As defined, NULL ve CURRENT\_TIMESTAMP seçenekleri vardır. Eğer varsayılan olarak

- ✓ Kendinizin tanımlandığı bir değerın 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.

Name	Type	Length/Values	Default
ogrnno	VARCHAR	8	None
yas	TINYINT	2	As defined: 6
ortalama	DECIMAL	4,2	NULL
tarihsaat	DATETIME		CURRENT_TIME

## 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.

Name	Type	Length/Values	Default	Collation
adres	TEXT	255	NULL	utf8_unicode_ci

## Attributes

Veri tanımlaması yaparken integer(int) negatif değer alınabilir. Sadece pozitif değerli veri girişi yapılması isteniyorsa Attributes özelliği “UNSIGNED” olarak değiştirilmelidir.

Name	Type	Length/Values	Default	Collation	Attributes
ortalama	DECIMAL	4,2	NULL		UNSIGNED

## NULL

Eğer sütuna veri girişi zorunlu tutulmak istenilmiyorsa null seçeneğine tik atılması gerekir. Böylece ilk veri girişinde sütun doldurulmadan boş çözülebilir.

Name	Type	Length/Values	Default	Collation	Attributes	Null
ortalama	DECIMAL	4,2	NULL		UNSIGNED	<input checked="" type="checkbox"/>

Bir veri tabanı içerisinde tablo oluştururken önemli olan 7 özellik ve veri türleri görüldü. Okul veri tabanı içerisinde öğrenci varlığı ile ilgili oluşturulacak tablo şekildeki gibidir.

Name	Type	Length/Values	Default	Collation	Attributes	Null
OgrNo	VARCHAR	8	None	utf8_turkish_ci		<input type="checkbox"/>
Adi	VARCHAR	50	NULL	utf8_unicode_ci		<input checked="" type="checkbox"/>
Soyadi	VARCHAR	50	NULL	utf8mb4_unicode		<input checked="" type="checkbox"/>
Yasi	TINYINT	2	As defined: 6		UNSIGNED	<input type="checkbox"/>
DogumTarihi	DATE		NULL			<input checked="" type="checkbox"/>
Adresi	TEXT		NULL	utf8_unicode_ci		<input checked="" type="checkbox"/>
Dersi	VARCHAR	50	NULL	utf8_turkish_ci		<input checked="" type="checkbox"/>
Yaz1	TINYINT	3	NULL		UNSIGNED	<input checked="" type="checkbox"/>
Yaz2	TINYINT	3	NULL		UNSIGNED	<input checked="" type="checkbox"/>
Yaz3	TINYINT	3	NULL		UNSIGNED	<input checked="" type="checkbox"/>
Ortalama	DECIMAL	5,2	As defined: 0.00			<input type="checkbox"/>
Durum	TINYINT	1	NULL			<input checked="" type="checkbox"/>

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
<input type="checkbox"/>	1	OgrNo	varchar(8)		No	None
<input type="checkbox"/>	2	Adi	varchar(50)		Yes	NULL
<input type="checkbox"/>	3	Soyadi	varchar(50)		Yes	NULL
<input type="checkbox"/>	4	Yasi	tinyint(2)	UNSIGNED	No	6
<input type="checkbox"/>	5	DogumTarihi	date		Yes	NULL
<input type="checkbox"/>	6	Adresi	text		Yes	NULL
<input type="checkbox"/>	7	Dersi	varchar(50)		No	None
<input type="checkbox"/>	8	Yaz1	tinyint(3)	UNSIGNED	Yes	NULL
<input type="checkbox"/>	9	Yaz2	tinyint(3)	UNSIGNED	Yes	NULL
<input type="checkbox"/>	10	Yaz3	tinyint(3)	UNSIGNED	Yes	NULL
<input type="checkbox"/>	11	Ortalama	decimal(5,2)		No	0.00
<input type="checkbox"/>	12	Durum	tinyint(1)		Yes	NULL

Check all    With selected:

column(s)

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 durumunda kayıt girişlerinin yapılamaması istenir. Bu durumda, her iki sütunun seçilerek tablonun altındaki Primary butonuna tıklanması gerekmektedir.

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
<input checked="" type="checkbox"/>	1	OgrNo				No	None	Change
<input type="checkbox"/>	2	Adi			Yes	NULL		Change
<input type="checkbox"/>	3	Soyadi			Yes	NULL		Change
<input type="checkbox"/>	4	Yasi		UNSIGNED	No	6		Change
<input type="checkbox"/>	5	DogumTarihi			Yes	NULL		Change
<input type="checkbox"/>	6	Adresi			Yes	NULL		Change
<input checked="" type="checkbox"/>	7	Dersi			No	None		Change
<input type="checkbox"/>	8	Yaz1		UNSIGNED	Yes	NULL		Change
<input type="checkbox"/>	9	Yaz2		UNSIGNED	Yes	NULL		Change
<input type="checkbox"/>	10	Yaz3		UNSIGNED	Yes	NULL		Change
<input type="checkbox"/>	11	Ortalama			No	0.00		Change
<input type="checkbox"/>	12	Durum			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 kayıta 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 istediğiniz tabloyu seçtikten sonra, Insert Sekmesine tıklayınız. Daha sonra alanlara değerleri yazınız ve Go butonuna basınız.

Column	Type	Function	Null	Value
OgrNo	varchar(8)			01012424
Adi	varchar(50)		<input type="checkbox"/>	Selvi
Soyadi	varchar(50)		<input type="checkbox"/>	Güzel
Yasi	tinyint(2) unsigned			15
DogumTarihi	date		<input checked="" type="checkbox"/>	
Adresi	text		<input checked="" type="checkbox"/>	
Dersi	varchar(50)			Matematik
Yaz1	tinyint(3) unsigned		<input type="checkbox"/>	50
Yaz2	tinyint(3) unsigned		<input type="checkbox"/>	60
Yaz3	tinyint(3) unsigned		<input type="checkbox"/>	90
Ortalama	decimal(5,2)			66.66
Durum	tinyint(1)		<input type="checkbox"/>	

Go

### 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.

Show all | Number of rows: 25 | Filter rows: Selvi

Sort by key: None

+ Options

	OgrNo	Adi	Soyadi	Yasi	DogumTarihi	Adresi	Dersi	Yaz1	Yaz2	Yaz3	Ortalama	Durum
<input type="checkbox"/>	01012424	Selvi	Güzel	15	2002-04-21	Güçlükaya mah. Fizik		75	85	100	86.66	1
<input type="checkbox"/>	01012424	Selvi	Güzel	15	2002-04-21	Güçlükaya Mah. Kimya		97	100	87	94.66	NULL
<input type="checkbox"/>	01012424	Selvi	Güzel	15	NULL	NULL Matematik		50	60	90	66.66	0

### Kayıt Güncelleme

Herhangi bir kaydı güncellemek istediğinizde, kaydın solundaki Edit butonuna tıkladıktan sonra kayıt eklemede yaptığınız gibi yeni değer girebilir ve güncelleme işlemi yapabilirsiniz.

		OgrNo	Adi	Soyadi	Yasi	DogumTarihi	Adresi
<input type="checkbox"/>	Edit Copy Delete	01012424	Selvi	Güzel	15	2002-04-21	Güçlükaya Mah.
<input checked="" type="checkbox"/>	Edit Copy Delete	01012424	Selvi	Güzel	15	NULL	NULL

Check all With selected: Edit Copy Delete Export

### Kayıt Silme

Herhangi bir kaydı silmek istediğinizde, kaydın solundaki Delete butonuna tıkladıktan sonra karşınıza gelen ekrandaki uyarıyı onaylayarak silme işlemi yapabilirsiniz.

		OgrNo	Adi	Soyadi
<input type="checkbox"/>	Edit Copy Delete	01012323	Mete	Yıldırım
<input type="checkbox"/>	Edit Copy Delete	01012323	Mete	Yıldırım
<input type="checkbox"/>	Edit Copy Delete	01012424	Selvi	Güzel
<input type="checkbox"/>	Edit Copy Delete	01012424	Selvi	Güzel
<input checked="" type="checkbox"/>	Edit Copy Delete	01012424	Selvi	Güzel

**Confirm**

Do you really want to execute "DELETE FROM `ogrenci` WHERE `ogrenci`.`OgrNo` = '01012424' AND `ogrenci`.`Dersi` = 'Matematik'?"

OK Cancel

### SQL Kodları

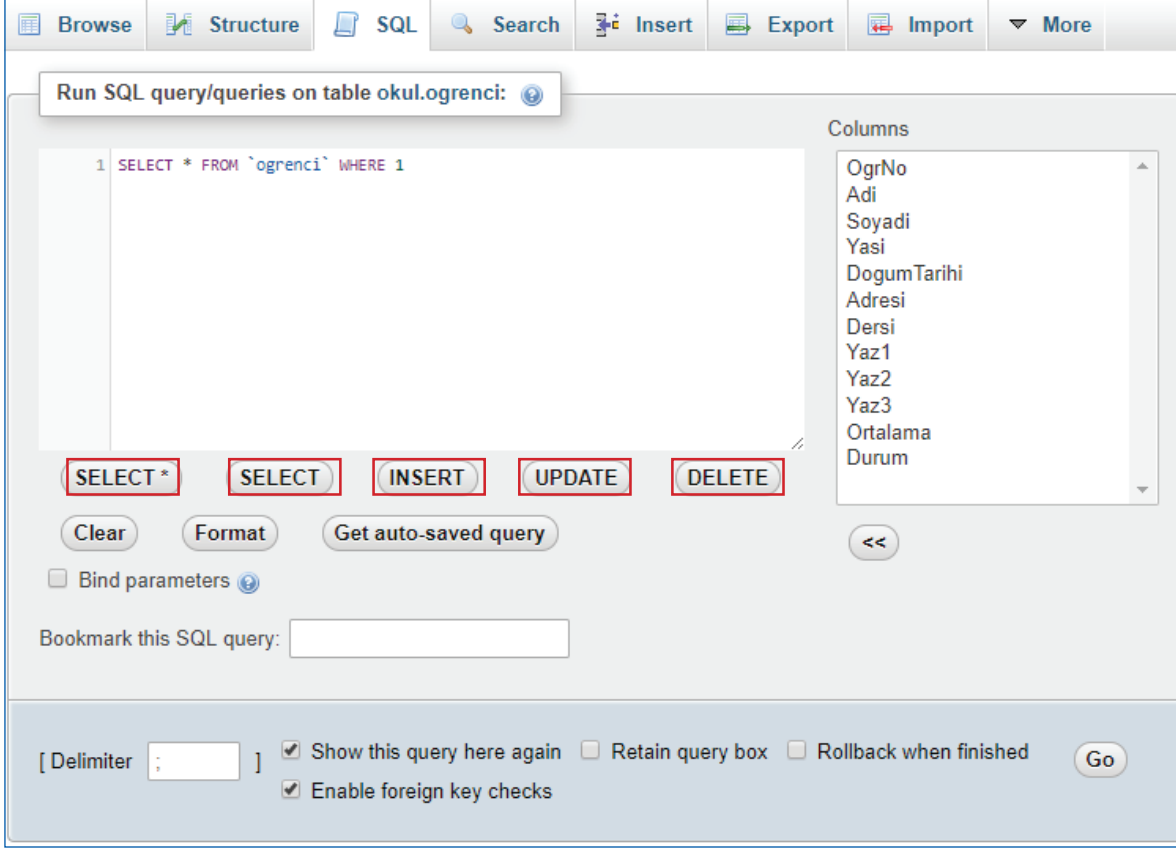
phpMyAdmin kullanıcılara kod yazımı olmadan kolay bir veri yönetimi sunar. Peki, arka planda çalışan kod nasıldır?

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. Gerekğinde 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 ünite, 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.



The screenshot shows a SQL query editor interface. At the top, there are tabs for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', and 'More'. Below the tabs, there is a header that says 'Run SQL query/queries on table okul.ogrenci:'. The main area contains a text box with the SQL query: '1 SELECT \* FROM `ogrenci` WHERE 1'. To the right of the text box is a 'Columns' list containing: 'OgrNo', 'Adi', 'Soyadi', 'Yasi', 'DogumTarihi', 'Adresi', 'Dersi', 'Yaz1', 'Yaz2', 'Yaz3', 'Ortalama', and 'Durum'. Below the text box are buttons for 'SELECT \*', 'SELECT', 'INSERT', 'UPDATE', and 'DELETE'. There are also buttons for 'Clear', 'Format', and 'Get auto-saved query'. A checkbox labeled 'Bind parameters' is checked. Below that is a text box for 'Bookmark this SQL query:'. At the bottom, there is a section for query options: '[ Delimiter ; ]', a checked checkbox for 'Show this query here again', an unchecked checkbox for 'Retain query box', an unchecked checkbox for 'Rollback when finished', and a 'Go' button. There is also a checked checkbox for 'Enable foreign key checks'.

## SELECT

Veri seçimi için kullanılan SQL komutudur.

```
SELECT sutun1, sutun2, ...  
FROM tablo_adi  
WHERE koşul  
ORDER BY sutun1, sutun2, ... ASC|DESC;
```

Bir tablo içerisinde 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 sutun1 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.','Müzik','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 kosul;
```

```
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 Adres='Gölbasi Mah. Oğuzlar ÇORUM';
```

*/\* Koşulsuz bir update kodu (Where olmadan) yazarsanız, tablonuz içerisindeki bütün kayıtların, belirtmiş olduğunuz sütun yada sütunların (Ör: Adres) değeri aynı olur. Update kodunu bu şekilde kullanırken çok dikkatli olmalısınız!\*/*

## DELETE

Veri silme için kullanılan SQL komutudur.

```
DELETE * FROM tablo_adi
WHERE kosul;
```

```
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 ogrenci
```

*/\* 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 içerisinde 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 konuşurmak 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 ünite de 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.

### Örnek 1

Adınız ...  
Soyadınız ...  
Düşünceleriniz ...

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 kullanıcılarından 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
//ilk 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 ögesi
    içerisindeki butona tıklanmadı.</p>";
}
?>

```

## Sayfa İlk Açıldığında

Örnek 3

İsim:

E-posta:

Sayfa POST edilmedi. Bir diğer ifadeyle methodu post olan bir form ögesi içerisindeki butona tıklanmadı.

## Veri Girişi Yapıp Göndere Basıldığında

Örnek 3

İsim:

E-posta:

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
//ilk 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 ögesi
içerisindeki butona tıklanmadı.</p>";
}
?>
```

← → ↻ 🏠 ⓘ localhost/phpsite/test.php?ad=Mutlu&eposta=mutlu%40test.com

Örnek 3

İsim:

E-posta:

Sistem **Mutlu** adlı kişiye **mutlu@test.com** eposta adresini kullanarak mail atacak.

### Dikkat!

- Taşınan verilerin ismi ve değeri ile birlikte tarayıcının URL satırında görüntülediği 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.

İsim:  
Soyisim:  
E-mail:  
Cinsiyet:  
 Erkek  
 Kadın  
Mesaj:  
Formu gönder

### 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>
```

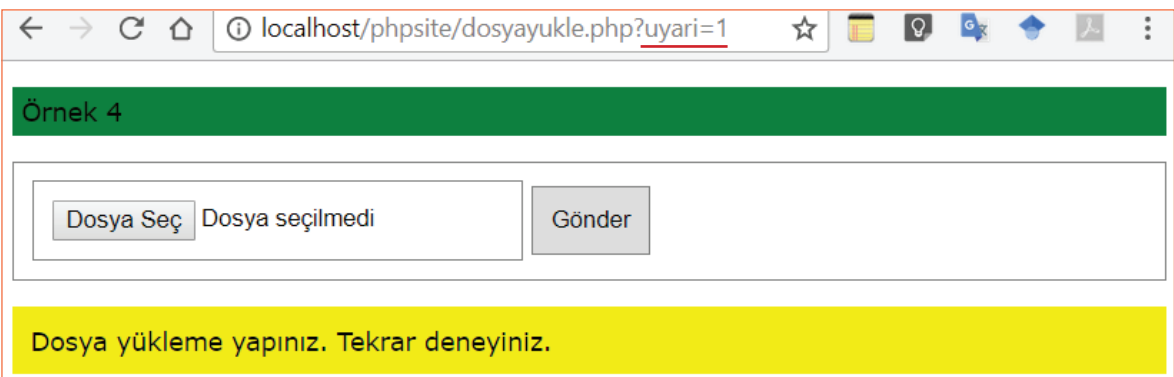
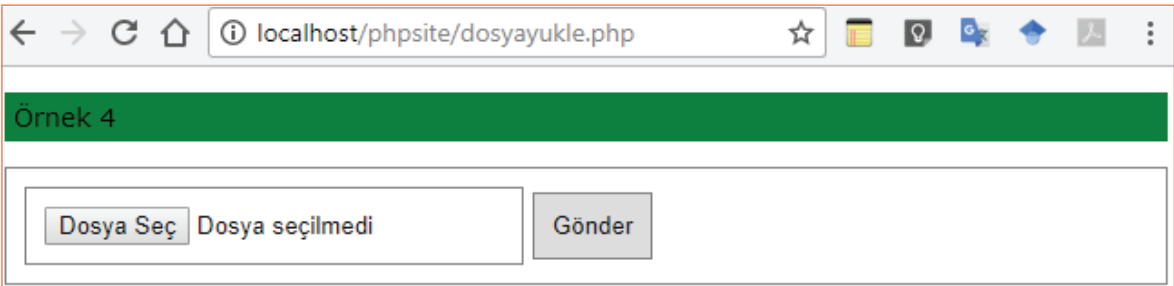
## CSS Kodları (dosyayukle.php)

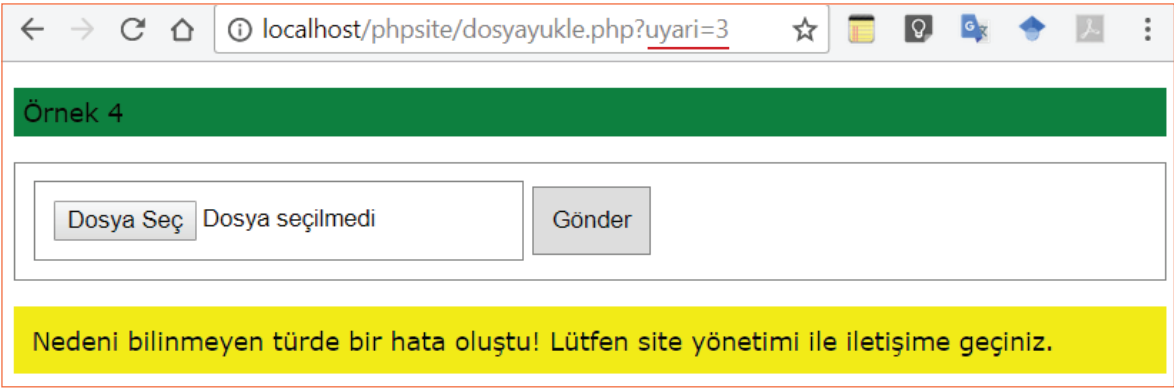
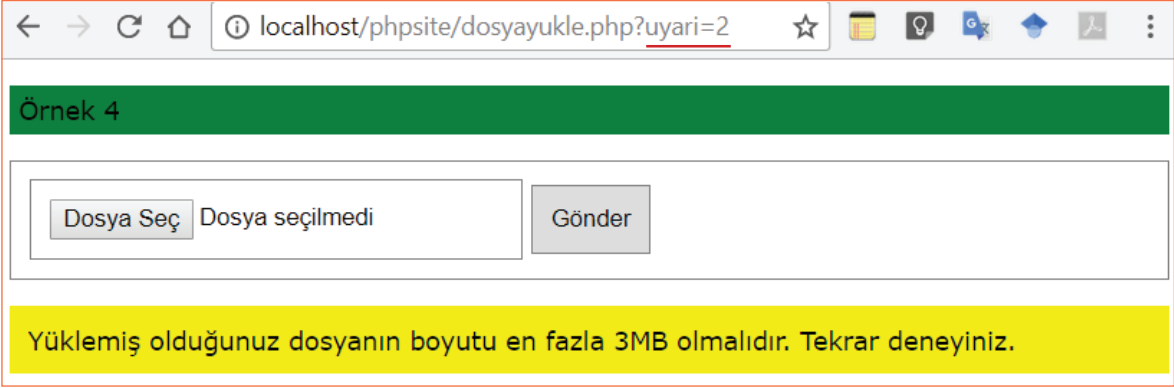
```
<style>
  body    {font-family:Verdana;font-size:14px;}
  .baslik {padding:5px; background-color:green;}
  form { border:1px solid grey; padding:10px;}
  input { border:1px solid grey; padding:10px;}
  .uyari {padding:10px; background-color:yellow;}
</style>
```

## 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
// 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 algorithmada, 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

```

$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 algorithmadaki yerlerine uygun şekilde yazalım. Daha sonra sayfamızı test edelim.

#### Örnek 4

Dosyanız başarılı şekilde sunucuya yüklendi!

[Yeni Dosya Yükle](#)

**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 kullanıcı 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.

The image shows a web browser window displaying a registration form titled 'uyeol.php'. The form is set against an orange background. It contains five input fields: 'Kullanıcı Adı', 'Şifreniz', 'Adı-Soyadı', 'E-Posta', and 'Özgeçmiş'. A 'Kayıt Ol' button is positioned at the bottom center of the form.

### 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 platformları birbiri ile konuşturmalıyız. Bunu yaparken php kodlarını kullanacağız. İşlem adımlarımız şekildeki gibidir.



- mysql\_connect



- mysql\_select\_db



- mysql\_query



- mysql\_close

## Veritabanına Bağlan

```
<?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 adı ve şifresi ile
// sunucuya bağlanabilirsiniz.

$baglanti = mysql_connect($veritabanisunucusu, $vt_kullanicisi, $vt_kullanicisi_sifre);
// Veritabanımıza mysql_connect kodu ile bağlanıyoruz.
// Kodla beraber, veri tabanı sunucusunun adı ya da ip adresi
// veritabanı kullanıcısı ve veritabanı kullanıcısının şifresi
// belirtilmelidir.
// Kod çalıştığı anda 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

```
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 karşılaşılma ise, artık sorgumuzu gönderebiliriz. Senaryomuz gereği kayıt ekleme işlemini yapacağız. Bu işlem için, kullanıcı 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.

$insert_sql= "insert into kullanıcı (kullanıcıAdi, sifre, adıSoyadı, ePosta, ozgecmisi)
values ($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 işleminin gerçekleşip gerçekleşmediğini kontrol ediyoruz.
```

### Bağlantıyı Kapat

Kayıt ekleme işlemimiz de bittiğine göre artık veri tabanı bağlantımızı kapatalım.

```
mysql_close($bağlantı);
// 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**

```
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.

```
if ($baglanti==true) {  
    echo "Veritabanı bağlantısı sağlandı!<br/>";  
  
    mysql_query("SET NAMES UTF8");  
    // Karakter setinin UTF- olarak ayarlanması  
  
    // Kodun devamı ...
```

## Üye olma ekranının görünümü

uyeol.php

Kullanıcı Adı	<input type="text" value="dnzyil"/>
Şifreniz	<input type="password" value="...."/>
Adı-Soyadı	<input type="text" value="Denizer Yıldırım"/>
E-Posta	<input type="text" value="test@test.com"/>
Özgeçmiş	<input type="text" value="Doğum tarihim ... .&lt;br/&gt;Eğitim durumum ... .&lt;br/&gt;... görevlerde bulundum."/>
	<input type="button" value="Kayıt Ol"/>

Veritabanı bağlantısı sağlandı!  
Veritabanı seçimi başarılı!  
Kayıt başarılı şekilde eklendi.

## 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.

giris.php

Kullanıcı Adı

Şifre

## 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



- mysql\_connect



- mysql\_select\_db



- mysql\_query



- mysql\_close

Veri ekleme işleminde olduğu gibi Veritabanına Bağlanma, Veri tabanı Seçme ve bağlantı kapama işlemlerini aynı şekilde yapıyoruz.

```
// --Form verilerini çekme--
$yadi = "".$_POST["kadi"]."";
// Form verilerinin öncesinde ve sonrasında
// ' işaretleri eklediğimiz dikkatinizi çekmiştir.
// Veritabanına sql sorgusu göndereceğimiz için
// değerleri sorgu biçimine uygun hale getirdik.

$sql = "select * from kullanıcı where kullanıcıAdi= $yadi";
// Kullanıcı adı, giriş.php sayfasında
// kadi isimli form elementine girilen değer olan bir kaydı
// getirecek sql cümlesi...

$sorgulamaislemi = mysql_query($sql);
// Son olarak oluşturmuş olduğumuz select cümlecığının
// 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 kayıta içerebilir.
// Tersisi durumda ise false döndürülür.
```

```
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!";
    }
}
```

```
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!!!

    $satisir = mysql_fetch_assoc($sorgulamaislemi);
    // ilk olarak kaynağımız içerisindeki verilere,
    // kolayca ulaşabilmemiz için
    // kaynağımızı mysql_fetch_assoc() ile bir diziye dönüştürüyoruz.

    if ($satisir['sifre'] == $_POST["sifre"]) {
    } else {
        // Kullanıcının girmiş olduğu kadi veritabanında yok demektir.
        echo "Kullanıcı Adı Bulunamadı!";
    }
}
```

```

if ($satis['sifre'] == $_POST["sifre"]) {
    // Veritabanındaki şifre ile kullanıcının girmiş olduğu şifre eşleşiyorsa;
    // kullanıcının rolü kontrol edilir.
    if ($satis["rolu"] == 1) {
    } else {
        // Kullanıcının girmiş olduğu şifre ile veritabanındaki şifre eşleşmiyorsa;
        echo "Şifrenizi Yanlış Girdiniz!";
    }
}

```

```

if ($satis["rolu"] == 1) {
    // kullanıcının rolü 1 (yönetici) ise;
    //oturum başlatılıp yönetim.php sayfasına yönlendirme sağlanabilir.
    session_start(); // oturumu başlat
    $_SESSION["adsoyad"] = $satis["adiSoyadi"];
    header("Location: yönetim.php");
}else{
    // kullanıcının rolü 0 (yönetici değil) ise;
    echo "Yönetici Rolünüz Atanmamıştır. Giriş Yapamazsınız.";
}

```

Sayfayı test ettiğinizde, eğer tüm kodlarınız içerisinde herhangi bir sorunla karşılaşırsa, 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. Veritabanı bağlantısı ve veritabanı seçimi
<?php ### 5. Veri güncelleme ve silme işlemleri
<?php ### 4. Veritabanındaki kullanıcı 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
    session_start();
    // session kullanımını başlatıyoruz.

    // giris.php sayfasında;
    // kullanıcı adı ve şifresi doğrulanan kişinin
    // adsoyad için bir session açmıştık.
    // Not: kullanıcı idsi için de açabilirdik.

    if (isset($_SESSION["adsoyad"]) == false)
    {
        // eğer daha önce adsoyad 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 ### Veritabanı 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 kurulamadı! Hata:".mysql_error());}

?>
```

## 3. Veri tabanı bağlantısının kapatılması

```
<?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 ### Veritabanındaki kullanıcı tablosundan tüm kayıtların getirilmesi

$sql = "select * from kullanıcı";
// Veritabanındaki kullanıcı 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

```
### 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>Özgeçmiş</td>";
echo "<td>Rolü</td>";
echo "<td>Düzenleme</td>";
echo "<td>Silme</td>";
// tr etiketinin altında td etiketleri
// ilk satır sütunların başlıklarını içermektedir.

echo "</tr>";
// ilk satırın kapatılması
```

```

### Kayıtların sayfada gösterimi
### Tablo içeriğinin veritabanından döndürülen veri kaynağı içerisindeki
### ($sorgulamaislemi) verilerle doldurulması
while($satisir = mysql_fetch_assoc($sorgulamaislemi)) {
    // $sorgulamaislemi kaynağındaki veriler bitene kadar
    // her seferinde $satisir değişkenine
    // bir kaydın bilgileri dizi olarak getirilir.
    // Kaynaktaki veriler bittiğinde while döngüsü sona erer.

    echo "<tr>";
    echo "<td>" . $satisir["kullaniciAdi"]. "</td>";
    echo "<td>" . $satisir["sifre"]. "</td>";
    echo "<td>" . $satisir["adiSoyadi"]. "</td>";
    echo "<td>" . $satisir["ePosta"]. "</td>";
    echo "<td>" . $satisir["ozgecmisi"]. "</td>";
    echo "<td>" . $satisir["rolu"]. "</td>";
    // while döngüsü sonlandırılana kadar,
    // tablonun ilk satırını oluştururken yapmış olduğumuz gibi
    // tabloya bir kayıt eklenir ve sütunların içeriği
    // $satisir dizi elemanları ile doldurulur.

    echo '<td><a href="yonetim.php?eylem=duzenle&kadi='
    .$satisir['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='
    .$satisir['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

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	<a href="#">Düzenle</a>	<a href="#">Sil</a>
Myildiri	09876	Mutlu Yıldırım	happy@test.com	Ankara Üniversitesi Siyasal Bilimler ve kamu Yönetimi mezunuyum.	0	<a href="#">Düzenle</a>	<a href="#">Sil</a>
Sguzel	345987	Selvi Güzel Yıldırım	selvi@test.com	MEB Fen ve Teknoloji Öğretmeni...	0	<a href="#">Düzenle</a>	<a href="#">Sil</a>

## 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. Dolayısıyla ilk olarak sayfamızda get yöntemi ile taşınan bu isimlerde veriler olup olmadığını kontrol etmemiz gerekmektedir. Bu bölümdeki kodlamamızın genel algoritması aşağıda gösterilmektedir.



```
<?php ### 5. Veri güncelleme ve silme işlemleri

    if (isset($_GET["kadi"])&&isset($_GET["eylem"]))
    {
        if ($_GET["eylem"]=="duzenle"){
            ### 1.Veritabanındaki kullanıcı tablosundaki tüm kayıtları getiren sql cümlesi...
        }
    }else if ($_GET["eylem"]=="sil"){
        ### 3. Veri tabanından veri silme işlemi
    }
    ### 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 öğelerine değerler yazıyorduk. Şimdi ise değerleri girilmiş verimiz üzerinde değişiklik yapmak istiyoruz. Dolayısı ile, güncelleme yapmak istediğimiz kaydın sütun değerlerini veri tabanından çekerek, form elemanları içerisine yazmamız gerekmektedir.

```
### Veri Güncelleme İşlemleri
$sql = "select * from kullanıcı where kullanıcıAdi='".$_GET["kadi"].'";
// Veritabanındaki kullanıcı tablosundaki tüm kayıtları getiren sql cümlesi...

$sorgulamaislemi = mysql_query($sql);
// select cümleciğininin 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 içeriğininin kullanıcı tablosundaki verilerle doldurulması
    $satisir = mysql_fetch_assoc($sorgulamaislemi);
    // $sorgulamaislemi kaynağında aynı kullanıcı adı ile birden fazla kayıt olamayacağı
    için while döngüsü açmadık.
    // kullanıcı tablomuzda birincil anahtar olarak kullanıcıAdi sütununu atamıştık.

    echo "<tr>";
    echo "<td>Kullanıcı Adı: <input name=kadi type=text value='".$satisir['kullanıcıAdi'].
    "'/></td>";
    echo "<td>Şifresi:<input name=sifre type=text value='".$satisir['sifre']."'/></td>";
    echo "<td>Adı Soyadı: <input name=adsoyad type=text value='".$satisir['adıSoyadı'].
    "'/></td>";
    echo "<td>E-postası:<input name=eposta type=text value='".$satisir['ePosta']."'/></td>";
    echo "<td>Özgeçmiş:<textarea name=ozgecmis>".$satisir['ozgecmisi'].</textarea></td>";
    echo "<td>Rolü:<input name=rol type=text value='".$satisir['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

<b>Kullanıcı Adı:</b> Myildiri	<b>Şifresi:</b> 09876	<b>Adı Soyadı:</b> Mutlu Yıldırım	<b>E-postası:</b> happy@test.com	<b>Özgeçmişi:</b> Ankara Üniversitesi Siyasal Bilimler ve kamu Yönetimi	<b>Rolü:</b> 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	<a href="#">Düzenle</a>	<a href="#">Sil</a>
Myildiri	09876	Mutlu Yıldırım	happy@test.com	Ankara Üniversitesi Siyasal Bilimler ve kamu Yönetimi mezunuyum.	0	<a href="#">Düzenle</a>	<a href="#">Sil</a>
Sguzel	345987	Selvi Güzel Yıldırım	selvi@test.com	MEB Fen ve Teknoloji Öğretmeni...	0	<a href="#">Düzenle</a>	<a href="#">Sil</a>

## 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.

    $updatesql = "Update  kullanıcı set kullanıcıAdi=$kadi, sifre=$sifre
, adıSoyadı=$adsoyad, ePosta=$eposta, ozgecmisi=$ozgecmis,rolu=$rol
where kullanıcıAdi=$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.

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	<a href="#">Düzenle</a>	<a href="#">Sil</a>
Myildiri	09876	Mutlu Yıldırım	mutluyildirim@test.com	Ankara Üniversitesi Siyasal Bilimler ve kamu Yönetimi mezunuyum.	1	<a href="#">Düzenle</a>	<a href="#">Sil</a>
Sguzel	345987	Selvi Güzel Yıldırım	selvi@test.com	MEB Fen ve Teknoloji Öğretmeni...	0	<a href="#">Düzenle</a>	<a href="#">Sil</a>

## Veri Silme İşlemi

```
### Veri Silme İşlemleri
$yadi= "". $_GET["yadi"]. "";
// delete sql cümlesi için gerekli olan değeri
// uygun biçimde yeni değişkenlere aktarıyoruz.

$silmesql ="Delete from kullanıcı where kullanıcıAdi='". $_GET["yadi"].
"";
// 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 tabanındaki verilerin güncel hali ekranda gösterilecektir. Örneğin üç kaydımız vardı. “Myildiri” kullanıcı isimli veriyi sildiğimizde ekranda sadece iki kayıt gösterilmelidir.

## Ekran Görüntüsü

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	<a href="#">Düzenle</a>	<a href="#">Sil</a>
Sguzel	345987	Selvi Güzel Yıldırım	selvi@test.com	MEB Fen ve Teknoloji Öğretmeni...	0	<a href="#">Düzenle</a>	<a href="#">Sil</a>

## 7. WEB TABANLI PROJE GELİŐTİRME

### 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 olduğlarınızı 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 kullanılacağını 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 yapabilesidir.

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? İnternette 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 kullanıcılarına daha uygun olur?

En basit ve sade şekilde sitenin amacına uygun bir tasarımı kağıt üzerinde çizin. Sitenizde kaç sayfa olmasını bekliyorsanız bu sayfalar arasındaki bağlantıları da çizin. 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 deri 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 paylaşılacağı 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

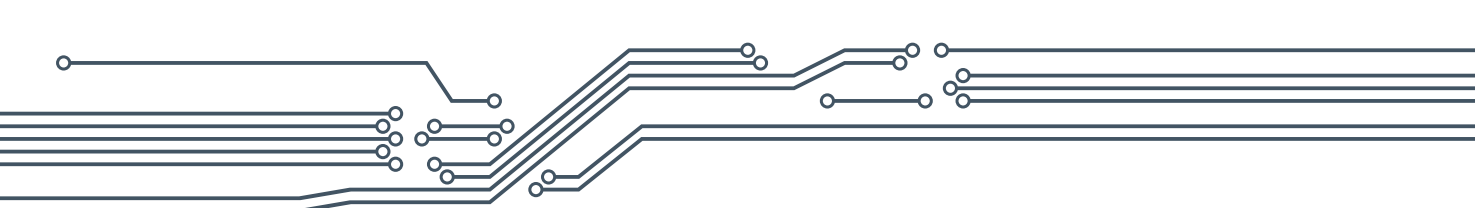
Bilişim öğretmeni için sizlere vermiş olduğu ödevleri kolayca takip ederek değerlendirebileceği bir web sitesi geliştirebilirsiniz. Bu siteyi basitçe senaryolaştıracak olursak;

- 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 tıkladığında, sizlerin videoya çekmiş oldukları kodlama örnekleri, tıklanan başlığa göre veri tabanından çağrılır.

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 laboratuvarında bulunan araçları listeleyebilecek, kolayca güncelleştirilmesini sağlayabilecek ve istenildiğinde bu araçlar ile ilgili tanıtıcı bilgilere de ulaşılacak 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. Site-nize kullanıcılar üye olabilir ve kendi deneyimlerini paylaşabilir.

### **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 PROGRAMLAMA

## 1. Mobil Uygulama Geliştirmeye Giriş

İnternet bilgiye erişim süreçleri üzerinde çok önemli değişikliklere yol açmıştır. Bilgi kaynaklarına erişim ağ üzerinden gerçekleşmeye başladığı andan itibaren mekândan 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. 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.

### 1.1. Mobil Programlamadaki Temel Kavramlar

Mobil programlama, uygulamanın çalışacağı platformun özelliklerine uygun olarak özellikle çalışma performansı açısından dikkatli bir geliştirme sürecine ihtiyaç duymaktadır. Mobil cihazların sınırlı donanım özellikleri ve kısıtlı enerji kaynakları nedeniyle geleneksel program geliştirme yaklaşımının dışına çıkılması gerekmektedir.

### 1.2. Uygulama Geliştirirken Kullanılan Tasarım Yapıları

Mobil uygulama geliştirirken kullanılacak 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 hibrit olarak adlandırılan yerel ve Web tabanlı uygulamaların iç içe kullanıldığı yapılar da bulunmaktadır.

## 2. MOBİL DONANIM



## 2. Mobil Donanım

Mobil donanım, standart bilgisayar mimarisi ile paralellik içerir ve daha fazlası olarak çeşitli sensörlere sahiptir.

### 2.1. Donanım Bileşenleri

Mobil donanımlarda bulunan temel bileşenler aşağıda listelenmiştir:

- a. İşlemci
- b. Anakart
- c. Bellek
- ç. Ekran
- d. Depolama alanı

Bu temel bileşenlere ek olarak çeşitli çoklu ortam bileşenleri de mobil cihazların olmazsa olmaz donanımları arasına girmiştir. Bu bileşenlerden bazıları (kamaralar, mikrofonlar vb.) aynı zamanda algılayıcı (sensör) olarak da kullanılmaktadır:

- a. Ön kamera
- b. Arka kamera
- c. Hoparlör
- ç. Mikrofon

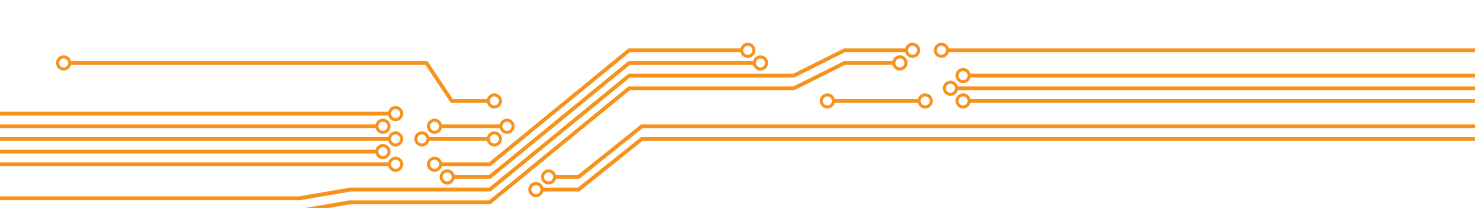
Mobil cihazların dış dünya ile iletişim kurabilmesi açısından gerekli olan bağlantı modülleri de önemli mobil donanım bileşenleri olarak öne çıkmaktadır:

- a. Kablosuz ağ bağlantısı
- b. Bluetooth
- c. GPS
- ç. NFC
- d. USB
- e. USB Type-C

Tüm bunlara ek olarak mobil donanımı belki de diğer bilgisayar sistemlerinden ayıran en önemli özellik sahip oldukları algılayıcılardır (sensörler). 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 algılayıcılar sağlamaktadır.

### 2.2. 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 bulunmaktadır. Bu merkezi işlem birimi günümüz teknolojisinin izin verdiği sınırlar çerçevesinde çoğunlukla çoklu 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.

### 2.3. Donanım Bileşenlerinin Programlanması

Donanım bileşenlerinin programlanması, genel olarak ilgili donanım bileşeni için oluşturulmuş kütüphanenin uygulama içerisinde çağırılarak kullanılması şeklinde gerçekleştirilir. Mobil cihazlarda çalışacak işletim sistemlerinin geliştirme ortamlarında donanıma özel olarak hazırlanmış fonksiyonlar donanım kütüphanesi olarak adlandırılır.

### 2.4. Mobil Uygulama Geliştirme Araçları

Mobil uygulama geliştirirken kullanılacak araçları üç kategoride değerlendirmek mümkün olabilir. Bu kategoriler:

1. Yazılım geliştirme ortamı
2. Uygulama test ortamı
3. Uygulama yayınlama platformu

Her kategori içerisinde yer alacak araç öncelikle mobil uygulamanın çalışacağı işletim sistemi açısından farklılık gösterebilmektedir. Yazılım geliştiriciler, geliştirme aracı seçiminde hedefledikleri işletim sistemini veya uygulama platformunu dikkate alarak araç seçiminde bulunurlar.

### 2.5. Mobil İşletim Sistemleri

Mobil işletim sistemi, diğer tüm bilgisayar sistemlerinde olduğu gibi cihazın en etkin şekilde işletilebilmesi için donanıma özel olarak geliştirilmiş ana yazılımdır. Mobil cihazların donanım yapılarındaki farklılıklar, onların geleneksel bilgisayarlarda kullanılan işletim sistemlerini kullanmalarına olanak vermez. Bu açıdan gerek algılayıcıların kontrolü açısından, gerekse de işlemci gücü vb. gibi sınırlılıklar açısından mobil cihazlara özel olarak geliştirilen işletim sistemleri kullanılmaktadır.

### 2.6. Günümüzün Mobil İşletim Sistemleri

Mobil cihazların gelişim süreci içerisinde ana akım platformlara bağlı olarak birçok işletim sistemi kullanılmıştır. Bu işletim sistemleri içerisinde bazıları gelişimlerine devam ediyorken bazıları ise artık kullanılmamaktadır. Uzun yıllar mobil telefon pazarında lider konumda bulunan Nokia firmasının cihazlarında kullanılan Symbian işletim sistemi, artık kullanılmayan mobil işletim sistemleri arasında en bilinenidir.



Günümüzde en çok tercih edilen mobil işletim sistemleri aşağıda listelenmiştir:

- Android
- iOS
- Windows Phone

Bu işletim sistemleri aynı zamanda mobil cihazlar için ana kategorileri de oluşturmaktadır. Google tarafından birçok farklı donanım üreticisinin kullanımına sunulan Android, çok çeşitli cihazlarda kullanılması açısından yaygınlığı en fazla olan mobil işletim sistemidir. Bunu sadece Apple tarafından üretilen mobil cihazlarda kullanılabilen iOS işletim sistemi takip etmektedir. Pazar payı günümüzde en düşük olan işletim sistemi ise Microsoft tarafından farklı donanım üreticilerinin de kullanımına sunulan Windows Phone işletim sistemidir.

### 2.7. Mobil Cihazlarda Yer Alan Sensörler

Mobil donanımı belki de diğer bilgisayar sistemlerinde ayıran en önemli özellik sahip oldukları algılayıcılardır (sensörler). 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 algılayıcılar sağlamaktadır. Mobil donanımlar içerisinde en çok yer verilen ve artık neredeyse standart hâle dönüşen algılayıcılar aşağıda listelenmiştir:

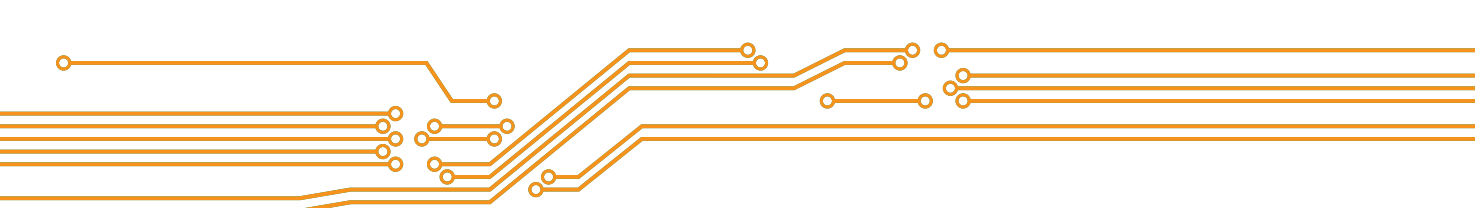
- a. Parmak izi sensörü
- b. Üç eksenli jiroskop
- c. İvmeölçer
- ç. Yakınlık sensörü
- d. Ortam ışığı sensörü
- e. GPS

### 2.8. 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 içerisinde çağırılması ile birlikte ilgili sensöre erişim sağlanır ve kontrol edilebilir bir yapı ortaya çıkar.

### 2.9. Emülatörlerin Çalışma Mantıkları

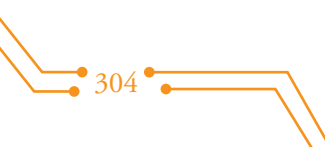
Emülatörler, fiziki donanıma sahip olan cihazların işlevlerini taklit edebilen yazılımlar olarak ifade edilebilirler. Geliştirilen uygulamaların test edilebilmesi için farklı donanım özelliklerine sahip birçok cihaz yerine bu şekilde yapılandırılabilen yazılımların kullanılması hem zaman hem de bütçeleme açısından önemli avantajlar sağlar. Buna ek olarak fiziki donanımlarda rahatlıkla oluşturamadığımız birçok anlık durumu emülatörlerde oluşturabiliriz. Örneğin bir mobil cihazın pili bitmek üzereyken yapacağı işlemleri test edebilmek için emülatörü buna göre yapılandırmak mümkün iken fiziki donanıma sahip bir cihazda pilin bitmesini beklememiz gerekir.



Emülatörlerin sağlamış olduđu birçok avantaja rağmen geliştirme ortamı olarak kullanılan bilgisayar sistemine ek yük getiriyor olması en önemli dezavantajları arasında yer alır. Buna ek olarak her zaman için emülatörde sorunsuz çalışan bir uygulamanın fiziki donanım üzerinde sorun yaşaması mümkündür. Bu açıdan özellikle ticari projelerde geliştirme aşamalarında sıkça kullanılan emülatörler uygulama hayata geçirilmeye yakın yerlerini fiziki donanımlara bırakırlar.

## 2.10. Geliştirilen Mobil Uygulamaların Emülatörlerde Çalıştırılması

Uygulama geliştirme araçlarının içinde entegre olarak yer alan emülatörlerin kullanımını oldukça basittir. Uygulama geliştirildikten sonra derlenip emülatöre otomatik olarak gönderilir ve emülatör, geliştirme için kullanılan bilgisayar içerisinde sanki gerçek bir mobil cihaz gibi kullanılabilir.



### 3. UYGULAMA GELİŐTİRME

### 3. 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

#### 3.1. Uygulamaların Yaşam Döngüleri

Uygulama geliştirme yaşam döngüleri, uygulama geliştirme mimarisinde tercih edilen yönetime göre şekillenir ve bu tercih mobil uygulama geliştirme araçlarının seçimini de etkiler.

#### 3.2. 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.

#### 3.3. Yerel (Native) Uygulamalar İçin Yaşam Döngüsü

Bu uygulamaların yaşam döngüsü, tamamen hedef platforma yönelik olarak uygulamanın geliştirilmesi süreçlerini içerir. Seçilen hedef platforma (ör.: Android, 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. Bunun nedeni, Android platformunun geliştirme ortamı olarak kişisel bilgisayarlardaki işletim sistemleri konusunda esnek oluşudur. Mobil cihazlar için geliştirilecek bir Android uygulaması, Linux, Windows ve MacOS işletim sistemleri kullanılarak geliştirilebilir. Bu durum maalesef iOS ve Windows Phone için geçerli değildir. Bu platformlarda yerel geliştirme yapılabilmesi ancak iOS için MacOS, Windows Phone için ise Windows işletim sisteminin kullanılması ile mümkündür.

### 3.4. 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 faydalanılabilesidir. 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 diğel platform için de yeniden kodlama yapılmaksızın kullanılabilmesidir.

## 4. MOBİL UYGULAMA GELİŞTİRME

## 4. 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.

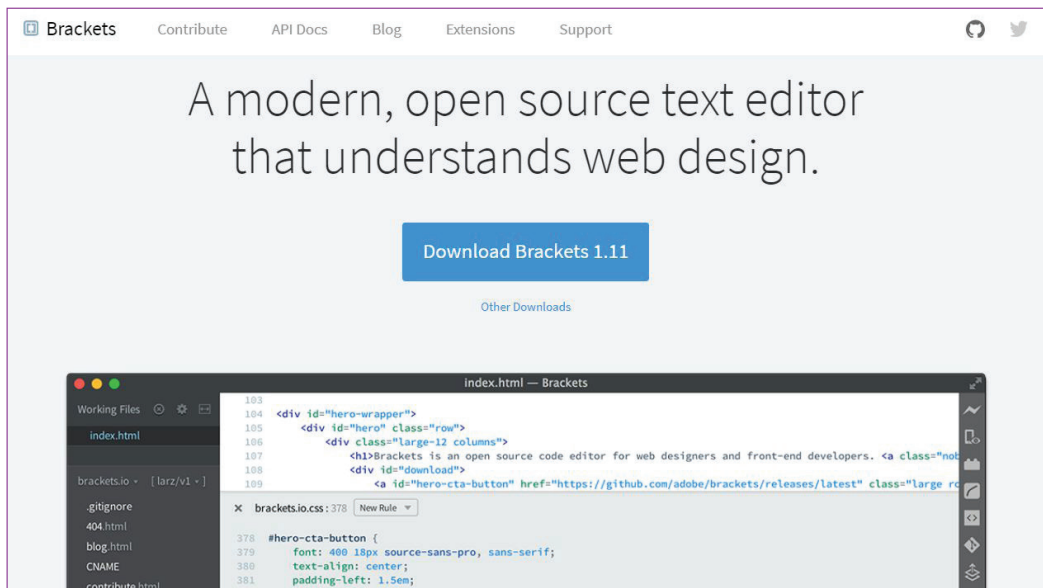
### 4.1. Web Tabanlı Mobil Uygulama Geliştirme

Mobil cihazlarda yaşanan teknolojik gelişmeler ve özellikle işlemci hızının artması, masaüstü bilgisayarlardaki İnternet kullanım deneyimine benzer bir deneyimin mobil cihazlarda da yaşanmasına olanak sağlamaktadır. Bu aşamada mobil cihazların özellikle ekran boyutları dikkate alınarak geliştirilen Web tabanlı uygulamalar bir mobil uygulama olarak çalışabilmekte ve mobil işletim sisteminden bağımsız olarak kullanılabilir.

#### 4.1.1. Web Tabanlı Mobil Uygulama Geliştirme Ortamı

Web tabanlı mobil uygulama geliştirme ortamı olarak herhangi bir metin editörü kullanılabilir. Metin editörü seçiminde dikkat edilmesi gereken en önemli nokta editörün Web tabanlı uygulama geliştirme açısından verdiği desteklerdir (kod renklendirme, kod tamamlama vb. gibi).

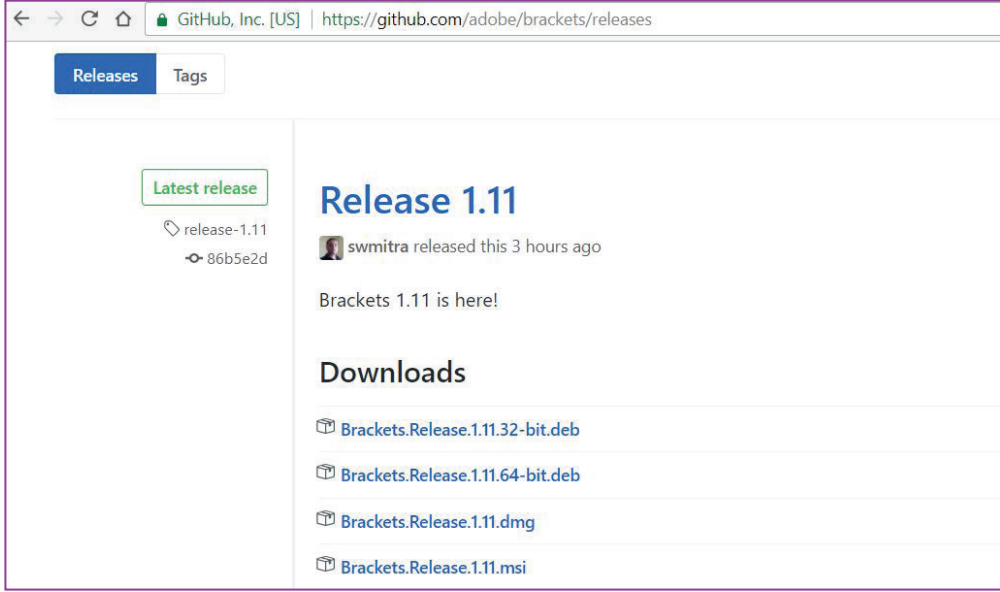
Bu ders kapsamında açık kaynak kodlu olan ve ücretsiz olarak sunulan Adobe firmasının Brackets isimli metin editörü kullanılacaktır. Bu editör <http://brackets.io/> adresinden tüm popüler işletim sistemleri için indirilebilir ve kişisel bilgisayarlara kurulabilir (Şekil 1). Kurulum gerçekleşikten sonra arayüz Türkçe olarak kullanılabilir.



Şekil 1: Adobe Brackets metin editörü İnternet sitesi (<http://brackets.io/>)

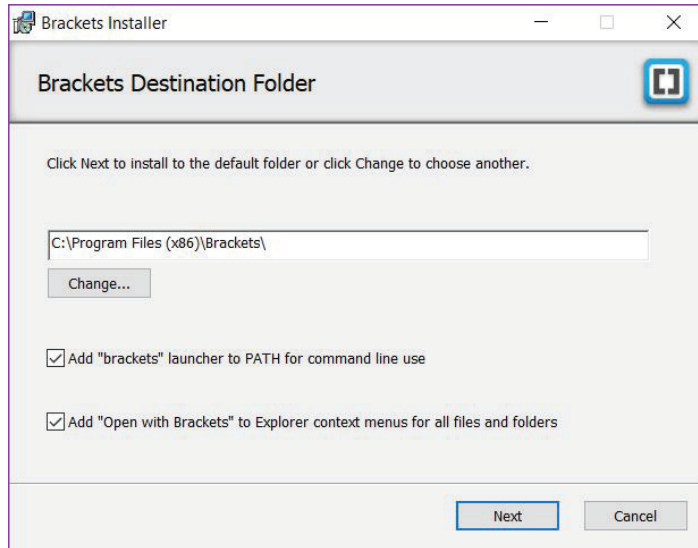
Brackets Web sitesindeki ana sayfada yer alan “Download Brackets” (Brackets’ı İndir) isimli butona

tıklandığında kurulum dosyasının indirilme işlemi başlatılmış olur. O an hangi işletim sistemi kullanılarak indirme işlemini gerçekleştiriliyorsa o işletim sistemine uygun kurulum dosyası indirilmeye başlanacaktır (ör.: Windows işletim sistemi için “Brackets.Release.1.11.msi”). Desteklenen diğer işletim sistemleri için sunulan kurulum dosyalarını ise “Other Downloads” (Diğer İndirmeler) bağlantısına tıklanarak görüntülenebilir (Şekil 2).



Şekil 2: Diğer işletim sistemleri için kurulum dosyaları

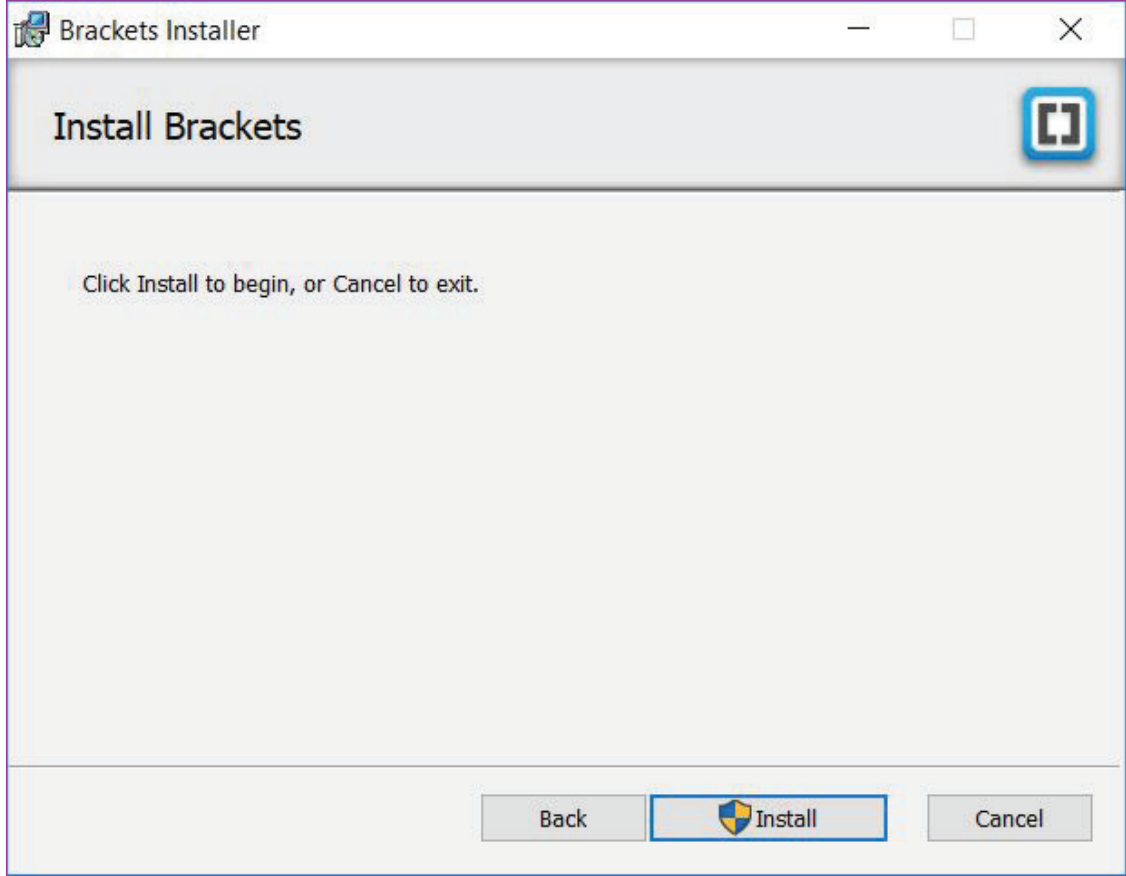
İnternet tarayıcısı aracılığıyla indirilmiş olan kurulum dosyasına “İndirilenler” adlı klasörden erişebilirsiniz. Bu klasör işletim sisteminize ve İnternet tarayıcınıza göre farklı isimlerde ve sabit diskiniz üzerinde farklı konumlarda olabilir. Kurulum dosyasına çift tıklandığında kurulum otomatik olarak başlatılacaktır (Şekil 3).



Şekil 3: Kurulum

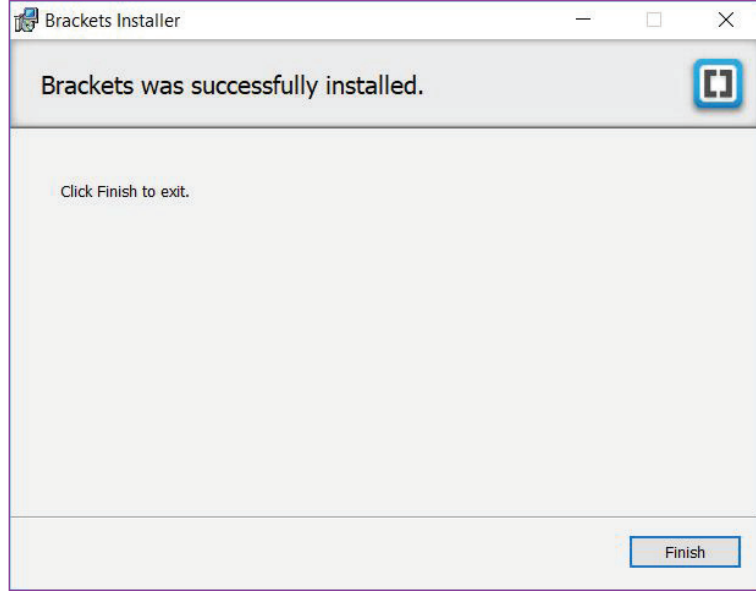


Kurulum yazılımındaki ayarlar standart olarak bırakılabilir. Sabit disk üzerinde farklı bir alana kurulumun gerçekleşmesi isteniyorsa bu bölümden yeni bir alan tanımlaması yapılabilir. “Next” (İlerle) butonuna tıklayarak bir sonraki aşamaya geçilir. Bir sonraki aşamada “Install” (Kur) butonuna tıklayarak kurulum gerçekleştirilir (Şekil 4).



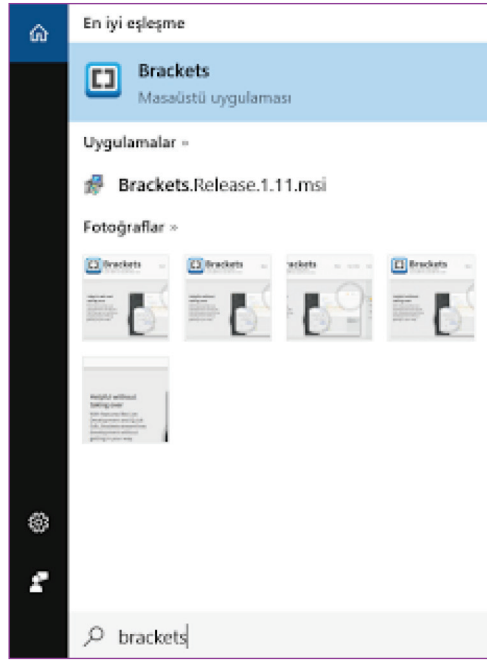
Şekil 4: “Install” (Kur)

Kurulum gerçekleştikten sonra Şekil 5’te yer alan programın sorunsuz kurulduğu ile ilgili bilgi ekranı görüntüleniyorsa kurulum tamamlanmış demektir. “Finish” (Tamamlandı) butonuna tıklayarak kurulum programından çıkılabilir. Brackets programı artık kullanılabilir durumdadır.



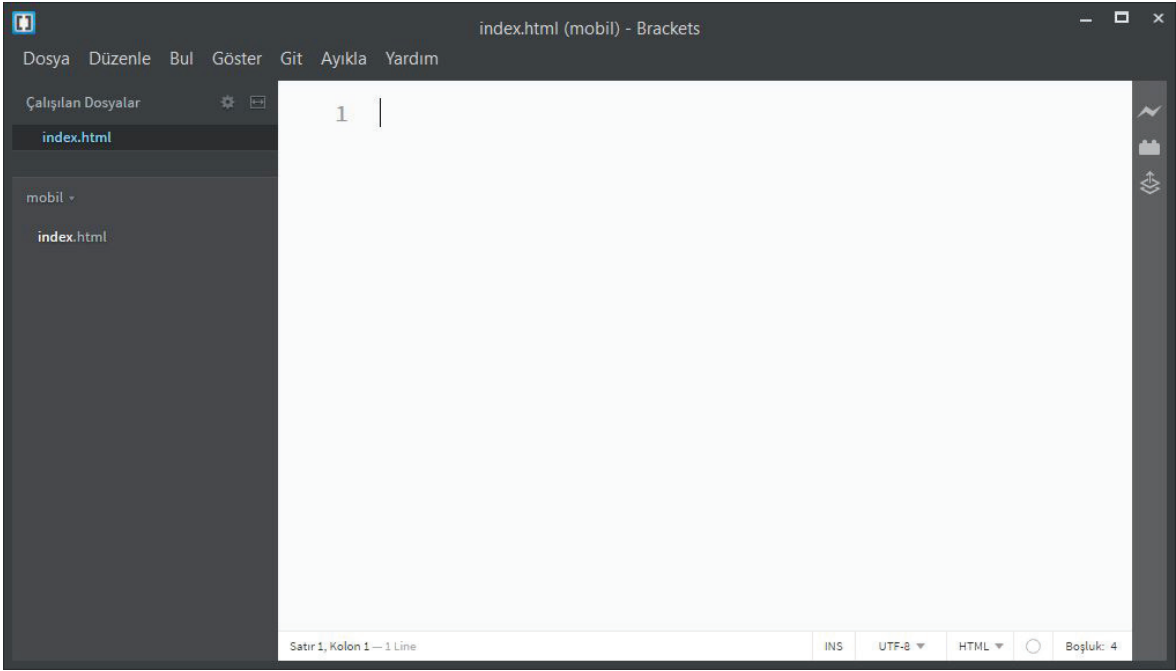
**Şekil 5:** Kurulumun tamamlandığını belirten bilgi ekranı

Kurulum işlemi tamamlanan Brackets yazılımına bilgisayardaki diğer programların da yer aldığı liste içerisinde erişilebilir. Yazılımı başlatmak için bilgisayar içerisinde “Brackets” kelimesi aratılarak da yazılıma erişilebilir (Şekil 6).



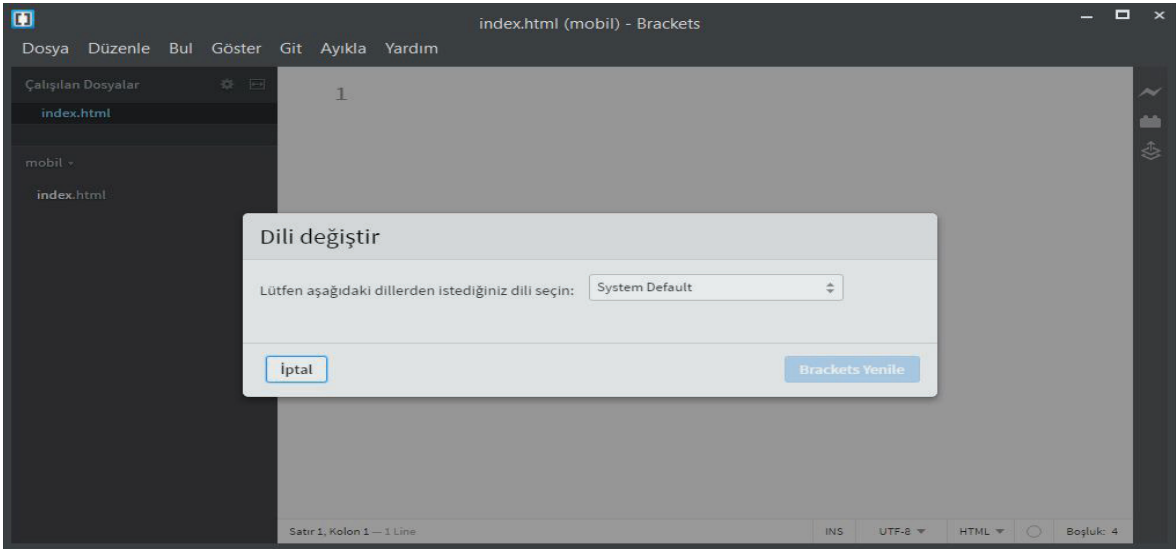
**Şekil 6:** Brackets yazılımının aratılması ve arama sonucu

Liste içerisinde ya da arama sonuçlarından Brackets ikonuna tıklandığında yazılım çalışacak ve Şekil 7’deki arayüz görüntülenecektir.



**Şekil 7:** Brackets yazılımının arayüzü

Brackets, kurulu olduğu işletim sisteminin dil ayarlarını algılayarak otomatik olarak arayüzünü o dile göre yapılandırır. Arayüzün dil seçimi isteğe göre menüler yardımıyla da yapılabilir. Arayüzün dilini değiştirmek için “Ayıkla” menü grubundaki “Dili Değiştir” bölümü kullanılır (Şekil 8).



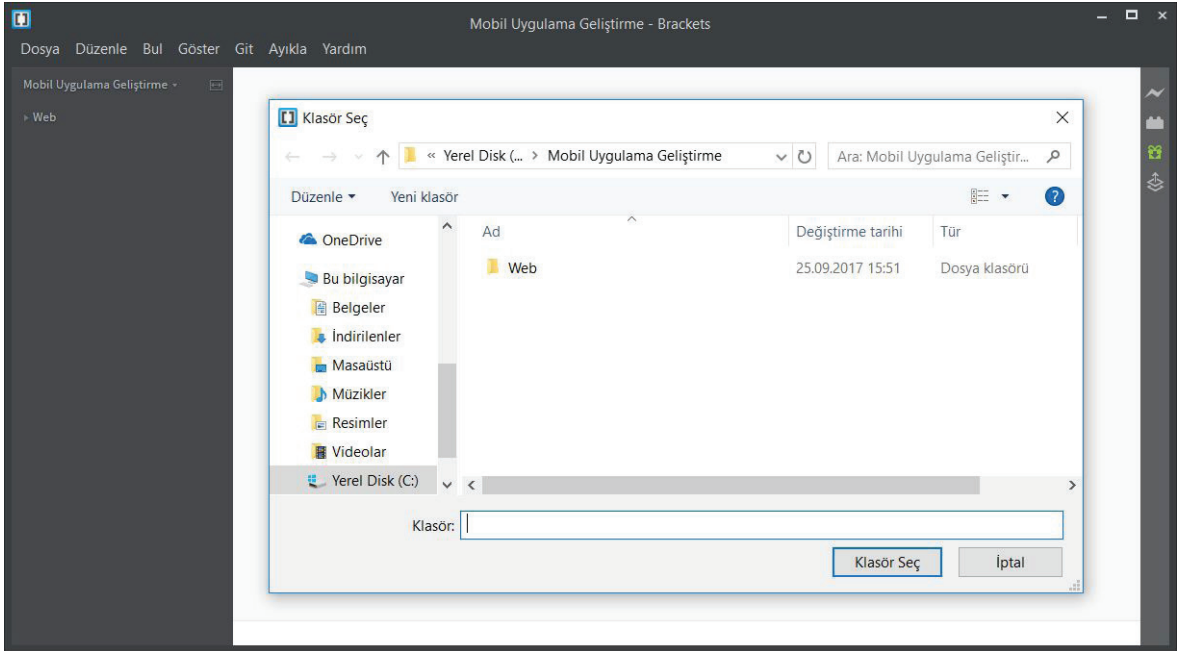
**Şekil 8:** Arayüzde dil değişikliği yapılması

Brackets, temel olarak iki ana bölümden oluşmaktadır. Bu bölümler solda yer alan klasör ve dosya görüntüleme alanı ve sağda yer alan kod yazma ve görüntüleme alanıdır. Mobil uygulama geliştirmeye

başlamadan önce bilgisayardaki diğer dosya ve klasörlerle uygulama dosyalarının karışmasını engellemek için bu çalışmaya özel bir klasör oluşturmak ve o klasörün içinde çalışmak uygun olacaktır.

Bu ders kapsamında geliştirilecek uygulamaların depolanması için sabit diskte “Mobil Uygulama Geliştirme” adlı ana bir klasör ve bu klasörün içerisinde de uygulama geliştirme mimarilerine bağlı olarak alt klasörler oluşturulmuştur. Web tabanlı uygulamaların yer alacağı klasörün adı “Web”dir.

Brackets açıldıktan sonra yapılması gereken ilk şey çalışma alanı olarak oluşturulmuş klasörün seçilmesidir. Bu seçim işlemi “Dosya” menü grubu altındaki “Klasörü Aç” (Ctrl+Alt+O) ögesine tıklayarak gerçekleştirilir (Şekil 9). Klasör seçimi bir kere gerçekleştirildikten sonra Brackets her açılışında aynı klasörün içeriğini gösterir. Brackets’ı her açışta yeniden bu işlemin gerçekleştirilmesine gerek kalmaz.



**Şekil 9:** Çalışmaların yapılacağı klasörün brackets içerisinde tanımlanması

Klasör seçimi ile birlikte Web tabanlı mobil uygulama geliştirme ortamının hazırlık aşaması da tamamlanmış olur. Bu aşamadan sonra artık uygulama geliştirmeye başlanabilir.

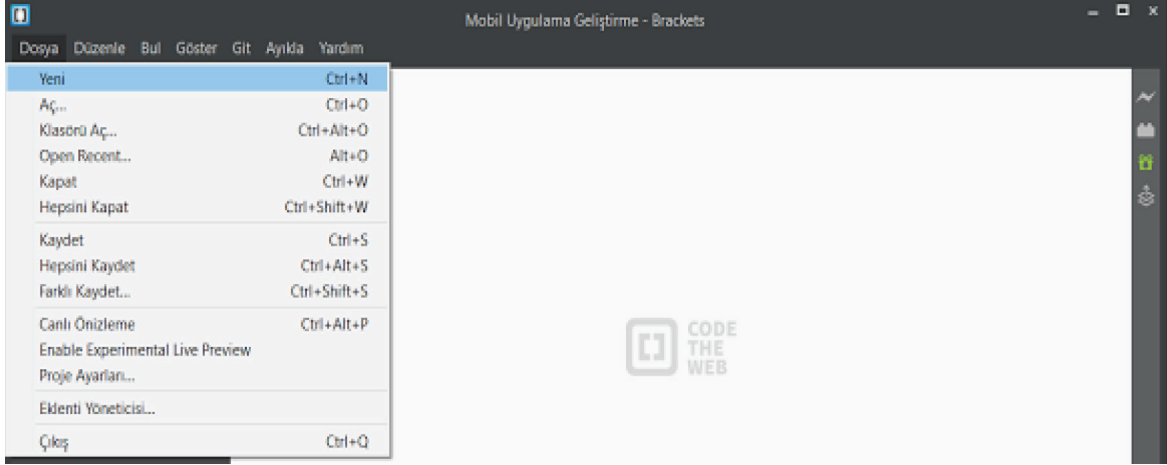
#### 4.1.2. Web Tabanlı Örnek Bir Mobil Uygulama Geliştirme

Web tabanlı mobil uygulama geliştirmeye en iyi başlangıç bu iş için özel olarak geliştirilmiş yazılım kütüphanelerinin kullanılması ile olabilir. HTML5, CSS3 ve JavaScript’in birlikte kullanımı ile oluşturulmuş birçok farklı yazılım kütüphanesi bulunmasına rağmen en fazla kullanılan, dokümantasyonu ve örnekleri en iyi hazırlanmış kütüphanelerden biri JQuery Mobile adlı yazılım kütüphanesidir. JQuery Mobile ile ilgili detaylı bilgi, dokümantasyon ve örnekler <https://jquerymobile.com/> adresinden erişilebilir.

Örnek mobil uygulamamızda JQuery Mobile kütüphanelerini kullanarak basit bir sözlük uygulaması gerçekleştireceğiz. Bu sözlük uygulaması kelime listesi ve bu kelime listesi içerisinde bir arama bölümü içerecek şekilde tasarlanmış olacak. Şimdi adım adım bu uygulamayı geliştirelim.

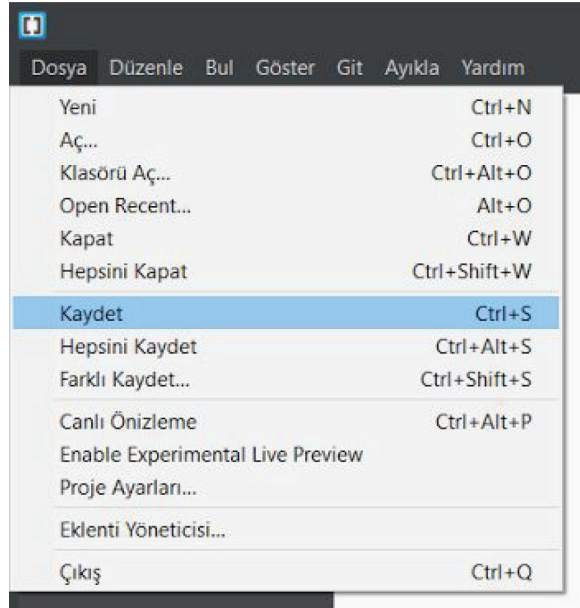
Öncelikle kelime listesinin ve arama formunun yer alacağı ana sayfayı oluşturalım. Web tabanlı uy-

gulamalarda aynen Web tasarımında olduğu gibi ana sayfanın adının “index.html” olarak belirlenmesi gerekmektedir. Bu şekilde bir klasör içerisinde duran uygulamayı yorumlayacak olan mobil İnternet tarayıcı hangi dosyayı başlangıç dosyası olarak kullanacağını belirleyebilecektir. Brackets içerisinde “index.html” isimli dosyayı “Dosya” menü grubu altındaki “Yeni” (Ctrl+N) ögesine tıklayarak oluşturabilirsiniz (Şekil 10).



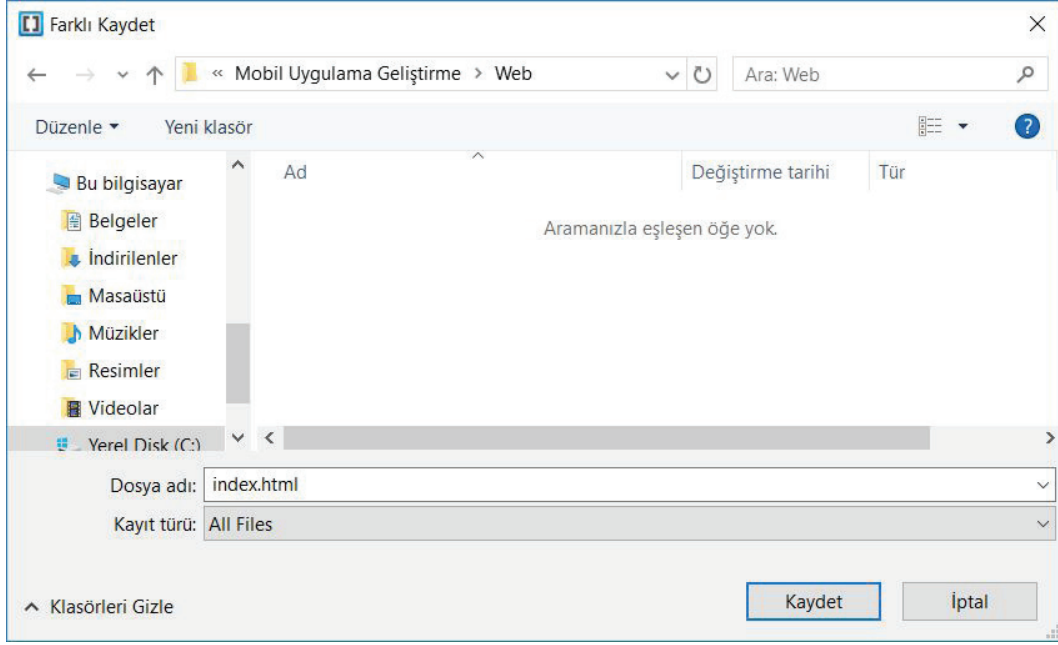
Şekil 10: Yeni bir dosya oluşturulması

Yeni bir dosya oluşturduğunuzda bu dosya “Adsız” adıyla oluşturulur. Oluşturulan bu dosyayı “Dosya” menü grubu altındaki “Kaydet” (Ctrl+S) ögesine tıklayarak “index.html” adıyla içinde çalıştığınızı klasörün altına kaydediniz (Şekil 11).



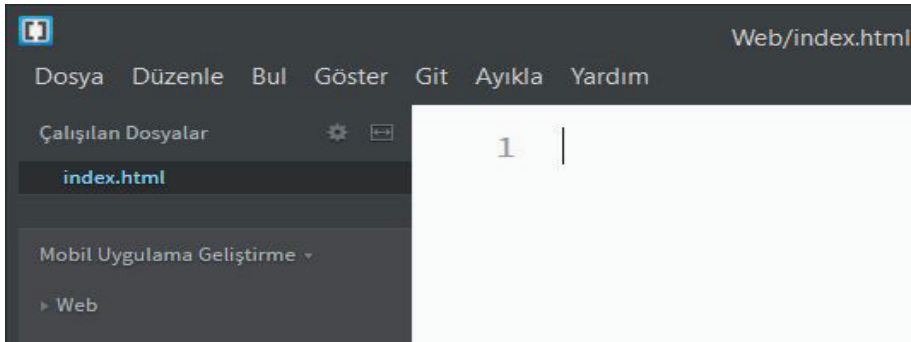
Şekil 11: Oluşturulan dosyanın kaydedilmesi

Bu örnek uygulamada çalışmalar “Mobil Uygulama Geliştirme” ana klasörü içerisindeki “Web” adlı alt klasöre kaydedilmektedir (Şekil 12).



**Şekil 12:** Dosya kaydedilirken klasör seçimi ve dosya isimlendirilmesi

Dosyanın oluşturulması ve isimlendirme yapılarak kayıt işleminin tamamlanmasının ardından kod yazımı ile ilgili bölüme geçiş yapılabilir. Öncelikle Brackets’ın sol bölümünde yer alan dosya bölümünde index.html adlı dosyanın seçili olduğunu kontrol edin. Brackets’ın üst bölümünde şu an düzenleme için açık olan dosyanın ismi yazmalıdır. Aynı zamanda yine sol üst bölümde “Çalışılan Dosyalar” adı altında index.html adlı dosyanın listelenmesi gerekmektedir (Şekil 13).



**Şekil 13:** “index.html” adlı dosyanın düzenlemeye hazır hale getirilmesi

Şimdi “index.html” içerisinde ihtiyacımız olan kodları yazmaya başlayalım. Aşağıdaki kod bloğu standart olarak HTML kodlamada kullanılan kod bloğudur (Şekil 14).

```

1  <!DOCTYPE html> |
2  <html>
3  <head>
4      <title>Page Title</title>
5  </head>
6
7  <body>
8
9  </body>
10 </html>

```

Şekil 14: Standart HTML kod bloğu

Bu kod bloğunda 1. satırda dokümanın tipi tanımlanmaktadır. 2. satırdan 10. satıra kadar yer alan bölüm HTML olarak tanımlanmıştır. HTML içerisindeki tanımlamalar etiket adı verilen yapıların açılıp kapatılması ile gerçekleştirilir. Örnek uygulamamızın 2. satırında <html> ifadesi ile HTML etiketi açılmakta ve 10. satırda </html> ifadesi ile HTML etiketi kapatılmaktadır. Aynı etiketin açılıp kapatılması arasında kalan tüm kodlar o etiketin genel özellikleri ile tanımlanmış olur. Yine örnek uygulamamızdan yola çıkarak HTML dokümanının üst bölümü olarak tanımlanmak istenen alanda HEAD etiketi açılıp kapatılmıştır. <head> ile </head> arasında kalan tüm kodlar HEAD (başlık) etiketinin özelliğine sahip olacak şekilde tanımlanmıştır.

Bir HTML dokümanında iki ana bölüm bulunmaktadır. Bu bölümler başlık (head) bölümü ve Gövde (body) bölümüdür. Başlık bölümünde İnternet tarayıcısına özel bir takım tanımlamalar yer alır. Bu tanımlamalar, sayfanın başlığı, metin kodlaması, kullanılacak biçimlendirme dosyalarının (CSS) ve JavaScript kütüphanelerinin adresleri ve birtakım üst veri alanlarıdır.

Örnek uygulamamızda başlık (<head>) etiketi içerisinde aşağıdaki tanımlamaların yapılması gerekmektedir. Bu tanımlamalar JQuery Mobile biçimlendirme dosyalarının ve kod kütüphanelerinin adreslerini içermektedir (Şekil 15).

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Örnek Mobil Uygulama</title>
5      <meta charset="utf-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1">
7      <link rel="stylesheet" href="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css" />
8      <script src="http://code.jquery.com/jquery-1.11.1.min.js"></script>
9      <script src="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js">
10 </script>
11
12 </head>
13
14 <body>
15 </body>
16 </html>

```

Şekil 15: Başlık bölümü tanımlamaları

Başlık bölümü içerisindeki kod satırlarını detaylandırmamız gerekirse

- 4. satır uygulamanın başlığını tanımlamaktadır ve mobil İnternet tarayıcının üst bölümünde uygulamanın adının yer almasını sağlayacaktır.
- 5. satır uluslararası dil kodlaması olarak belirlenmiş olan UTF-8'in sayfanın dil kodu olarak belirlenmesini sağlar. Bu satır özellikle Latin alfabesi dışındaki harflerin kullanımı açısından önemlidir.
- 6. satır mobil İnternet tarayıcısına hazırlanan sayfanın boyutları ile ilgili bir düzenleme yapmasını söyler. Bu şekilde boyutlandırma açısından ideal bir oran yakalanmaya çalışılır.
- 7. satır JQuery Mobile biçimlendirme dosyasının adresini içerir. Bu sayede belirli bir temaya göre yazı biçimleri (fontlar), boyutları, renkler, form ve diğer HTML bileşenlerinin biçimsel özellikleri belirlenmiş olur.
- 8. satır JQuery'nin ana kütüphanesinin adresini içermektedir. Burada bu çalışma çerçevesi (framework) için kullanılan JavaScript kütüphanesi bulunmaktadır.
- 9. satır JQuery'nin mobil cihazlar için geliştirdiği JavaScript kütüphanesinin adresini içerir.

Başlık bölümü tamamlandığına ve gerekli tanımlar yapıldığına göre artık uygulamanın kullanıcılar tarafından görüntülenecek kısmı olan “gövde” (body) üzerinde çalışmalara başlanabilir. Uygulamamızın gövde kısmında bir arama formu ve bir kelime listesi için gerekli kodlar yer alacaktır. Bu kodlar şekil 15-a'da gösterilmektedir.

```
12 <body>
13 <form>
14 <input id="filter-for-listview" data-type="search" placeholder="Arama
    yapmak için birkaç harf giriniz...">
15 </form>
16 <ul data-role="listview" data-inset="true" data-filter="true" data-
    input="#filter-for-listview">
17 <li><a href="#">Armut</a></li>
18 <li><a href="#">Elma</a></li>
19 <li><a href="#">Muz</a></li>
20 <li><a href="#">Şeftali</a></li>
21 <li><a href="#">Üzüm</a></li>
22 </ul>
23 </body>
```

Şekil 15-a: Uygulamanın gövde bölümünde yer alan kodlar

Gövde bölümü içerisindeki kod satırlarını detaylandırmamız gerekirse

- 13 ve 15. satırlar arasında yer alan kodlar bir HTML formunu tanımlamaktadır. Bu form JQuery Mobile JavaScript kütüphanesi içerisindeki “filter-for-listview” adlı fonksiyonu da içinde barındıracak şekilde yapılandırılmıştır. Bu fonksiyon listeyi süzmemize imkan sağlayacak ve bu sayede basit bir arama işlemi gerçekleştirilebilecektir. Arama kutucuğu içerisinde yer alan varsayılan mesajı da yine bu bölümdeki “placeholder” değişkenine atanan değer ile değiştirmek mümkündür.
- 16 ve 22. satırlar arasında yer alan kodlar, kelimelerin listelenmesini sağlamaktadır. Listelenen kelimelere bir Web sayfası ya da Web sitesi adresi atamak da mümkündür. Böylece tıklanan kelime ile ilgili bir detay sayfasına ya da bir başka siteye yönlendirme yapılabilir.



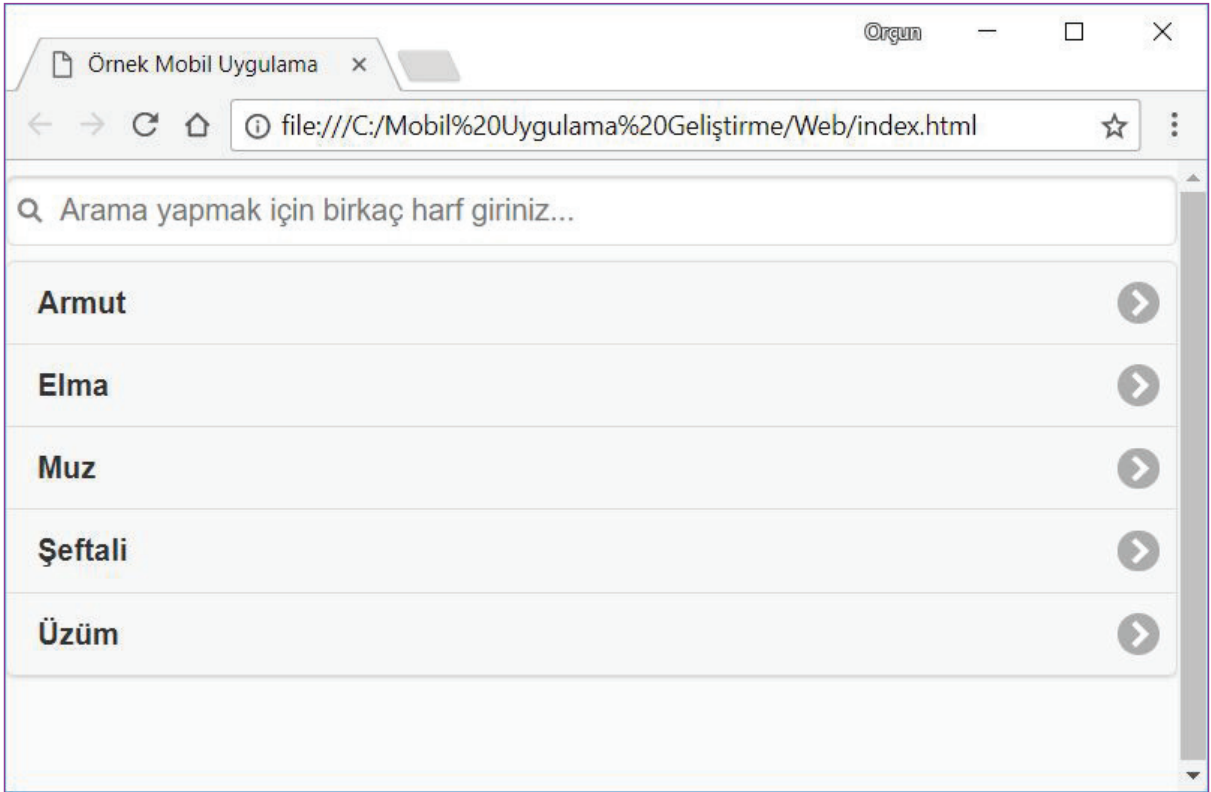
Uygulamamızın temel özelliklerini tamamladığımızı göre ilk test işlemini gerçekleştirebiliriz. Bu aşamada uygulama içerisinde kullanılan CSS ve JavaScript kütüphanelerini İnternet üzerinden kullandığımız için test işlemini gerçekleştireceğimiz cihazımızda İnternet bağlantısının olması gerekmektedir. Uygulamanın İnternet bağlantısına ihtiyaç duymasını istemiyorsak CSS ve JavaScript kütüphanelerini JQuery Mobile’ın İnternet sitesinden (<http://jquerymobile.com/download/>) bilgisayarımıza indirip proje klasörü içerisine kaydetmeliyiz. Daha sonra kaydettiğimiz bu dosyalara HTML sayfası içerisinde bağlantı sağlamalıyız.

Test işlemi için öncelikle kendi kişisel bilgisayarımızda yer alan herhangi bir İnternet tarayıcısını kullanabiliriz. Daha sonra uygulamamızı istersek dosyaları İnternet üzerine yüklediğimiz bir lokasyondan istersek de uygulama dosyalarını direkt mobil cihazın içerisine yükleyerek test edebiliriz.

#### 4.1.3. Mobil Uygulamanın Kişisel Bilgisayar Üzerinde Yer Alan İnternet Tarayıcı ile Test Edilmesi

Test işlemini kişisel bilgisayar üzerinde gerçekleştirmek için takip edebileceğimiz iki yol bulunmaktadır. Bunlardan birincisi eğer bilgisayarımızda Google Chrome tarayıcı yüklü ise Brackets içerisindeki canlı önizleme (Live Preview) özelliğini kullanmak, diğeri ise direkt olarak bilgisayarımızın “Dosya Gezgini” yardımı ile uygulamamızın başlangıç dosyası olan “index.html” dosyasına çift tıklamak. Bu iki yol da İnternet tarayıcıda uygulamamızın açılmasını ve test edebilmemizi sağlayacaktır.

“Dosya Gezgini” ile proje klasörüne girip “index.html” dosyasına çift tıkladığımızda açılan uygulamamızın görüntüsü Şekil 16’daki gibi olmalıdır.

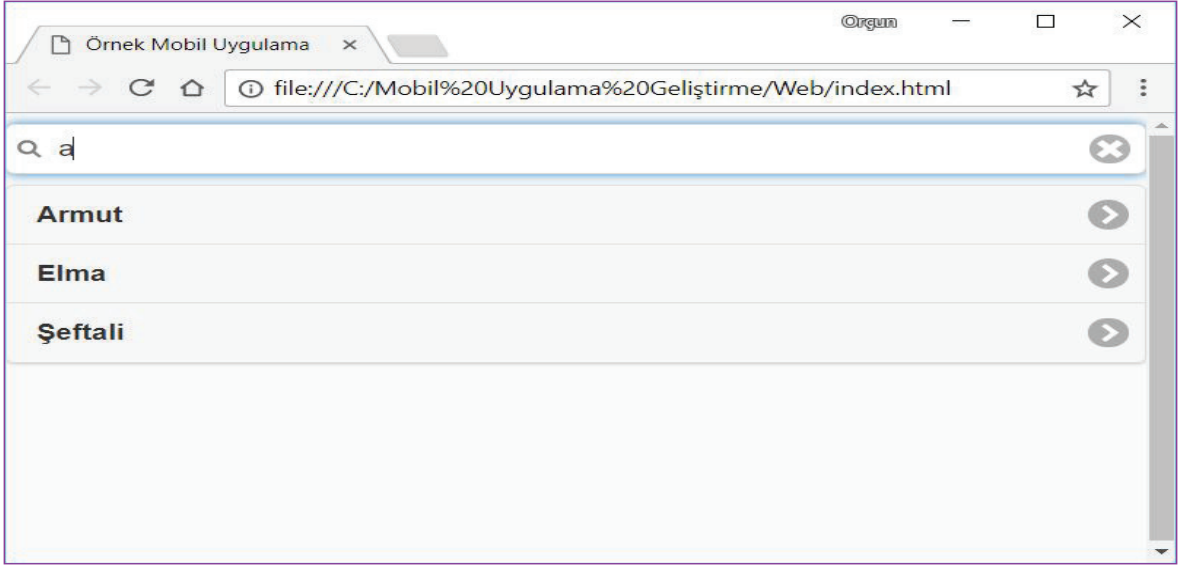


Şekil 16: Mobil uygulamanın İnternet tarayıcısındaki görünümü

Bilgisayarınızda varsayılan İnternet tarayıcınızın hangisi olduğuna bağlı olarak uygulamanız ilgili tarayıcıda görüntülenir.

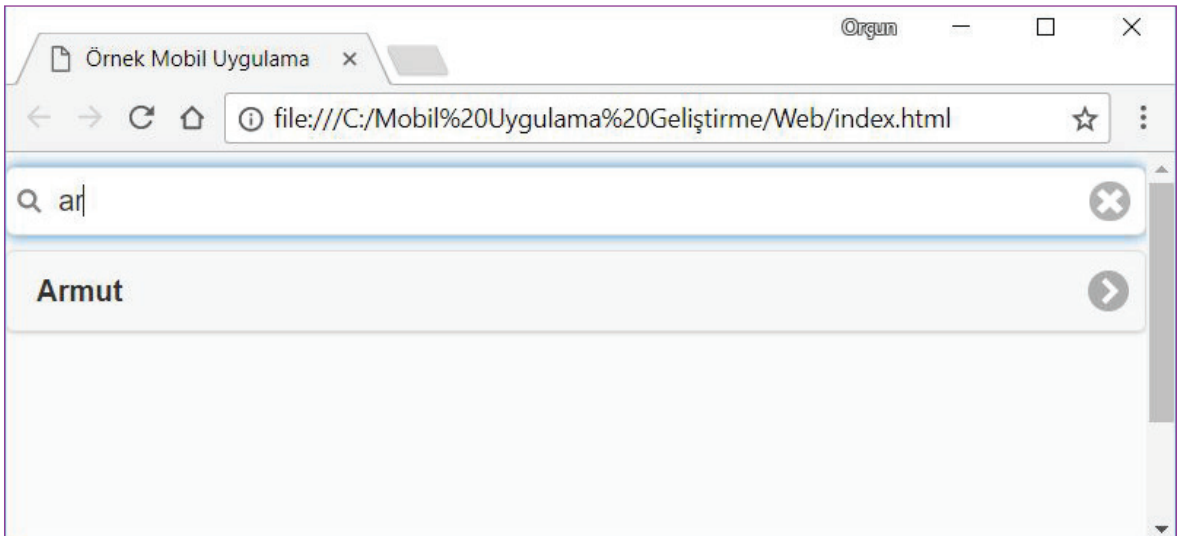
Farklı İnternet tarayıcılarda uygulamanızı test etmek için “Dosya Gezgini”nde “index.html” dosyasının üzerine sağ klik yaparak “Birlikte aç” bölümünden farklı bir İnternet tarayıcı seçebilirsiniz.

Şimdi uygulamamızın arama bölümünü test edelim. Örneğin arama kutucuğuna “a” harfi yazdığımızda içinde “a” harfi bulunan kelimeler haricindeki kelimeler listeden çıkartılacaktır (Şekil 17).



Şekil 17: Arama kutucuğuna harf girişi ile süzme işleminin gerçekleştirilmesi

Uygulama ekranında görüldüğü gibi “a” harfi girildikten sonra kelime listesinde yer alan “Üzüm” ve “Muz” içlerinde “a” harfini barındırmadığı için listeden çıkartıldı. Şimdi “a” harfine bir de “r” harfi ekleyerek devam edelim. Bu durumda listede sadece “Armut”un kaldığını göreceksiniz (Şekil 18).



Şekil 18: İkinci harfin eklenmesi ile listenin yeniden oluşması

Örnek uygulamamızı 5 kelimedenden oluşacak şekilde hazırlamıştık. Uygulamaya yeni kelimeler eklemek için Şekil 19’da işaretli satırı (ör.: 18. satır) çoğaltarak yeni kelimelere göre düzenlememiz yeterli olacaktır (Şekil 20).

```
<body>
  <form>
    <input id="filter-for-listview" data-type="search" placeholder="Arama
      yapmak için birkaç harf giriniz...">
  </form>
  <ul data-role="listview" data-inset="true" data-filter="true" data-
    input="#filter-for-listview">
    <li><a href="#">Armut</a></li>
    <li><a href="#">Elma</a></li>
    <li><a href="#">Muz</a></li>
    <li><a href="#">Şeftali</a></li>
    <li><a href="#">Üzüm</a></li>
  </ul>
</body>
```

Şekil 19: Uygulamaya yeni kelimelerin eklenmesi

```
<body>
  <form>
    <input id="filter-for-listview" data-type="search" placeholder="Arama
      yapmak için birkaç harf giriniz...">
  </form>
  <ul data-role="listview" data-inset="true" data-filter="true" data-
    input="#filter-for-listview">
    <li><a href="#">Armut</a></li>
    <li><a href="#">Elma</a></li>
    <li><a href="#">Erik</a></li>
    <li><a href="#">Muz</a></li>
    <li><a href="#">Şeftali</a></li>
    <li><a href="#">Üzüm</a></li>
  </ul>
</body>
```

Şekil 20: Yeni kelime olarak “Erik”in listeye eklenmesi

Artık uygulamamızda 6 sözcük yer almakta. İstenilen sayıda sözcüğü bu şekilde eklenip çıkartılması bu sayede mümkün olmaktadır.

Uygulamamızdaki bir diğer özellik ise seçilen kelime ile ilgili detaylı bilgi verilebilecek bir sayfaya ya da bir harici Web sitesine bağlantı sağlanabilmesidir. Bu bağlantıyı sağlayabilmek için kod bölümünde ilgili kelimenin bulunduğu satırda bir düzenleme yapmamız gerekmektedir. Bu düzenleme HTML içerisinde bağlantı vermek için kullanılan “<a>” etiketinin içindeki “href” tanımlamasında yapılmalıdır.

Örnek uygulamamızda ilk önce “Armut” kelimesi üzerinde bir düzenleme yapalım. Kullanıcı “Armut” kelimesini tıkladığı zaman Türk Dil Kurumunun (TDK) Güncel Türkçe Sözlüğü’ndeki armut kelimesinin arama sonucuna yönlendirilmiş olsun. Bu arama sonucunu elde edebilmek için <http://www.tdk.gov.tr/> adresi ziyaret edilmeli ve buradaki sözlükte “Armut“ kelimesi aratılmalıdır.

Arama sonucunda gelen sayfanın adresi İnternet tarayıcısından kopyalanarak mobil uygulamamızın kod bölümünde ilgili yere yapıştırılır (Şekil 21).

```
<body>
  <form>
    <input id="filter-for-listview" data-type="search" placeholder="Arama yapmak için birkaç harf giriniz...">
  </form>
  <ul data-role="listview" data-inset="true" data-filter="true" data-input="#filter-for-listview">
    <li><a href="http://www.tdk.gov.tr/index.php?option=com_gts&kelime=ARMUT">Armut</a></li>
    <li><a href="#">Elma</a></li>
    <li><a href="#">Erik</a></li>
    <li><a href="#">Muz</a></li>
    <li><a href="#">Şeftali</a></li>
    <li><a href="#">Üzüm</a></li>
  </ul>
</body>
```

**Şekil 21:** Yönlendirme yapılacak harici Web sitesinin adresinin kod içine yerleştirilmesi

Yönlendirme yapılacak adres kod bölümüne eklendikten sonra uygulama test edildiğinde armut üzerine tıklandığında TDK'nin ilgili sayfasına ulaşıldığı görülür (Şekil 22).



**Şekil 22:** Yönlendirme yapılan sayfanın İnternet tarayıcısında görüntülenmesi

#### 4.1.4. Mobil Uygulamanın Mobil Cihaz Üzerinde Yer Alan İnternet Tarayıcı ile Test Edilmesi

Önceki bölümde bilgisayarımızın içindeki İnternet tarayıcı ile yapmış olduğumuz testi şimdi mobil cihazımızın içinde yer alan İnternet tarayıcı ile yapalım. Bunu gerçekleştirirken iki farklı yol izleyebiliriz. Bu yollardan ilki mobil uygulamamızı oluşturan dosyaları İnternet üzerindeki bir lokasyona aktarmak ve mobil İnternet tarayıcı ile bu lokasyonu ziyaret ederek mobil uygulamayı görüntülemek. İkinci yol ise uygulamayı oluşturan dosyaları mobil cihazımızın depolama alanına kopyalamak ve buradan mobil İnternet tarayıcı yardımı ile uygulamayı görüntülemek.

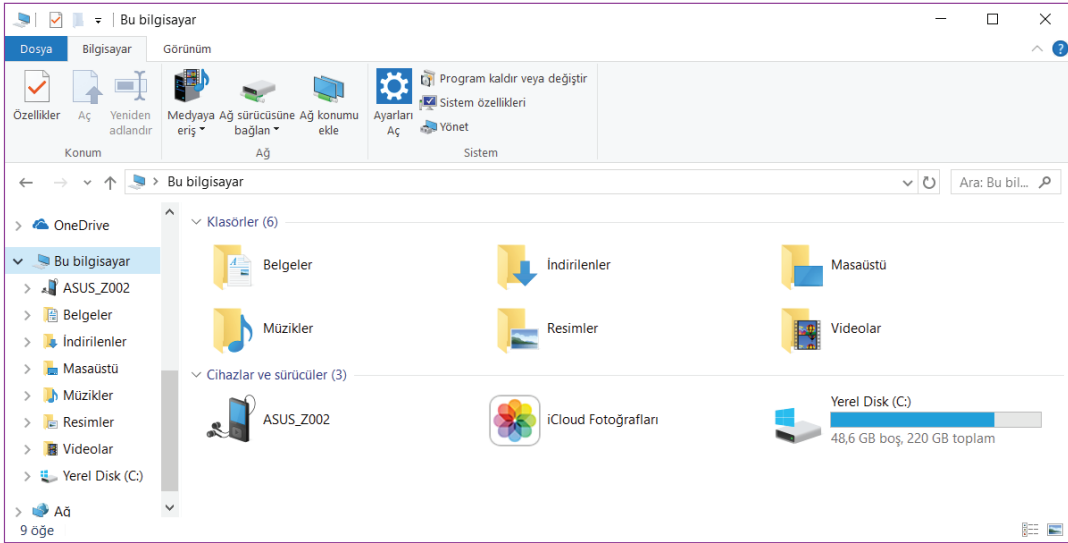
Yukarıda belirtilen ilk yol için bir İnternet sitesine, bu İnternet sitesine erişim haklarına ve dosyaları aktarabileceğimiz bir yazılıma (FTP yazılımı) ihtiyacımız var. Bunların temin edilmesinde sorunlar

olabileceğinden yola çıkarak uygulamamızı mobil cihaz üzerinde ikinci yolu kullanarak test edeceğiz.

Öncelikle mobil cihazımızın şarj kablosunu cihazımızın şarj çıkışına ve kişisel bilgisayarımızın USB girişine bağlayarak fiziksel bağlantısını kuruyoruz. Bu aşamada mobil cihazın açık konumda olması gerekiyor, aksi halde bilgisayarın mobil cihazı algılaması mümkün olmaz.

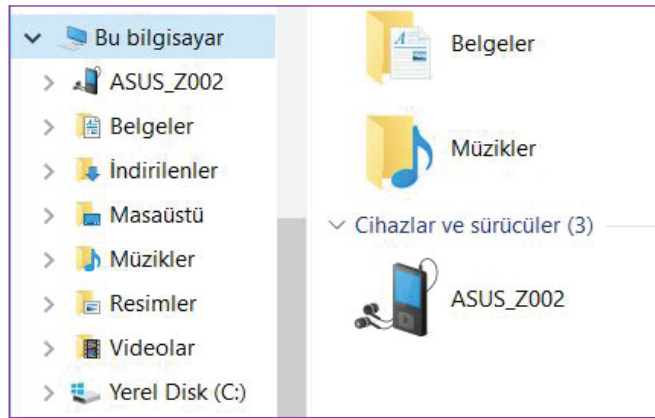
Fiziksel bağlantı kurulduktan sonra kimi mobil cihazlar bağlantının sağlanabilmesi için bir onay ekranı görüntüler, bu ekrandan onay verilmesi ile birlikte bağlantı tam olarak sağlanır. Bazı durumlarda ise mobil cihazın ekran kilidinin açılması gerekmektedir.

Gerekli izinler verildikten sonra bilgisayarınızın Dosya Gezini uygulaması aracılığıyla mobil cihazınızı görüntüleyebiliyor olmanız gerekir (Şekil 23).



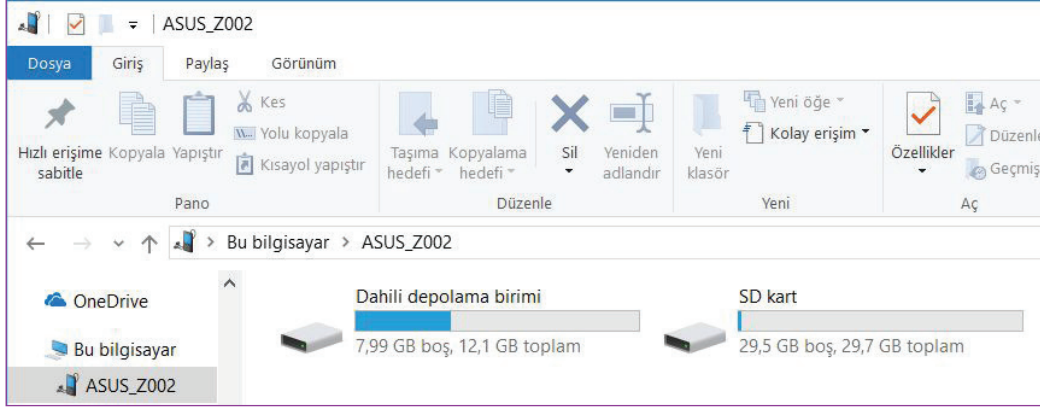
Şekil 23: Dosya Gezini aracılığıyla mobil cihazın görüntülenmesi

Dosya Gezini içerisinde mobil cihazınızın varsayılan olarak görüntüleneceği yer “Bu bilgisayar” adlı bilgisayarınızın sabit disklerinin de görüntülendiği bölümdür. Mobil cihazınız marka ve model adı ile “Cihazlar ve sürücüler” altında görüntüleniyor olmalıdır (Şekil 24).



Şekil 24: Mobil cihazın marka ve model adı ile görüntülenmesi

Bu aşamada mobil cihazınızı temsil eden ikona çift tıklayarak mobil cihazınızın depolama alanları görüntülenir. Mobil cihazınızda tek bir depolama alanı olabileceği gibi birden çok dahili depolama alanı ve / veya dışarıdan eklenebilen (ör.: SD kart vb. gibi) depolama alanları olabilir (Şekil 25).

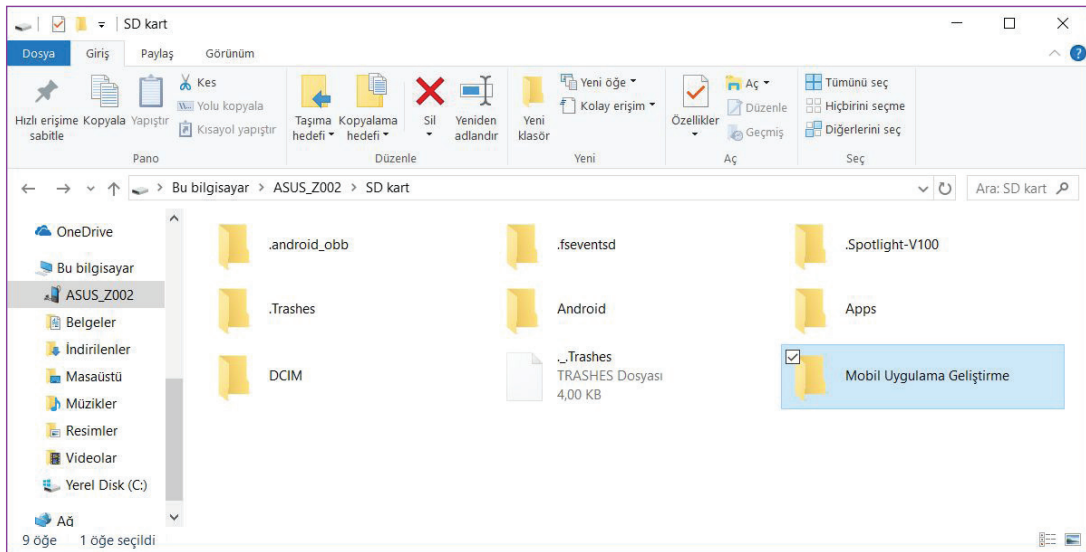


Şekil 25: Mobil cihazın depolama alanları.

Şekil 25'te görüldüğü gibi test cihazımızda bir dahili ve bir de SD kart olmak üzere 2 farklı depolama alanı bulunmakta. Bu testimizde uygulama dosyalarını kopyalamak için SD kart alanını kullanacağız. Bunun için SD kartı temsil eden ikona çift tıklıyoruz.

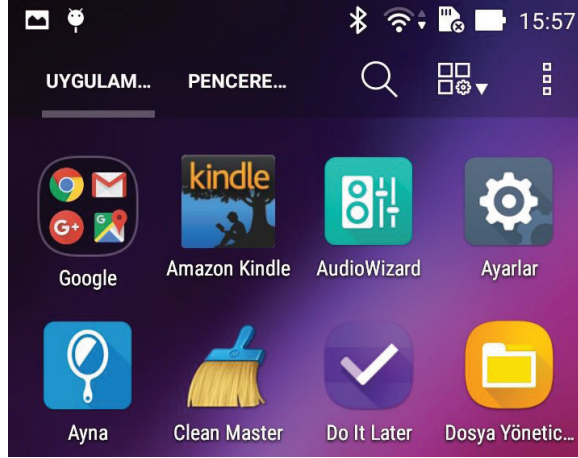
SD kart cihazımızın özelliklerine göre farklı dosyalar ve klasörler içerebilir. Bu dosya ve klasörler arasında mobil uygulamamızı rahatça bulabilmek için tüm çalışmalarımızı kaydettiğimiz ana klasör olan "Mobil Uygulama Geliştirme" adlı klasörü bilgisayarımızın sabit diskinde yer alan lokasyonundan kopyalayarak SD kart içerisine yapıştırmamız yeterli olacaktır. Kopyalama ve yapıştırma işlemi tamamlandığında mobil cihazımızın SD kart bölümünün görünümü Şekil 26'daki gibi olmalıdır.

Geliştirdiğimiz uygulamanın klasör ve dosyaları dâhilî depolama alanına da yerleştirilebilir, burada dâhilî depolama alanı ve SD kart depolama alanı arasında bir fark bulunmamaktadır.



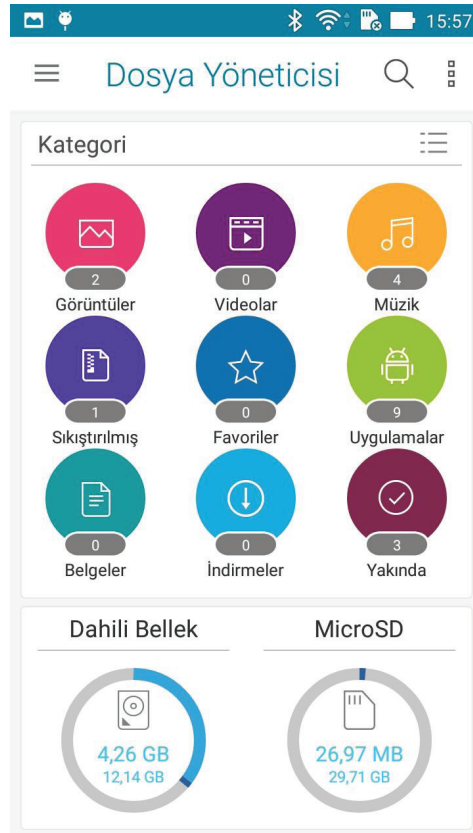
Şekil 26: "Mobil Uygulama Geliştirme" klasörünün SD karta kopyalanması

Mobil uygulamamızı oluşturan dosya ve klasörlerin mobil cihaza aktarılma işlemi tamamlandı. Artık testi gerçekleştirmek için mobil cihazımızın uygulamaları içinde yer alan dosya yöneticisi ya da benzeri bir program yardımıyla uygulama dosyalarına ulaşabiliriz (Şekil 27).



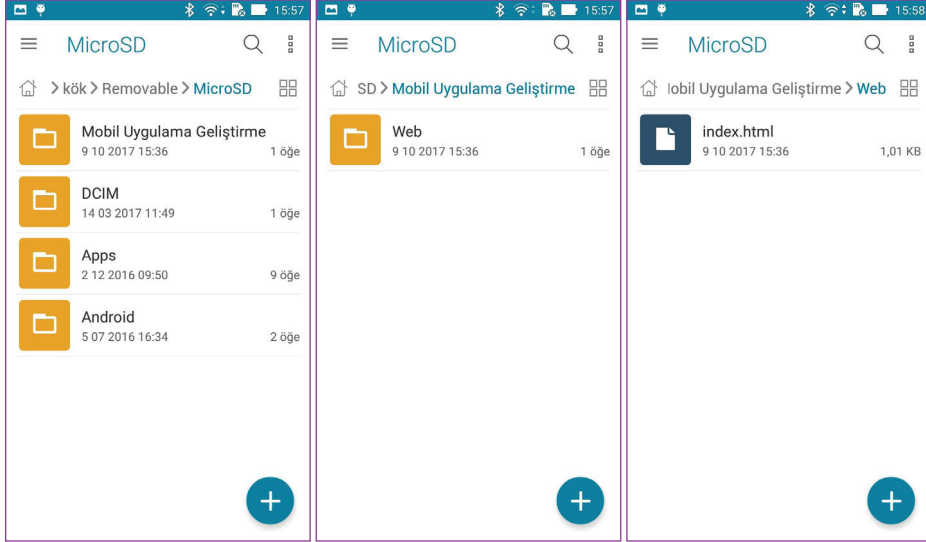
Şekil 27: Mobil cihazın uygulamalar bölümü

Dosya yöneticisi uygulamasına giriş yaptıktan sonra Micro SD kartın olduğu bölüme giriş yapıyoruz (Şekil 28).



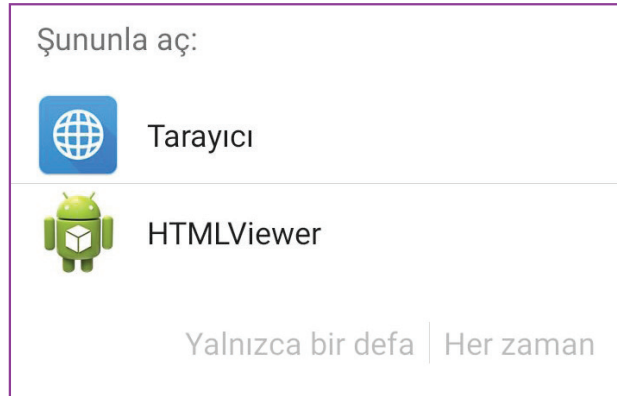
Şekil 28: Dosya Yöneticisi uygulaması ve mobil cihazın depolama alanları

Micro SD kart içerisine giriş yaptıktan sonra sırasıyla “Mobil Uygulama Geliştirme” ve “Web” klasörlerinin içine girerek uygulamamızın yer aldığı klasöre kadar ilerlemiş oluyoruz (Şekil 29).



Şekil 29: Mobil uygulamanın yer aldığı klasöre erişim

“index.html” adlı dosyayı gördüğümüzde artık bu dosyanın üzerine dokunarak uygulamamızın testini gerçekleştirebiliriz. Mobil cihazınızda HTML uzantılı dosyaları görüntüleyebilecek birden fazla uygulama yer alabilir. Bu durumda istediğiniz herhangi bir İnternet tarayıcıyı test için seçebilirsiniz (ör.: Standart Android İnternet tarayıcı, Chrome, Opera Mobil). Bu çalışmamızda mobil cihazımızda yer alan Standart Android İnternet tarayıcısını (ilk sıradaki “Tarayıcı”) seçiyoruz (Şekil 30).



Şekil 30: Test için tarayıcının seçilmesi

Görüntüleyiciyi seçtiğinizde otomatik olarak ilgili uygulama açılacak ve mobil uygulamanın yer aldığı HTML dosyası ekranda kullanılabilir şekilde yer alacaktır (Şekil 30-a).

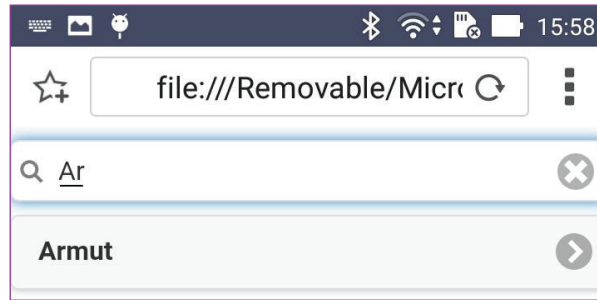




Şekil 30-a: Mobil tarayıcıda uygulamanın görüntülenmesi

Bu aşamada artık mobil cihazımızda uygulamamız kullanıma hazır hâle gelmiş oldu. Daha önce kişisel bilgisayarımızda test sırasında gerçekleştirdiğimiz işlemleri şimdi mobil cihazımız üzerinde deneyebiliriz.

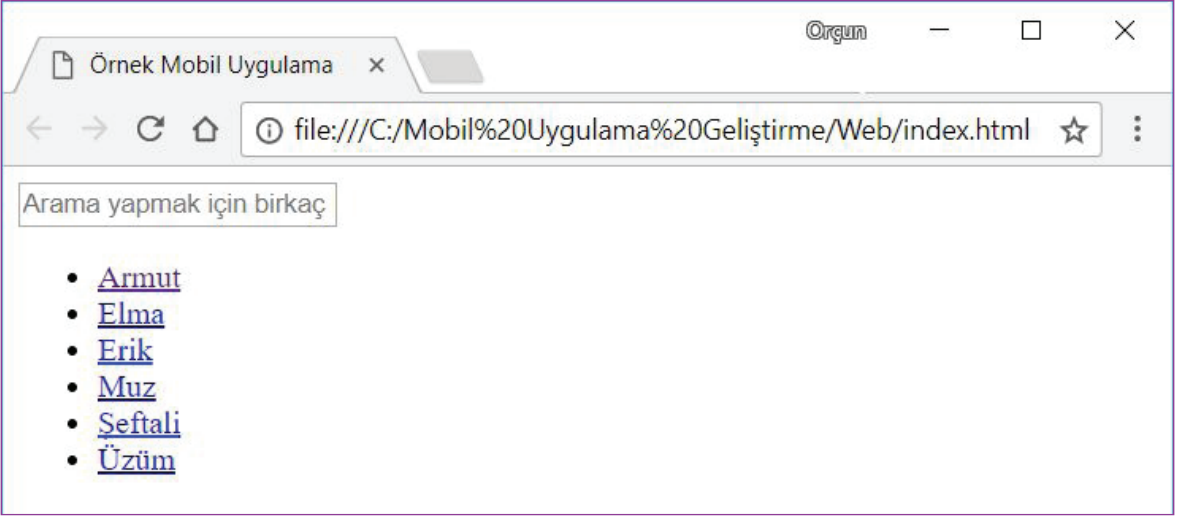
Arama kutucuğuna birkaç harf girdiğimizde listenin filtrelendiğini ve güncellendiğini görüntüleyebiliyor olmanız gerekir (Şekil 31).



Şekil 31: Arama kutucuğuna harf girilerek test işleminin gerçekleştirilmesi

#### 4.1.5. Mobil Uygulamada Test Aşamasında Yaşanabilecek Sorunlar ve Sorunların Çözümleri

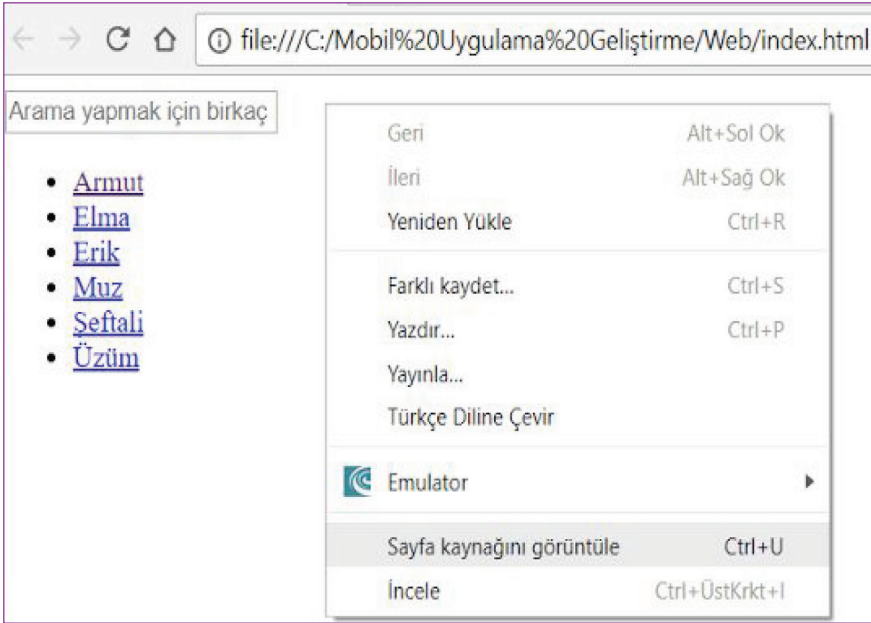
Web tabanlı bir mobil uygulama geliştirme sırasında yaşanabilecek sorunlardan biri uygulamanızda kullandığınız yazılım kütüphanelerine (örnek uygulamamızda kullanılan JQuery, JQuery Mobile vb. gibi) uygulama içerisinden erişilememesidir. Bu durumun temel sebepleri İnternet bağlantısı ile ilgili sorunlar ya da kütüphane adreslerinin yanlış yazılması olabilir. Her iki durumda da, gerekli olan Javascript ve CSS dosyalarına ulaşamadığı için uygulama tarayıcıda görüntülenecek ancak hem biçimsel olarak hem de fonksiyon açısından istediğimiz sonucu vermeyecektir (Şekil 32).



**Şekil 32:** Gerekli kütüphanelere erişemeyen uygulamanın İnternet tarayıcıdaki görüntüsü

Şekil 32'de görüldüğü gibi daha önceki testimize göre biçimsel olarak bozulmuş bir arayüz ile karşı karşıyayız; aynı zamanda arama yapmak istediğimizde filtreleme işlemi de gerçekleşmeyecektir. Bu aşamada çözüm için öncelikle İnternet bağlantımızın olup olmadığını kontrol etmemiz, daha sonra ise harici olarak bağlantı kurduğumuz yazılım kütüphanelerinin adreslerini kontrol etmemiz gerekir. İnternet bağlantımızın olduğunu varsayarak harici dosyaların adreslerini nasıl kontrol edebileceğimize bakalım.

Kontrol işlemi için sorun yaşadığımız HTML sayfasının boş bir bölümünde sağ klik yaparak "Sayfa kaynağını görüntüle" menü ögesine tıklayalım (Şekil 33).



**Şekil 33:** HTML sayfasının kaynağının görüntülenmesi için gerekli komutun verilmesi

İnternet tarayıcımızda açılan yeni bir sekmede HTML sayfasının kaynak kodu görüntülenecektir. Bu kaynak kod bizim Brackets içerisinde yazdığımız kodun aynısıdır ancak içerisindeki bağlantılar tıklanabilir durumdadır ve bu sayede haricî bağlantılı olan dosyalara erişilebiliyor mu, bunu test etmemiz mümkün hâle gelir (Şekil 34).



```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Örnek Mobil Uygulama</title>
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <link rel="stylesheet" href="code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css" />
8   <script src="code.jquery.com/jquery-1.11.1.min.js"></script>
9   <script src="code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js"></script>
10 </head>
11

```

Şekil 34: HTML kaynak kodun İnternet tarayıcıda görüntülenmesi

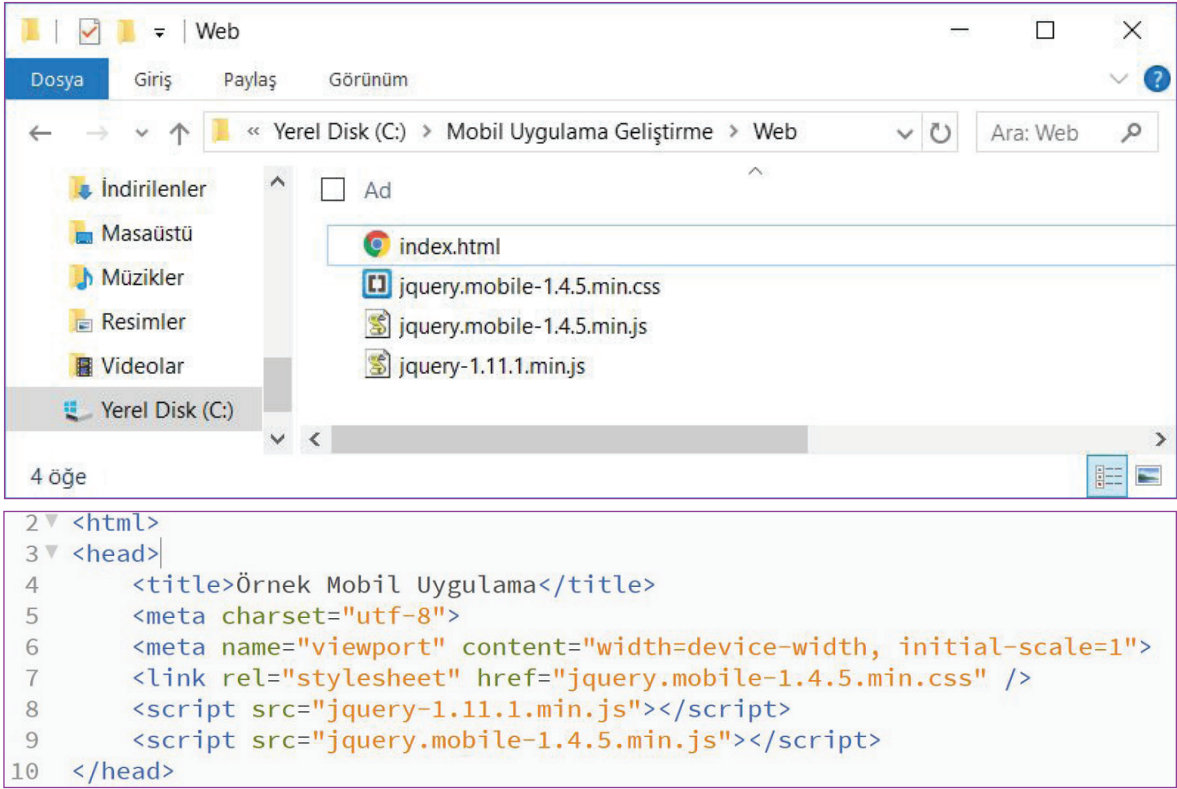
Kaynak kod içerisinde yer alan 7, 8 ve 9. satırlar, haricî kütüphanelerin erişim adreslerini içermektedir. Şimdi bu adreslerin üzerine tıklayarak “css” ve “js” uzantılı dosyalara erişilebiliyor muyuz kontrol edelim.



Şekil 35: Dosyanın bulunamadığına dair İnternet tarayıcıda yer alan hata mesajı

Şekil 35’te görüntülediği gibi dosya bulunamadı hata mesajı ile karşılaşıldı. Bunun sebebi haricî dosyaların bağlantı adreslerinin yanlış olmasıdır (İnternet bağlantınızın olduğu varsayılmaktadır). Örnek kodda bağlantı adreslerinin başında bulunan “http://” ön eki silinmiştir. Bu da dosyaya erişimi engellemektedir. Haricî dosyalarının bağlantı adreslerini Brackets’den güncelleyelim ve tekrar erişim sağlanabilecek bir yapıya kavuşturalım. Adresler güncellendiğinde sayfa kaynağındaki bağlantılara tıkladığımızda ilgili dosyayı İnternet tarayıcınızda görüntüleyebiliyor olmanız gerekir (Şekil 36).

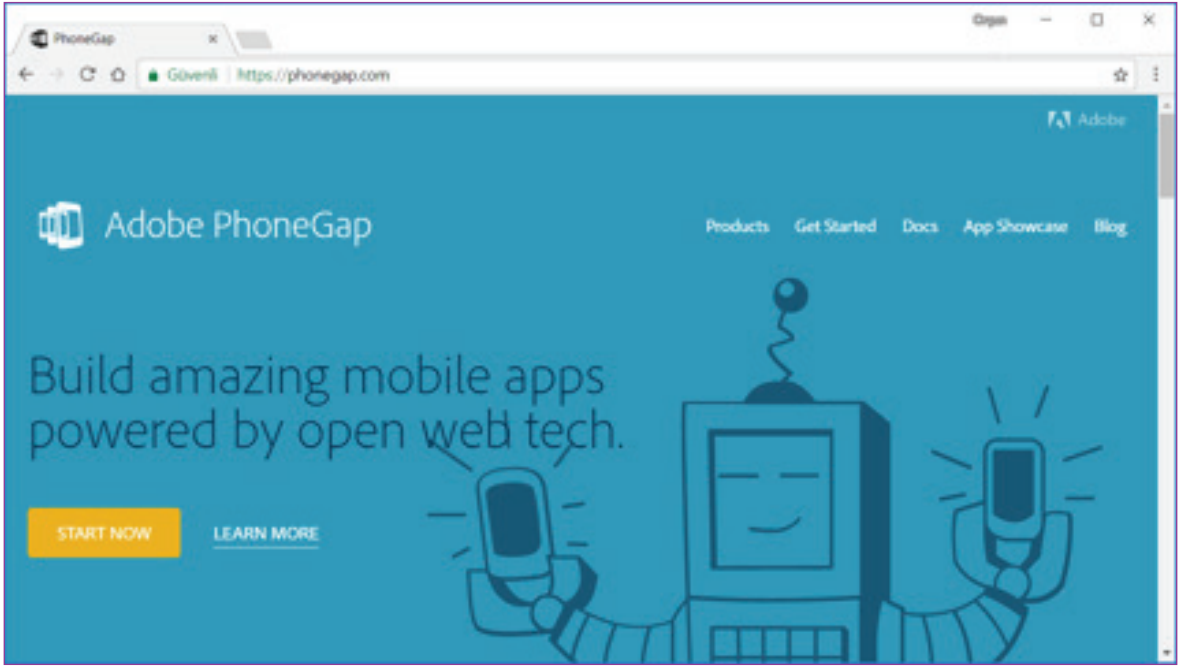




**Şekil 37:** Kütüphane dosyalarının yeni lokasyonuna göre adreslerin güncellenmesi

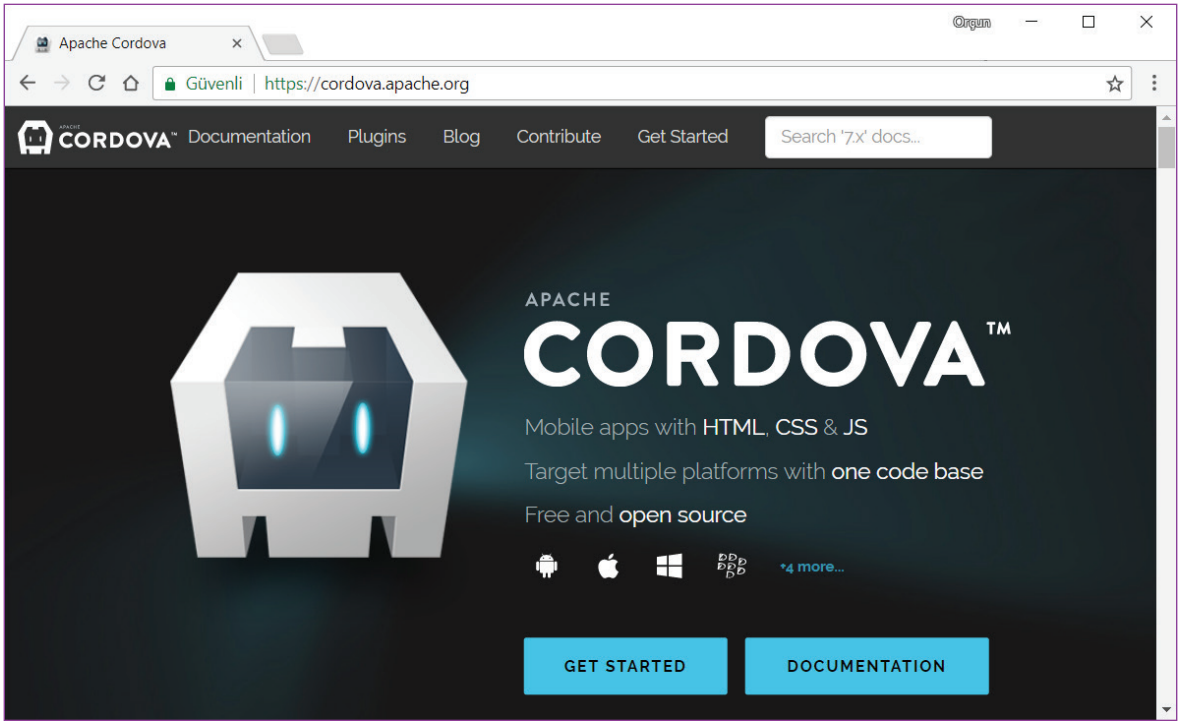
## 4.2. Web Tabanlı Mobil Uygulamanın Yerel (Native) Bir Uygulama Olarak Paketlenmesi

Mobil uygulamaların Web tabanlı olarak geliştirilmesi her ne kadar süreci hızlandırır da birtakım sınırlılıkları da beraberinde getirmektedir. Hem Web tabanlı geliştirme ortamının avantajlarından yararlanmak, hem de beraberinde getirdiği sınırlılıkları en aza indirmek için geliştirilen uygulamanın sanki Yerel (Native) bir uygulamaymış gibi paketlenip mobil cihazlara yüklenebilmesi mümkündür. Bu şekilde paketleme işlemi yapan birçok platform bulunmaktadır. Bu platformların en gelişmiş olanlarından biri Adobe PhoneGap'tir. PhoneGap'in geliştiricisi Nitobi firmasının Adobe tarafından satın alınmasıyla PhoneGap iki farklı platform üzerinde geliştirilmeye devam etmektedir. Bu platformlardan biri Adobe'nin geliştirme çalışmalarına devam ettiği "Adobe PhoneGap"tir (Şekil 37-a).



Şekil 37-a: Adobe PhoneGap (<https://phonegap.com/>)

Diğer platform ise PhoneGap'in kaynak kodlarının Apache Yazılım Vakfı'na (Apache Software Foundation) devredilmesiyle ortaya çıkan ve vakıf tarafından desteklenen Cordova Projesi'dir (Şekil 38).

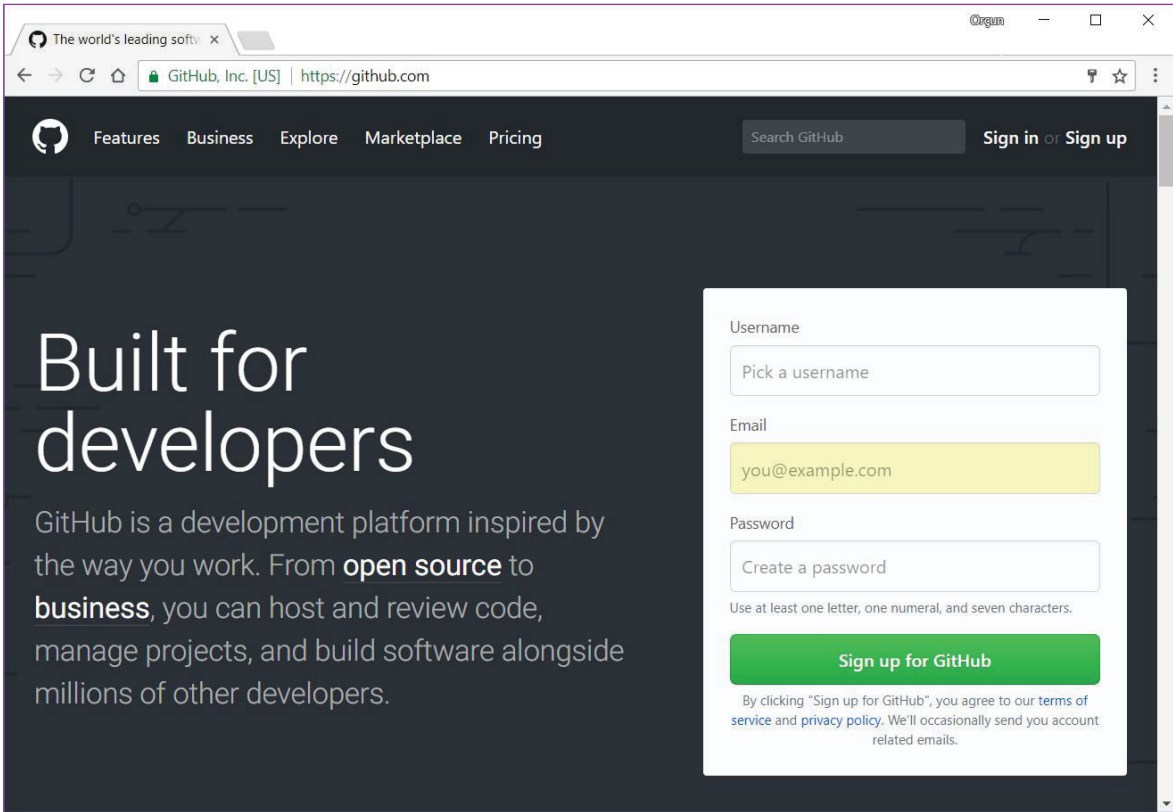


Şekil 38: Apache Cordova Projesi (<https://cordova.apache.org/>)

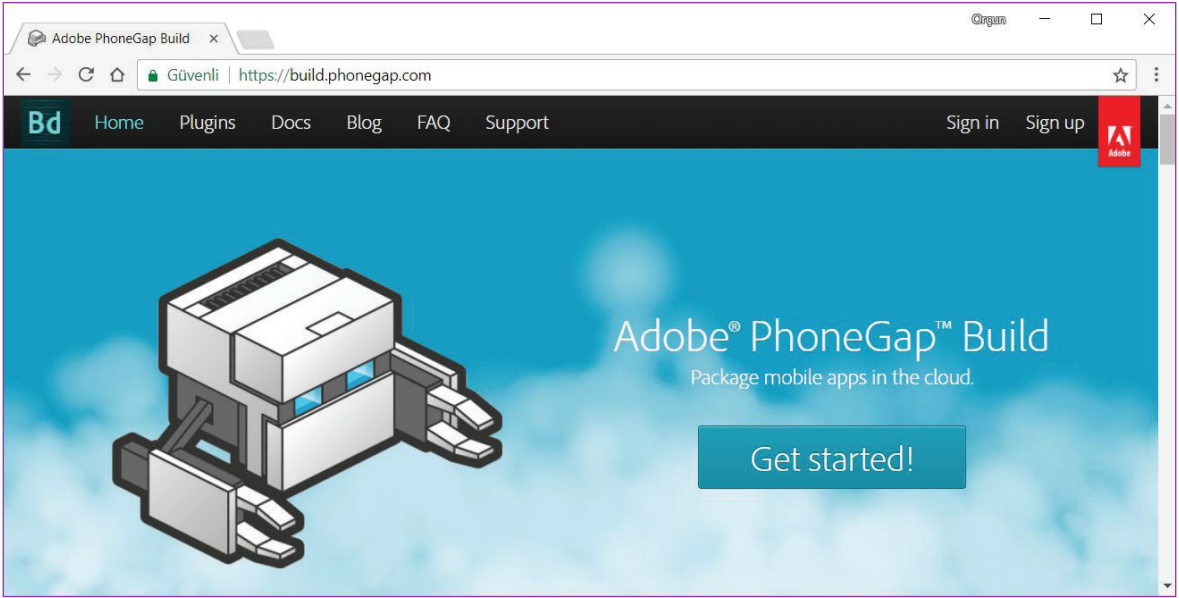
Apache Cordova, açık kaynak kodlu bir proje olması açısından, Adobe PhoneGap ise desteklediği yazılım kütüphanelerinin zenginliği açısından avantajlar sunmaktadır. Buna ek olarak Adobe PhoneGap paketleme işlemini kişisel bilgisayarınıza herhangi bir özel yazılım yüklemenize gerek kalmadan bulut üzerinde gerçekleştirebilmektedir. Yine Web tabanlı bir emülatörü de bulunan Adobe PhoneGap bu hizmetleri ücretsiz olarak vermektedir. Bu çalışmamızda geliştirdiğimiz Web tabanlı mobil uygulamayı Adobe PhoneGap platformu yardımıyla paketleyerek mobil cihazımıza yükleyeceğiz.

#### 4.2.1. Mobil Uygulamanın Paketlenmesi İçin Kullanılacak Bulut Platformları

Yukarıda da belirtildiği gibi Adobe PhoneGap yardımıyla tüm süreç bulut üzerinde bir İnternet tarayıcısı yardımıyla gerçekleştirilebilmektedir. Bu süreci yönetebilmek için iki farklı bulut platformunda kullanıcı hesabına ihtiyacımız bulunmaktadır. Bu bulut platformlarından ilki dosyalarımızın yükleneceği ve depolanacağı GitHub kod deposudur (Şekil 39).



Şekil 39: GitHub kod deposu (<https://github.com/>)



**Şekil 40:** Adobe PhoneGap Built (<https://build.phonegap.com/>)

Her iki bulut platformu için de kullanıcı üyeliği ücretsizdir. Geçerli bir e-posta adresinizin olması durumunda kullanıcı hesabınızı kolaylıkla oluşturabilirsiniz.

İlk olarak GitHub platformu üzerinde kullanıcı hesabımızı oluşturalım ve giriş yaparak mobil uygulamamızın dosyalarını kod deposuna yükleyelim. İnternet tarayıcımızdan github.com adresine giriş yaptığımızda yeni bir hesap oluşturmak için sayfanın sağ tarafında yer alan formu doldurarak “Sign up for Github” düğmesine tıklayalım (Şekil 41).

Username

Email

Password

Use at least one letter, one numeral, and seven characters.

[Sign up for GitHub](#)

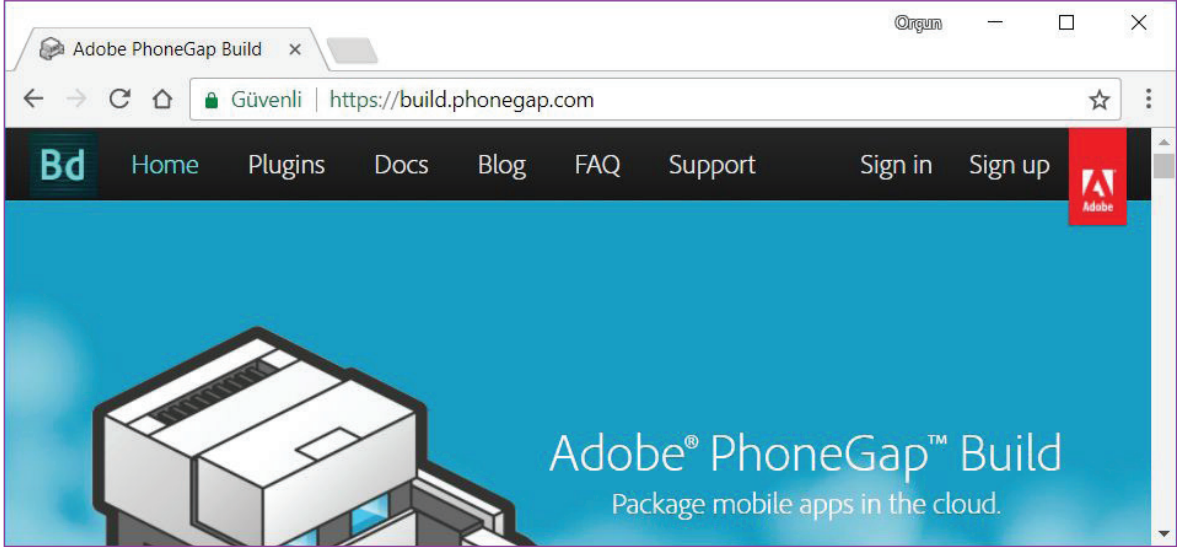
By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you account related emails.

**Şekil 41:** Yeni bir kullanıcı hesabı oluşturmak için kullanılacak iletişim formu



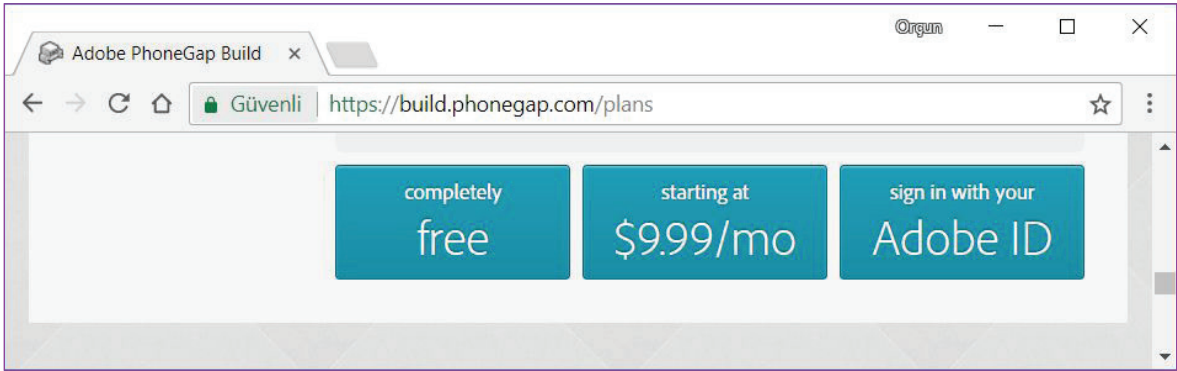
Kayıt işlemi sırasında istenen bilgiler arasında Kullanıcı Adı (Username), E-Posta (Email) ve Şifre (Password) yer almakta. Bu bilgileri dikkatli bir şekilde doldurmakta fayda var çünkü daha sonra sisteme giriş yaparken bu bilgilere ihtiyacınız olacak.

Şimdi ise Adobe Built için gerekli olan kullanıcı hesabını oluşturalım. <https://build.phonegap.com/> adresine giriş yaptığımızda sağ üst köşede “Sign in” (Giriş yap) ve “Sign up” (Kayıt yaptır) seçenekleri bulunmakta. Bu seçeneklerden “Sign up” (Kayıt yaptır) seçeneğini tıklayalım (Şekil 42).



**Şekil 42:** Adobe Built Web sitesinde “Sign up” (Kayıt yaptır) menü öğesinin seçilmesi

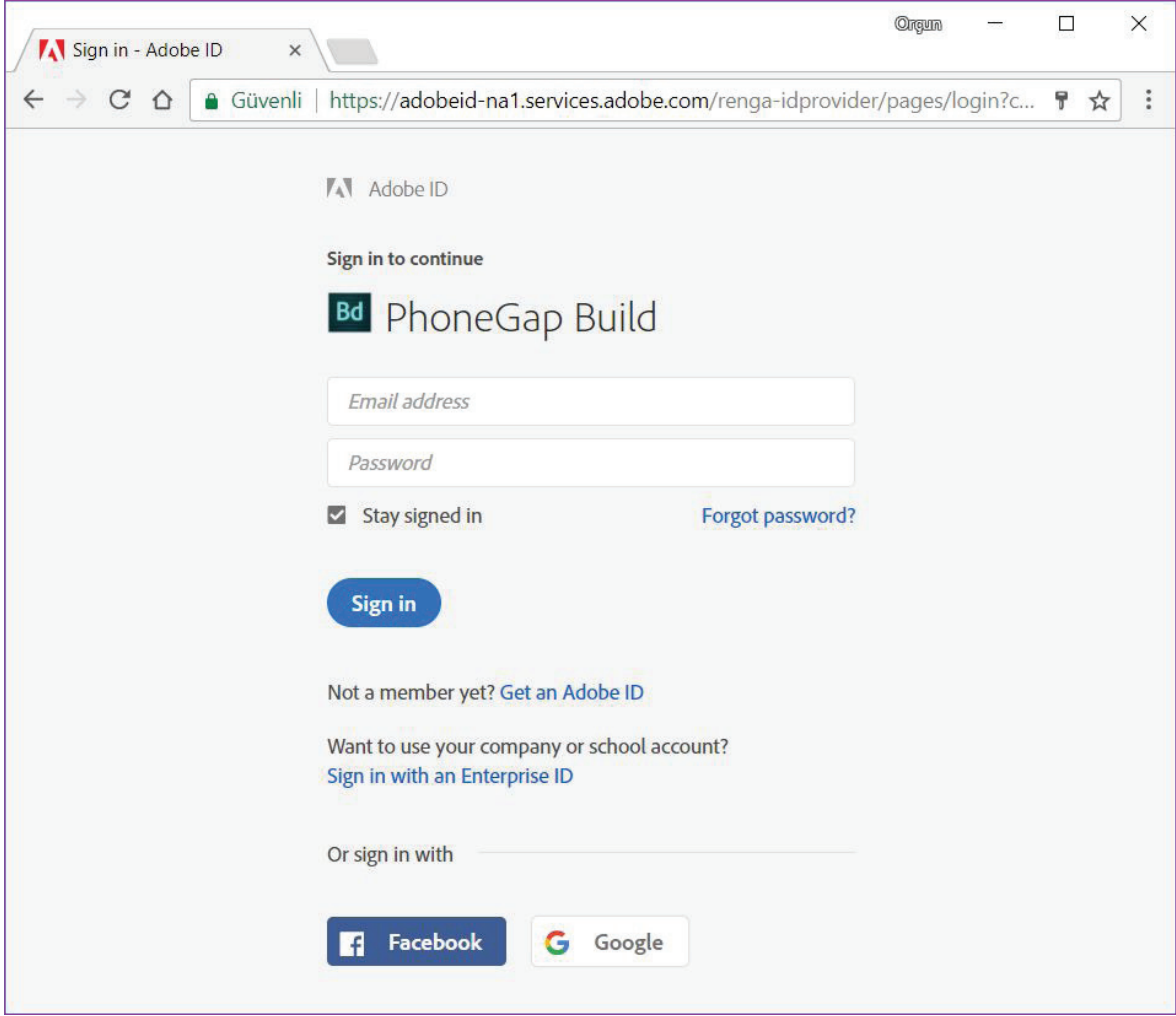
“Sign up” (Kayıt yaptır) menü öğesine tıklandığında platformu kullanım için sunulan farklı seçeneklerin yer aldığı bir sayfaya ulaşılır. Bu sayfada ücretsiz olan planı “Completely free” düğmesine basarak seçelim (Şekil 43).



**Şekil 43:** Ücretsiz olan kullanım planının seçilmesi

Kullanım planı seçimi sonrasında sisteme giriş için kullanıcı adı ve şifresinin olduğu bölüm ekrana gelecektir. Bu ekranda eğer daha önceden bir Adobe Hesabı varsa bu hesap ile giriş yapılabilir, Facebook ve / veya Google hesapları ile giriş yapılabilir ya da yeni bir Adobe Hesabı oluşturmak için “Get an

Adobe ID” (Bir Adobe Kimliđi al) bağlantısına tıklanır (Şekil 44).



The screenshot shows a web browser window with the title "Sign in - Adobe ID". The address bar shows a secure connection to "https://adobeid-na1.services.adobe.com/renga-idprovider/pages/login?c...". The page content includes the Adobe ID logo, the text "Sign in to continue", and the "PhoneGap Build" logo. There are two input fields for "Email address" and "Password". A checkbox labeled "Stay signed in" is checked, and a link "Forgot password?" is visible. A blue "Sign in" button is present. Below the button, there are links for "Not a member yet? Get an Adobe ID" and "Want to use your company or school account? Sign in with an Enterprise ID". At the bottom, there are social login options for "Facebook" and "Google".

**Şekil 44:** Adobe Built platformu için kullanıcı hesabının seçilmesi ya da oluşturulması

Yeni bir Adobe Hesabı oluşturmak için “Get an Adobe ID” (Bir Adobe Kimliđi al) bağlantısına tıklanıldığı zaman kullanıcı hesabının oluşturulacağı iletişim formu görüntülenir. Bu iletişim formundaki bilgileri dikkatli bir şekilde doldurduktan sonra “Sign up” (Kayıt ol) düğmesine tıklayarak kayıt işlemi tamamlanır (Şekil 45).

Adobe ID

Sign up to continue

**Bd** PhoneGap Build

First name

Last name

Email address

Password

Turkey

Stay informed about Adobe products and services. [Learn more.](#)

I have read and agree to the [Terms of Use](#) and [Privacy Policy](#).

**Sign up**

Already have an Adobe ID? [Sign In](#)

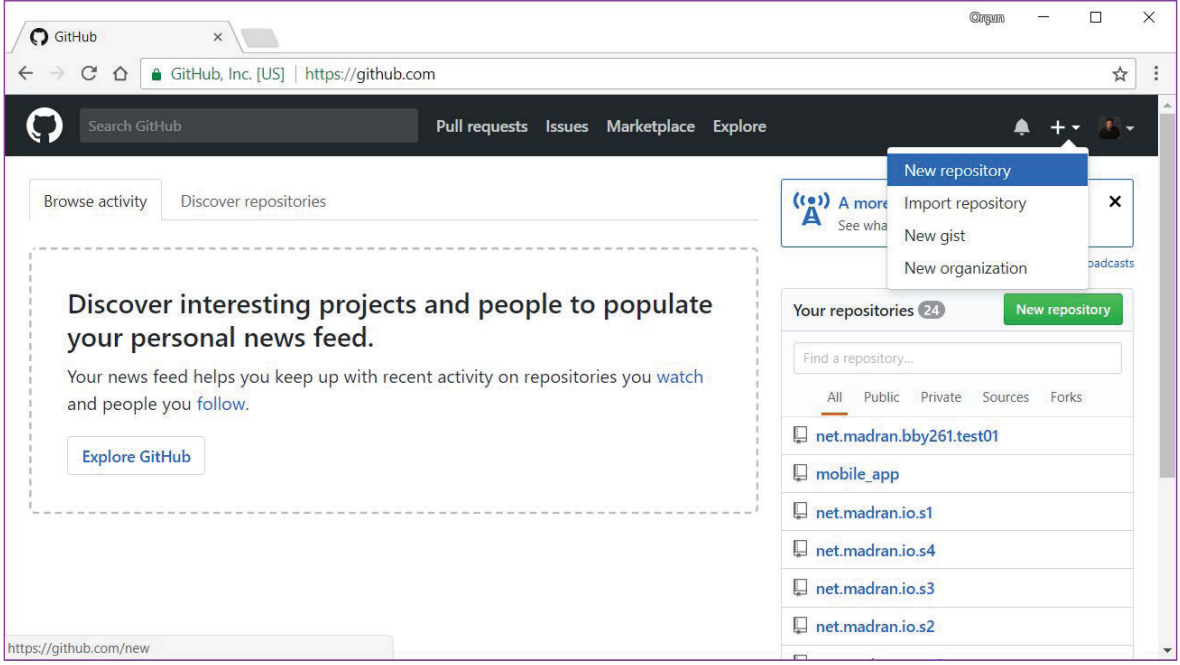
**Şekil 45:** Adobe ID (Adobe Kimliği) için iletişim formu

Bu ve benzeri kayıt işlemlerinde bulut platformları çoğunlukla kayıt olurken verilen e-posta adresine bir doğrulama postası gönderir ve bu e-posta mesajının içerisindeki bağlantıya tıklayarak kullanıcı hesabının aktif hâle getirilmesini isterler. Bu açıdan kayıt sırasında e-posta adresi forma yazılırken doğruluğunu kontrol etmek çok önemlidir.

#### 4.2.2. Mobil Uygulamanın GitHub Kod Deposuna Yüklenmesi

Geliştirilen uygulamanın GitHub kod deposuna yüklenebilmesi için daha önce oluşturulan kullanıcı hesabı bilgileri (kullanıcı adı ya da e-posta ve şifre) ile platforma giriş işlemi gerçekleştirilir. Giriş panelinin görüntülenebilmesi için <https://github.com/> ana sayfasının sağ üst köşesinde yer alan “Sign in” (Giriş yap) düğmesine tıklanır.

Giriş işlemi gerçekleştirildiğinde öncelikli olarak kod deposu içerisinde yeni bir depo oluşturulması gerekmektedir. Yeni bir deponun oluşturulması için sayfanın sağ üst köşesindeki “+” işaretine tıklayarak “New repository” (Yeni depo) menü öğesi seçilmeli ya da doğrudan sayfanın sağ bölümünde yer alan “New repository” (Yeni depo) düğmesine tıklanmalıdır (Şekil 46).



**Şekil 46:** “New repository” (Yeni depo) oluşturulması

GitHub içerisinde birden çok kod deposu oluşturulabilir, her bir depo bir projeyi / uygulamayı temsil edecektir. Aynı uygulamanın birden fazla sürümü de bir kod deposu içinde yer alabilir.

Yeni bir depo oluşturmak için Şekil 47’de görüntülenen form aracılığıyla projenin yer alacağı depo ile ilgili temel bilgiler girilir. Bu bilgiler arasında en önemlisi deponun adının yer aldığı “Repository name” alanıdır. Bir depo adı bir kullanıcı hesabı içerisinde ancak bir kere kullanılabilir. Depo adını oluştururken boşluk, Türkçe karakter ve noktalama işaretlerini kullanmamak gerekir (noktalama işaretlerinden bazıları kullanılabilir, ör.: nokta).

Depo adının bir başka önemi ise bu adın mobil cihazlar için oluşturulan uygulama marketlerinde paket adı olarak kullanılacak olmasıdır. Bu açıdan bir mobil uygulama resmî olarak uygulama marketlerde yer alacaksa, bu marketlerin belirlemiş olduğu paket adı özelliklerine uygun olarak paket adının (dolayısıyla depo adının) belirlenmesi gerekir. Bu genelde uygulamayı geliştiren organizasyonun Web sitesi olarak kullandığı alan adının tersten kullanımı şeklinde bir algoritma içerir ve bu şekilde dünya üzerinde benzersiz bir paket adı oluşturulabilir.

Create a New Repository x

GitHub, Inc. [US] | https://github.com/new

Search GitHub Pull requests Issues Marketplace Explore

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner:  Repository name:

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-computing-machine](#).

Description (optional):

**Public**  
Anyone can see this repository. You choose who can commit.

**Private**  
You choose who can see and commit to this repository.

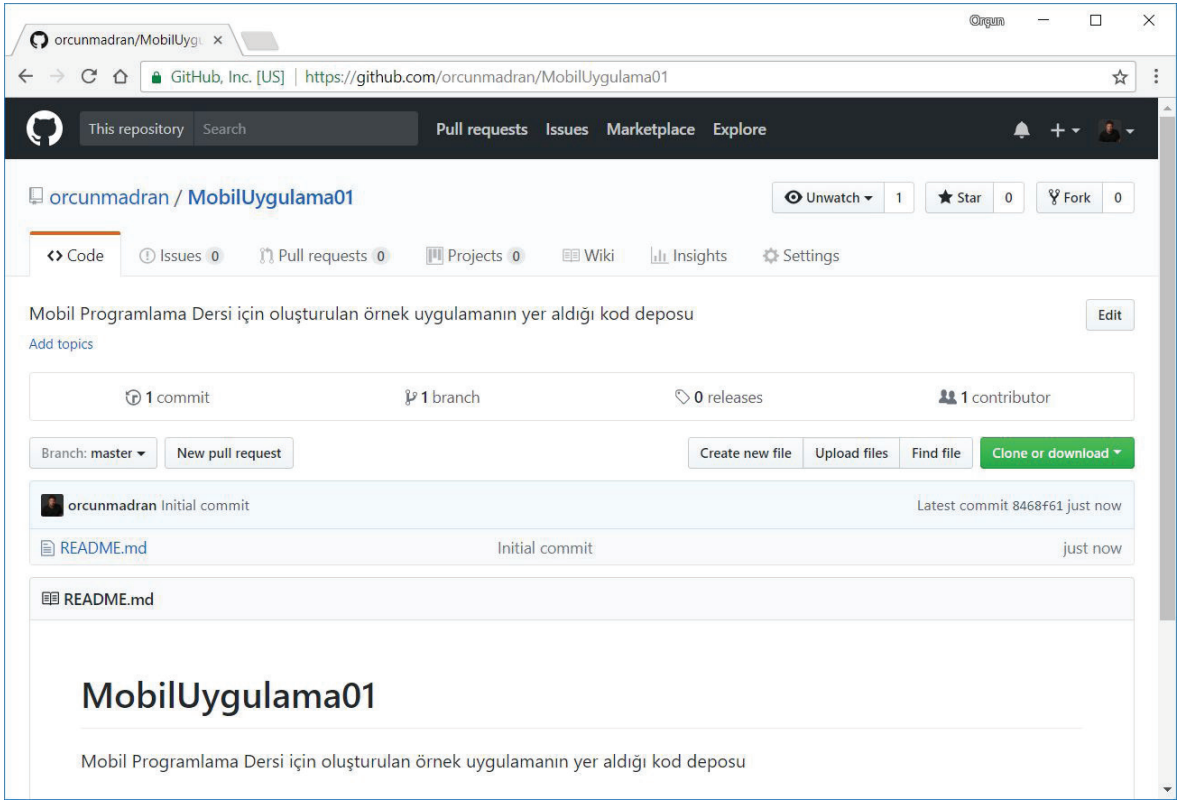
**Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore:  Add a license:

Şekil 47: Yeni depo oluşturulması için kullanılacak form

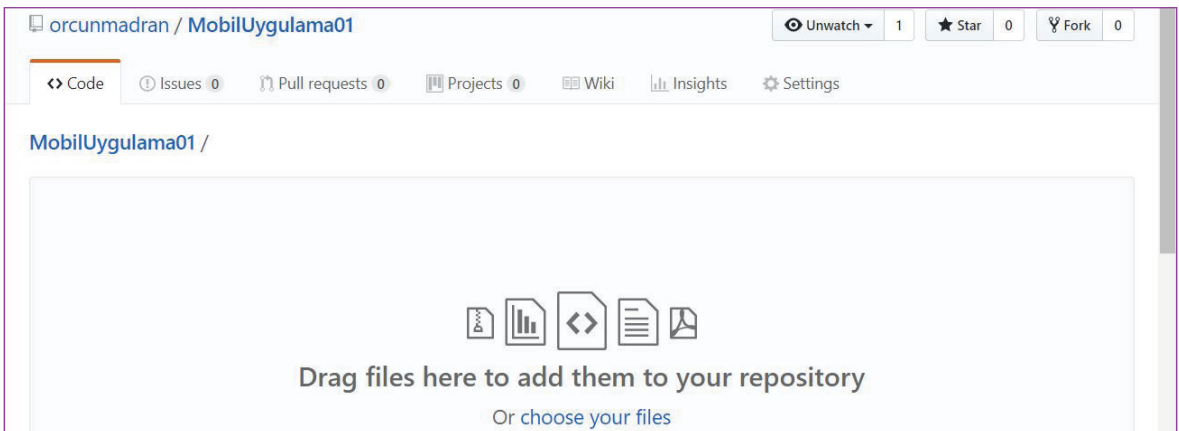
Kod deposu oluşturulurken depo hakkında genel bir bilgi verilmesi amacıyla “README” (BENİOKU) adlı bir dosyanın da oluşturulmasında fayda vardır. Şekil 47’de yer alan formdaki “Initilize this repository with a README” seçeneği işaretlenerek bu dosyanın otomatik olarak oluşturulması sağlanabilir. Bu şekilde kod deposuna Web üzerinden direkt dosya yüklenmesini sağlayacak başlangıç işlemleri de tamamlanmış olur.

“Create repository” (deponu oluştur) düğmesine tıkladığında yeni deponun oluşturulması için gerekli komut verilmiş olur ve eğer form içeriği ile ilgili bir sorun yoksa Şekil 48’deki yeni kod deposunun içeriğini gösteren ekran görüntülenir.



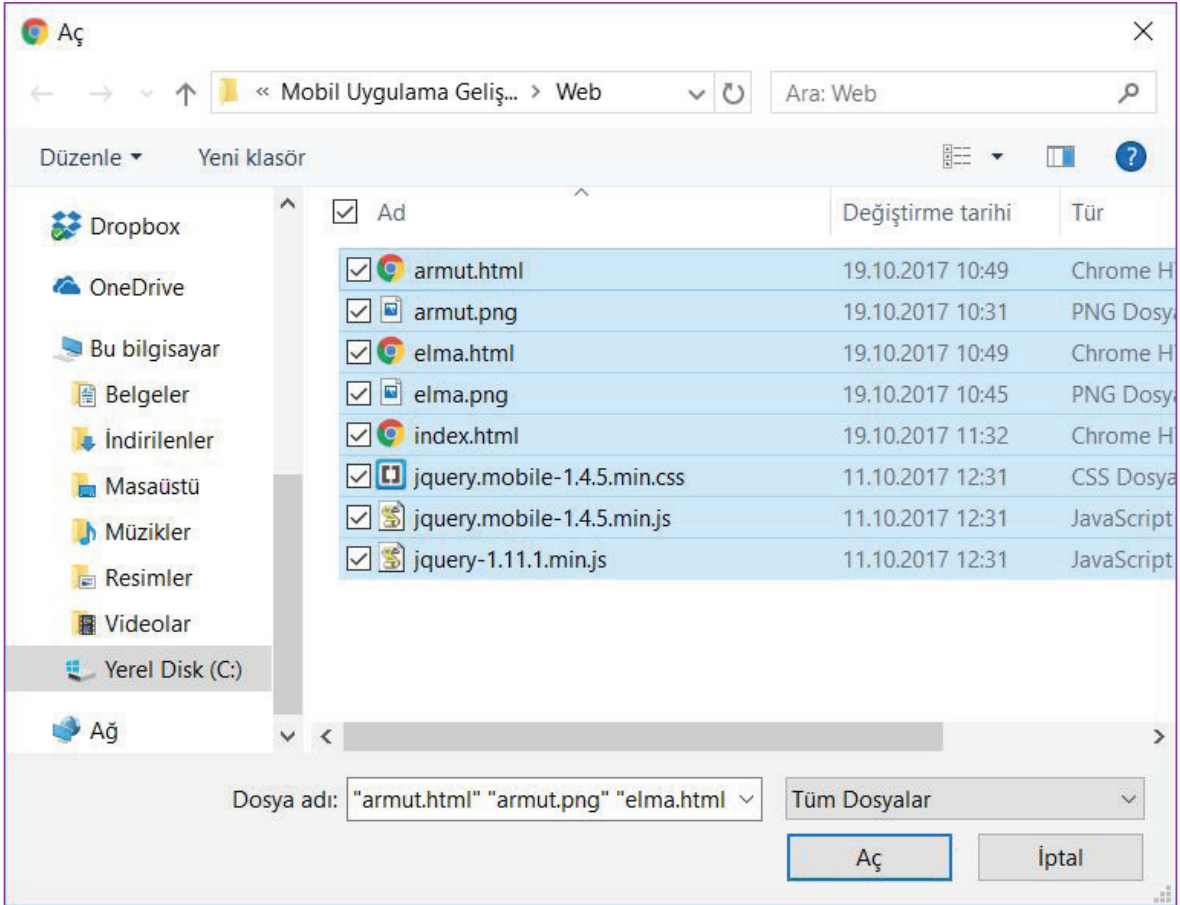
Şekil 48: Oluşturulan yeni deponun görüntülenmesi

Yeni deponun içinde şu an için sadece uygulama ile ilgili genel bilgilerin yer aldığı README.md dosyası bulunmaktadır. Uygulamayı oluşturan tüm dosyaların bu kod deposuna yüklenmesi gerekmektedir. Bu yükleme işlemi dosya listesinin yer aldığı bölümün hemen sağ üst köşesinde yer alan “Upload files” (Dosyaları yükle) düğmesi ile yapılır. Düğmeye tıklandığında Şekil 49’da yer alan ekran görüntülenir.



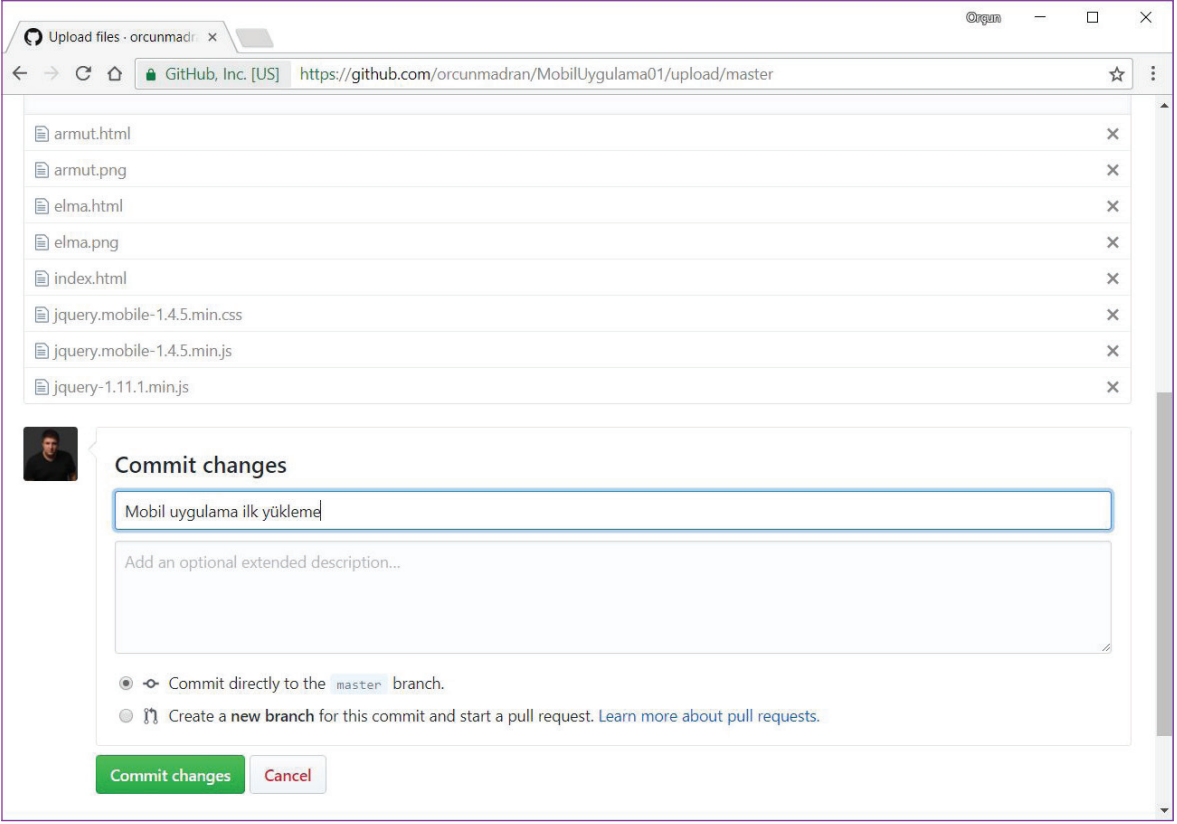
Şekil 49: Dosyaların yükleneceği arayüz

Bu ekran yardımıyla iki farklı şekilde dosya yükleme işlemi gerçekleştirilebilir, dosyaları seçtikten sonra sürükleyip gri bölüme bırakarak ya da bilgisayardan dosyaları seçerek. Orta bölümde yer alan “choose your files” (dosyalarınızı seçiniz) bağlantısına tıkladığınızda dosyaları seçebileceğiniz iletişim kutusu görüntülenecektir (Şekil 50).



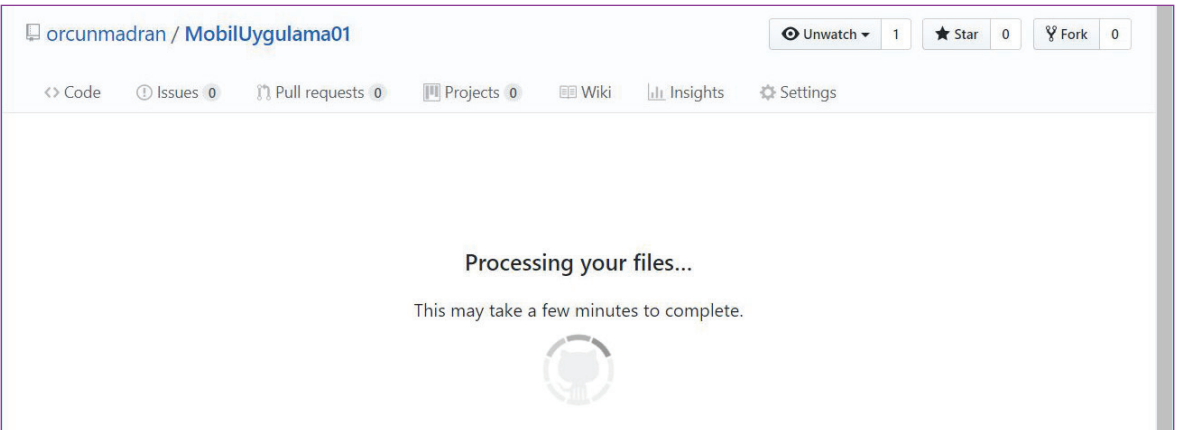
Şekil 50: Dosyaların seçileceği iletişim kutusu

Tüm dosyalar işaretlendikten sonra “Aç” düğmesine tıklanır ve yükleme işlemi öncesi dosya seçimi tamamlanmış olur. Kod deposuna dosyaların teslim edilmesi için bir mesaj eşliğinde sürecin onaylanması istenir.



Şekil 51: Değişikliklerin kod deposuna teslim edilmesi

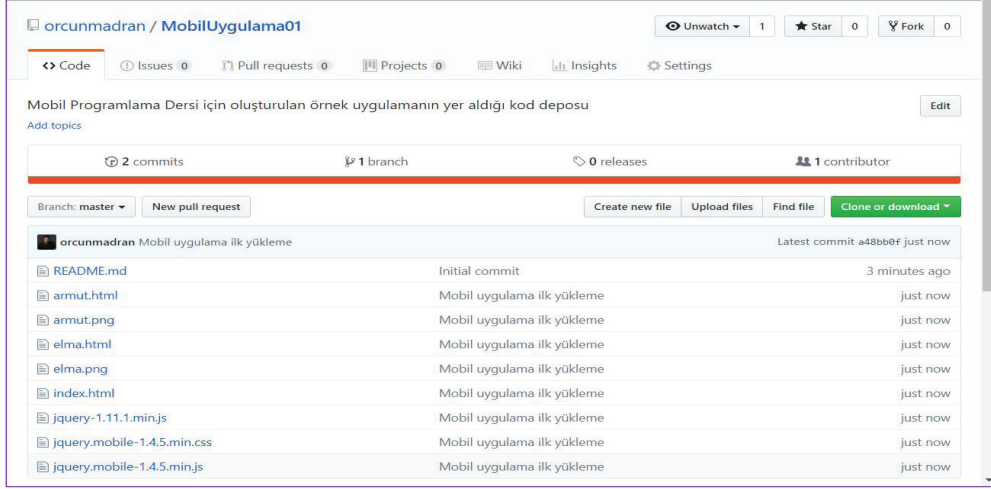
Şekil 51’de yer alan forma yükleme işlemi ile ilgili kısa bir bilgi girdikten sonra “Commit changes” (değişiklikleri teslim et) düğmesine tıklandığında kod deposuna uygulama dosyalarının gönderilme işlemi başlatılır. İnternet bağlantısının hızına bağlı olarak dosyaların gönderilme işlemi bir süre devam eder (Şekil 52).



Şekil 52: Dosyaların gönderilme süresi



Dosyalar kod deposuna teslim edildiğinde tüm uygulama bileşenleri Şekil 53'teki gibi görüntülenir. Kodlarda yapılacak herhangi bir değişiklik sonrası sadece değişiklik yapılan dosyanın yeniden kod deposuna gönderilmesi yeterlidir.



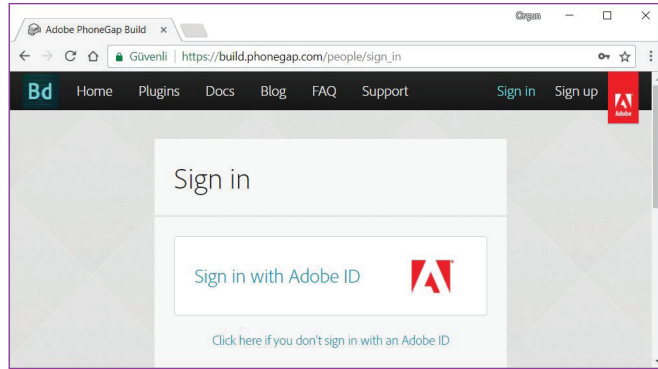
Şekil 53: Kod deposunun dosyalar teslim edildikten sonraki görünümü

Bu ders kapsamında geliştirilen “MobilUygulama01” adlı örnek uygulamaya GitHub üzerinden aşağıdaki adres yardımıyla ulaşabilirsiniz. Dosyaları ve içeriklerini görüntüleyebilir, kendi kişisel bilgisayarınıza indirebilirsiniz:

<https://github.com/orcunmadran/MobilUygulama01>

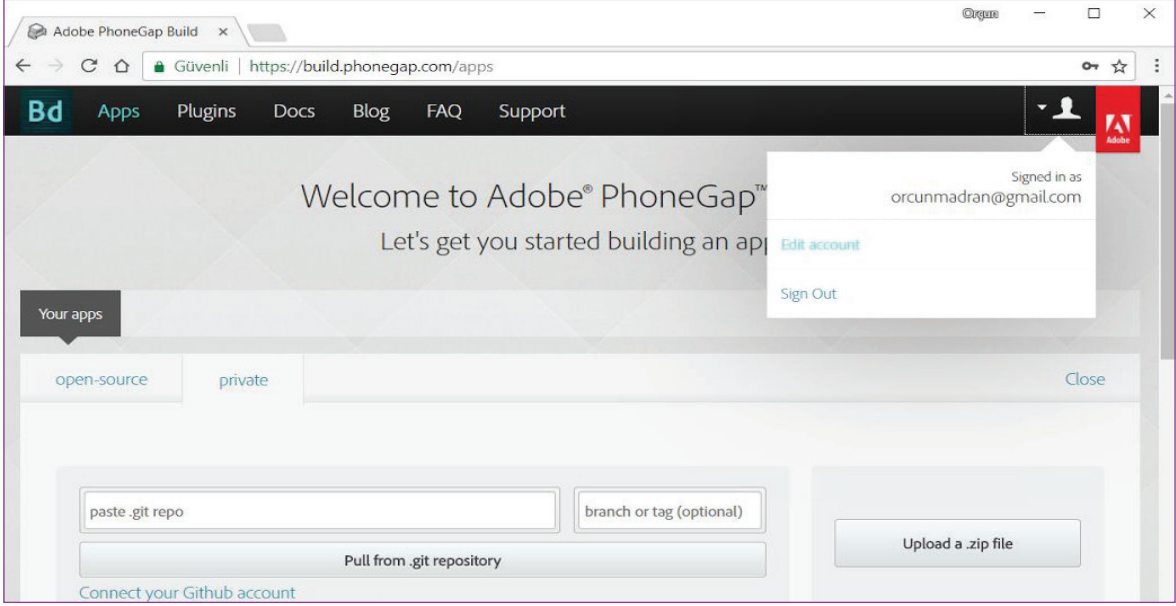
#### 4.2.3. Paketleme İşleminin Gerçekleştirilmesi İçin PhoneGap Build Platformunun Kullanılması

GitHub kod deposuna uygulama dosyalarının yüklenmesi ile paketleme işleminde kullanılacak dosyalar için yapılan ön hazırlık tamamlanmış olur. Bu aşamada paketleme işleminin gerçekleştirileceği PhoneGap Build (<https://build.phonegap.com/>) platformuna ücretsiz oluşturulan Adobe ID (Adobe Kimliği) ile giriş yapılır (Şekil 54).



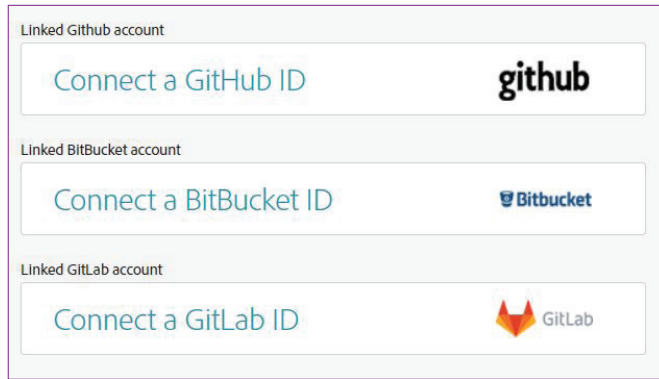
Şekil 54: Adobe ID ile PhoneGap Built platformuna giriş yapılması

Giriş yapıldıktan sonra sağ üst köşede bulunan platform ile ilgili hesap ayarlarının düzenlendiği bölümden “Edit account” (Hesabı düzenle) öğesi seçilerek GitHub ile bağlantının kurulacağı sayfa görüntülenir (Şekil 55).



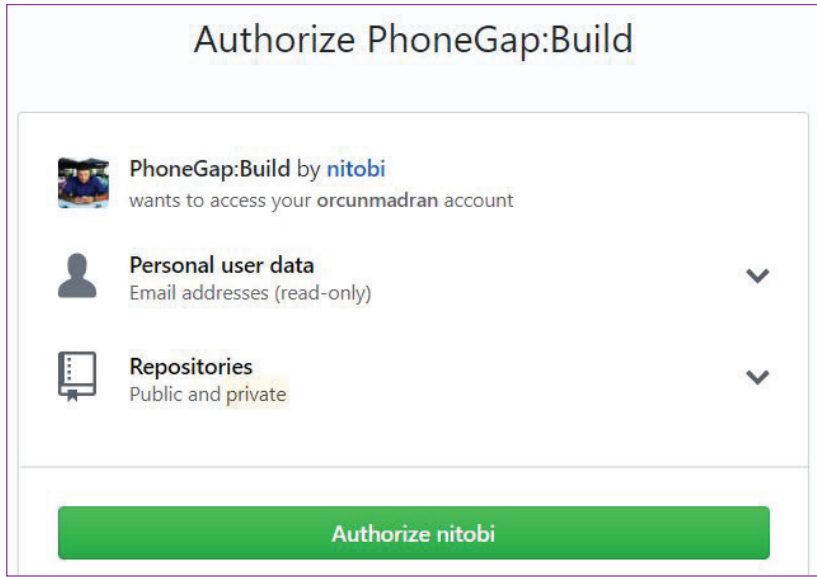
Şekil 55: GitHub ile bağlantının kurulacağı sayfanın görüntülenmesi

Şekil 56’da görüntülenen sayfadan bağlantı kurulmak istenen platform seçilir. Mobil uygulamamız ile ilgili dosyalar GitHub kod deposunda olduğu için Github platformunun üzerine tıklayarak sonraki aşamaya geçilir.



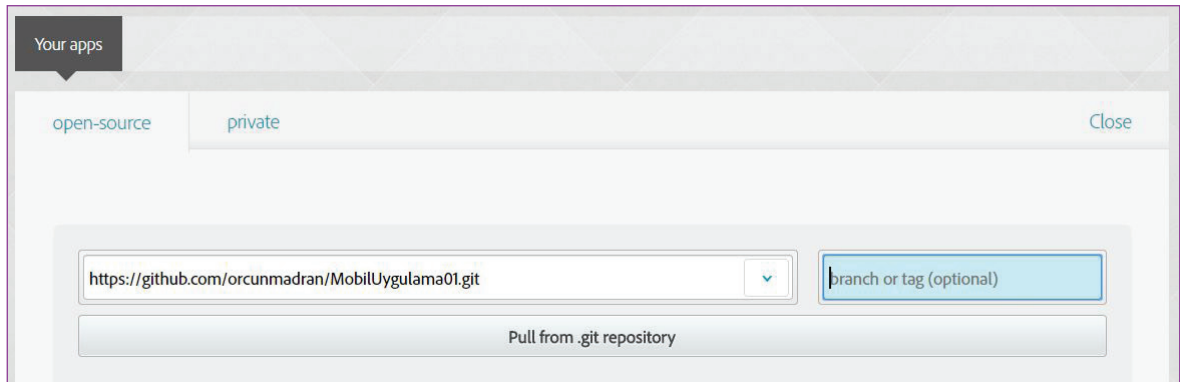
Şekil 56: Bağlantı kurulacak platformun seçilmesi

Seçim işleminden sonra PhoneGap Build ve Github platformları arasında bağlantı kurulmasına ilişkin iletişim sayfası görüntülenir ve bu sayfada işleme onay verilmesi istenir (Şekil 57).



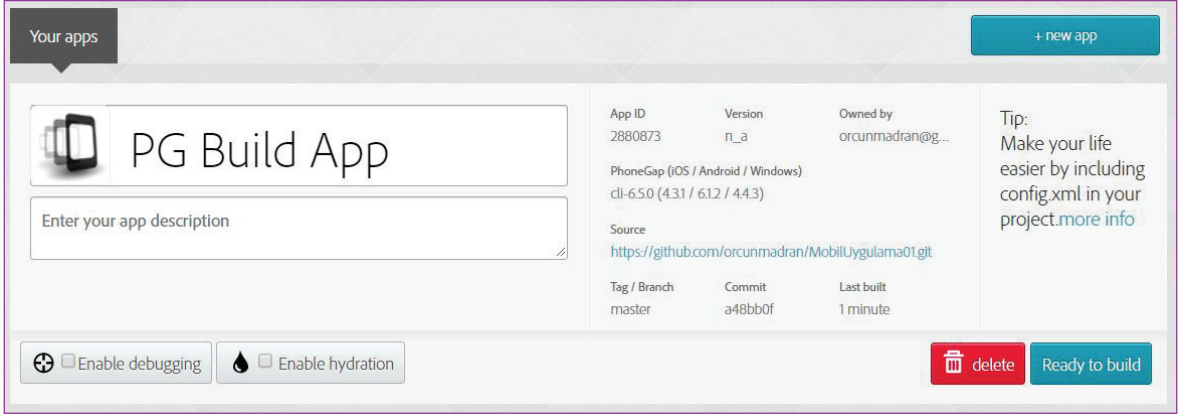
Şekil 57: Bağlantının kurulabilmesi için onay verilmesi

Onay işlemi verildikten sonra PhoneGap Build platformuna otomatik olarak geri döndüğünde “Your GitHub account has been linked successfully.” (GitHub hesabınız ile başarılı bir şekilde bağlantı kuruldu.) mesajı görüntülenir. PhoneGap Build ana sayfasına dönüş yapıldığında “Your apps” (Uygulamalarınız) altındaki “open-source” (açık-kaynak) sekmesine tıklanır ve “find existing repo / paste .git repo” (varolan depoyu bul) açılır kutusundan GitHub’daki ilgili kod deposu seçilir (Şekil 58).



Şekil 58: GitHub kod deposunun seçilmesi

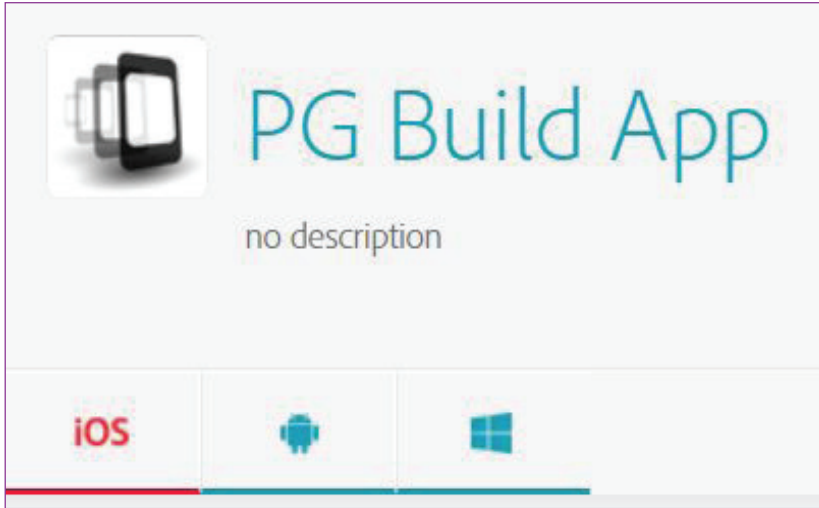
Kod deposunun seçilmesinin ardından “Pull from .git repository” (git deposundan çek) butonuna tıklanarak GitHub’da yer alan uygulama dosyalarının paketleme işlemi için PhoneGap Build’e aktarılması sağlanır. Dosyaların aktarılma işlemi tamamlandıktan sonra uygulama ile ilgili genel bilgilerin yer aldığı sayfa görüntülenir (Şekil 59).



**Şekil 59:** PhoneGap Build'e aktarılan uygulama ile ilgili genel bilgilerin yer aldığı sayfa

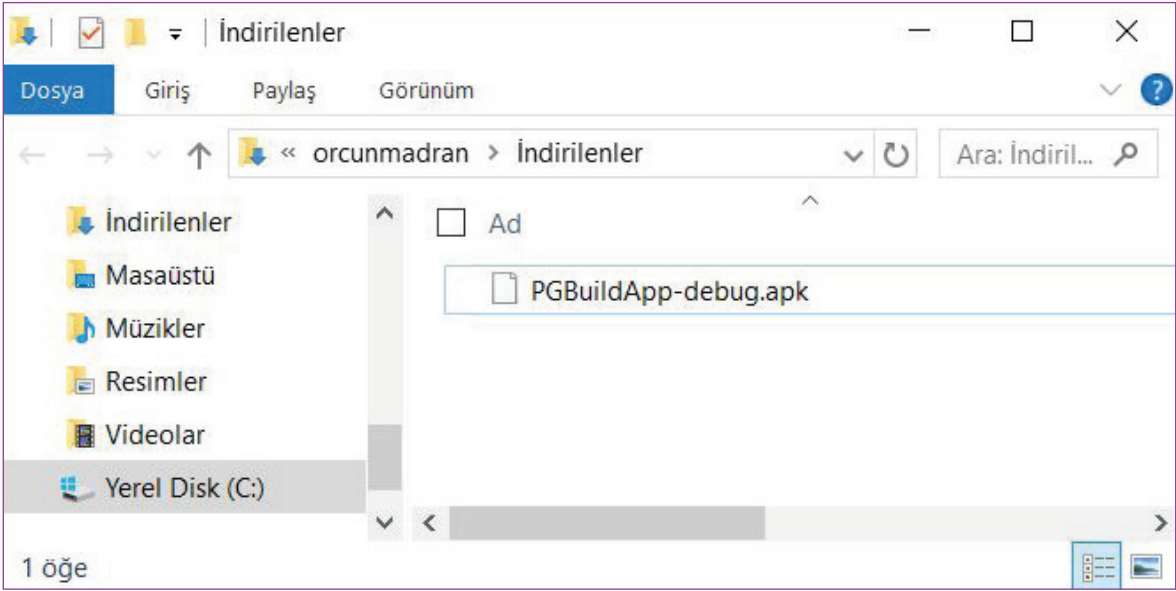
Şekil 59'da görüntülenen sayfa içerisinde uygulamanın adı değiştirilebilir, kısa bir açıklama eklenebilir ve "Ready to build" (Paketlemeye hazır) düğmesine tıklanarak mobil cihaza uygulamanın yüklenmesi için "APK" uzantılı dosya oluşturulabilir.

Paketleme işlemi tamamlandığında Şekil 60'da yer alan platform ikonlarına tıkladığında ilgili platformun kurulum dosyası (ör.: \*.apk) tarayıcı yardımıyla kişisel bilgisayara indirilir.



**Şekil 60:** Kurulum dosyasının indirilmesi

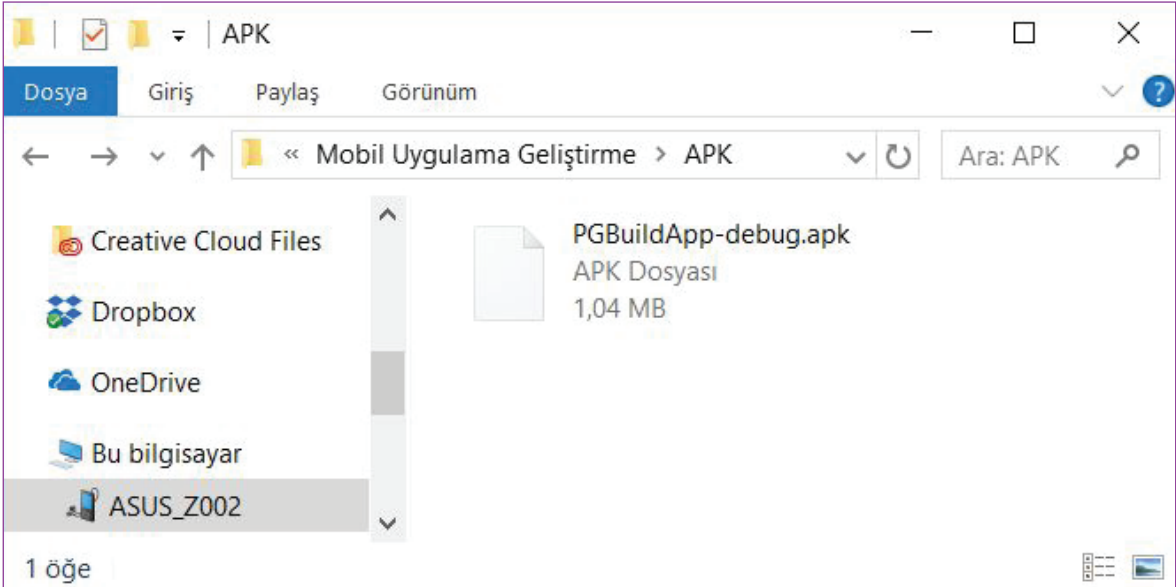
Platform ikonlarından ortada bulunan Android ikonuna tıkladığında "PGBuildApp-debug.apk" kişisel bilgisayara indirilir. Bu dosyayı Dosya Gezgini yardımıyla bulup mobil cihazınızın içine yükleyerek buradan kurulumu gerçekleştirilir ve mobil cihaz üzerinde kullanıma hazır hâle gelir (Şekil 61).



Şekil 61: İndirilen kurulum dosyasına Dosya Gezgini ile erişilmesi

#### 4.2.4. APK Uzantılı Kurulum Dosyasının Mobil Cihaza Aktarılması ve Kurulumun Gerçekleştirilmesi

APK uzantılı kurulum dosyasını mobil cihaza aktarmak için mobil cihazın USB kablosunu kişisel bilgisayara takmak yeterli olacaktır. Bağlantı sağlandıktan sonra APK uzantılı dosya daha sonra mobil cihaz üzerinden erişimin kolay olabileceği bir lokasyona kopyalanır (Şekil 62).

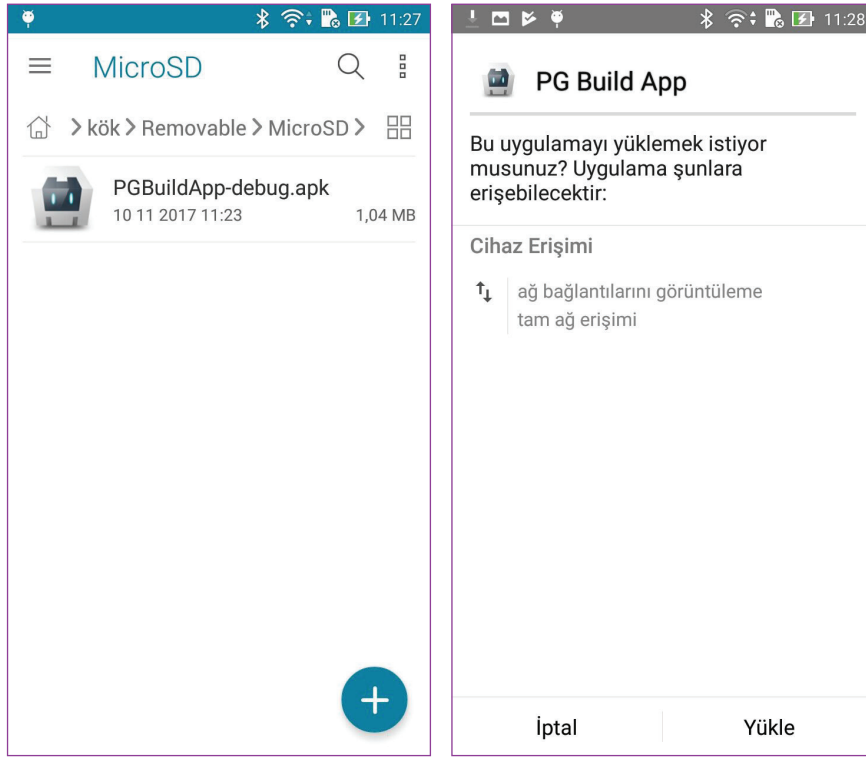


Şekil 62: APK uzantılı kurulum dosyasının mobil cihaz içerisine aktarılması

APK uzantılı kurulum dosyası mobil cihaza aktarıldıktan sonra mobil cihazdaki dosya yöneticisi ya da benzeri bir program yardımıyla dosyaya ulaşılır ve çalıştırılır. Bu aşamada dikkat edilmesi gereken nokta, resmî uygulama marketi dışından bir kurulum dosyasının mobil cihazda çalışması için gerekli izinlerin verilmesi gerekliliğidir.

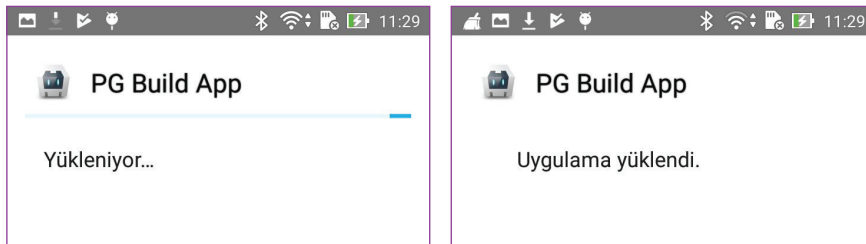
Geliştirici seçenekleri olarak tanımlayabileceğimiz bir dizi özelliğin kontrolüne mobil cihazdaki “Ayarlar” altında yer alan “Geliştirici Seçenekleri” bölümünden ulaşılır.

Kurulum dosyası çalıştırıldığında “Bu uygulamayı yüklemek istiyor musunuz? Uygulama şunlara erişebilecektir:” şeklinde bir onay bölümü mobil cihazda görüntülenir. Ekranın sağ alt tarafında “Yükle” butonuna tıklayarak uygulamanın mobil cihaza kurulumu başlatılmış olur (Şekil 63).



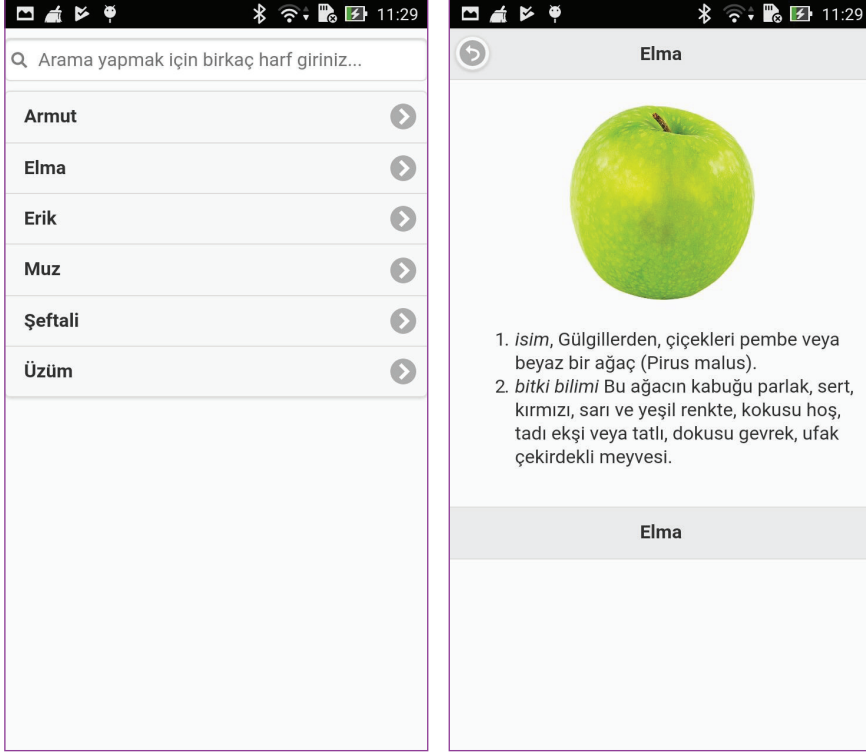
Şekil 63: Kurum dosyasına erişim ve yükleme işleminin başlatılması

Uygulamanın boyutuna ve mobil cihazın işlemci gücüne bağlı olarak yükleme işlemi belli bir süre alacaktır. Yükleme işlemi tamamlandığında “Uygulama yüklendi.” mesajı ekranda görüntülenir (Şekil 64).



Şekil 64: Uygulamanın yüklenme süreci ve yükleme işleminin tamamlanması

Uygulama artık program listesi içerisinde yer almaktadır. Uygulama çalıştırıldığında Şekil 65'teki gibi görüntülenir.



Şekil 65: Mobil uygulamanın cihazda görüntülenmesi

## Faydalanılabilecek Çevrim İçi Kaynaklar

- Adobe PhoneGap ana portalı: <https://phonegap.com/>
- Adobe PhoneGap Build: <https://build.phonegap.com/>
- Adobe PhoneGap Emulate: <http://emulate.phonegap.com/>
- Android platformu ana portalı: <https://www.android.com/>
- Android platformu geliştirici portalı: <https://developer.android.com/>
- Android platformu için tasarım çalışmaları: <https://developer.android.com/design>
- Android platformu için geliştirme çalışmaları: <https://developer.android.com/develop>
- Apache Cordova Projesi: <https://cordova.apache.org/>
- Brackets Metin Editörü: <http://brackets.io/>
- GitHub kod deposu: <https://github.com/>
- HTML5 eğitim ve referans portalı: <https://www.w3schools.com/html/default.asp>
- iOS platformu geliştirici portalı: <https://developer.apple.com/ios/>
- JQuery JavaScript kod kütüphanesi: <http://jquery.com/>
- JQuery Mobile kod kütüphanesi: <http://jquerymobile.com/>
- JQuery Mobile uygulama örnekleri: <http://jquerymobile.com/demos/>
- Windows Phone geliştirici portalı: <https://developer.microsoft.com/en-us/windows>