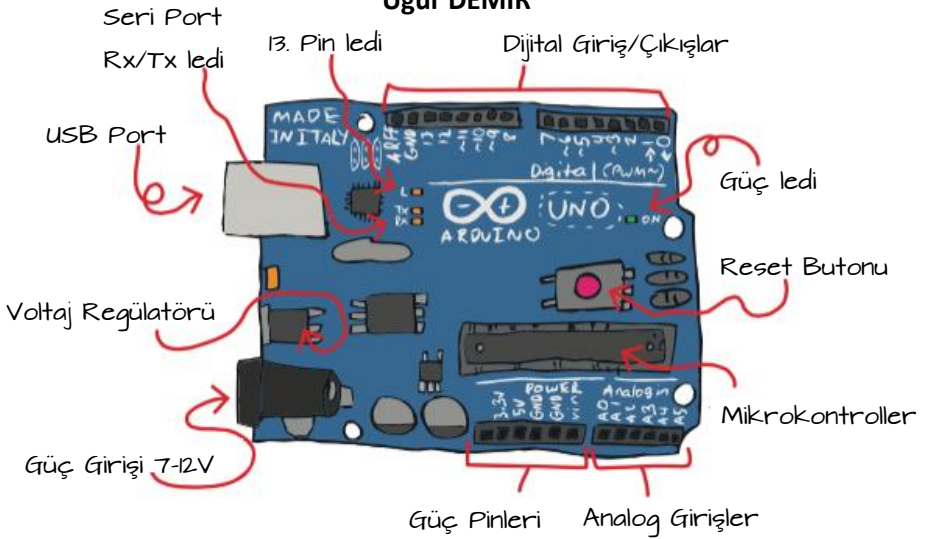


ARDUINO

PROGRAMLAMA KİTABI

Bir parça kod!

Uğur DEMİR



if / else

switch / case

random

analogRead

digitalWrite

interrupt

I2C / SPI

int

Kapak Fotoğrafi: Arduino Uno

Copyright © 2016 Uğur DEMİR.

Bu kitabın yazımı Arduino resmi sitesindeki Reference bölümü referans alınarak yazar tarafından derlenerek yazılmıştır.

Her hakkı saklıdır.

Arduino Programlama Kitabı: Bir parka kod!

Copyright © 2016 Uğur DEMİR

Her hakkı saklıdır.

Yazarla irtibat

x12pfu@gmail.com

Ücretsiz PDF

<http://www.ugrdmr.wordpress.com>

ISBN:-

Uğur Demir

1992 yılında Eskişehir de doğdum. Lise öğrenimimi Türk Telekom Anadolu Teknik Lisesi Elektrik Elektronik Bölümü, lisans eğitimimi Sakarya Üniversitesi Elektrik Elektronik Mühendisliğini başarıyla bitirdim. Yaklaşık bir senedir Sakarya Üniversitesi Teknokent’de özel bir firmada Ar-Ge R & D Mühendisi olarak çalıştım. Bir senedir de Gömülü Sistemler Uzmanı olarak çalışmaktayım. Yaklaşık beş senedir Arduino, Beaglebone ve Raspberry Pi ile çeşitli projeler hayata geçirdim. Blog sitemde yaptığım bazı projeleri açık kaynak kod olarak paylaşıyorum. Open Source açık kaynak kod olarak paylaştığım ilk kitabımdır. Biraz da olsa açık kaynak kodlu platforma faydamız olabilirse ne mutlu.

Teşekkür

Bu kitabın yazımında bana destek olan Mekatronik Mühendisliği okuyan kardeşim Okan KOÇOĞLU’na, bana ilham veren sevdiğim, hayat yoldaşım Sakine KÖKLÜ’ ye ve bana bu fikri aşıl原因an Elektronik Mühendisi, Genel Müdürüm Sayın Orhan NERGİZ’e teşekkürlerimi sunarım.

İçindekiler

1. Giriş	1
1.1. Microkontroller nedir?	2
1.2. Arduino Özellikleri	4
1.3. Arduino Yazılımının Yüklenmesi.....	6
1.4. İlk Program	9
2. Program Yapısı	10
2.1. void setup().....	10
2.2. void loop()	10
2.3. #define.....	12
2.4. #include	12
3. Kontrol Yapısı	13
3.1. if.....	13
3.2. if/else.....	15
3.3. switch/case	16
3.4. while	18
3.5. do/while	20
3.6. break.....	21
3.7. continue.....	22
3.8. return	22
3.9. go to	24
4. Söz Dizimi	24
4.1. Noktalı Virgül (;).....	24
4.2. Süslü Parantez ({}).	25
4.3. Çift Slash(//).	26

4.4.	Yıldızlı Slash(/**/)	26
5.	Aritmetik Operatörler	26
5.1.	Toplama, Çıkarma, Çarpma, Bölme	26
6.	Karşılaştırma Operatörleri	27
6.1.	==(eşit eşit) , != (eşit değil), < (küçük), > (büyük), <= (küçük eşit), >= (büyük eşit)	27
7.	Boolean Operatörleri	28
7.1.	&& (and)	28
7.2.	(or)	30
7.3.	!(not).....	30
8.	Birleşik Operatörler	31
8.1.	++ (arttırma), -- (azaltma)	31
8.2.	+=, -=, *=, /=, %=	31
8.3.	&= (Bitsel Lojik Ve)	32
8.4.	= (Bitsel Lojik Veya)	33
9.	Sabitler	34
9.1.	HIGH LOW	34
9.2.	INPUT OUTPUT	34
9.3.	true false	35
9.4.	integer constants	36
9.5.	U & L	37
9.6.	floating point constants	37
10.	Dönüşümler	37
10.1.	char	37
10.2.	byte	37
10.3.	int	37

10.4. unsigned int	38
10.5. long	38
10.6. float	38
10.7. string.....	38
10.8. substring()	38
11. Değişken kapsamaları	40
11.1. static	40
11.2. volatile	40
11.3. const	40
11.4. PROGMEM	40
11.5. sizeof()	42
12. Dijital Giriş Çıkışlar	43
12.1. pinMode(pin,mod)	43
12.2. digitalWrite(pin,değer)	43
12.3. digitalRead(pin)	43
13. Analog Giriş Çıkışlar	45
13.1. analogRead(pin,mod)	45
13.2. analogWrite(pin,değer)	47
13.3. analogReference(tip)	48
14. Gecikmeler	49
14.1. delay(milisaniye)	49
14.2. unsigned long millis()	49
14.3. delayMicroseconds(mikrosaniye).....	51
15. Matematiksel İşlevler	51
15.1. min(x,y)	51
15.2. max(x,y)	52

15.3. abs(x)	52
15.4. constrain(x,a,b)	52
15.5. map()	53
15.6. pow(a,b)	53
15.7. sqrt(a,b)	53
16. Trigonometri	54
16.1. sin(rad)	54
16.2. cos(rad)	54
16.3. tan(rad)	54
17. Karakterler	54
17.1. isAlphaNumeric()	55
17.2. isAlpha()	55
17.3. isAscii().....	55
17.4. isWhiteSpace().....	55
17.5. isControl()	55
17.6. isDigit()	55
17.7. isGraph().....	55
17.8. isPrintable()	55
17.9. isPunct().....	55
17.10. isSpace()	55
17.11. isUpperCase()	55
17.12. isHexadecimalDigit()	55
18. Seri Haberleşme	58
18.1. Serial.begin(hızı)	58
18.2. int Serial.available()	59
18.3. int Serial.read().....	59

18.4. Serial.flush()	60
18.5. Serial.print(data)	62
18.6. Serial.println(data)	62
19. İnterruptlar (Kesmeler)	62
19.1. interrupts()	62
19.2. noInterrupts()	64
20. Random Sayılar	67
20.1. randomSeed()	67
20.2. random(min,max)	68
21. Gelişmiş Giriş Çıkışlar	68
21.1. tone()	68
21.2. noTone()	69
21.3. shiftOut()	69
22. Kütüphaneler	72
22.1. EEPROM	72
23. Haberleşme Protokolleri	77
23.1. I2C Veri Yolu	77
23.2. SPI Veri Yolu	81
24. Arduino Detaylı Pin Yapısı	85
25. ASCII Kodları	86
Kodlar Listesi	88
Şekiller ve Tablolar Listesi	89

1. Giriş

Arduino Programlama Kitabı – Bir parça kod!

Bu kitabın amacı Arduino Atmel Atmega 328P Mikrodenetleyici kullanarak özellikle Arduino için C programlama dilini öğrenmek için yazılmıştır. C genel amaçlı bir programlama dilidir. Geliştirildiği ortam olan UNIX sistemi ile yakından ilgilidir. Derleyici ve işletim sistemleri yazmak için oldukça kullanışlı olması nedeni ile “sistem programlama dili” olarak adlandırılır.

C iyi düzenlenmiş programların geliştirilebilmesi için gerekli temel akış kontrolü sunar. Bunlar gruplama, karar verme(if-else), olası durumlarından biri dallanma(switch), döngülerde koşul sınamanın başta (for,while) ve döngülerden ani çıkma (break) olarak sayılabilir.

Arduino programlanırken C dilinin yanı sıra asıl beslendiği yer kütüphanelerimizde yazıldığı C++ dilidir. Bizim kullandığımız fonksiyonlar arkada çalışan ve programımıza en başta eklediğimiz kütüphanelerden gelmektedir.

Bu kitapta Arduino için yazılmış özel kütüphanelerde kullanacağımız hazır fonksiyonlar ve C nin temeli anlatılmıştır.

Arduino projelerimizde ister uzaya göndereceğimiz araç yapalım ister akıllı saat ister led yakıp söndürelim programlama yaparken önceliğimiz her zaman basitlik olmalıdır. Parçaları birleştirerek, basitten başlayarak bütüne varmak bir parça kod ile başlayacaktır.

Bir parça kod ile harikalar yaratabiliriz.

Hadi başlayalım.

1.1. Microkontroller nedir?

Mikro kontroller adından da anlaşılacağı üzere mikro kontroller, mikro düzeyde denetleyici olarak tabir edilen programlayarak kontrol edilebildiğimiz aslında dijital bir bilgisayardır.

Bir mikro denetleyici, komple bir bilgisayarın (Merkezi işlem birimi, hafıza ve giriş - çıkışlar) tek bir entegre devre üzerinde üretilmiş halidir. Kısıtlı miktarda olmakla birlikte yeterince hafıza birimlerine ve giriş – çıkış uçlarına sahip olmaları sayesinde tek başlarına çalışabildikleri gibi donanımı oluşturan diğer elektronik devrelerle irtibat kurabilir, uygulamanın gerektirdiği fonksiyonları gerçekleştirebilirler. Üzerinde analog-dijital çevirici gibi entegre devreler barındırmaları sayesinde algılayıcılardan her türlü verinin toplanması ve işlenmesinde kullanılabilirler. Ufak ve düşük maliyetli olmaları gömülü uygulamalarda tercih edilmelerini sağlamaktadır.

Günümüzde mikro işlemci ve mikro denetleyiciler üreten irili ufaklı pekçok firma bulunmaktadır. Bunlara örnek olarak INTEL, MOTOROLA, AMD, PHILIPS, SIEMENS, TEXAS INS., DALLAS, ATMEL, MICROCHIP, HITACHI, MITSUBISHI, SGSTHOMSON, ANALOG DEVICES, NATIONAL gibi firmalar sayılabilir.

Arduino günümüzde en çok tercih edilen bir mikro denetleyicidir. Yüksek seviyeli C dili ile geliştirilen AVR mimarisi ile tasarlanmıştır. Açık kaynak kodlu olması, donanım ve yazılımına kolay bir şekilde ulaşılabilmesi ve pic diğer mikro kontroller gibi çıplak olmayışı programlanabilmesinin bir parça kod ile sağlanması bunun nedenlerindedir.

Bir mikro denetleyicide bulunması gereken özellikler şunlardır:

- Programlanabilir dijital paralel giriş/çıkış.
- Programlanabilir analog giriş/çıkış.
- Seri giriş/çıkış (senkron, asenkron ve cihaz denetimi).
- Motor veya servo kontrol için pals sinyali çıkışı.
- Harici giriş vasıtasıyla kesme.
- Timer vasıtası ile kesme.
- Harici bellek ara birimi.
- Harici bus arabirimi(PC ISA gibi).
- Dâhili bellek tipi (ROM, EPROM, EEPROM).
- Dâhili RAM seçeneği.
- Kayan nokta hesaplaması.

Bir mikro denetleyicide bir komutun işlenme süreci 4 aşamada gerçekleştirilir.

- 1.Alma (Fetch)

Hafızaya yüklenmiş olan program komutlarını alır.

- 2.Kod Çözme (Decode)

Yazmaçtaki komutları kod çözücü(decoder) yardımıyla çözer.

- 3.Uygulama (Execution)

Çözülen komutları uygular ve bu işlemi sürekli tekrar eder.

- 4.İşlemi tamamlama (Complete Process)

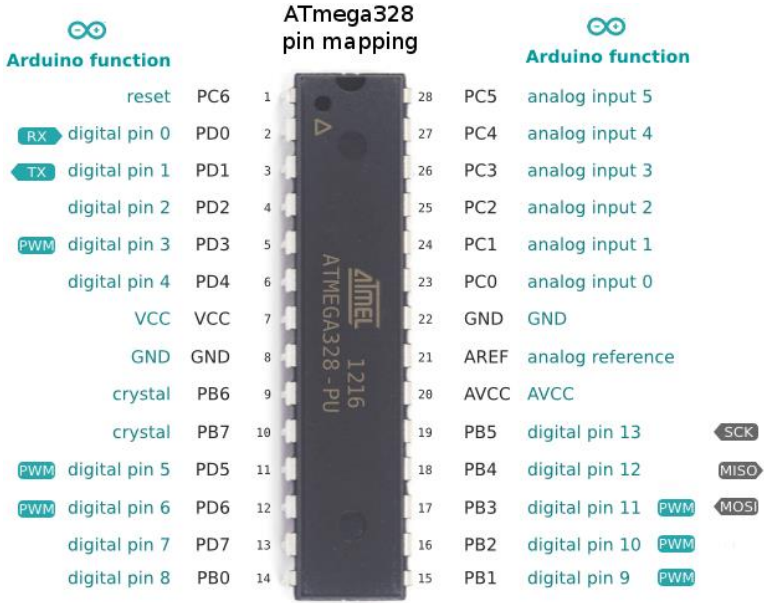
İşlemi tamamlama sürecidir. Bazı komutlarda işlem sonucunu W yada file register'a yazma süreci olarak düşünülmüştür, bazı komutlarda ise bu süreç içerisinde işlem yapılmaz.

1.2. Arduino Özellikleri

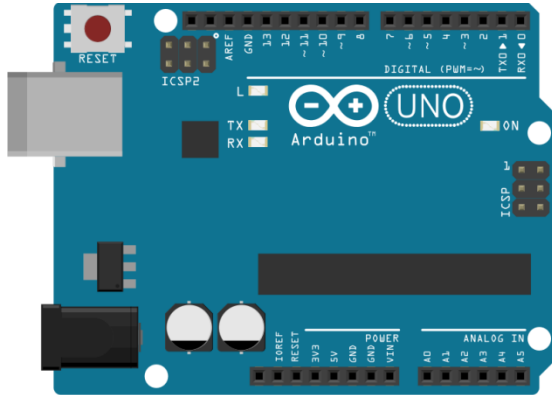
Arduino Uno Atmel Atmega 328P mikro denetleyicisine sahip bir bordurdur. USB bağlantı girişine, güç jak girişine, reset butonuna sahiptir. Bir mikro denetleyicide bulunması gereken her şeye sahiptir.

Mikro denetleyici	Atmega328P
Çalışma gerilimi	5V
Giriş gerilimi (önerilen)	7-12V
Giriş gerilimi (limit)	6-20V
Dijital giriş/çıkış pini	14 adet
PWM giriş/çıkış pini	6 adet
Analog giriş pini	6 adet
Giriş/çıkış pin başına dc akım	20mA
3.3V için DC akım	50mA
Flash bellek	32 KB
Sram	2KB
EEPROM	1 KB
Saat Hızı	16 MHz
Uzunluk	68.6 mm
Genişlik	53.4 mm
Ağırlık	25 g

Şekil- 1.1 – Arduino Uno Özellikleri



Şekil - 1.2 – Atmega 328P Pin isimleri

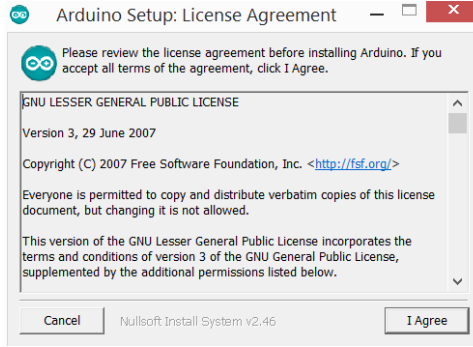


Şekil - 1.3 – Arduino Uno kartı

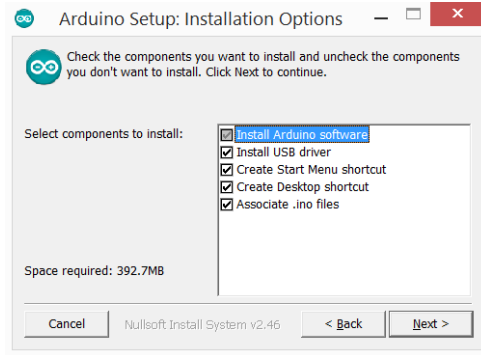
1.3. Arduino Yazılımının Yüklenmesi

Arduino yazılımının yüklenmesi için öncelikle internet tarayıcımıza <https://www.arduino.cc/en/Main/Software> linki girilir. Hangi işletim sistemine sahipsek Windows, Mac OS X, Linux dan birini seçerek indirme işlemini gerçekleştiririz.

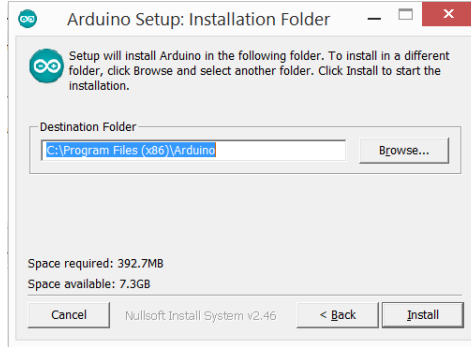
İndirilen *arduino-1.6.7-windows.exe* ye sağ tıklayıp yönetici olarak çalıştır diyelim.



Şekil - 1.4 – Yazılım Yüklenmesi (I Agree)

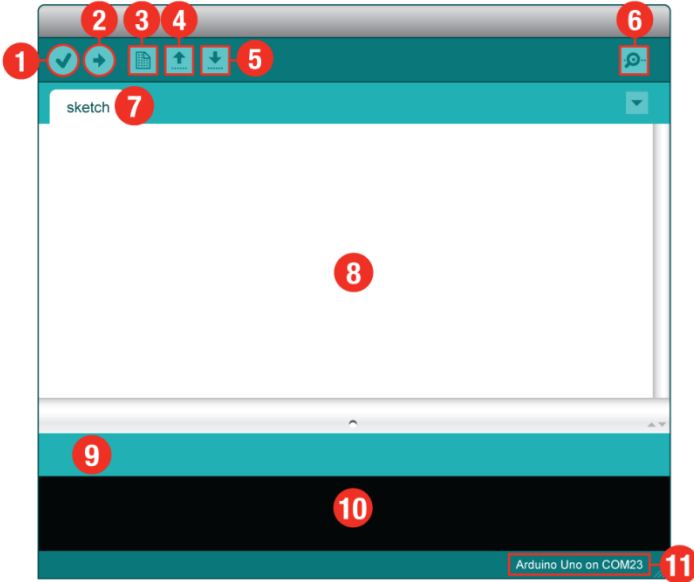


Şekil - 1.5 – Yazılım Yüklenmesi2 (Next)



Şekil - 1.6 – Yazılım Yüklenmesi3 (Install)

Kurulum gerçekleştiikten sonra açılan ekran:



Şekil - 1.7 - Arduino IDE

- 1.Derleme:** Yazdığımız programı derler hataları bulur.
- 2.Yükleme:** Yazdığımız kodu derler, Arduino içine atar.
- 3.Yeni:** Yeni çalışma sayfası açar.
- 4.Açma:** Kayıtlı bir programı açar.
- 5.Kaydetme:** Yazdığımız programı kaydeder.
- 6.Seri Monitör:** Arduino ile seri iletişim yaparak ekran açar.
- 7.Sketch:** Yazdığımız programın dosya ismi.
- 8.Boş alan:** Yazacağımız program alanı.
- 9.Gösterge:** Yaptığı işlemin ilerleme durumunu gösterir.
- 10.Rapor:** Derleme sonucu yapılan hataların veya programımızın yükleme sonrası mikro denetleyicide kapladığı alanı gösterir.
- 11.Gösterge:** Bilgisayarımıza usb ile bağladığımız Arduino'nun bağlandığı portu ve hangi Arduino modeli ile çalışıyorsa onu gösterir.
 - Arduino Programını Türkçe yapmak
 - File / Preferences / Editor language / Türk (Turkish)
 - Arduino Programında ekran numaralarını aktif etme
 - File / Preferences den Display line numbers aktif edin.
 - Arduino Programında Arduino modelini değiştirme
 - Araçlar/ Kart / Çalıştığınız Arduino modeli seçin.
 - Arduino Programında port değiştirme
 - Araçlar / Port / çalıştığınız portu seçin.
 - Arduino Programında kütüphane ekleme
 - Taslak / Include Library / eklemek istediğiniz kütüphaneyi seçiniz.

1.4. İlk Program

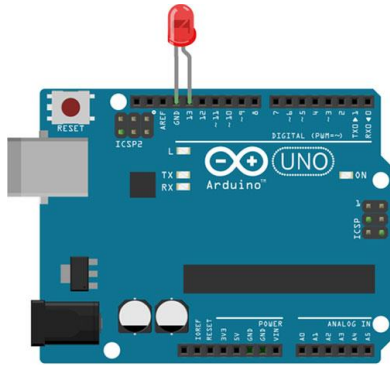
Arduino programımızı yükleyip, usb bağlantılarımızı yaptıktan sonra ilk uygulamamız olan blink, led yakıp söndürme örneğini hiç kod yazmadan yapalım. Öncelikle Arduino programından;

Dosya/ Örnekler/ Basics/ Blink i seçelim derleyerek programı Arduino içine atalım. Led bağlantısı aşağıdaki gibi olacaktır.

```
void setup() {           //ana kurulumlar
  pinMode(13, OUTPUT);   //13. dijital pin çıkış
}

void loop() {           // sonsuz dongu
  digitalWrite(13, HIGH); // led 5v seviyesinde
  delay(1000);           // 1 saniye bekle
  digitalWrite(13, LOW); // led 0v seviyesinde
  delay(1000);           // 1 saniye bekle
}
```

Kod - 1 – Blink



Made with  Fritzing.org

Şekil - 1.8 - Blink Led Devre Şeması

2. Program Yapısı

2.1. void setup()

Setup() fonksiyonu program yüklenilip enerji verildikten veya reset atıldıktan sonra 1 defa çalışır. Bu fonksiyon içine yazdıklarımız pin modları, kütüphaneyi başlatma ve değişkenlerdir.

Örnek

```
int buton = 3; // butonu 3. pine tanımladık

void setup () {

  Serial.begin (9600); // Seri haberleşme

  pinMode (buton, INPUT); // Pin modu tanımladık

}

void loop () {

  }
```

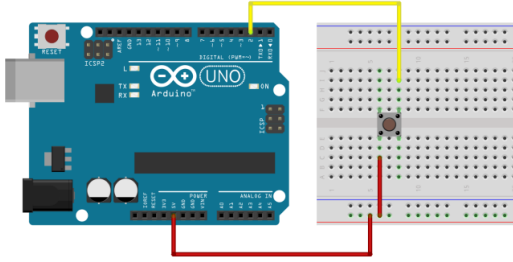
Kod - 2 - Setup

2.2. void loop()

Setup() fonksiyonumuz tamamlandıktan sonra loop fonksiyonumuza geçer ve burada sonsuz döngü içinde yazdığımız programı çalıştırır.

Örnek

Şekildeki devreyi kurup program mantığını anlamaya çalışalım.



Şekil - 2.1 - Loop

```
int buton = 3; // butonu 3. pine tanımladık

void setup() { Serial.begin(9600); // seri haberleşme
  pinMode(buton, INPUT); } // 3. pin giriş oldu

void loop(){
  if (digitalRead(butonPin) == HIGH)
    Serial.write('Basıldı'); // Seri ekrana yaz
  else
    Serial.write('Basıldı');
  delay(1000);
}
```

Kod - 3 - Loop

Program akışını sırayla açıklamak gerekirse;

1. Buton adlı int değişkenimizi 3. pine tanımladık.
2. void setup() içinde pinMode ile 3. pini giriş atadık. Serial.begin ile seri haberleşmeyi açtık.
3. Void loop() ile sonsuz döngü içine girdik. 3. pini dijitalRead ile okuduk.
4. if() ile kontrol ettik. Basıldıysa ekrana Serial.write ile yazdırdık.
5. delay(1000) ile 1 saniye beklettik.
6. delay ile 1 saniye bekledikten sonra program loop içerisinde olduğundan 3. açıklamadan itibaren kendini tekrar edecektir.

2.3. #define

#define ön işlemci komutu olup, bir isim yerine başka bir ismi değişimini sağlar.

Örnek

```
#define ledpin 13;
```

denildiğinde programda ledpin gördüğü yere 13 rakamını yerleştirecektir.

2.4. #include

#include programımız dışındaki kütüphanelere erişmek için kullanılır. Programımızda SPI kütüphanemizi kullanmak ve onun içerisindeki komutlara ulaşmak istediğimizde program başına tanımlamamız 2 şekilde olabilir. (“.”) yada (<.>)

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

3. Kontrol Yapısı

3.1. if

#if Türkçe karşılığı eğer demektir. Mesela eğer butona basıldıysa ledi yak vb. durumlarda veya karşılaştırmalarda kullanılırız.

Örnek

Şekildeki devreyi kurup if yapısını anlamaya çalışalım.

```
int buton = 7; // butonu 3. pine tanımladık

int led = 9; // ledi 9. pine tanımladık

void setup() {

  pinMode(buton, INPUT); // 3. pin giriş oldu

  pinMode(led, OUTPUT); // 9. pin çıkış oldu

}

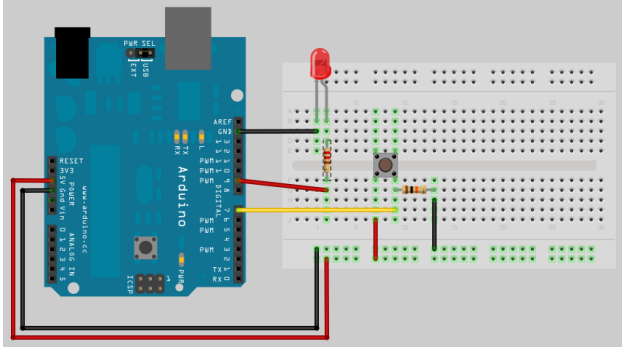
void loop(){      //sonsuz döngü

  if (digitalRead(buton) == HIGH) { //okunan buton 1 ise

    digitalWrite(led, HIGH); // ledi yak

  }

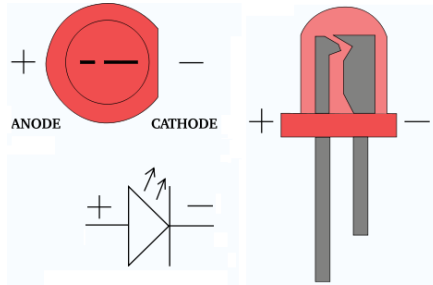
}
```



Şekil - 3.1 – Buton, Led Devre Şeması

Not

Led yapısı



Şekil - 3.2 -Led Diyot Pin Yapısı

3.2. if/else

#if else deęimi kořullu ifade y¼r¼tmek iin kullanılır. İf eęer demektir else, deęil ise demektir. if ve else birlikte kullanılır. Programlama dilinde else tek bařına kullanılamaz.

Örnek

Bir önceki devremizi kurup else yapısını anlayalım.

```
int buton = 7; // butonu 3. pine tanımladık
int led = 9; // ledi 9. pine tanımladık
void setup() {
  pinMode(buton, INPUT); // 3. pin giriş oldu
  pinMode(led, OUTPUT); } // 9. pin çıkış oldu
void loop(){ //sonsuz döngü
  if (digitalRead(buton) == HIGH) { //okunan buton 1 ise
    digitalWrite(led, HIGH);} // ledi yak
  else { // deęil ise
    digitalWrite(led, LOW); }} // ledi söndür
```

Kod - 5 – if/else

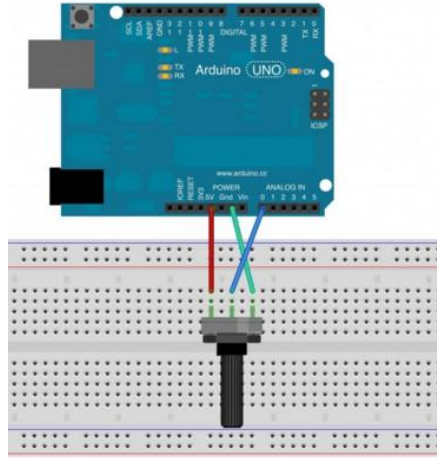
3.3. switch/case

#switch/case bir ifadenin sabit değerlerinden birisiyle eşleşip eşleşmediğini test eden çok yönlü bir karar verme yapısıdır.

Bir switch/case yapısından çıkışı sağlamak ya da sonlandırmak için break yada return kullanılır.

Örnek

Aşağıdaki devreyi kurup switch/case yapısını anlamaya çalışalım.



Şekil - 3.2 – Potansiyometre Devre Şeması

Not

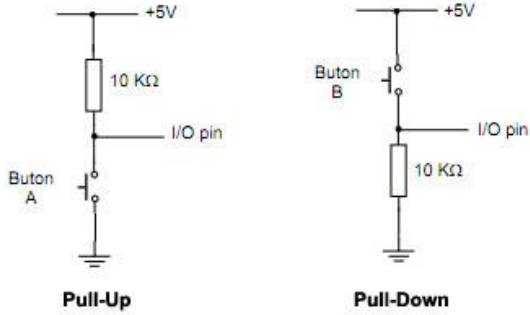
Arduino analogRead fonksiyonu, **ADC (Analog to Digital Converter) 10 bit** olduğu için maximum $2^{10} = 1024$, ve 0 dahil olduğu için de 0-1023 arası değerler alır.

```
void setup() {  
    Serial.begin(9600); //Seri haberleşme  
}  
void loop() {  
    int okunandeger = analogRead(A0);  
  
    //map fonksiyonu okuduğumuz değeri 0-1023 aralığını 2 //ye  
    bölür 0-511 ise menzile 0 değerini verir. 512-1023 //ise  
    menzile 1 değerini verir.  
  
    int menzil= map(okunandeger, 0, 1023, 0, 1);  
  
    switch (menzil) {  
  
    case 0: //Pot döndürülmesi 0-49% arasında ise  
        Serial.println("düşük"); //Seri port ekranına yazdır  
        break; // döngüden çıkış  
  
    case 1: // Pot döndürülmesi 50-100% arasında ise  
        Serial.println("yüksek");  
        break; } // döngüden çıkış  
  
    delay(1); // stabil çalışması için 1 milisaniye bekleme  
}
```

Kod - 6 – switch/case

Not

Pull-up dirençler; elektronik devrelerde lojik sistemlere girişlerin (input), eğer dışarıdan bağlı cihazların bağlantısı kesildiyse, umulan lojik seviyelerde kalmalarını sağlamada kullanılır.



Şekil - 3.3 - Pull Up/ Pull Down Dirençler

3.4. while

```
while (ifade) {
```

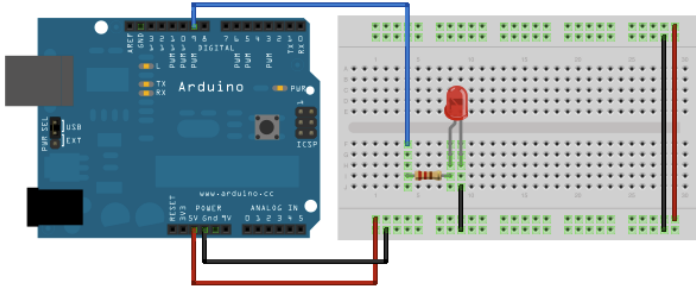
```
// İfadeler
```

```
}
```

#while (ifade) döngüsü ifadenin doğru olduğu durumlarda iki süslü parantez içindeki ifadeleri baştan aşağı çalıştırır, tekrar başa döner çalıştırır. Sonsuz döngü içine girer.

Örnek

Aşağıdaki devrede while döngüsüne girecek ledi yakacak, 1 saniye bekleyip ledi söndürecek ve ifadeyi 1 arttıracaktır. While döngüsünde 49 kere aynı işlemi yaptıktan sonra tekrar while döngüsüne girmeyecektir.



Şekil 3.23 – Led Devre Şeması

```
int ifade=0;
int led=0;
void setup() { }

void loop() { while (ifade<50)

digitalWrite(led, HIGH); // ledi yak

delay(1000); //1 saniye bekle
digitalWrite(led, LOW); // ledi söndür
ifade=ifade+1;
}
```

3.5. do/while

#do-while döngüsü öncelikle do parantez içindeki işlemi yapar ve while ile yapılan işlemi kontrol eder.

Örnek

Aşağıdaki devrede do döngüsüne girecek sayıyı beş arttıracak, seri ekranımıza yazdıracak while ile durumu kontrol edecektir. Sayımız 25 ten büyük olduğu zaman döngüden çıkacaktır.

```
void setup() {  
  int sayi = 0;  
  
  Serial.begin(9600); // seri haberleşme  
  
  do { // 25 e kadar 5 er sayma  
    sayi = sayi+ 5;  
  
    Serial.print("sayi = ");  
  
    Serial.println(sayi);  
  
    delay(500);          // 500ms bekleme  
  } while (sayi< 25);  }  
  
void loop() {  
  }  
}
```

Kod – 8 – do/ while

3.6. break

Break do, for, ve while döngülerinden döngü çalışması bittiğinde döngü dışına çıkmak için kullanılır. Switch case yapısında da kullanılır.

Örnek

Aşağıdaki programda sensör değeri 300 den büyükse döngü dışına çıkarılır.

```
for (x = 0; x < 255; x ++)  
{  
    analogWrite(PWMPin, x);  
    sens = analogRead(sensorPin);  
    if (sens > 300){  
        x = 0;  
        break;  
    }  
    delay(50);  
}
```

Kod – 9 – break

3.7. continue

Continue do, for, while döngülerinde bir satırın, işlem yapılmadan geçilmesini istediğimiz durumlarda kullanılır.

```
for (x = 0; x < 255; x ++)  
{  
    if (x > 40 && x < 120){  
        continue; // x değeri 120den büyük ve  
        120 den küçük ise bu alanda bir şey yapma  
    }  
}
```

Kod – 10 – continue

3.8. return

Return bir fonksiyon sonlandırılmak istenirse return ile döndürülecek değer belirtilir. Returnun ikinci bir kullanımı da belli bir yerden sonra kodlar çalışmasın istiyorsak return kullanırız.

Örnek

Aşağıdaki program parçasında okuduğumuz sensör değeri 250 den büyük ise 1 değerini döndürür.

250 den küçük ise sıfır değerini döndürür.

```
int sensorkontrol () {  
    if(analogRead(0) > 250) {  
        return1 ;  
    }  
    else  
        return 0;  
}
```

Kod – 11 – return1

2.kullanım yolu:

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

Kod – 12 – return2

3.9. go to

Go to program akışını istediğimiz yöre yönlendirir.

```
void loop() {  
    int x = analogRead(0);  
    if (x < 200) {  
        goto burdandevam;  
    }  
    burdandevam:  
    delay(1000);  
    // istenilen kod  
}
```

Kod – 13 – go to

4. Söz Dizimi

4.1. Noktalı Virgül (;)

C program dilinde her satır programından sonra noktalı virgül konulması gerekir. Noktalı virgül konulmadığı yerlerde derleme yaparken hata verir.

4.2. Süslü Parantez ({})

Fonksiyonlarda, döngülerde ve koşullu ifadeleri bildirirken süslü parantez kullanılır. İç içe olan fonksiyonlarda en dıştaki süslü parantezin en baştaki fonksiyona ait olduğu dikkat edilmelidir aksi takdirde derleme yaparken hata verecektir.

Örnek

```
void fonksiyonum()  
{  
    //yapılacaklar }  
while () {  
    //yapılacaklar }  
do {  
    //yapılacaklar  
}  
for (x=0;x<=10;x++)  
{  
    //yapılacaklar }
```

4.3. Çift Slash(//)

Program satırından, program başında ve ya herhangi bir yerde açıklama yapmak istiyorsak çift slash kullanarak yaparız. Çift slash dan sonra yazılanlar program kodu olarak alınmaz.

4.4. Yıldızlı Slash(/**/)

*/** Buraya yazdıklarım satır, sütun dahil program olarak alınmayacak açıklama olarak alınacaktır. **/*

5. Aritmetik Operatörler

5.1. Toplama, Çıkarma, Çarpma, Bölme

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

Örnek

```
int x,y,i,r;
```

```
y = x + 3;
```

```
x = x - 7;
```

```
i = j * 6;
```

```
r = r / 5;
```

Kod – 15 – Toplama, Çıkarma, Çarpma, Bölme

6. Karşılaştırma Operatörleri

6.1. ==(eşit eşit) , != (eşit değil), < (küçük), > (büyük), <= (küçük eşittir), >= (büyük eşittir)

Genellikle if içerisinde karşılaştırma yaparken kullandığımız operatörlerdir.

```
if (degisken > 50)
{
    //değişken 50 den büyükse buraya girer.
}
```

Örnek

```
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)
```

Kod - 16- Eşit eşit, Eşit değil, Küçüktür, Büyüktür, Küçük eşittir, Büyük eşittir.

7. Boolean Operatörleri

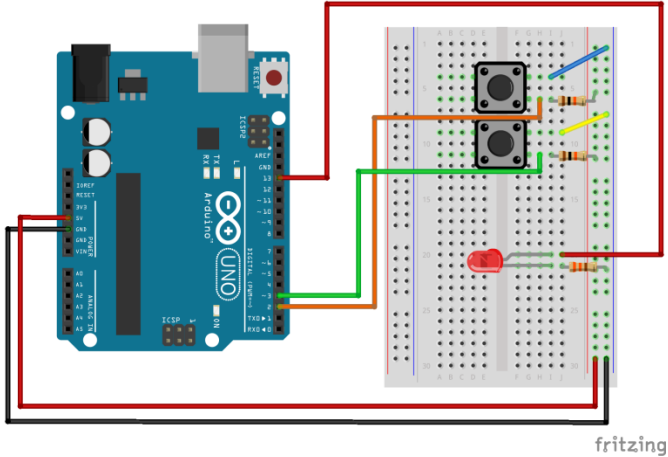
Boolean ifadeleri genellikle if yapısının içerisinde ve şart gerektiren durumlarda kullanılır.

7.1. && (Mantıksal Ve)

Her iki işlem de doğru ise if şartı sağlanır ve if içerisindeki komut çalıştırılır. Herhangi bir durum ya da ikisi de false, yanlış sonuç ise if yapısına girilmez.

Örnek

Aşağıdaki devreyi kurup her iki butona basıldığında ledin yanmasını sağlayalım. Programda 2 buton ve 1 led tanımlayıp her iki butona basılması durumunda oku1 ve oku2 HIGH olacağından if bloğunun içine girer ve ledi yakar.



Şekil 7.1 – Led Buton Devre Şeması

```
int buton1 = 2; // butonu 2. pine tanımladık

int buton2 = 3; // butonu 3. pine tanımladık

int led = 13; // ledi 13. pine tanımladık

void setup() {

  pinMode(buton1, INPUT); // 2. pin giriş oldu

  pinMode(buton2, INPUT); // 3. pin giriş oldu

  pinMode(led, OUTPUT); // 13. pin çıkış oldu

}

void loop(){ //sonsuz döngü

int oku1= digitalRead(2);

int oku2= digitalRead(3);

if (oku1 == HIGH && oku2 == HIGH) {

  digitalWrite(led, HIGH); // ledi yak

}

}
```

7.2. || (Mantıksal Veya)

Her iki işlemden herhangi birisi doğruluk şartını taşıyorsa if yapısının içine girer.

Yukarıdaki örneğe göre herhangi butonlardan birisine basıldığında ledin yanmasını istiyorsak veya kullanmalıyız.

```
if (oku1 == HIGH || oku2 == HIGH) { }
```

7.3. ! (Mantıksal Değil)

Değer olarak verilen ifadenin sıfır olma durumudur. İfadenin sıfır olma şartı sağlanıyor ise if yapısının içine girer.

Buton1 e basılmıyor ise ledi söndürür.

Örnek

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

Kod – 18 – Mantıksal Değil

8. Birleşik Operatörler

8.1. ++ (arttırma), -- (azaltma)

Artırım veya azaltma yapmamızı sağlar.

++x veya x++ kullanımı: $x=x+1$ ile aynıdır yani maç sayıyı 1 arttırmaktır.

--x veya x--kullanımı: $x=x-1$ ile aynıdır yani maç sayıyı 1 arttırmaktır.

y = ++x //ilk işlem yapar sonra atama.
y = x++ //ilk atama yapar sonra işlem.

8.2. += , -= , *= , /= , %=

x += y; // $x = x + y$;

x -= y; // $x = x - y$;

x *= y; // $x = x * y$;

x /= y; // $x = x / y$;

x %= y; // $x = x \% y$; //mod alma

**Mod alma x in y ile bölümünden kalan sayıdır. $x=10\%3$; dersek $x=1$ olacaktır.

Örnek

```

x = 1; olsun
x += 4; // x=5 olur
x -= 3; // x=2 olur
x *= 10; // x =10 olur
x /= 2; // x=5 olur.
x %= 5; // x=0 olur

```

Kod – 19 – += , -= , *= , /= , %=

8.3. &= (Bitsel Lojik Ve)

İşleme giren bitlerin ve sini verir. Yani eğer her iki işlenenin *i* inci bitleri 1 ise, sonucun *i* inci biti de 1 olur.

0 0 1 1 operand1

0 1 0 1 operand2

0 0 0 1 (operand1 & operand2) – sonuç

Örnek

```
int a = 92; // İkili: 000000001011100
int b = 101; // İkili: 000000001100101
int c = a & b; // 000000001000100 veya ondalık
68.
```

Kod – 20 – Bitsel Lojik Ve

8.4. |= (Bitsel Lojik Veya)

İşleme giren bitlerin ve sini verir. Yani eğer her hangi işlenenin i inci bitlerinden birisi 1 ise, sonucun i inci biti de 1 olur.

0 0 1 1 operand1

0 1 0 1 operand2

0 1 1 1 (operand1 | operand2) – sonuç

Operator	Adı
$a \& b$	And (ve)
$a b$	Or (veya)
$a \wedge b$	Xor (dışarmalı veya)
$\sim a$	Not (değil)
$n \ll p$	left shift (sola kayma)
$n \gg p$	right shift (sağa kayma)
$n \ggg p$	right shift (sağa p hane kayma)

Tablo – 8.1 – Bitsel Operatörler

9. Sabitler

9.1. HIGH | LOW

Okuma veya yazma yaparken dijital pine verilen aktif veya/pasif durumudur. HIGH ile çıkışı aktif etmiş oluruz yani 5 V, LOW ile pasif yaparak 0 V vermiş oluruz.

```
int led= 13;
```

```
digitalWrite(led, HIGH); // ledi yak
```

```
digitalWrite(led, LOW); // ledi söndür
```

9.2. INPUT | OUTPUT

Pin modumuzun giriş mi? çıkış mı olacağını belirleriz. Sensör okurken giriş, yada çıkış tanımlamamız bizim için yararlı olur. Setup fonksiyonumuz içinde tanımlarız.

Örnek

```
int led=13;

int buton=5;

void setup(){

  pinMode(led, OUTPUT); // Çıkış tanımladık.

  pinMode(buton, INPUT); // Giriş tanımladık. }
```

Kod – 21 – INPUT,OUTPUT

9.3. true | false

Arduino da doğru yada yanlış göstermek için kullanılan mantıksal tanımlamadır.

false 0 (sıfır) olarak tanımlanır.(LOW)

true 1 olarak tanımlanır.(HIGH)

Örnek

```
int a = true; // a doğru
int b = false; // b yanlış
int led=13; // led 13. pine tanımladık
int buton=10; // butonu 10. pine
void setup() { } //ana kurulum
void loop() { // sonsuz döngü
if (buton == a) { // butona basıldı mı?
digitalWrite(led, HIGH); // led e 5v ver
delay(1000); //1 saniye bekle
digitalWrite(led, LOW); } //ledi söndür
}
```

Kod – 22 – true / false

9.4. integer constants

Sayı sistemleri için kullanılır.

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

Tablo – 9.1 – Sayı Sistemleri

Decimal kullandığımız 10luk sayı sistemidir.

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

Binary ikili sayı sistemidirç 0 ve 1 vardır.

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

Octal sekizlik sayı sistemidir. 0 dan 7 ye kadar.

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

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 edilir.

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

9.5. U & L

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

100000L , 32767ul, 33u gibi kullanımları vardır.

9.6. floating point constants

Kayan nokta sabitleri olarak nitelendirilir. Sayıları daha okunabilir hale getirmek için kullanılır. “E” ve “e” olarak kabul edilir.

67e-12 = $67,0 * 10^{-12}$ 0,000000000067dir.

10. Dönüşümler

10.1. char

```
char myChar = 'A';
```

```
char myChar = 65; // eşdeğeri
```

10.2. byte

```
b = B10010; // "B" binary formatı
```

(B10010 = 18 ondalık)

10.3. int

```
int led= 3;
```

10.4. unsigned int

```
unsigned int led= 3;
```

10.5. long

```
long sensor = 186000L;
```

10.6. float

```
float sensorkalibre = 1.117;
```

10.7. string

```
char Str2[6] = {'d', 'e', 'n', 'e', 'm', 'e'};
```

```
char Str6[15] = "deneme";
```

10.8 substring()

String içerisindeki kelimedenden kaç karakter alacağımızı belirtir.

```
String cumle = "Bir parca kod";  
    if (cumle.substring(3,9) == "parca") {  
        //  
    }
```

Kod – 23 – substring()

Veri Türü	Açıklama	Bellek Boyutu
boolean	Mantıksal veri türü (1 yada 0)	1 byte
char	Tek karakteri içeren veri türü	1 byte
byte	0-255 arası pozitif sayı saklar.	8bit=1byte
int	Tam sayıları saklar. İşaretli: (-32,768 ile 32,767) İşaretsiz: (0 ile 65535)	2 byte
unsigned int	Tam sayıları saklar. (0 ile 4294967295)	4 byte
long	4 byte lik işaretli tam sayıları saklar. (-2147483648ile+2147483647)	4 byte
unsigned long	4 byte lik işaretli tam sayıları saklar. (0 ile 4294967295)	4 byte
float	4 byte lik işaretli ondalık sayıları saklar.	4 byte
double	İşaretli ondalık sayılar $5.0 * 10^{-324}$ - $1.7 * 10^{308}$	8 byte
string	Char veri türünden olan karakterler toplamıdır.	Her eleman 1 byte
array	Her birine indeks numarası ile ulaşılan aynı türdeki veri topluluğudur.	Değişken

Tablo – 10.1 – Dönüşümler Tablosu

11. Değişken kapsamaları

11.1. static

Static tanımlanan değişken bellekte tutulur ve daha sonra kullanmak istersek yeniden oluşturulmaz bellekten çağırılır.

```
static int sayi=0;
```

11.2. volatile

Volatile ile tanımlanan değişkenler Arduino' nun ram bölgesine kaydedilir. Kullanmak istersek direk ram bellekten okunur. Volatile kullanıyorsak değişkenin değerini interrupt ile değiştirmemiz gerekir.

```
volatile int led=LOW;
```

11.3. const

Const ile oluşturulan değişken sabitlenir ve değeri daha sonradan değiştirilemez.

```
const float fi= 1.61; //Altın oran
```

```
const float pi= 3.14; // Pi sayısı
```

11.4. PROGMEM

Bilgiyi SRAM bellek yerine Flash bellekte depolamaya yarar. Arduino da programımızda uzun char yazmaya kalkarsak SRAM sorun çıkarabilir. Bu yüzden çok uzun metinleri Flash belleğe kaydederiz.

Kullanabilmemiz için programa `#include <avr/pgmspace.h>` kütüphanesini eklemeliyiz.

Örnek

```
#include <avr/pgmspace.h>

const char signMessage[] PROGMEM = {"UZUN
CÜMLELER BU SEKILDE FLASH BELLEGE
KAYDEDILIR...."};

int k=0;

char myChar;

void setup() {

    Serial.begin(9600);

    while (!Serial);

    int len = strlen_P(signMessage);

    for (k = 0; k < len; k++)

    {

        myChar = pgm_read_byte_near(signMessage + k);

        Serial.print(myChar);    }

    Serial.println(); }

void loop() {

}
```

Kod – 24 – PROGMEM Kullanımı

11.5. sizeof()

Size of operatörü değişkenin kaç bayt olduğunu verir.

Örnek

```
char mesaj[] = "test denemesi";  
  
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); }
```

```
0 = t  
1 = e  
2 = s  
3 = t  
4 =  
5 = d  
6 = e  
7 = n  
8 = e  
9 = m  
10 = e  
11 = s  
12 = i
```

Kod – 25 – sizeof

Şekil – 11.1 – sizeof çıktısı

12. Dijital Giriş Çıkışlar

12.1. pinMode(pin,mod)

INPUT, OUTPUT mod olarak, hangi pinin giriş yada çıkış olacağını tanımladığımız komuttur.

```
pinMode(3, OUTPUT); // Çıkış tanımladık.
```

```
pinMode(4, INPUT); // Giriş tanımladık.
```

12.2. digitalWrite(pin,değer)

HIGH, LOW değer olarak, hangi pinin aktif yada pasif olacağını tanımladığımız komuttur.

```
digitalWrite(13, HIGH); //13. pin aktif
```

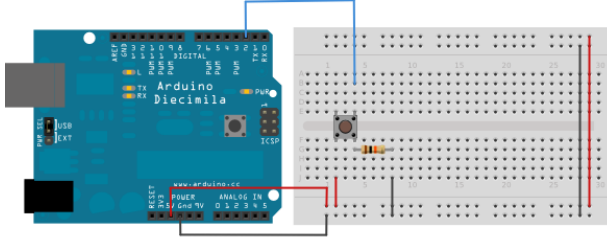
```
digitalWrite(13, LOW); //13. pin pasif
```

12.3. digitalRead(pin)

Dijital pinin aktif mi pasif mi olduğunu yani HIGH mı LOW mu olduğunu belirleriz.

Örnek

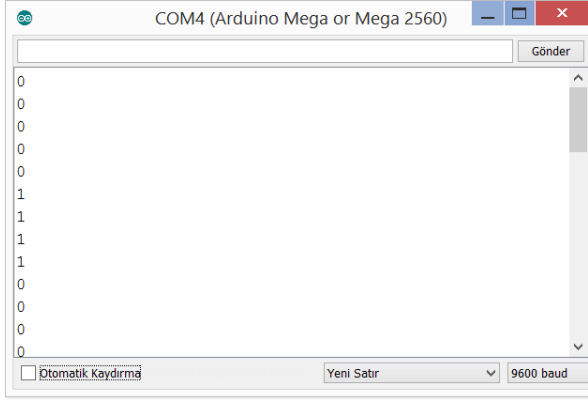
Butona basılıp basılmadığını seri port ekranında görelim. 0 basılmadı 1 basıldı.



Şekil -12.1 - Led Devre Şeması

```
int buton = 2; //dijital 2 ye tanımladık.  
  
void setup() {  
  Serial.begin(9600); //Seri haberleşme hızı  
  pinMode(buton, INPUT); //buton giriş oldu  
}  
  
void loop() {  
  int butondurumu = digitalRead(buton);  
  Serial.println(butondurumu);  
  delay(1);  
}
```

Kod – 26 – digitalRead



Şekil – 12.1 – Program çıktısı

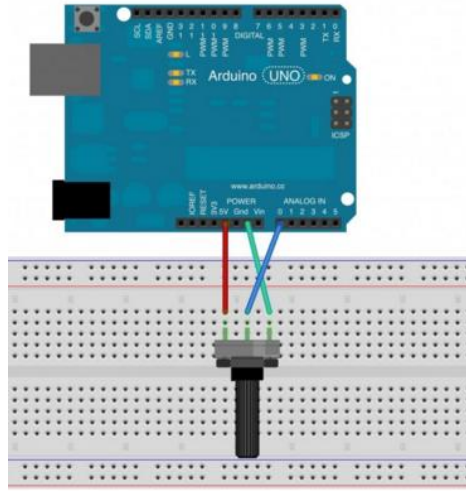
13. Analog Giriş Çıktılar

13.1. analogRead(pin,mod)

Arduino Uno'da 6 adet, Mini ve Nano'da 8 adet ve Mega'da 16 adet 10 bit analog to digital converter vardır. Yani analog bir girişi dijitale çevirerek okuruz. 10 bit olması hassasiyeti gösterir. $2^{10}=1024$, buda demek oluyor ki bir ölçümü 0-1023 parçaya böler ve bize hassasiyetliği sağlar.

Örnek

Potansiyometre ile düşürdüğümüz gerilimi ölçelim 10K pot kullanabiliriz.



Şekil -12.1 - Potansiyometre Devre Şeması

```
void setup() {  
    Serial.begin(9600);  
}  
void loop() {  
    int sensor = analogRead(A0);  
    float volt = sensor * (5.0 / 1023.0);  
    Serial.println(volt);  
}
```

Kod – 27 – analogRead

13.2. analogWrite(pin,değer) - PWM

Değişen led ışığı ya da çeşitli hızlarda küçük motor sürmek için kullanılır.

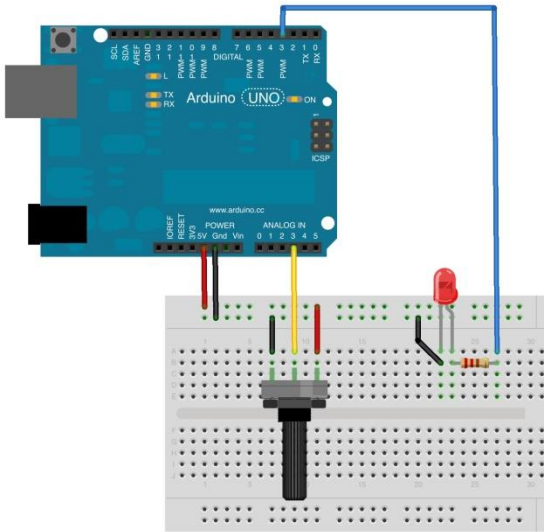
Arduino Uno'da bu komut 3, 5, 6, 9, 10. pin

Arduino Mega'da 11. 2-13 ve 44 - 46. pin

Arduino Due 2 den 13. pine kadar destekler.

Örnek

Potansiyometre ile led parlaklığını ayarlayalım.



Şekil -12.1 – Potansiyometre - Led Devre Şeması


```
int led = 3;

int analogPin = 3;

int okunan= 0;

void setup(){

  pinMode(led, OUTPUT); }

void loop(){

  int okunan = analogRead(analogPin);

  analogWrite(led, okunan / 4);

}
```

Kod – 28 – analogWrite

13.3. analogReference(tip)

Analog giriş için kullandığımız referans voltajını yapılandırır. Analog girişten 5V geliyorsa 10 bit için okuduğumuz 11 1111 1111'dir. Decimal olarak 1023'e tekabül eder.

5V için;

$(5V - 0V) / 1023 = 0,00488'$ dir.

3.3V için;

$(3.3V - 0V) / 1023 = 0,00322'$ dir.

- DEFAULT: olarak Arduino bordlarında 5V ya da 3,3V tur.
- INTERNAL: 1,1 volt Atmega168 ya da Atmega 328'de geçerlidir. Mega hariç.
- INTERNAL1V1: 1.1V sadece Arduino Mega.
- INTERNAL2V56: 5.56V sadece Arduino Mega.
- EXTERNAL: Bordun üzerinde bulunan AREF pin (0 - 5V) referans kullanılabilir.

14. Gecikmeler

14.1. delay(milisaniye)

Program akışına milisaniye cinsinden bekletme verir. 1 saniyelik gecikmeye ihtiyacımız var o zaman delay(1000); kullanırız. Şunu unutmamalıyız ki delay Arduino'yu tamamen durdurur, delay ileri seviye programlarda tavsiye edilmeyen bir durumdur. Örneğin bir butonumuzun 1'er saniye aralıklarla yanıp sönsün aynı zamanda butona basılıp basılmadığını kontrol etmek istersek edemeyiz. Program sağlıklı olmaz. Bunu çözmek için interruptlar veya timerlar kullanılır.

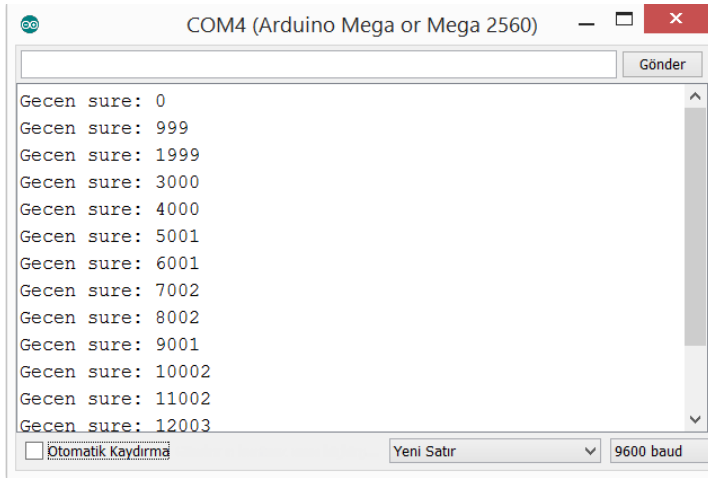
14.2. unsigned long millis()

Arduino programı yüklenilip enerji verildikten sonra süre saymaya başlayan komuttur. Maximum 50 gün sayabilir bu süre sonunda taşma ile sıfırlanır.

Örnek

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

Kod – 29 – millis



Şekil – 14.1 – millis Program Çıktısı

14.3. delayMicroseconds(mikrosaniye)

Program akışına mikrosaniye cinsinden bekletme verir.

Örnek

```
int led = 8;           // digital pin 8 giriş

void setup(){         //ana kurulum

  pinMode(led, OUTPUT); // led çıkış tanımı

}

void loop(){          //sonsuz döngü

  digitalWrite(led, HIGH); //led yansın

  delayMicroseconds(50) ;// bekle

  digitalWrite(led, LOW); // led sönsün

  delayMicroseconds(50); // bekle

}
```

Kod – 30 – delayMicroseconds

15. Matematiksel İşlevler

15.1. min(x,y)

X ve y sayısal değerlerinden en küçük olanı seçer.

enkucuk=min(20,30); //en küçük 20 olarak belirler.

15.2. max(x,y)

X ve y sayısal değerlerinden en büyük olanı seçer.

```
enbuyuk=max(20,30);
```

```
//en büyük 30 olarak belirlir.
```

15.3. abs(x)

Bir sayının mutlak değerini hesaplar. Sayı sıfırdan küçükse pozitif değerini, sıfırdan büyükse aynı sayıyı verir.

```
int k= -20;
```

```
int m= 20;
```

```
deger=abs(k);
```

```
//değer 20 olarak belirlenir.
```

```
deger=abs(m);
```

```
//değer 20 olarak belirlenir.
```

15.4. constrain(x,a,b)

x: herhangi türdeki bir sayı

a: alt aralık

b: üst aralık

x değeri a ile b arasında ise çıktımız x olacaktır. Eğer alt aralıktan küçükse çıktımız a, üst aralıktan büyükse çıktımız b olacaktır.

```
sensor = constrain(sensor, 10, 150);
```

// Sensor değeri 10 ile 150 arasında kalır. 10 dan küçükse çıktımız 10, 150 den büyükse 150 olur. 10 ile 150 arasında bir değer ise o değer çıktımız olur.

15.5. map()

Sensor değerimizi belli aralıkta tutarak istediğimiz aralığa dönüştürmemizi sağlar.

Diyelim ki bizim sensörden okuduğumuz değer 0 ile 1023 arasında ama bize 0 ile 255 arası değer gerekiyor bunun için;

```
int sonuc=map(sensor,0,1023,0,255);
```

kullanırız.

15.6. pow(a,b)

Bir sayının karesini almak için kullanılır. A taban b ise üst dür.

```
sonuc= pow(3^2)
```

//sonuç 9 olacaktır.

15.7. sqrt(a,b)

Bir sayının karekökünü almak için kullanılır. A taban b ise üst dür.

```
sonuc= sqrt(25)
```

//sonuç 5 olacaktır.

16. Trigonometri

Trigonometrik işlemleri yaptırabilmemiz için;

#include "math.h" kütüphanesini kullanmamız gerekir.

16.1. sin(rad)

Radyan bir açının sinüsünü hesaplar. Sonuç 0 ile -1 arasında olacaktır.

16.2. cos(rad)

Radyan bir açının kosinüsünü hesaplar. Sonuç 0 ile -1 arasında olacaktır.

16.3. tan(rad)

Radyan bir açının tanjantını hesaplar. Sonuç 0 ile -1 arasında olacaktır.

17. Karakterler

17.1. isAlphaNumeric()

Bilindiği gibi 0 - 9 decimal sayıları göstermek için çeşitli binary kodları kullanılır. Alfabetik ve nümerik karakterlerin elde edilmesinde büyük sayılar kullanılır (binary). Bu kodlar alphanumeric bazen de alphameric kodlar olarak ifade edilir. Karakterin alphanumeric olup olmadığını kontrol eder.

17.2. isAlpha()

Karakterin alpha olup olmadığını kontrol eder.

17.3. isAscii()

Karakterin Ascii tablosundaki değerini verir.

17.4. isWhiteSpace()

Bir karakter bir boşluk olup olmadığını kontrol eder.

17.5. isControl()

Karakterin kontrol karakteri olup olmadığını kontrol eder.

17.6. isDigit()

Karakterin dijital karakter olup olmadığını kontrol eder.

17.7. isGraph()

Karakterin grafik karakter olup olmadığını kontrol eder.

17.8. isPrintable()

Yazdırılabilir bir karakter olup olmadığını kontrol eder.

17.9. isPunct()

Karakterin noktalama işareti olup olmadığını kontrol eder.

17.10. isSpace()

Karakterin boşluk olup olmadığını kontrol eder.

17.11. isUpperCase()

Bir karakter mi bir harf mi olup olmadığını kontrol eder.

17.12. isHexadecimalDigit()

Bir karakter geçerli bir onaltılık rakam olup olmadığını kontrol eder.

Örnek

```
void setup() {
  Serial.begin(9600); //Seri haberleşme
  while (!Serial) {
    ; // seri port haberleşmesinin beklenmesi
  }
  Serial.println("Bir bayt yazın ve analizini görelim:");
  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("Alphanumeric Degeri");
    }
    if (isAlpha(thisChar)) {
      Serial.println("Bu Alfabetik!");
    }
    if (isAscii(thisChar)) {
      Serial.println(" Bu ASCII karakter.");
    }

    if (isWhitespace(thisChar)) {
      Serial.println(" Bu whitespace");
    }
  }
}
```

```
}  
if (isControl(thisChar)) {  
    Serial.println("Bu kontrol karakteri");  
}  
if (isDigit(thisChar)) {  
    Serial.println("Bu numeric digit");  
}  
if (isGraph(thisChar)) {  
    Serial.println("Bosluk degil yazdirilabilir karakter");  
}  
if (isLowerCase(thisChar)) {  
    Serial.println("Daha 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("Ust durum ");  
}  
if (isHexadecimalDigit(thisChar)) {  
    Serial.println("Gecerli hexadecimaldigit var (i.e. 0 - 9, a - F,  
or A - F)");  
}  
Serial.println();  
Serial.println("Baska bir karakter girin:");  
Serial.println();  
}  
}
```

```
Gonderilen: '1'  ASCII Degeri: 49
Alphanumeric Degeri
  Bu ASCII karakter.
Bu numeric digit
Bosluk degil yazdirilabilir karakter
Yazdirilabilir
Gecerli hexadecimaldigit var (i.e. 0 - 9, a - F, or A - F)

Baska bir karakter girin:
```

Şekil – 17.1 – Karakter Analizi Çıktısı

18. Seri Haberleşme

18.1. Serial.begin(hız)

Bilgisayar ile arduino arasında seri iletişimi başlatmak için void setup() fonksiyonu altında seri iletişim açılır. Serial.begin(9600); buradaki 9600 alınan ve gönderilen bilgilerin, verilerin hızıdır. Arduino tarafına 9600 baud rate yazdıysak bilgisayar tarafında da aynı baud rate ile çalışmalıyız. Çünkü veri kaybı yaşarız.

300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, ve 115200 baud ratelerden birini seçebiliriz.

Seri iletişim halindeyken arduinodaki rx, tx, yani dijital 0, 1 pinleri kullanılamaz.

Örnek

Arduino Mega için 3 adet seri portu olduğundan;

Serial.begin(9600); Serial1.begin(9600);

Serial2.begin(9600); Serial3.begin(9600); kullanılabilir.

```
void setup(){  
    Serial.begin(9600);  
    Serial1.begin(38400);  
    Serial2.begin(19200);  
    Serial3.begin(4800);  
    Serial.println("Test");  
    Serial1.println("Serial 1");  
    Serial2.println("Serial 2");  
    Serial3.println("Serial 3");  
}  
void loop() {  
}
```

Kod – 32 – Serial.begin

18.2. int Serial.available()

Veri gelmeye başladığı zaman seri portu okumaya açar.

18.3. int Serial.read())

Seri datalar gelmeye başladığı zaman o dataları okur.

Örnek

```
int gelenbyte = 0; // gelen data

void setup() {

    Serial.begin(9600); // seri haberleşme hızı

}

void loop() {

    //data gelmeye başladığında

    if (Serial.available() > 0) {

        // bytları oku

        incomingByte = Serial.read();

        //ekrana yaz

        Serial.print("Denildi ki: ");

        Serial.println(gelenbyte, DEC);    }

}
```

Kod – 33 – Serial.read

18.4. Serial.flush()

Giden seri iletim datalarının tamamlanmasını bekler.

Örnek

```
int x = 0; // variable
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.print("NO FORMAT"); // prints a label
  Serial.print("\t"); // tab boşluk verir.
  Serial.print("DEC"); //DEC cinsten yazar.
  Serial.print("\t");
  Serial.print("HEX"); //HEX cinsten yazar.
  Serial.print("\t");
  Serial.print("OCT"); //OCT cinsten yazar
  Serial.print("\t");
  Serial.print("BIN"); //BIN cinsten yazar.
  Serial.print("\t");

  for(x=0; x< 64; x++){ // 0 dan 64 e sayar.

    Serial.print(x); //x değerini yazar
    Serial.print("\t");

    Serial.print(x, DEC); //x in DEC cinsini yazar.
    Serial.print("\t");

    Serial.print(x, HEX); //x in HEX cinsini yazar.
    Serial.print("\t");

    Serial.print(x, OCT); //x in OCT cinsini yazar.
    Serial.print("\t");

    Serial.println(x, BIN); //x in BIN cinsini yazar.

    delay(200); //200 ms bekleme
  }
  Serial.println(""); return;
}
```

Kod – 34 – Serial.print

NO	FORMAT	DEC	HEX	OCT
1	1	1	1	1
2	2	2	2	10
3	3	3	3	11
4	4	4	4	100
5	5	5	5	101
6	6	6	6	110
7	7	7	7	111
8	8	8	10	1000
9	9	9	11	1001
10	10	A	12	1010
11	11	B	13	1011
12	12	C	14	1100

Şekil – 18.1 – Serial.print Ekran Çıktısı

18.5. Serial.print(data)

Data yazılan yere çift tırnak içerisinde yazdığımız yazıları ya da değişkenleri yazar.

18.6. Serial.println(data)

Data yazılan yere çift tırnak içerisinde yazdığımız yazıları ya da değişkenlerin sonuna satır sonu ekler böylece alt satıra gönderecektir.

19. İnterruptlar (Kesmeler)

19.1. interrupts()

İnterruptlar **donanım interruptu** ve **timer interruptu** olmak üzere ikiye ayrılır. İnterrupt olayını şöyle düşünebiliriz. Arduino her bir saniye aralıklarla butonu yakıp söndürüyor. Bunu

yaparken aynı zamanda butona basılıp basılmadığını kontrol etmek istiyorsak interrupt kullanırız. Elektronikte ve arduino da önemli bir konudur.

Donanım Interruptu:

Bord	Dijital Pin Interrupt
Uno, Nano, Mini, other 328-based	2, 3
Mega, Mega2560, MegaADK	2, 3, 18, 19, 20, 21
Micro, Leonardo, other 32u4-based	0, 1, 2, 3, 7
Zero	Tüm pinler, 4 hariç
Due	Tüm pinler

Tablo – 18.1 – Interrupt pin Tablosu

`attachInterrupt(pin, fonksiyon, mod) ;`

Pin, hangi interrupt pinini kullandığımız.

Foksiyon, interrupt tetiklendiğinde yapılacak işlemler fonksiyonu.

Mod, inerruptun nasıl tetikleneceğidir.

Modlar:

-LOW: İnterrupt pininin Low seviyede olmasıyla

-CHANGE: İnterrupt pininin değişimi ile 0v dan 5v a geçişi ya da 5 den sıfıra geçişi.

-RISING: İnterrupt pininin 0v dan 5v a geçmesi, yükselen kenar.

-FALLİNG: İnterrupt pininin 5v den 0v geçmesi, düşen kenar.

-**HIGH**: İnterrupt pininin 5v da olması

Kullanımı:

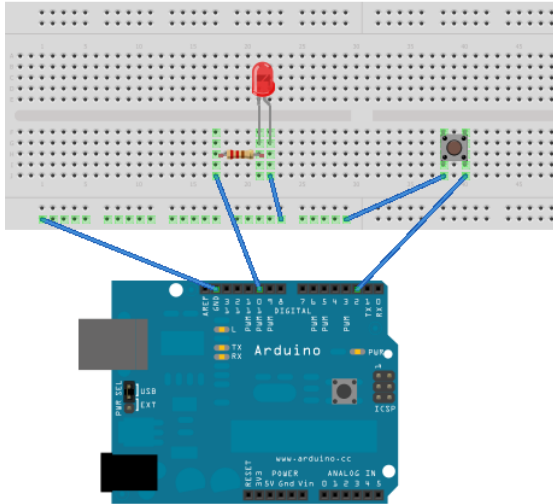
attachInterrupt(2, kontrol, RISING) ;

Yukarıdaki kullanımda 2 ile hangi pinin interrupt olacağını, RISING ile yükselen kenarda tetiklenmesi gerektiğini, tetiklendiğinde kontrol fonksiyonuna gitmesi gerektiğini belirttik.

19.2. noInterrupts()

Kesmeleri devre dışı bırakır.

Örnek



Şekil -19.1 – Buton- Led Devre Şeması

```
int pin = 10;

volatile int durum = LOW;

void setup()
{ pinMode(pin, OUTPUT);
  digitalWrite(2, HIGH);
  attachInterrupt(0, kontrollet, FALLING);
}

void loop()
{ }

void kontrollet()
{
  if(durum == LOW) {
    durum = HIGH;
  } else {
    durum = LOW;
  }
  digitalWrite(pin, durum);
}
```

Kod – 35 – Donanım Interrupt

Timer Interruptu:

Zamanlayıcı interruptları olarak adlandırılır. Kendi kendilerine bir süre sonunda tetikleme yapabiliriz. Örneğin bir ledi 1 er saniye aralıklarla yakıp söndürelim ve her 3 saniyede bir başka bir ledi yakıp söndürmek yada başka bir buton durumunu kontrol etmek istersek timer interruptlarını kullanırız. Timer' lar, counter register'ler sayinde sayaç olarak çalışırlar ve programdan bağımsızdır. Timerler sayaç olarak çalışması için osilatör kullanırlar. Osilatör kare dalga üretir. Arduino Uno 16MHz osilatör kullandığı için periyot süresi 63ns dir.

Timer interruptu kullanmak için hazır kütüphane olan TimerONE kütüphanesini kullanarak basit bir şekilde kullanabiliriz.

Örnek

```
#include <TimerOne.h>

void setup()    {

    pinMode(13, OUTPUT);

    Timer1.initialize(100000); // 1 saniyede
    tetiklenmesi

    Timer1.attachInterrupt( kontrol );      }

void loop() { }

void kontrol()

{    digitalWrite( 13, digitalRead( 13 ) ^ 1 ); }
```

20. Random Sayılar

20.1. randomSeed()

Rasgele sayı üretilirken bir Seed değeri alınır. Bu algoritmalarla uzun bir sayı listesi hesaplanır. Seed belirtilmezse şu anki zamanı alır ve sayılar hep aynı sırada random üretilir.

Örnek

0 ile 300 arasında rasgele sayı üretecektir.

```
long rasgelesayi;  
  
void setup(){  
  randomSeed (analogRead (0));  
  Serial.begin(9600);  
}  
  
void loop(){  
  rasgelesayi = random(300);  
  Serial.println(rasgelesayi);  
  delay(1000);  
}
```

Kod – 37 – Random Sayı Üretme

20.2. random(min,max)

randomSeed() gibi parantez içine 2 adet sayı yazılır minimum ve maximum olacak şekilde, minimum ve maksimum belirtilen sayılar arasında random rasgele sayı üretir.

Örnek

10 ile 20 arasında rasgele sayı üretecektir.

```
long rasgelesayi;

void setup()    { Serial.begin(9600); }

void loop(){

  rasgelesayi = random(10,20);

  Serial.println(rasgelesayi);

  delay(1000);

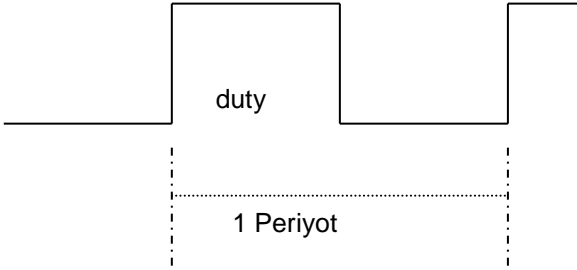
}
```

Kod – 38 – Random Sayı Üretme2

21. Gelişmiş Giriş Çıkışlar

21.1. tone()

Kare dalga üretmemize yarar. İstenilen frekansta ve istenilen pini ayarlayabiliriz. Verilen sinyal %50 duty saykıla sahiptir. Duty saykıl 1 periyot boyunca HIGH ve LOW kalma süresidir.



Şekil - 21.1 – Kare Dalga Sinyali

21.2. noTone()

Kare dalga üretimini sonlandırmamıza yarar.

```
noTone(pin);
```

21.3. shiftOut()

74HC95 etegresi için yazılmış özel bir koddur.

```
shiftOut(dataPin, clockPin, LSBFIRST, led);
```

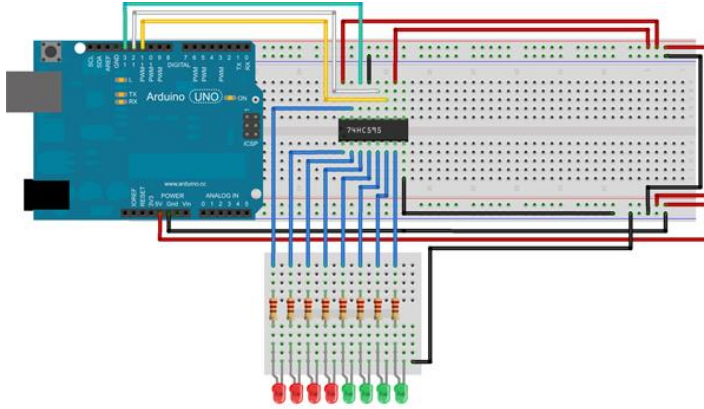
-dataPin: data pinini belirler.

-clockPin: clock pinini belirler.

-LSBFIRST: en küçük bitten başlamasıdır.

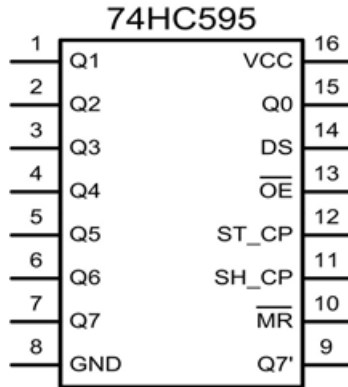
-led: Bayt değişkenidir.

Örnek



Şekil - 21.2 – 74HC95 Entegresi İle Led Devresi

74HC95 entegresi bir tür registerdendir ve registerler içerisinde lojik kapılarını ve FF(flip-flop) devrelerini içlerinde barındırırlar.



Şekil - 21.3 – 74HC95 Entegresi Pin Yapısı

```
int latchPin = 12;

int clockPin = 11;

int dataPin = 13;

byte led = 0;

int currentLED = 0;

void setup()

{ pinMode(latchPin, OUTPUT);

  pinMode(dataPin, OUTPUT);

  pinMode(clockPin, OUTPUT);

  led = 0; }

void loop() {

  led = 0;

  if (currentLED == 7) { currentLED = 0; }

  else {

    currentLED++; }

  bitSet(led, currentLED);

  digitalWrite(latchPin, LOW);

  shiftOut(dataPin, clockPin, LSBFIRST, led);

  digitalWrite(latchPin, HIGH);

  delay(250); }
```

Kod – 39 – shiftOut

22. Kütüphaneler

22.1. EEPROM

EEPROM kütüphanesi **Electronically Erasable Programmable Read-Only Memory** , elektroniksel olarak silinebilir programlanabilir, ve sadece okunabilir bellek demektir. Kalıcı hafıza olarak tabir edebiliriz. Arduino da elektrik gitse bile bazı önemli bilgileri eepromda saklayabiliriz. Hafıza boyutu olarak Arduino' nun farklı modellerine göre farklı boyutlara sahiptir.

Atmega328(kalsik arduino) :1024 byte

Atmega168, Atmega8 :512 byte

Atmega1280, Atmega2560 :4096byte (1KB)

hafıza alanlarına sahiptir.

EEPROM hafızasına yazacağımız maksimum integer sayı olarak, Arduino Uno 8 bit ve $2^8=256$ olduğundan 0-255 arası maximum 255 sayısını girebiliriz. Biz bir EEPROM hafızasına 280 sayısı yazamayız. Bunun çözümü olarak sayıyı çözümleyerek birlik, ondalık ve yüzlük olarak ayırarak her bir ondalığı farklı dijittlere yazarız ve sonra program içinde çözerek o sayıyı elde ederiz. Çözüm yollarından birisi budur.

Arduino Uno kullanıyorsak 1024 byte olduğundan 1024 e kadar hafıza hücrelerine yazabiliriz.

Fonksiyonlar

- **EEPROM.read (adres)**

Belirlediğimiz adresteki EEPROM baytını okur.

Örnek

Aşağıdaki örneği yükleyip Seri porttan EEPROM hücre değerlerimizi 0 dan 1023 e kadar okuyalım.

```
#include <EEPROM.h> // EEPROM kütüphanemiz
int hucre= 0;          //hücremiz 0 dan
baslasın
int veri;              //veri tanımladık

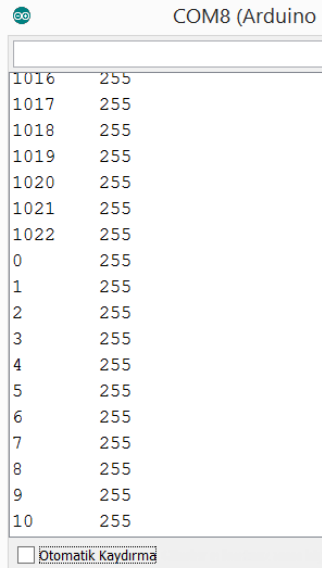
void setup(){
  Serial.begin(9600); //seri haberleşme hızımız
}

void loop() {
  veri = EEPROM.read(hucre); //eeprom hücrelerini oku
  veriye ata

  Serial.print(hucre); //hücreyi yaz
  Serial.print("\t");
  Serial.print(veri); //veriyi yaz
  Serial.println();

  hucre = hucre + 1; //hücreyi 1 arttır.

  if (hucre == 1024) //hücre 1024 olunca sıfırla
    hucre = 0;
  delay(500); //500 ms bekle
}
```



1016	255
1017	255
1018	255
1019	255
1020	255
1021	255
1022	255
0	255
1	255
2	255
3	255
4	255
5	255
6	255
7	255
8	255
9	255
10	255

Şekil - 22.1 – EEPROM Ekran Çıktısı

- **EEPROM.write ()**

Belirlediğimiz adresteki EEPROM' a istediğimiz veriyi yazar. Yaklaşık 3.3 ms yazma süresi vardır.

Örnek

```
#include <EEPROM.h>

void setup() {

  for (int i = 0; i < 255; i++)

    EEPROM.write(i, i);  }

void loop() {  }
```

Kod – 41 – EEPROM.write

- **EEPROM.update()**

Belirlediğimiz adresteki EEPROM' a istediğimiz veriyi yazarken eğer o adreste aynı veri kayıtlıysa yazmaz. Farklı bir veri kaydetmişsek o zaman üzerine yazar.

- **EEPROM.get ()**

Herhangi bir veri yada nesneyi okumamıza yarar.

- **EEPROM.put ()**

Herhangi bir veri tipini EEPROM a yazar.

- **EEPROM[adres]**

Bu operatör bir dizi tanımlayıcı gibi, EEPROM hücrelerini okuma, yada yazma olarak kullanılabilir.

Örnek

Random sayı üretip, sırayla EEPROM' a yazıp okuyalım.

```
#include <EEPROM.h>

int randomsayi;

void setup()      {

    Serial.begin(9600);

    randomSeed(analogRead(0)); }      //referans

void loop() {

    Serial.println("Random sayi bekleniyor");

    for (int i = 0; i < 1024; i++){ //i yi 1024 e kadar arttır.

        randomsayi=random(255); // 0-255 arası random sayi üret.

        EEPROM.write(i, randomsayi); //0 dan 1024 hücreesine kadar yaz.

        Serial.println();

        for (int a=0; a<1024; a++) //1024 e kadar tek tek arttır.

            { randomsayi = EEPROM.read(a); //a ıncı hücreyi oku randomsayiya ata

                Serial.print("EEPROM hucresi: ");

                Serial.print(a);

                Serial.print(" hafizaya alinan: ");

                Serial.println(randomsayi);

                delay(25); }

    }
```

23. Haberleşme Protokolleri

23.1. I2C Veri Yolu

I²C **Inter-Integrated Circuit** veri yolu en az sayıda pin kullanan, iki yönlü bir haberleşme protokolüdür. Hızları 400kbps e kadar çıkmaktadır. Herhangi bir shield yana sensörler arası dijital haberleşmeyi sağlar.

Haberleşme iki kablo ile sağlandığı için “Two-Wire” olarak adlandırılabilir. Kablolardan birisi data, diğeri clock kablosudur.

- Data kablosu (SDA) dir.
- Clock kablosu (SCL) dir.

Arduino da *I²C* kullanabilmemiz için;

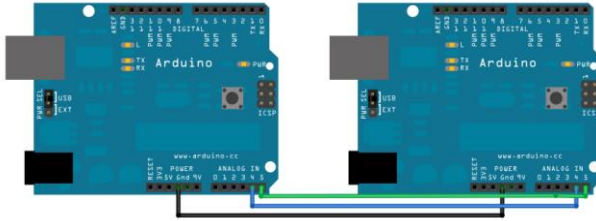
`#include <Wire.h>` kütüphanesini eklememiz gerekir. İletişimin gerçekleşebilmesi için cihazlardan birisi master , diğeri slave olmalıdır. Bunu biz kodlarla belirliyoruz.

Bord	<i>I²C</i> pinleri
Uno, Ethernet	A4 (SDA), A5 (SCL)
Mega2560	20 (SDA), 21 (SCL)
Leonardo	2 (SDA), 3 (SCL)
Due	20 (SDA), 21 (SCL), SDA1, SCL1

Tablo – 23.1 – I2C Haberleşme Pin Yapısı

Örnek

İki arduino arası i2c protokolü ile veri alışverişi yapalım.

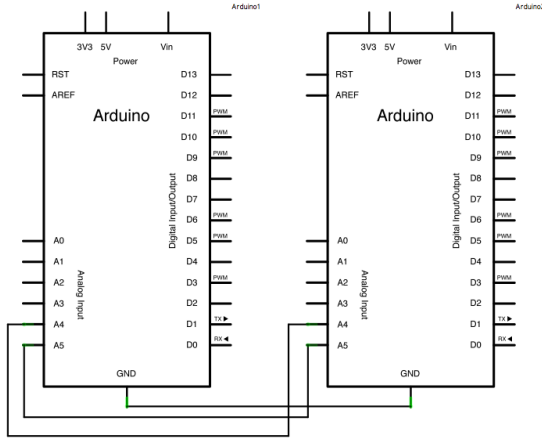


Şekil - 23.1 – I2C Haberleşme Bağlantısı 1

A4-A4 e,

A5-A5 e,

GND-GND ye.



Şekil - 23.2 – I2C Haberleşme Bağlantısı 2

Master kodu:

```
#include <Wire.h>

void setup() {
    Wire.begin(); // i2c haberleşmesi başlatma
}

byte x = 0;

void loop() {
    Wire.beginTransmission(8); // 8. cihaza ilet
    Wire.write("x is ");      // 5 byte gönderme
    Wire.write(x);            // 1 baye gönderme
    Wire.endTransmission(); // iletişimi durdur.

    x++; // x i 1 arttır.

    delay(500);
}
```


Slave kodu:

```
#include <Wire.h>

void setup() {
  Wire.begin(8);          // i2c haberleşmesini 8. adresle başlat
  Wire.onReceive(receiveEvent); // alıcı modda ol
  Serial.begin(9600);     // serial haberleşmeyi aç
}

void loop() {
  delay(100); }

void receiveEvent(int gelen) {
  while (1 < Wire.available()) { // gelen bayte varsa
    char c = Wire.read(); // gelen baytı oku c ye ata
    Serial.print(c);      // karakteri ekrana yaz
  }

  int x = Wire.read(); // gelen integeri oku
  Serial.println(x);    // seri porta yaz
}
```

Kod – 44 – I2C Slave

23.2. SPI Veri Yolu

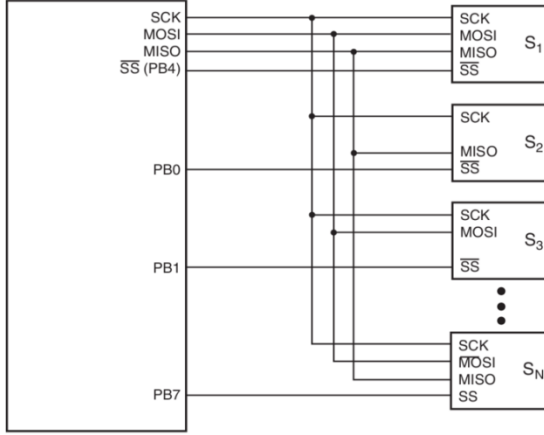
Serial Peripheral Interface (SPI) kısa mesafelerde hızlı veya birden fazla çevresel cihazlar ile iletişim kurmak için mikroişlemci tarafından kullanılan bir senkron seri veri protokolüdür. Ayrıca, iki mikroişlemcisi arasındaki iletişim için kullanılabilir. Motorola tarafından gerçekleştirilmiştir. Yüksek hızlara çıkabilir.

SPI bağlantısı için 4 adet pin gereklidir. Bunlar:

- **MISO** (Master In Slave Out) - master veri göndermek için Slave hattı,
- **MOSI** (Master Out Slave In) - çevre birimleri veri göndermek için ,
- **SCK** (Serial Clock) - master tarafından oluşturulan veri aktarımını senkronize etmek için saat darbesi
- **SS** (Slave Select) - Belirli aygıtları etkinleştirme ve devre dışı bırakmak için kullanabileceğiniz pin.

Bord	MOSI Pini	MISO Pini	SCK Pini	SS (Slave)	SS (Master)
Uno	11	12	13	10	
Mega	4	1	3	53	
Leonardo	4	1	3		
Due	4	1	3		4,10,52

Tablo – 23.2 – SPI Haberleşme Pin Yapısı

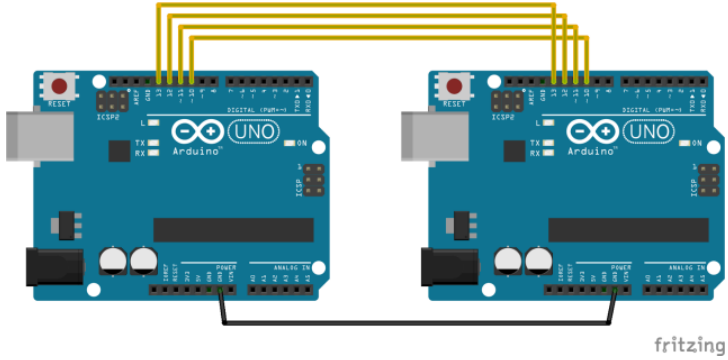


Şekil - 23.3 – SPI Haberleşme Bağlantısı 1

SPI veri yolunda iletişimi her zaman master başlatır.

Örnek

İki arduino arası SPI protokolü ile veri alışverişi yapalım.



Şekil - 23.4 – SPI Haberleşme Bağlantısı 2

Master kodu:

```
#include <SPI.h>

void setup (void)
{ digitalWrite(SS, HIGH);

  SPI.begin ();

  SPI.setClockDivider(SPI_CLOCK_DIV8);
}

void loop (void)
{
  char c;

  digitalWrite(SS, LOW); // ss pin 10

  for (const char * p = "deneme!\n" ; c = *p; p++)

    SPI.transfer (c);

  digitalWrite(SS, HIGH);

  delay (1000);
}
```

Kod – 45 – SPI Master

Slave kodu:

```
#include <SPI.h>

char gelen [100];
volatile byte pos;
volatile bool process_it;

void setup (void){

  Serial.begin (115200);
  SPCR |= bit (SPE);
  pinMode (MISO, OUTPUT);
  pos = 0;
  process_it = false;
  SPI.attachInterrupt();
}

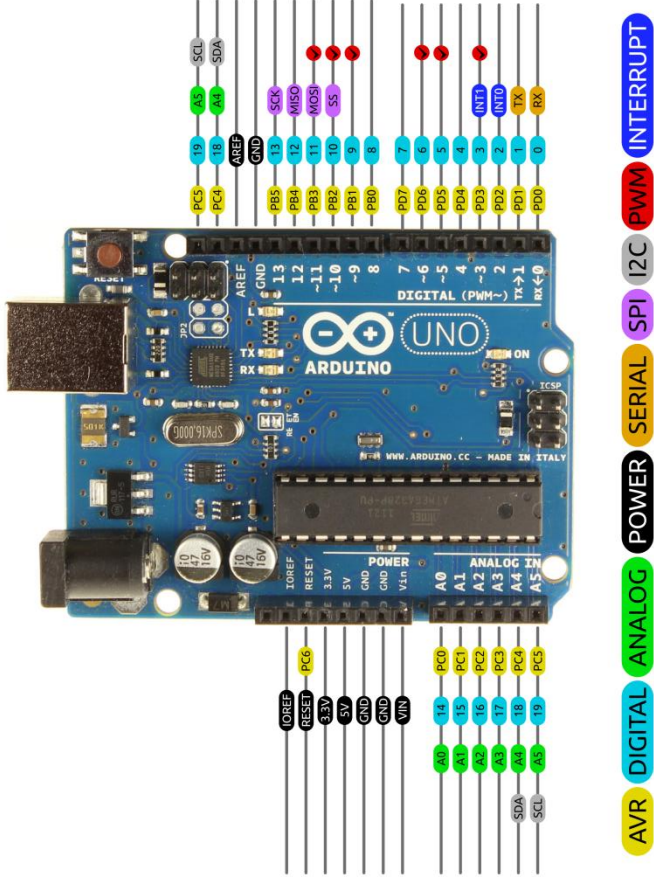
ISR (SPI_STC_vect)
{
  byte c = SPDR;
  // add to gelenfer if room
  if (pos < sizeof gelen){
    gelen [pos++] = c;
    if (c == '\n')process_it = true; }
}

void loop (void)
{
  if (process_it) {
    gelen [pos] = 0;
    Serial.println (gelen);
    pos = 0;
    process_it = false;
  }
}
```

Kod – 46– SPI Slave

24. Arduino Detaylı Pin Yapısı

Arduino Uno R3 Pinout



Şekil - 24.1 – Arduino Detaylı Pin Yapısı

25. ASCII Kodları

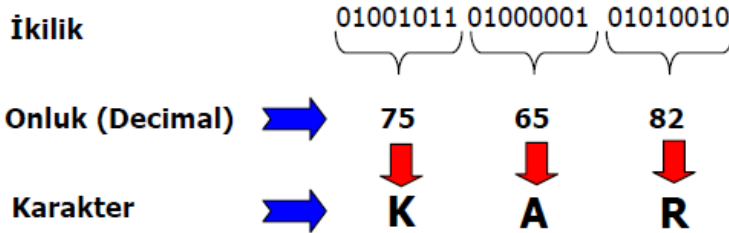
Sembollerin sayısal karşılıklarını belirleyerek, sayısal olmayan ya da alfabetik türdeki bilgiyi bilgisayarda temsil etmek amacıyla kullanılan kodlama sistemlerinden en yaygın olarak kullanılanı ASCII kodlama sistemidir.

ASCII sözcüğü **A**merican **S**tandard **C**ode For **I**nformation **I**nterchange sözcüklerinin koyu yazılmış ilk harflerinden oluşan yapay bir sözcüktür. ASCII kodlama sistemi her sembol için 8 bit kullanmaktadır. Sekiz bit kullanarak 0 ila 255 rakamları ile toplam 256 adet sembol temsil edilebilmektedir.

ASCII kodlama sistemi 1963 yılında tanımlanmıştır. Onluk sistemdeki (Decimal) ASCII kodlar ve karakter karşılıkları bu tablolarda

görülmektedir.

Örneğin KAR kelimesi 75 , 65 ve 82 nolu ASCII karakterlerdir ve bilgisayarda ikili sayı:



Şekil - 25.1 – ASCII Örnek

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(72	48	110	H	104	68	150	h
9	9	11		41	29	51)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Şekil - 25.1 – ASCII Tablosu

KODLAR LİSTESİ

- Kod -1 – Blink
- Kod - 2 – Setup
- Kod - 3 – Loop
- Kod - 4 – if
- Kod - 5 – if/else
- Kod - 6 – switch/case
- Kod - 7 – while
- Kod - 8 – do/ while
- Kod - 9 – break
- Kod - 10 – continue
- Kod - 11 – return1
- Kod - 12 – return2
- Kod - 13 – go to
- Kod - 14 – Süslü Parantez
- Kod - 15 – Toplama, Çıkarma, Çarpma, Bölme
- Kod - 16 – Eşit eşit, Eşit değil, Küçüktür, Büyüktür, Küçük eşittir, Büyük eşittir.
- Kod - 17 – Mantıksal Ve
- Kod - 18 – Mantıksal Değil
- Kod - 19 – += , -= , *= , /= , %=
- Kod - 20 – Bitisel Lojik Ve
- Kod - 21 – INPUT,OUTPUT
- Kod - 22 – true / false
- Kod - 23 – substring
- Kod - 24 – PROGMEM Kullanımı
- Kod - 25 – sizeof
- Kod - 26 – digitalRead
- Kod - 27 – analogRead
- Kod - 28 – analogWrite
- Kod - 29 – millis
- Kod - 30 – delayMicroseconds
- Kod - 31 – Bir Karakterin Analizi
- Kod - 32 – Serial.begin
- Kod - 33 – Serial.read
- Kod - 34 – Serial.print
- Kod - 35 – Donanım Interrupt
- Kod - 36 – Timer Interrupt
- Kod - 37 – Random Sayı Üretme
- Kod - 38 – Random Sayı Üretme2
- Kod - 39 – shiftOut
- Kod – 40 – EEPROM.read
- Kod - 41– EEPROM.write
- Kod - 42 – Kapsamlı EEPROM
- Kod - 43 – I2C Master
- Kod - 44 – I2C Slave
- Kod - 45 – SPI Master
- Kod - 46– SPI Slave

ŞEKİLLER Ve TABLOLAR LİSTESİ

- Şekil - 1.1 – Arduino Uno Özellikleri
- Şekil - 1.2 – Atmega 328P Pin isimleri
- Şekil - 1.3 – Arduino Uno kartı
- Şekil - 1.4 – Yazılım Yüklenmesi (I Agree)
- Şekil - 1.5 – Yazılım Yüklenmesi2 (Next)
- Şekil - 1.6 – Yazılım Yüklenmesi3 (Install)
- Şekil - 1.7 - Arduino IDE
- Şekil - 1.8 - Blink Led Devre Şeması
- Şekil - 2.1 – Loop
- Şekil - 3.1 – Buton, Led Devre Şeması
- Şekil - 3.2 – Potansiyometre Devre Şeması
- Şekil - 3.3 – Pull Up/ Pull Down Dirençler
- Şekil - 3.4 – Led Devre Şeması
- Şekil - 7.1 – Led Buton Devre Şeması
- Şekil - 11.1 – sizeof çıktısı
- Şekil - 12.1 - Led Devre Şeması
- Şekil - 12.1 – Program çıktısı
- Şekil - 12.1 - Potansiyometre Devre Şeması
- Şekil - 12.1 – Potansiyometre - Led Devre Şeması
- Şekil - 14.1 – millis Program Çıktısı
- Şekil - 17.1 – Karakter Analizi Çıktısı
- Şekil - 18.1 – Serial.print Ekran Çıktısı
- Şekil - 19.1 – Buton- Led Devre Şeması
- Şekil - 21.1 – Kare Dalga Sinyali
- Şekil - 21.2 – 74HC95 Entegresi İle Led Devresi
- Şekil - 21.3 – 74HC95 Entegresi Pin Yapısı
- Şekil - 22.1 – EEPROM Ekran Çıktısı
- Şekil - 23.1 – I2C Haberleşme Bağlantısı 1
- Şekil - 23.2 – I2C Haberleşme Bağlantısı 2
- Şekil - 23.3 – SPI Haberleşme Bağlantısı 1
- Şekil - 23.4 – SPI Haberleşme Bağlantısı 2
- Şekil - 24.1 – Arduino Detalı Pin Yapısı
- Şekil - 25.1 – ASCII Örnek

- Tablo – 8.1 – Bitsel Operatörler
- Tablo – 9.1 – Sayı Sistemleri
- Tablo – 10.1 – Dönüşümler Tablosu
- Tablo – 18.1 – Interrupt pin Tablosu
- Tablo – 23.1 – I2C Haberleşme Pin Yapısı
- Tablo – 23.2 – SPI Haberleşme Pin Yapısı