Arduino Programlama Ders İçeriği ve Konu Anlatımları

1.DERS

Arduino Nedir?

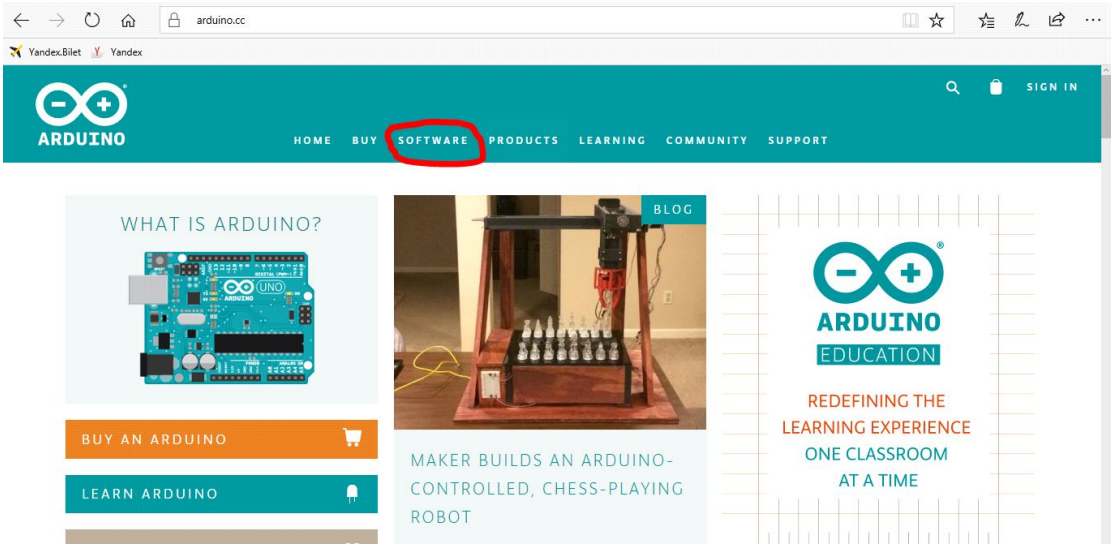
Arduino, mikrokontrolcü kartları ve yazılım paketinden oluşan bir programlama platformudur. Her kesimden insana hitap edebilmesi için kolaylık ön planda tutularak tasarlanmıştır. Kart üzerindeki mini bilgisayar (mikrokontrolcü), yazacağımız programa göre giriş ve çıkış bağlantılarını kontrol eder.

Arduino ile Neler Yapabiliriz?

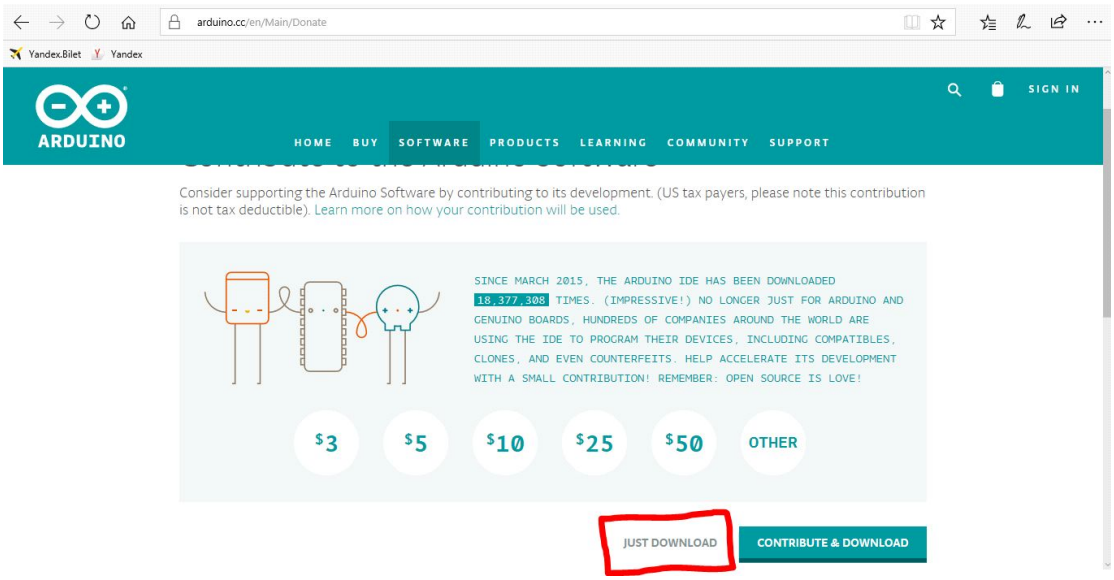
Arduino robotlar, insansız hava araçları (drone), akıllı ev otomasyonu projeleri gibi alanlarda sıklıkla tercih edilmektedir. Bunun yanı sıra Arduino ile yapabileceğiniz projelerin tek sınırı sizin hayal gücünüzdür. Aklınıza gelen hemen her çeşit projeyi Arduino kullanarak gerçekleştirebilirsiniz.

Arduino Yazılımının İndirilmesi

Arduino yazılımını indirmek için www.arduino.cc adresinden “Software” sekmesine gidiyoruz.

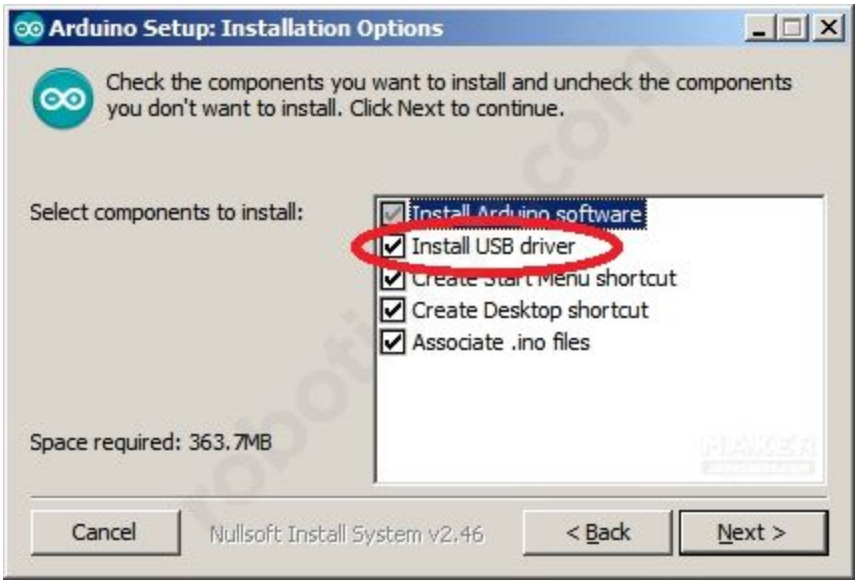


Software sekmesini tıkladıktan sonra, karşımıza işletim sistemimize göre olan dosyayı indireceğimiz ekran çıkıyor. Bu yazıyı hazırladığım sırada Arduino yazılımının en güncel sürümü 1.6.5 r2 idi. Windows kullananlar “Windows Installer” seçeneğini tıklayabilirler. Daha sonra bize bağış yapmamızı rica eden bir sayfa açılıyor. Tercihimize göre bağış yapabiliriz ya da “Just Download” seçeneği ile bağış yapmadan yazılımı indirebiliriz.



Arduino Sürücülerinin Yüklenmesi

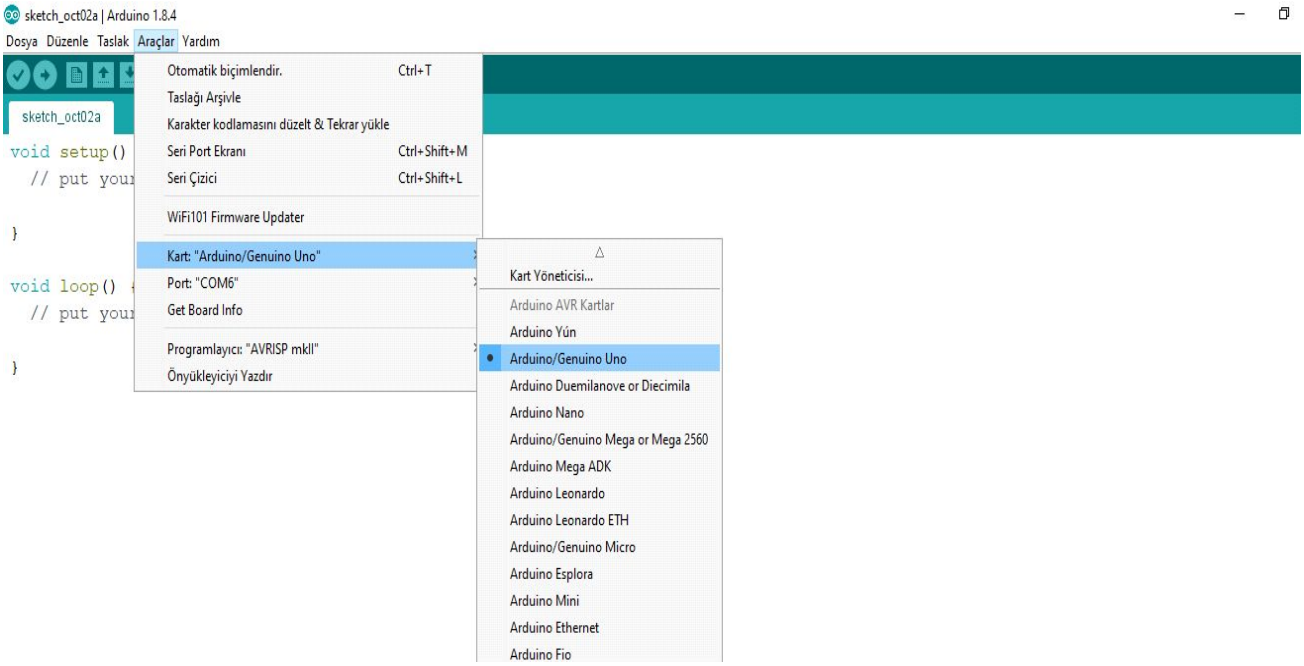
Bundan sonra yazılım kurulum dosyamız inmeye başlıyor. İndirme işlemi bittikten sonra dosyayı açarak kurulum işlemini başlatıyoruz. Kurulum sırasında çıkan “Install USB driver” seçeneğinin seçili olduğundan emin oluyoruz.



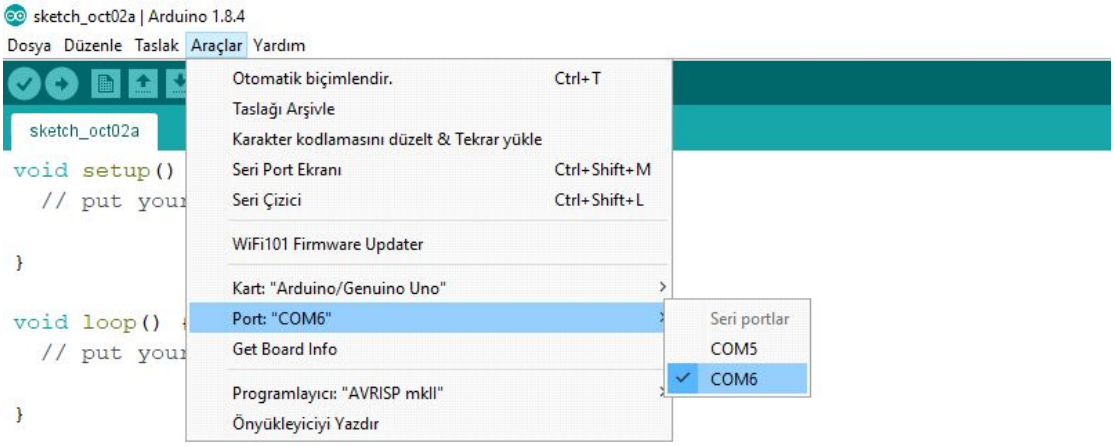
Kurulum işlemi bittikten sonra, kartımızı USB kablomuzla bilgisayarımıza bağlıyoruz. Bilgisayarımızda “Yeni donanım bulundu” penceresi açılıyor. Eğer sürücüler yazılımla birlikte kurulduysa, otomatik yükleme seçeneği Arduino’muzun sürücülerini otomatik olarak yükleyecektir.

Arduino Programının Bilgisayarımızda İlk Çalıştırılması

Artık Arduino programımızı açabiliriz. Programımızı açtıktan sonra ilk yapmamız gereken şey, programın Arduino UNO kartımızla çalışacak şekilde ayarlanmasıdır. Araçlar > Kart menüsünden Arduino UNO seçeneğini tıklıyoruz.

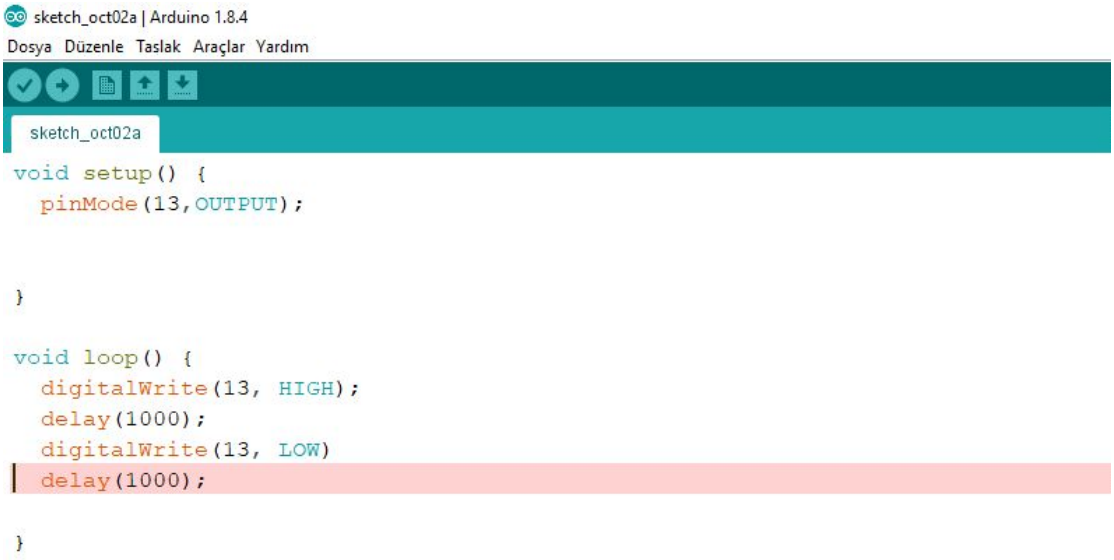


Daha sonra, yine Araçlar menüsünden Port alt menüsü altında Arduino’muzun bağlı göründüğü portu seçiyoruz. Bu port numarası, her bilgisayarda farklı olabilmektedir.



Programda void setup() kısmına yazacağımız fonksiyonlar, kart ilk enerji alıp çalıştığında sadece bir kere çalışır. Kullanacağımız giriş/çıkış pinlerini, seri port konfigürasyonunu vb. ayarları bu kısımda yapıyoruz. void loop() kısmında ise, setup fonksiyonundaki komutlar çalıştıktan sonra kartın enerjisi kesilene kadar sürekli çalışacak olan foknsiyonları barındırır.

Programımızı yazdıktan sonra kartımıza yüklemek istediğimizde, öncelikle “Kontrol Et” seçeneğine tıklıyoruz. Program, yazdığımız kodu öncelikle bilgisayarımızda bir klasöre kaydetmemizi istiyor, daha sonra da yazdığımız kodu derleyerek herhangi bir hata varsa bu hatayı bize bildiriyor.



Örneğin, bu kodda **digitalWrite** fonksiyonundan bir önceki komut olan **delay** komutunu yazdıktan sonra noktalı virgül (;) koymayı unuttuğumuz için bize bu satırla ilgili bir hata mesajı görüyoruz.

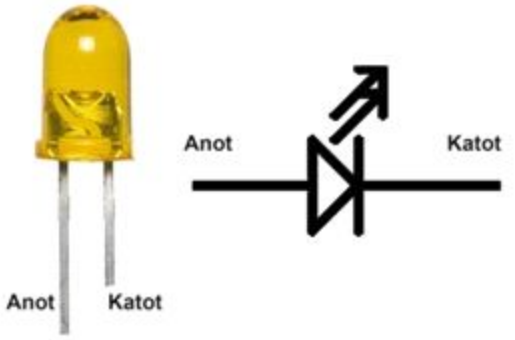
Eğer yazdığımız kodda bir hata yoksa ve Arduino kartımız bilgisayarımıza USB ile bağlıysa, "Yükle" seçeceğine tıklayarak kodumuzu kartımıza yükleyebiliyoruz.

2.DERS

LED YAKMA

LED Nedir?

LED, ışık yayan diyot anlamına gelen Light Emitting Diode sözcüğünün baş harflerinden oluşan bir kısaltmadır. Alışık olduğumuz ve çoğu projemizde kullandığımız 6V ile çalışan ufak ampullerin aksine LED'lerin anot ve katot olmak üzere iki farklı bacağı vardır. Bunlardan anodu pozitif gerilime yani + uca, katot ise negatif gerilime yani – uca ya da toprak hattına (GND, Ground) bağlanmalıdır.



Gerilim, Akım ve Ohm Yasası

Çeşitli devre elemanlarının farklı gerilim yani voltajlarda çalıştığını biliyoruz. Arduino kartımız ise 5V gerilimle çalışmaktadır. LED'imiz için ise bu durum biraz farklıdır. LED'in üzerinden geçecek maksimum akımın 20 mA (miliamper = amperin 1000'de 1'i) değerini geçmemesi gereklidir. Arduino'muz 5V ile çalışıyor demiştik. 5V değeri bize kartın çıkış gerilimini ifade etmektedir. Fakat LED 20 mA akıma ihtiyaç duymakta.

Eğer LED'imizi Arduino'ya doğrudan bağlayacak olursak, LED üzerinden kartın sağlayabileceği maksimum değerde akım geçecek ve LED'imiz veya kartımız bozulacaktır. Bunun için akım sınırlayıcı bir direnci LED'imize seri olarak bağlamamız gerekmekte. Peki bu direncin değeri nasıl belirlenecek? İşte burada Ohm Kanunu dediğimiz denklem devreye giriyor:

V = **i** x **R**

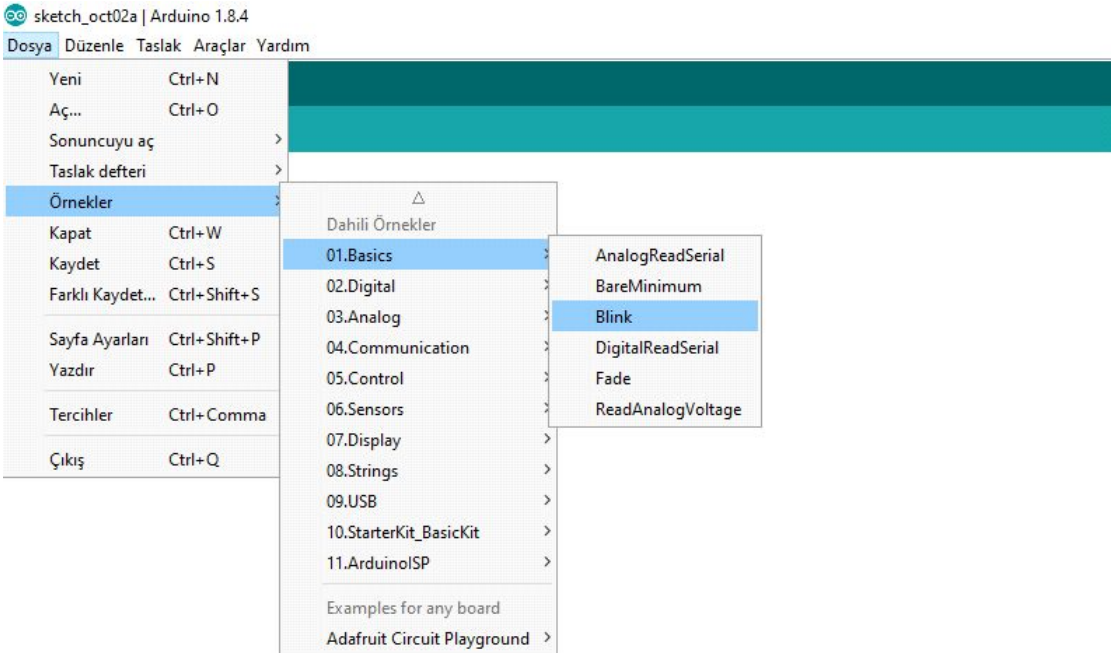
Bu denklemde V bize gerilimi, i akımı ve R ise direnci temsil ediyor. Eğer 20 mA akıma ihtiyaç duyan LED'i, Arduino'muzun 5V çıkış sağlayan pinlerinden birine bağlayacak olursak;

5V = **0,020A** x **R**

Denklemini elde etmiş oluruz. Bu denklemden **R**'yi çekecek olursak sonucu 250 buluruz. Bu demek oluyor ki LED'imizi 5V gerilimle kullanmak için 250 Ω (ohm) değerinde bir dirence ihtiyacımız var. Tam değeri doğru tutturmamız çok önemli değil, elimizde mevcut olan 220 Ω'luk direnci kullanabiliriz.

Arduino programımızı açıyoruz. Şu sırayı takip ederek “Blink” isimli örnek programı açıyoruz:

Dosya > Örnekler > 01.Basics > Blink



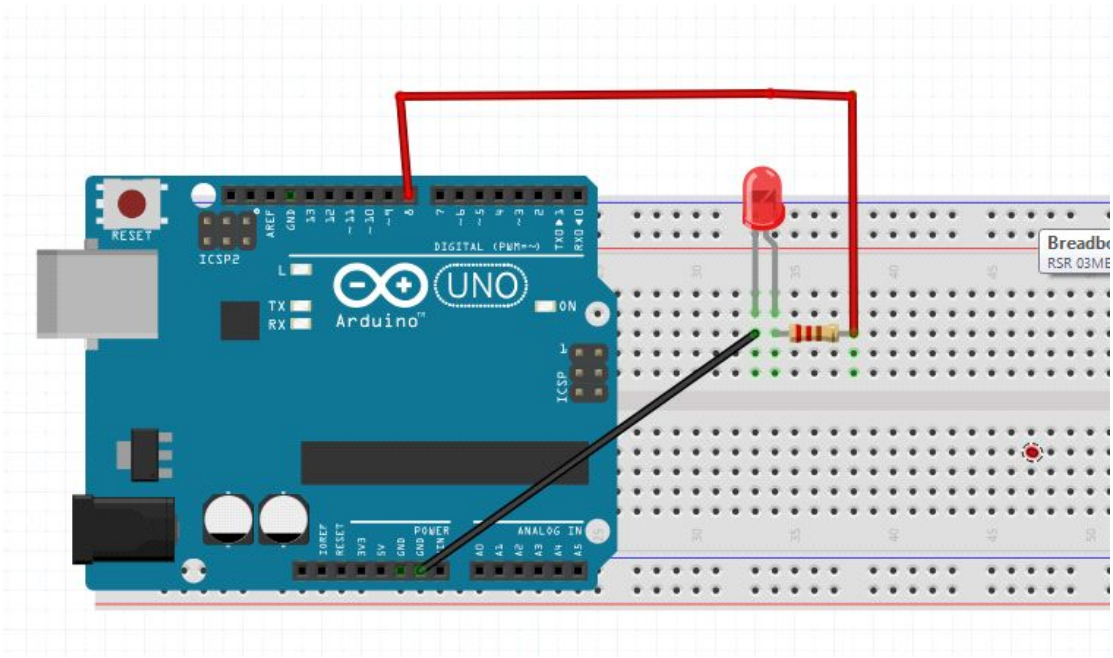

```
pinMode(8, OUTPUT);
```

Bu satır, kart üzerindeki 8 numaralı pini çıkış verecek şekilde ayarlıyor. Kullanacağımız pin çıkış veya giriş olarak belirlenmez ise programın devamında yazacağımız giriş veya çıkış fonksiyonları, o pini kullanamaz.

```
digitalWrite(8, HIGH);
delay(1000);
digitalWrite(8, LOW);
delay(1000);
```

Bu kısım ise öncelikle 8 numaralı pine HIGH lojik seviyesine, yani 5V’a ayarlıyor, 1000 milisaniye (1 saniyeye eşittir) hiçbir işlem yapmadan bekliyor ve bu sefer 8 numaralı pini lojik LOW yani 0V veya toprak hattı seviyesine ayarlıyor. Bu işlemi yaptıktan sonra mikokontrolcü, delay fonksiyonu sayesinde tekrardan 1 saniye hiçbir işlem yapmadan bekliyor.

Bu koddaki delay komutlarının sürelerini değiştirerek LED’in açık ve kapalı kaldığı süreleri değiştirebiliriz. Eğer başka bir pin kullanmak istersek tek yapmamız gereken pinMode ve digitalWrite fonksiyonlarında bulunan pin numarasını kullanmak istediğimiz pin numarası ile değiştirmek.

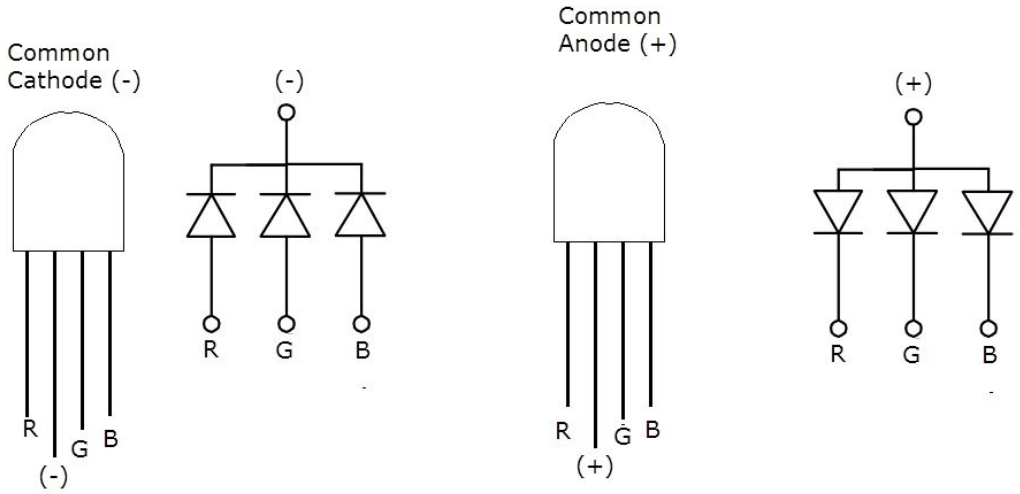

RESET
ICSP2
AREF
GND
DIGITAL (PWM~)
TXD
RXD
L
TX
RX
UNO
Arduino
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
A0
A1
A2
A3
A4
A5
Breadb
RSR 03ME

3.DERS

RGB LED KONTROLÜ

RGB LED Nedir?

RGB LED'ler, normal LED'lerden farklı olarak tek paket içerisinde 3 farklı renk (kırmızı, yeşil ve mavi) LED'i bir arada bulundurur. LED yakıp söndürme dersimizden hatırlayacak olursak LED'lerin anot ve katot uçları bulunuyordu. RGB LED'lerde ise LED'in üretim şekline göre anot veya katot bağlantıları ortak olarak bulunmaktadır.

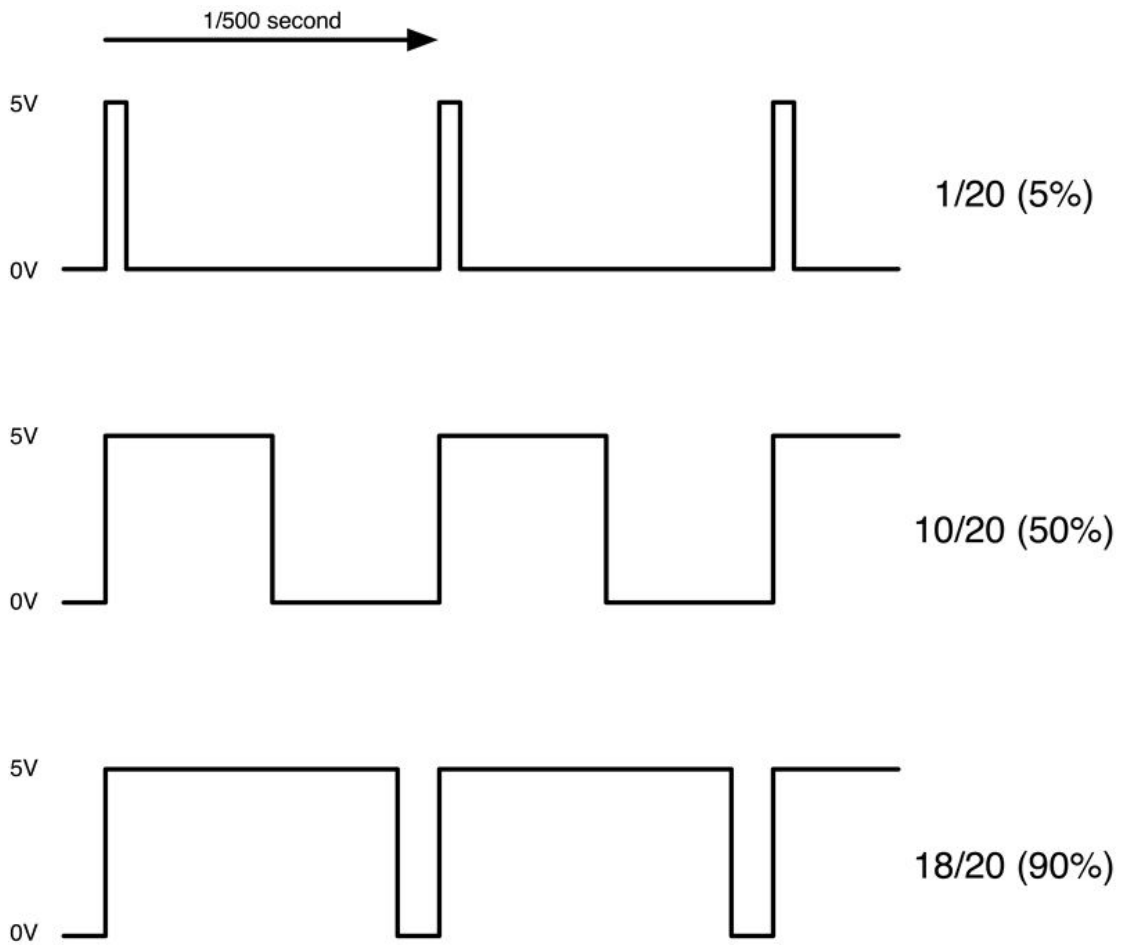


PWM ve Dijitalden Analog'a Dönüşüm

Bildiğimiz gibi **Arduino'muzun** giriş/çıkış pinlerinde kullanılan voltaj 5V seviyesinde. Bir önceki dersimizde LED'imizi 5V gerilimde 20 mA akım çekecek şekilde kartımıza bağlamıştık. Bu şekilde bağladığımızda LED'imiz olabilecek en parlak şekilde yanmaktaydı. Peki parlaklığı değiştirmek istersek ne yapmamız gerekir?

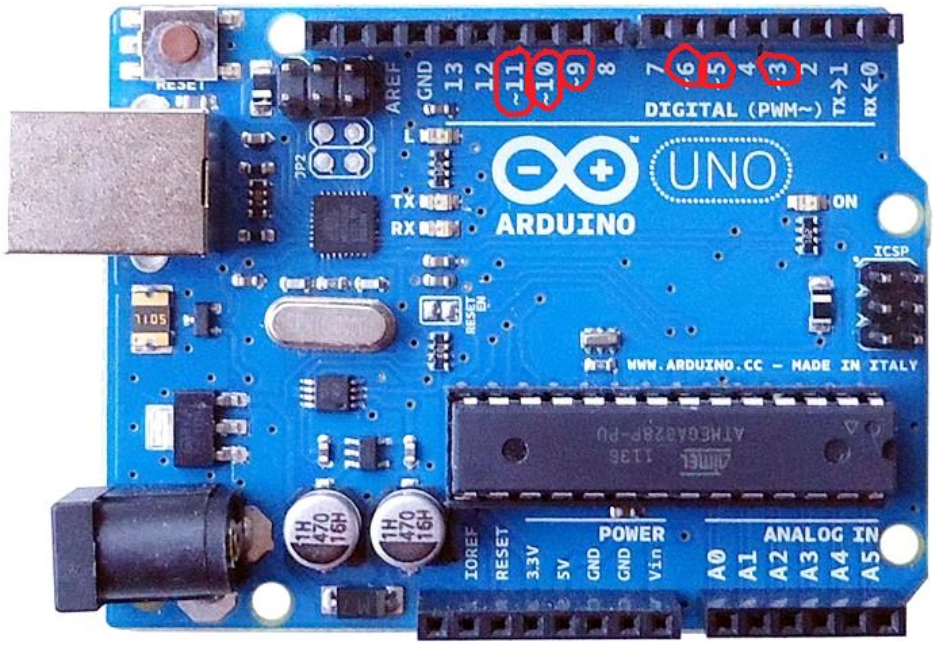
Sorunun cevabı aslında basit: gerilimi düşürmek. Eğer 5V ile çalışan LED'imizi daha düşük bir gerilimle, örneğin 3V ile çalıştırırsak parlaklığı azalacaktır. Fakat bu sefer de şunu sorabilirsiniz: Arduino çıkış gerilimi 5V değil miydi? Nasıl 3V çıkış alabiliriz?

Burada darbe genişliği modülasyonu (PWM – pulse width modulation) tekniğini kullanmamız gerekiyor. Bu yazıda darbe genişliği modülasyonu yerine kısaca PWM olarak bahsedeceğim. PWM, çıkışta aldığımız 5V gerilimi belirli bir zaman aralığında (Arduino'da genellikle saniyenin 1/500'ü) açıp kapatarak 0 ile 5V arasında istediğimiz gerilimi Arduino'nun çıkış pininden almamızı sağlar. Şu şekilde düşünelim: eğer bir LED çok hızlı bir şekilde yanıp sönerse gözümüz bu yanıp sönmeyi tam olarak algılayamaz ve parlaklığı daha düşük olarak algılar.



Yukarıdaki görselde gördüğümüz üzere, 5V'u 2 milisaniyelik sürenin sadece %5'lik kısmında açık olacak şekilde verirsek, elde ettiğimiz değer 5V'un %5'i; yani 0,25V olacaktır. Aynı şekilde eğer 2 ms sürenin yarısında (%50) açık kalacak şekilde ayarlarsak 2,5V elde ederiz.

Arduino UNO kartımızın tüm pinleri PWM çıkış yeteneğine sahip değildir. Kartımızın dijital pinlerinde pin numarasının önünde ~ işareti olan bazı pinler mevcuttur. Eğer PWM çıkış almak istiyorsak, bu pinleri kullanmak zorundayız. Bu pinler Arduino UNO için 3, 5, 6, 9, 10 ve 11 numaralı pinlerdir.



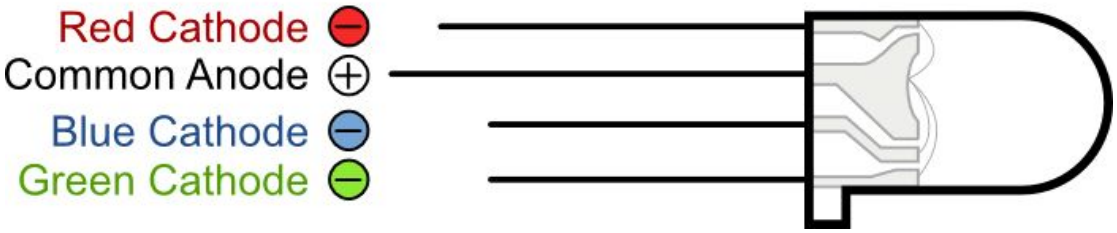
RGB LED İçin Arduino Kodu

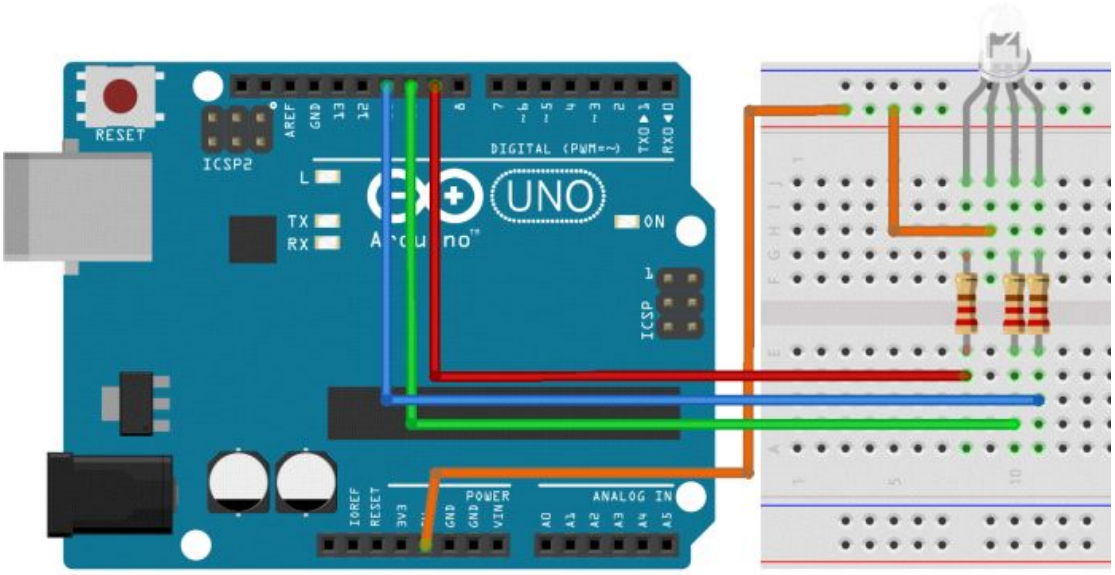
İlk dersimizde Arduino yazılımı içinde mevcut olan örnek kodu değiştirmiştik. Bu sefer kodumuzu kendimiz yazıyoruz:

```
int kirmiziPin = 9;
int yesilPin = 10;
int maviPin = 11;

void setup()
{
 pinMode(kirmiziPin, OUTPUT);
 pinMode(yesilPin, OUTPUT);
 pinMode(maviPin, OUTPUT);
}

void loop()
{
 renkAyarla(255, 0, 0); //kirmizi
 delay(1500);
 renkAyarla(0, 255, 0); //yesil
 delay(1500);
 renkAyarla(0, 0, 255); //mavi
 delay(1500);
 renkAyarla(255, 255, 0); //sari
 delay(1500);
 renkAyarla(80, 0, 80); //mor
 delay(1500);
 renkAyarla(0, 255, 255); //acik mavi
 delay(1500);
 renkAyarla(255, 255, 255); //beyaz
```

```
 delay(1500);
}

void renkAyarla(int kirmizi, int yesil, int mavi)

{
 kirmizi = 255 - kirmizi;
 yesil = 255 - yesil;
 mavi = 255 - mavi;
 analogWrite(kirmiziPin, kirmizi);
 analogWrite(yesilPin, yesil);
 analogWrite(maviPin, mavi);
}
```

RGB LED'imizin kırmızı bacağını 9 numaralı pine, yeşil bacağını 10 numaralı pine ve mavi pinini 11 numaralı pine 220 Ω'luk dirençler ile bağlıyoruz. Kullandığımız LED ortak anot yapıya sahip olduğu için anot bacağını da kartımızın 5V pinine bağlıyoruz.





Kodumuzun setup fonksiyonunda kullanacağımız pinleri çıkış olarak tanımlıyoruz. Ayrıca yazdığımız renkAyarla isimli fonksiyondaki analogWrite komutu, bize her bir PWM çıkış

pininden alacağımız voltajın yüksekliğini ayarlamamızı sağlıyor. analogWrite komutu şu şekilde kullanılıyor:

```
analogWrite(PWM çıkış pin numarası, 0-255 arası sayısal değer);
```

analogWrite komutunda 255 değeri maksimum çıkış voltajını yani 5V'u temsil ediyor. 0 ile 255 arası tüm değerler 0 – 5V arası voltaj değerlerine denk düşüyor. Örneğin analogWrite(9 , 80) komutu, 9 numaralı pinden **5V x (80/255) = 1,57V** geriliminde çıkış almamızı sağlıyor. Farklı parlaklıklardaki kırmızı mavi ve yeşil ışığı karıştırarak istediğimiz renkte ışık elde etmemiz bu sayede mümkün oluyor.

4.DERS

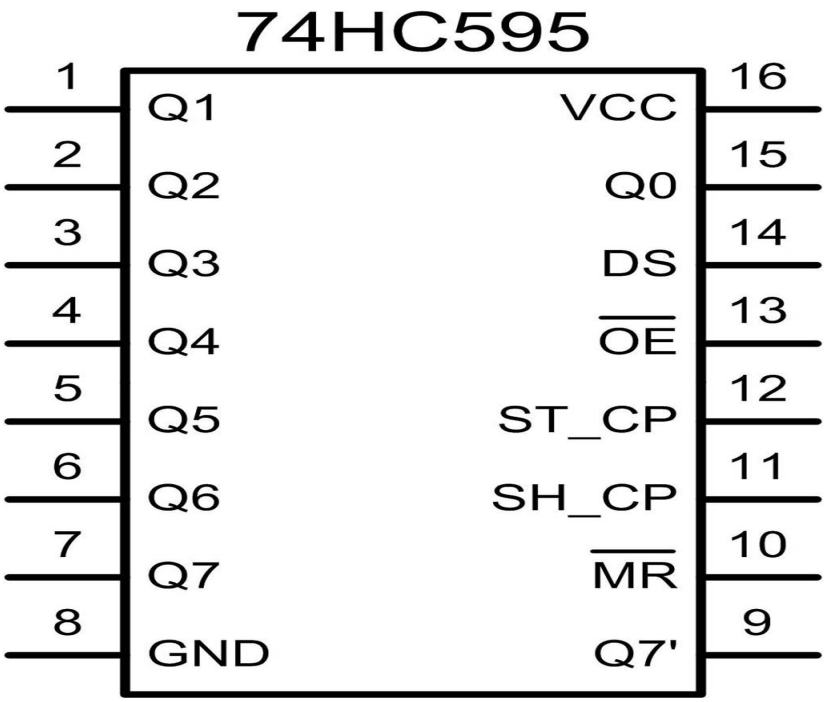
74HC595 SHİFT REGİSTER KULLANIMI

Arduino UNO kartımızda 13 adet dijital giriş/çıkış pini bulunmakta. Fakat bu pinlerin hepsini LED sürmek için kullanmak istemeyebiliriz. Yada LED haricinde diğer pinlere sensör, switch, buton v.b. diğer girişler bağlamış olabiliriz. Burada kullanacağımız metot sayesinde Arduino'da sadece 3 adet dijital çıkış pini kullanarak 8 adet LED'i kontrol etmemiz mümkün olacak.

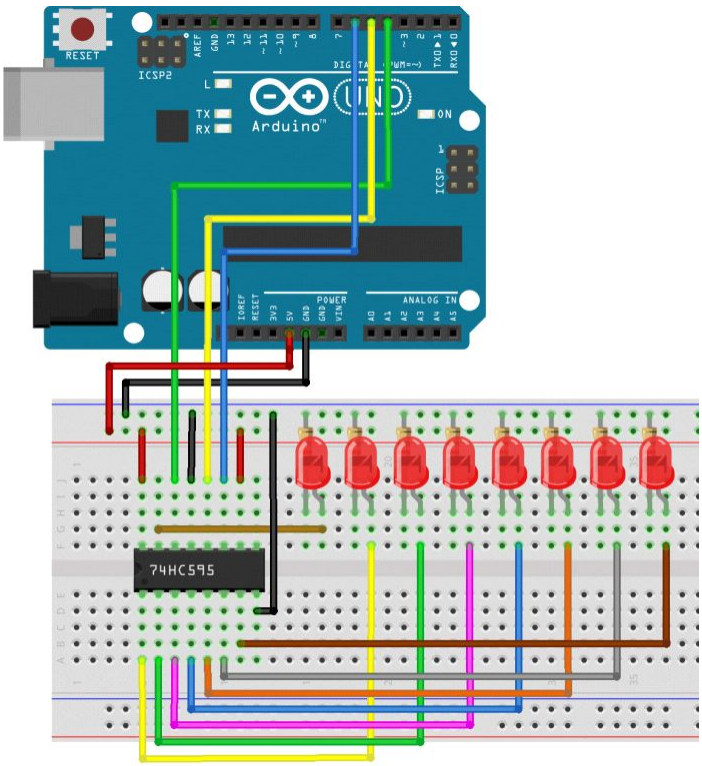
74HC595 Shift Register Entegresi

Shift register devresi, peş peşe bağlanmış flip-flop adındaki hafıza elemanı devrelerinden oluşur. Tiplerine göre paralel girişi seri çıkışa (parallel-in, serial-out, PISO) dönüştüren ya da bizim projemizde kullandığımız 74HC595 gibi seri girişi paralel çıkışa (serial-in, parallel-out, SIPO) dönüştüren işlevde olabilirler.

Kullandığımız entegrede Arduino'ya bağlayacağımız 3 adet bacağa sahiptir: Seri data girişi (DS, pin 14), latch (ST_CP, pin 12) ve clock pini (SH_CP, pin 11). Shift register şu şekilde çalışır: veri girişi yapılmak isteniyorsa latch pini lojik 0 seviyesine çekilir, data girişi yapılır (lojik 1 veya 0) ve saat darbesi (clock pulse) clock pinine uygulanır. Bu şekilde 8 bitlik veri tek bir seri giriş pininden girilir ve işlem sonucunda seri olarak girilen veri paralel çıkışlardan (Q0 – Q7 pinleri) alınmış olur.



Entegrenin LED’lere ve Arduino’ya bağlanması:



```
int latchPin = 5;
 int clockPin = 6;
 int dataPin = 4;
 byte leds = 0;
 void setup() {
 pinMode(latchPin, OUTPUT);
 pinMode(dataPin, OUTPUT);
 pinMode(clockPin, OUTPUT);
 }
 void loop() {
 leds = 0; registeraYaz();
 delay(500);
 for (int i = 0; i < 8; i++)
 {
bitSet(leds, i); registeraYaz();
 delay(500);
 }
for (int i = 8; i >= 0; i--)
 {
bitClear(leds, i);
 registeraYaz();
 delay(500);
 }
 }
void registeraYaz() {
 digitalWrite(latchPin, LOW);
 shiftOut(dataPin, clockPin, LSBFIRST, leds);
 digitalWrite(latchPin, HIGH);
 }
```

Kodumuzun ilk kısmında her zaman yaptığımız gibi çıkış pinlerimizi tanımladık. leds isimli 8 bitlik bir değişken tanımladık (byte tipindeki değişkenler 8 bit büyüklüğündedir). Bu baytın her bir biti, bizim shift register'ımızın çıkışına bağlı olan LED'leri temsil ediyor. registeraYaz fonksiyonumuz, shift register'ın çalışması için gerekli işlemleri yapıyor. loop fonksiyonumuzda bu fonksiyonu çağırarak leds değişkeninde yaptığımız değişiklikleri LED'lerimize aktarmak için

bu fonksiyonu çağırıyoruz. loop fonksiyonumuzda ise iki adet for döngüsü kullandık. İlk for döngüsü, leds değişkenimizdeki 8 bitten her birini sırayla 1 yaparak 00000001, 00000011, 00000111… şeklinde bir desen elde etmemizi sağlayacak. Her bir bit 1 olduktan sonra (11111111), ikinci for döngüsü başlayarak bu sefer bitleri 0'layarak 01111111, 00111111,00011111… desenini oluşturacak.

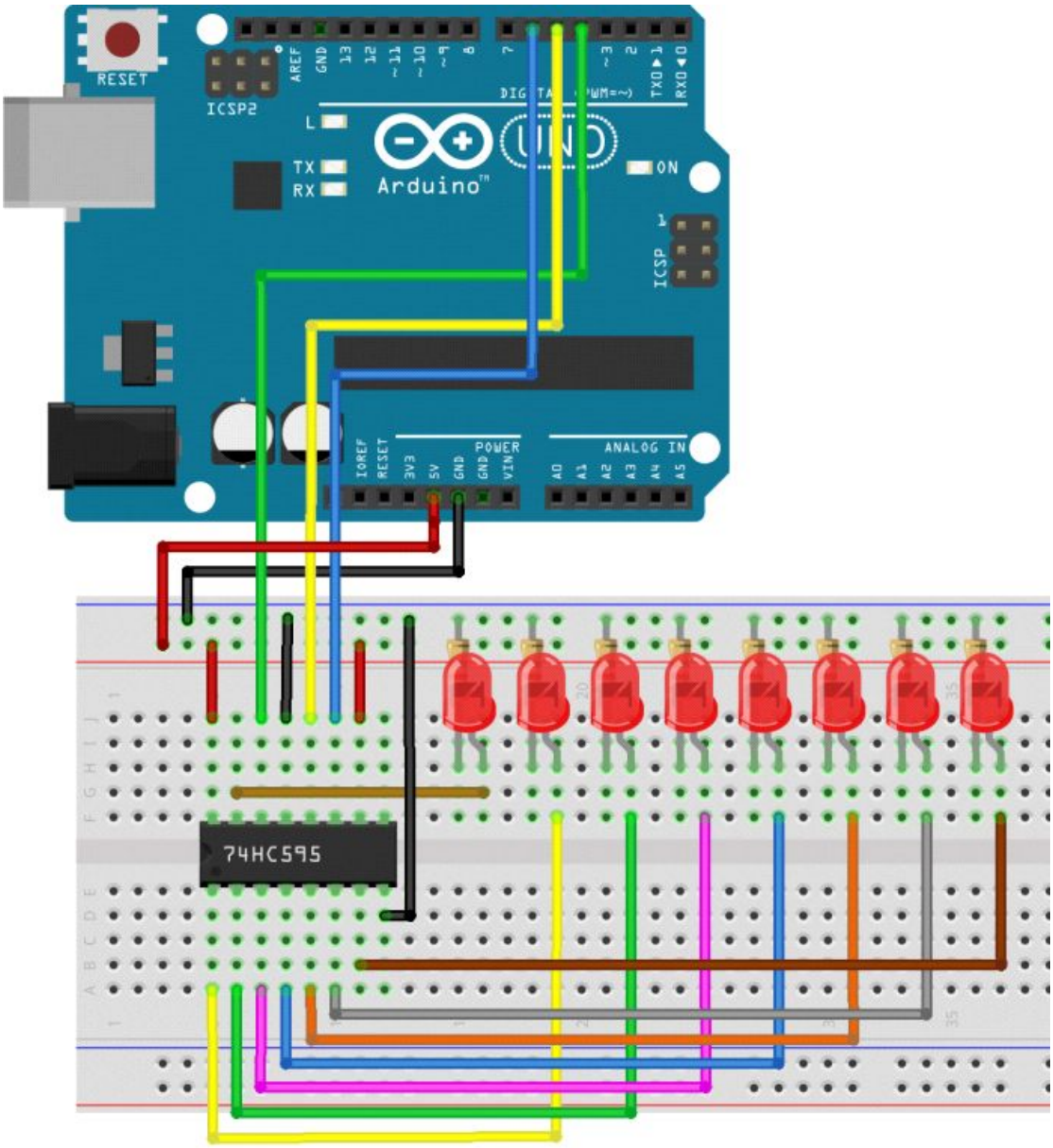
5.DERS

SERİ HABERLEŞME (UART)

Seri Haberleşme

Arduino bilgisayarımıza tanıtıldığında COM portu olarak tanınmıştı. Bu COM portu, seri haberleşme (UART) dediğimiz bir bağlantı türü için kullanılmaktadır. Bilgisayarımızla Arduino'yu haberleştirmek için Arduino yazılımında bulunan "Seri Port Ekranı"nı kullanacağız.

Devremizin bağlantı şeması;

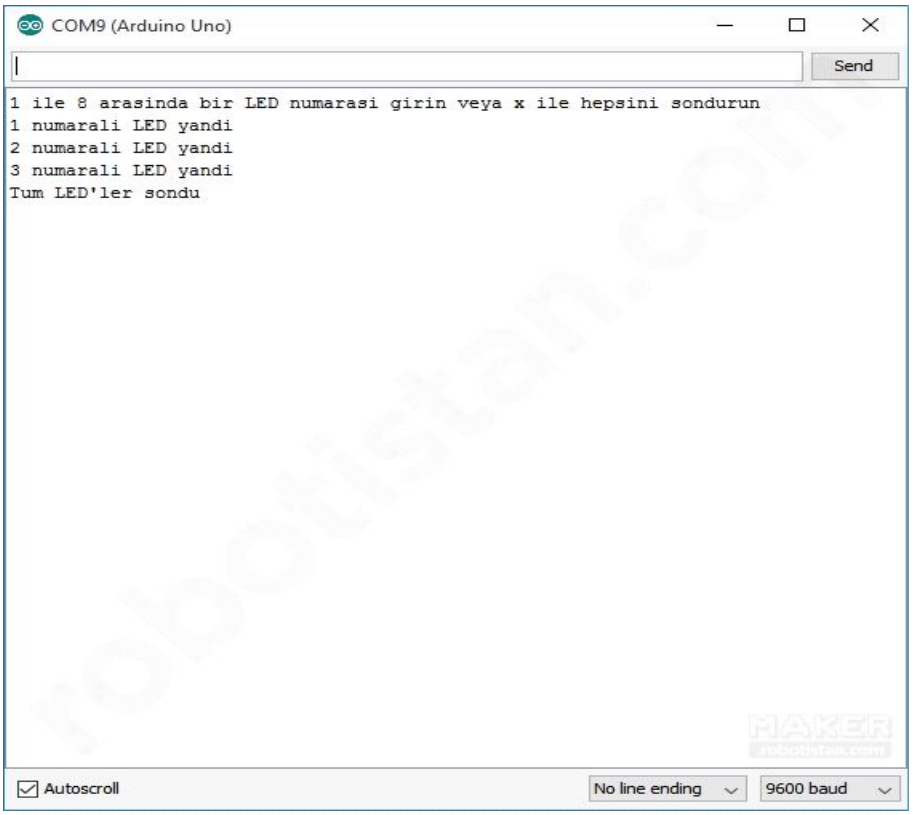


Örnek kod:

```
int latchPin = 5;
int clockPin = 6;
int dataPin = 4;

byte leds = 0;

void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
```

```
  pinMode(clockPin, OUTPUT);
  registeraYaz();
  Serial.begin(9600);
  while (! Serial);
  Serial.print("1 ile 8 arasinda bir LED numarasi girin veya ");
  Serial.println("x ile hepsini sondurun");
}

void loop()
{
  if (Serial.available())
  {
    char ch = Serial.read();
    if (ch >= '1' && ch <= '8')
    {
      int led = ch - '1';
      bitSet(leds, led);
      registeraYaz();
      Serial.print(led +1);
      Serial.println(" numarali LED yandi");
    }
    if (ch == 'x')
    {
      leds = 0;
      registeraYaz();
      Serial.println("Tum LED'ler sondu");
    }
  }
}

void registeraYaz()
{
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, LSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
}
```

Kodumuzun setup kısmında yer alan Serial.begin(9600) komutu, Arduino'muzun seri portunu 9600 baud'da (seri iletişim hızı) çalışacak şekilde ayarlıyor. Serial.print komutu ise Arduino'nun seri porttan bilgisayarla iletişime geçerek tırnak içinde yazan yazının seri port ekranında görünmesini sağlıyor.

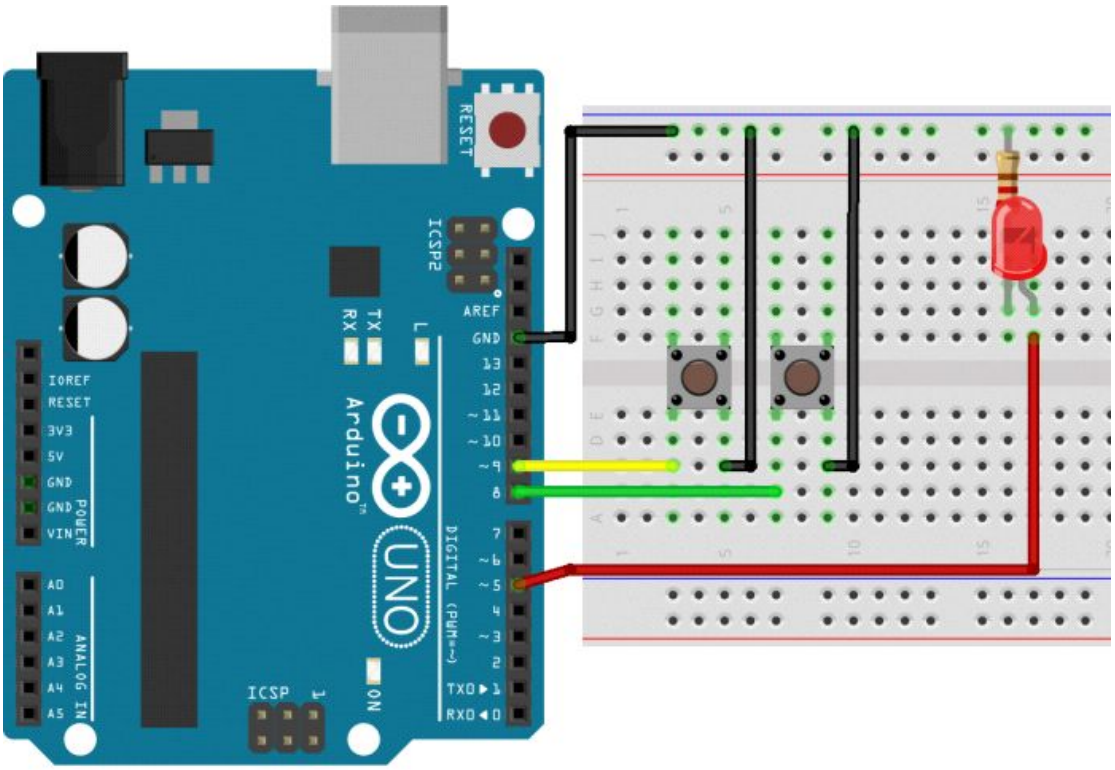


6.DERS

DİGİTAL GİRİŞLER

Buton kullanarak led yakıp söndürme uygulaması:

Arduino bağlantı şeması;



Örnek Kod:

```
int ledPin = 5;
int buttonApin = 9;
int buttonBpin = 8;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonApin, INPUT_PULLUP);
  pinMode(buttonBpin, INPUT_PULLUP);
}

void loop()
{
  if (digitalRead(buttonApin) == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }
  if (digitalRead(buttonBpin) == LOW)
  {
    digitalWrite(ledPin, LOW);
  }
}
```

Butonları bağladığımız pinleri tanımlarken sadece INPUT kullanmak yerine INPUT_PULLUP

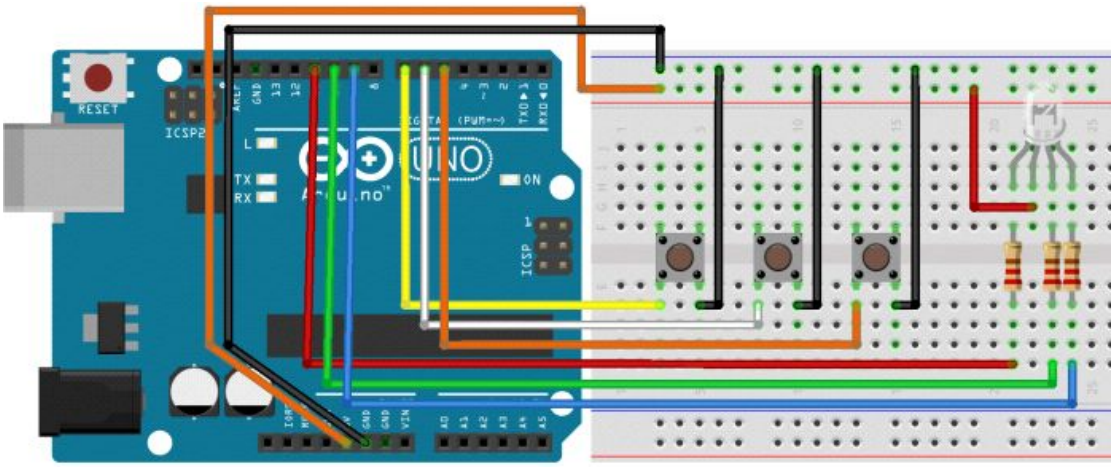
şeklinde bir tanımlama kullandık. Bu sayede Arduino kartımızın dijital pinlerine entegre olan pull-up direncini aktifleştirmiş oluyoruz. Peki pull-up direnci ne işe yarar?

Pull-up direnci, dijital pinleri giriş olarak kullandığımızda sinyalin bozulmamasını sağlar. Bu projemizde buton basılı değilken dijital pinden okunan değer 5V yani lojik HIGH seviyesidir. Pull-up direnci, buton basılıp değer LOW'a çekilmediği sürece bu pindeki gerilimin 5V'ta sabit kalmasını sağlar.

7.DERS

BUTON KONTROLLÜ RGB LED

Bağlantı şeması:



Örnek kod:

```
int kirmiziLEDPin = 11;
int yesilLEDPin = 10;
int maviLEDPin = 9;

int kirmiziSwitchPin = 7;
int yesilSwitchPin = 6;
int maviSwitchPin = 5;

int kirmizi = 0;
```

```
int mavi = 0;
int yesil = 0;

void setup()
{
  pinMode(kirmiziLEDPin, OUTPUT);
  pinMode(yesilLEDPin, OUTPUT);
  pinMode(maviLEDPin, OUTPUT);
  pinMode(kirmiziSwitchPin, INPUT_PULLUP);
  pinMode(yesilSwitchPin, INPUT_PULLUP);
  pinMode(maviSwitchPin, INPUT_PULLUP);
}

void loop()
{
  if (digitalRead(kirmiziSwitchPin) == LOW)
  {
    kirmizi ++;
    if (kirmizi > 255) {
      kirmizi = 0;
    }
  }
  if (digitalRead(yesilSwitchPin) == LOW)
  {
    yesil ++;
    if (yesil > 255) {
      yesil = 0;
    }
  }
  if (digitalRead(maviSwitchPin) == LOW)
  {
    mavi ++;
    if (mavi > 255) {
      mavi = 0;
    }
  }
  renkAyarla(kirmizi, yesil, mavi);
  delay(10);
}

void renkAyarla(int kirmizi, int yesil, int mavi)
{
  kirmizi = 255 - kirmizi; //ortak katot kullanılıyorsa silin
  yesil = 255 - yesil; //ortak katot kullanılıyorsa silin
  mavi = 255 - mavi; //ortak katot kullanılıyorsa silin
  analogWrite(kirmiziLEDPin, kirmizi);
  analogWrite(yesilLEDPin, yesil);
  analogWrite(maviLEDPin, mavi);
}
```

Yaptığımız uygulamada, her bir renk için parlaklığı arttıran bir push buton mevcut. Bu butonlara basıldığı sürece o rengin parlaklık değeri artıyor, 255'e ulaştığında ise tekrardan 0'lanıyor.

8.DERS

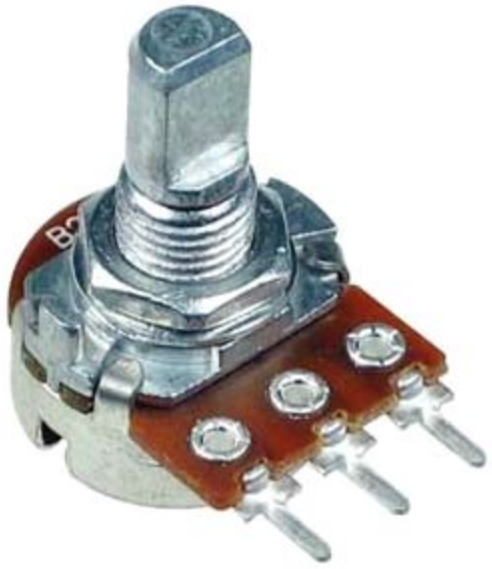
ANALOG GİRİŞLER

Analogdan Dijitale Çeviriciler (Analog-to-Digital Converter, ADC)

Arduino UNO kartındaki işlemcide, 10-bit çözünürlüğe sahip analogdan dijitale dönüştürücü (ADC – analog to digital converter) mecvuttur. Peki, bu 10-bit ne anlama geliyor? Bildiğimiz üzere Arduino'muzun mikrokontrolcüsü 5V gerilimle çalışmakta. Bu mikrokontrolcüde sahip olduğunu söylediğimiz 10-bit ADC, 0V ile 5V arası gerilimleri $2^{10} = 1024$ adım hassasiyet ile okuyabilir. Yani analog input pinlerinden birine vereceğimiz 0V gerilim bize 0 değerini; aynı şekilde 5V gerilim ise 1023 değerine denk düşüyor. Eğer 5V'tan daha düşük bir gerilimi referansımızın üst noktası olarak almamız gerekirse, Arduino üzerinde bulunan AREF pinini kullanmamız ve kodda küçük bir değişiklik yapmamız gerekir. Bu pine herhangi bir gerilim uygulamadığımız takdirde ADC'miz 0-5V arasında çalışacaktır.

Potansiyometre

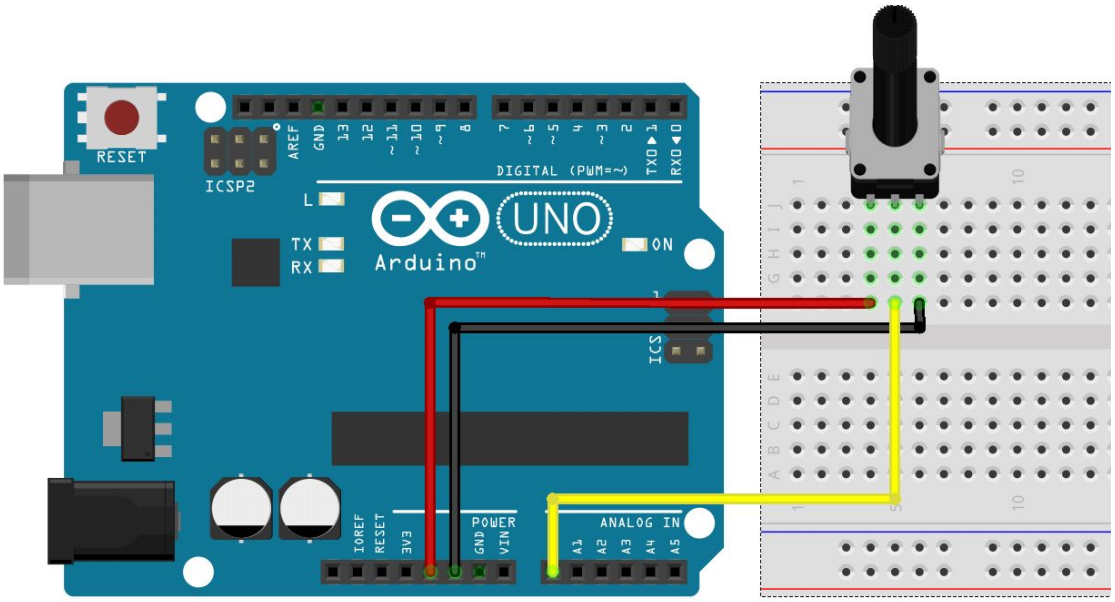
Potansiyometre, aslında çevremizde her gün kullandığımız cihazların neredeyse hepsinde mevcut olan bir devre elemanıdır. Örneğin, müzik setimizin ses seviyesini değiştirmek için çevirdiğimiz düğme bir potansiyometredir. En basit açıklama ile potansiyometre, değerini elimizle çevirerek ayarladığımız bir dirençtir. Mikrokontrolcü uygulamalarında ise genellikle gerilim bölücü olarak kullanılır.



Potansiyometreyi bir yöne çevirdiğimizde yan yana olan iki bacağının direnci değişir. Bunu bir multimetreyle ölçerek görebiliriz. Biz bu uygulamamızda ise 0 ile 5V arasında voltaj değişimini

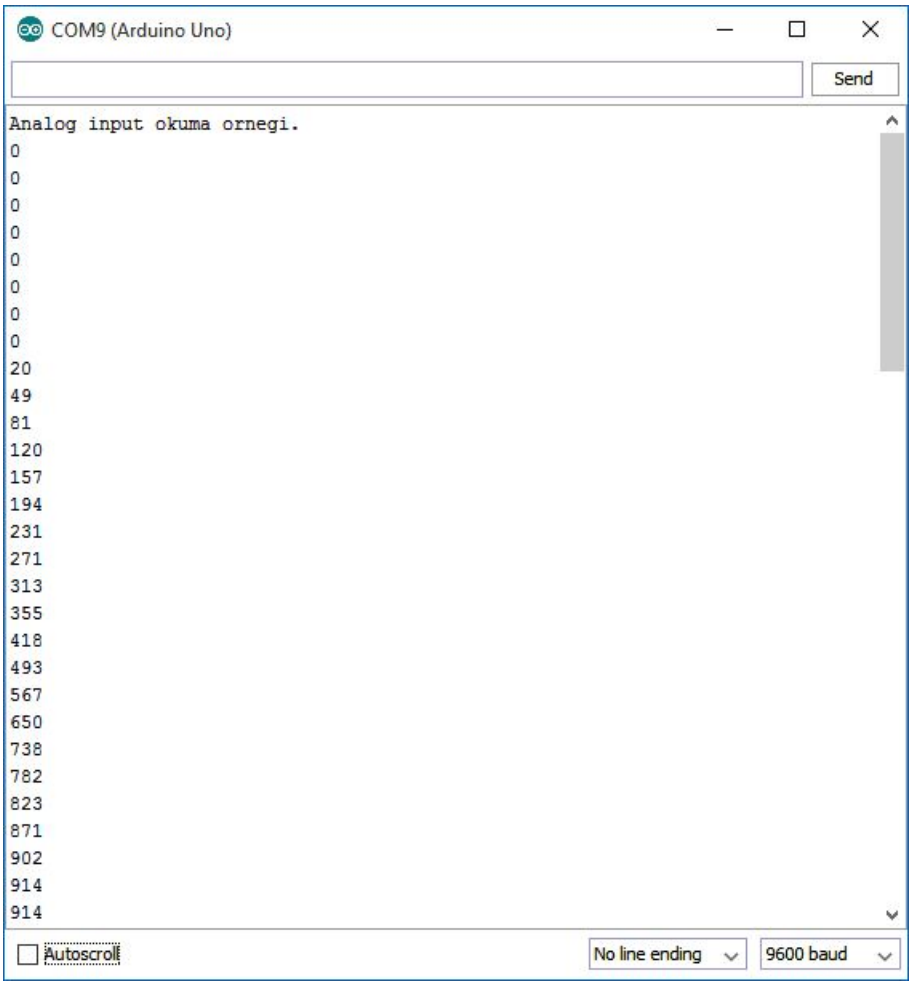
sağlamak için potansiyometre kullanacağız.

Uygulamamızın ilk adımında sadece analog giriş pininden gelen gerilimin sayısal karşılığını seri porttan okuma işlemini yapacağız. Devremizi bu şekilde kuruyoruz:

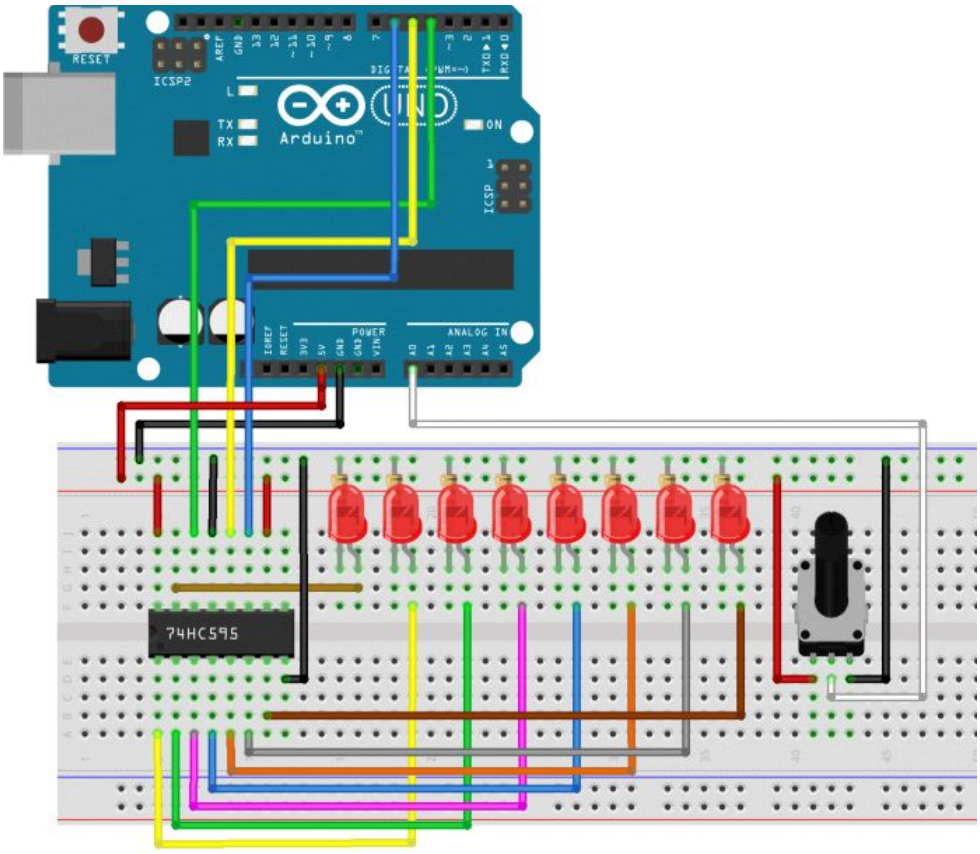


Bu şekilde bağlantı yapıldığında, potansiyometremizin orta bacağında 0-5V arası değişen bir gerilim alabiliriz. Şimdi yazacağımız kodla bu gerilimin sayısal karşılığını seri porttan görebileceğiz:

```
int potPin = A0;
void setup()
{
  Serial.begin(9600);
  Serial.println("Analog input okuma ornegi.");
}
void loop()
{
  int deger  = analogRead(potPin);
  Serial.println(deger);
  delay(100);
}
```



NOT: Analog giriş pininden okuduğumuz değere göre çoklu LED yakma uygulamasını potansiyometre ekleyerek de yapabilirsiniz.



Örnek Kod:

```
int potPin = 0;
int latchPin = 5;
int clockPin = 6;
int dataPin = 4;
int leds = 0;


void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}

void loop()
{
  int deger  = analogRead(potPin);
  int yanan_ledsayisi = deger / 114;  //1023 / 9
  leds = 0;
  for (int i = 0; i < yanan_ledsayisi; i++)
  {
    bitSet(leds, i);
```

```
  }
  registeraYaz();
}

void registeraYaz()
{
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, LSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
}
```

Bu kod, potansiyometremizin voltaj değerini 8 LED için ayrı adımlara bölerek, gerilim arttıkça daha fazla LED'in yanmasını sağlıyor. Bu uygulamadaki potansiyometreyi, analog çıkışlı başka elemanlarla değiştirerek sizler de farklı uygulamalar yapabilirsiniz. Örneğin LM35 sıcaklık sensörüyle bir sıcaklık seviyesi göstergesi ya da bir mikrofon ile birlikte kullanacağınız preamfi devresiyle ses şiddeti göstergesi (vumetre) yapmak mümkün.

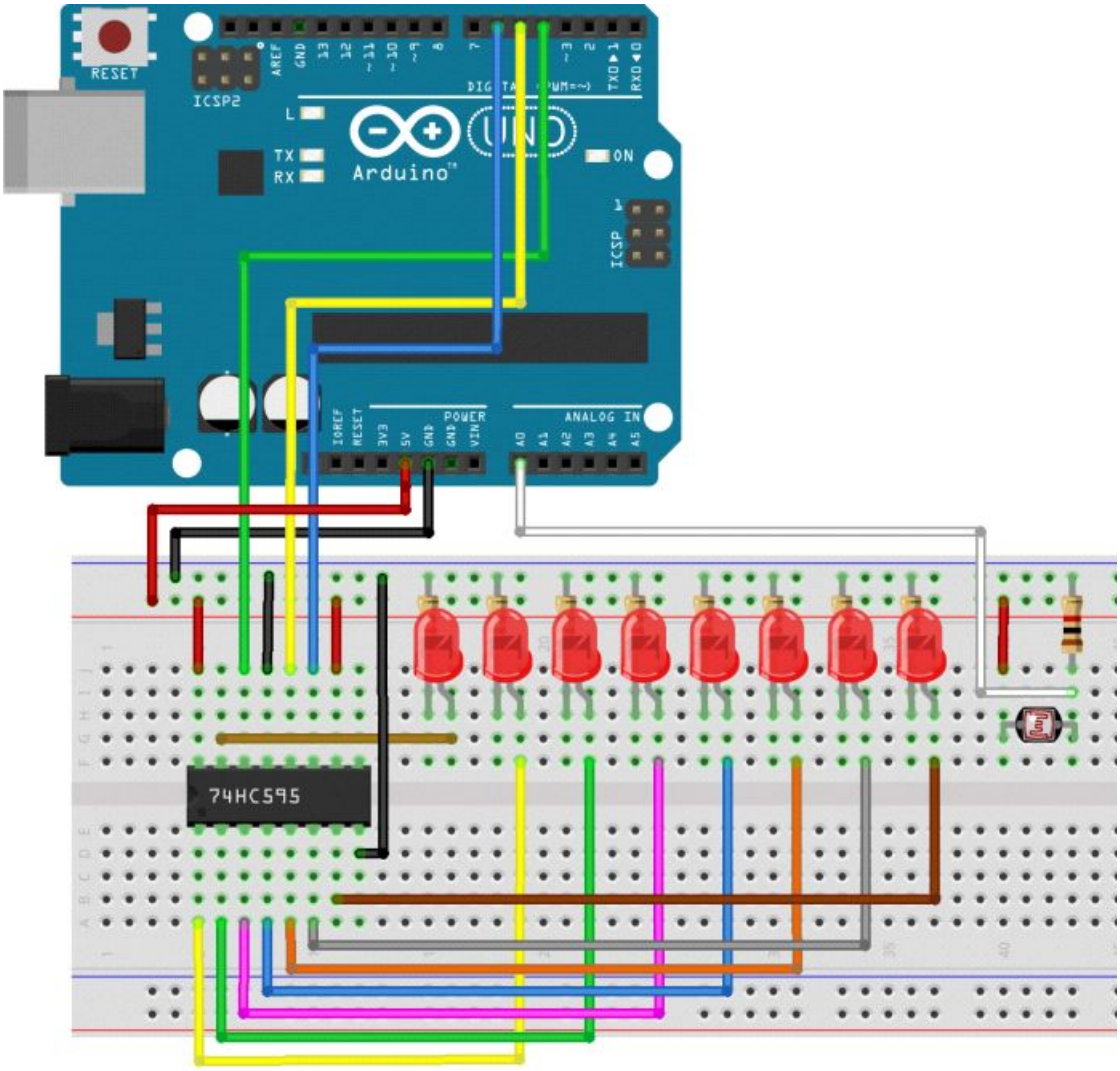
9.DERS

LDR İLE IŞIK ALGILAMA

Light Dependent Resistor (LDR, Foto direnç)

Foto direnç, üzerine düşen ışığın şiddetine bağlı olarak değişen dirence sahip bir elemandır. Direnci, üzerine düşen ışık miktarıyla ters orantılı olarak değişir. Gündelik kullandığımız çoğu elektronik alette "fotosel" ismiyle yaygın olarak kullanılır.

Bu uygulamadaki devre şeması:



Örnek Kod:

```
int potPin = 0;
int latchPin = 5;
int clockPin = 6;
int dataPin = 4;
int leds = 0;

void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
```

```
  pinMode(clockPin, OUTPUT);
}
void loop()
{
  int deger  = analogRead(potPin);
  int yanan_ledsayisi = deger / 114;  //1023 / 9
  leds = 0;
  for (int i = 0; i < yanan_ledsayisi; i++)
  {
    bitSet(leds, i);
  }
  registeraYaz();
}

void registeraYaz()
{
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, LSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
}
```

10.DERS

BUZZER İLE SES ÇIKIŞI ALMA

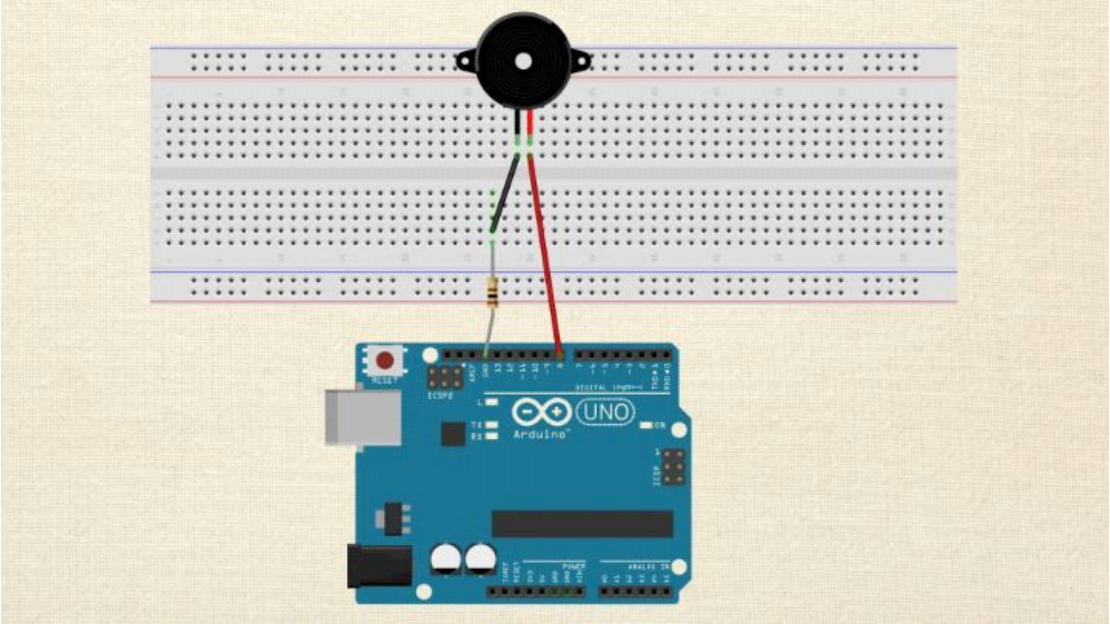
Buzzer dediğimiz devre elemanını ufak bir hoparlör olarak tanımlanabilir. Hoparlörler kadar yüksek ve detaylı ses üremeseler de, “bip” leme seslerini çıkartmada oldukça başarılıdırlar.

Notalar ve Frekans

Bildiğimiz gibi hemen hemen her sesin kendine ait bir notası vardır. Notaların da her birine ait belirli bir frekans değeri vardır. Frekans arttıkça çıkan ses tizleşir.

Bu uygulamada ilkokul müzik dersinden hatırladığımız do, re, mi, fa, sol, la ve si notalarını çalan bir kod hazırlayacağız. Notaların isim ve harf olmak üzere iki farklı gösterim şekli mevcuttur: C = do, D = re, E = mi, F = fa, G = sol, A = la ve B = si. “do” gibi sözcükler programlama dillerinde farklı komutlar tarafından kullanıldığı için, bu kodda notaların harf gösterimi kullanılır.

Devre şeması:



Örnek Kod:

```
int buzzerPin = 12;
int notaSayisi = 8;
int C = 262;
int D = 294;
int E = 330;
int F = 349;
int G = 392;
int A = 440;
int B = 494;
int C_ = 523;
int notalar[] = {C, D, E, F, G, A, B, C_};
void setup()
{
  for (int i = 0; i < notaSayisi; i++)
  {
    tone(buzzerPin, notalar[i]);
    delay(500);
    noTone(buzzerPin);
    delay(20);
  }
  noTone(buzzerPin);
}
```

```
void loop()
{
}
```

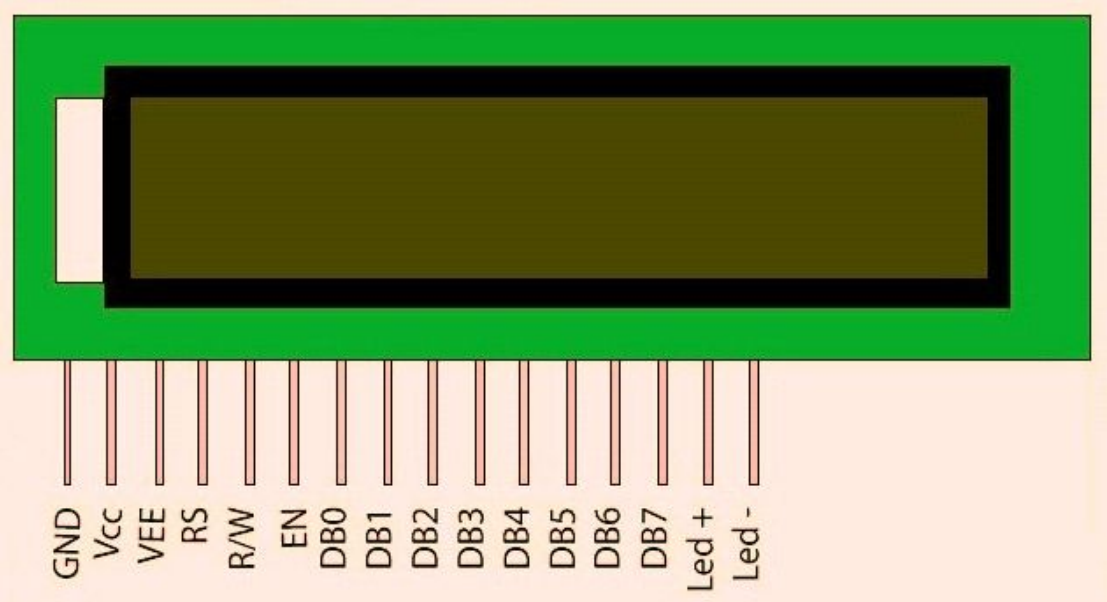
Kodun tamamı setup fonksiyonunda yer alıyor. Bunun anlamı, Arduino kartına enerji verdiğimizde ses sadece bir kere çıkacak ve daha sonra susacak. Tekrardan ses çıkmasını istiyorsak Arduino üzerindeki reset butonuna basmamız yeterli.

Kodu açıklamak gerekirse; her bir notanın frekans olarak karşılığı giriliyor. Daha sonra bu notları kalın do notasından ince do notasına sıralayarak bir array içine aldık. Bu array'in içeriğini ise bir for döngüsü kullanarak o array'in içeriğini okuttuk ve ses çıkışı için tone komutunu kullandık. Tone komutunu bir kez verdikten sonra Arduino, noTone komutuna gelene kadar çıkış vermeye devam eder.

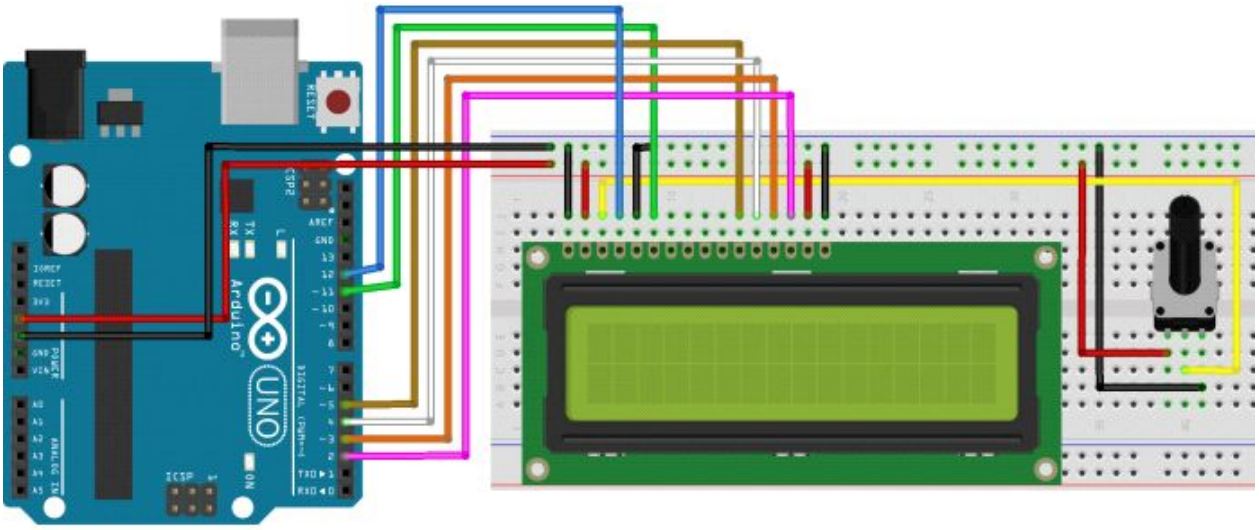
11.DERS

16×2 LCD EKRAN BAĞLAMA

LCD ekranda 16 adet pin bulunmakta. Kullanacağımız ekrana göre pinler ekranın üst, alt veya her iki tarafında da yer alabilir. Çok nadir olarak bazı ekranlarda ise arka aydınlatma ışığı bulunmadığından 14 adet pin yer almaktadır. 15 ve 16 numaralı pinler, ekran aydınlatması bulunan ekranlarda ışığı yakmak için kullanılır.



Bağlantı şeması:



Arduino programında LCD ekran için örnek kod mevcut. Bu kodu açmak için şu sırayı takip etmemiz gerekiyor:

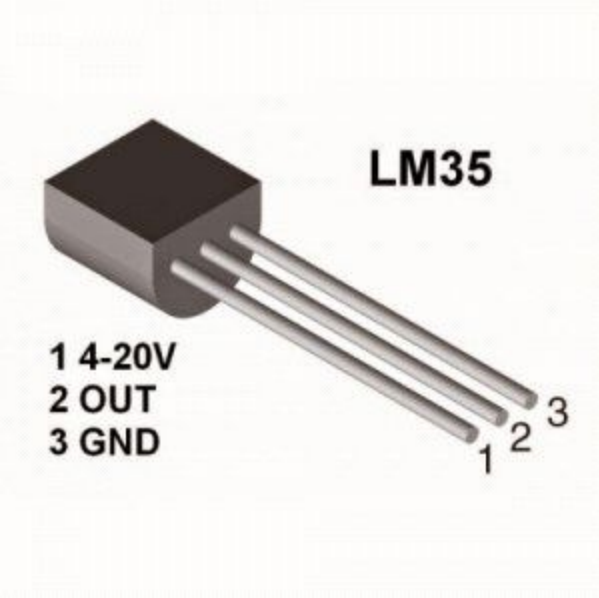
Dosya > Örnekler > LiquidCrystal > HelloWorld

Bu kod, ekranın ilk satırına “hello, world!”, ikinci satıra ise Arduino’ya enerji verildiğinden itibaren geçen süreyi saniye cinsinden yazıyor. Eğer ekranınızda herhangi bir görüntü alamıyorsanız, potansiyometreyi çevirerek ekranın kontrastını değiştirebilirsiniz.

12.DERS

SICAKLIK ÖLÇÜMÜ

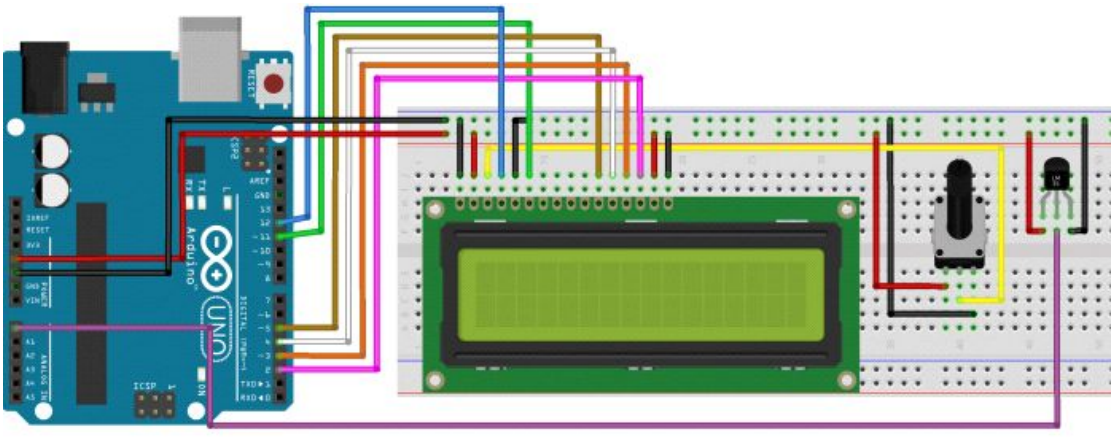
LM35 sıcaklık sensörü, hassas sıcaklık ölçümü yapan analog çıkışlı bir sıcaklık sensörüdür. 25 °C’de 0,5 °C hassasiyete sahip olan bu sensörü, Arduino’nun analog girişinden bağlayarak sıcaklık ölçümü yapılabilinmektedir.



Sensörün pin konfigürasyonu bu şekildedir. **TO-92 kılıf yapısına sahip bu sensörü transistör ile karıştırmak oldukça mümkündür, lütfen herhangi bir bağlantı yapmadan önce üzerinde yazanı okuyalım.** Bu sensör, 2 numaralı bacağından sıcaklıkla doğru orantılı olacak şekilde 0 ile 1V arasında gerilim çıkışı vermektedir. Arduino'un analog girişleri, biz aksini belirtmediğimiz sürece 0-5V arasında gerilimleri ölçmekte.

Eğer analog giriş aralığımızı 0-5V arasında bırakırsak, sensörün hassasiyet kabiliyetinin %80'lik kısmı boşa gitmiş olacak. Bunu engellemek için Arduino'muzun analog ölçümünü 0 ile 1,1V arasında yapacak şekilde ayarlamamız gerekli.

Bağlantı Şeması:



Örnek Kod:

```
#include <LiquidCrystal.h>

int lm35Pin = A0;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
  lcd.begin(16, 2);
  analogReference(INTERNAL);
  lcd.print("Sicaklik Olcumu:");
}

void loop()
{
  int sicaklikVolt = analogRead(lm35Pin);
  float sicaklikC = sicaklikC = sicaklikVolt / 9.31;
  lcd.setCursor(0, 1);
  lcd.print(sicaklikC);
  lcd.setCursor(6,1);
  lcd.print("\337C");
  delay(100);
}
```

Setup fonksiyonunda bulunan **analogReference(INTERNAL)**; komutu, Arduino'muzun analogda dijitale çeviricisinin 1,1V referans voltajı kullanmasını sağlıyor. Bu sayede, analog girişten okuyacağımız 0-1023 arası değerlerin her bir adımı 1,0742 mV (milivolt) gerilime denk düşüyor. LM35 sıcaklık sensörünün çıkış bacağında okuduğumuz her 10 mV, 1 °C sıcaklığa denk geldiğinden; 10 / 1,0742 bize yaklaşık olarak 9,31 değerini veriyor. Yani bu demek oluyor ki, analog girişten ölçtüğümüz değeri 9,31 ile çarparsak, elde ettiğimiz sonuç sensörün verdiği çıkış geriliminin santigrat cinsinden karşılığını elde ediyoruz.

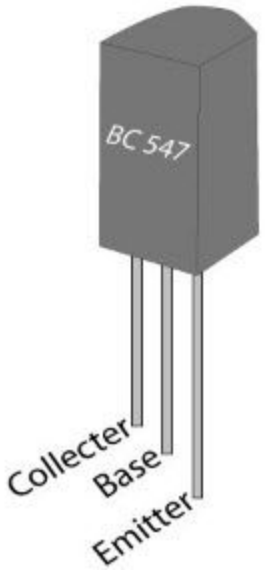
lcd.setCursor(0,1) komutu, LCD ekranımızın 1 numaralı satırının (ilk satır 0 olarak kabul edilir, yani 1 numaralı satır alt satır olmuş oluyor) 0'ıncı karakterine istediğimiz sonucu yazmamızı sağlıyor. Aynı şekilde ##.## şeklindeki sıcaklık değeri 5 karakter yer tuttuğu için bir karakter boşluk vererek 6'ıncı karaktere de santigrat derece sembolünü koyuyoruz (°C). Eğer sıcaklık ölçümü çok hızlı değişiyorsa, **delay()** komutunun alacağı parametre ile her bir ölçüm arasındaki süreyi arttırarak daha sabit bir değer elde edebiliriz.

13.DERS

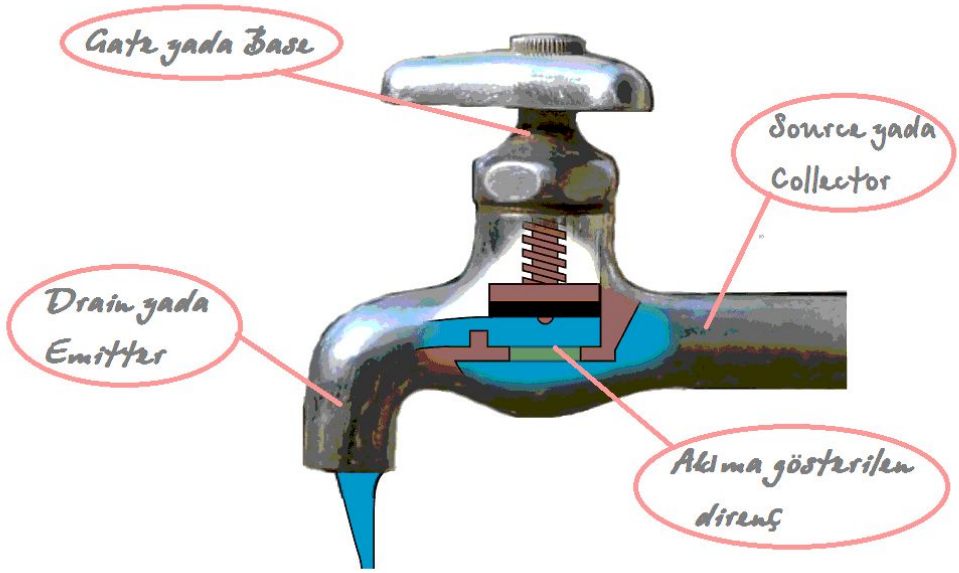
DC MOTOR HIZ KONTROLÜ

Bu uygulamada BJT tipi bir transistör kullanarak DC motorumuzun hızını kontrol edeceğiz. Peki neden bir transistöre ihtiyacımız var?

Arduino'muzun dijital pinlerinden alacağımız çıkış geriliminin 5V olduğunu biliyoruz. 9V DC motorumuz 5V gerilimle çalışabilir fakat motorun çekeceği akım, Arduino pinlerinden çekebileceğimiz en yüksek akım değeri olan 40 mA'in oldukça üzerinde olacaktır. Burada transistör devreye giriyor.

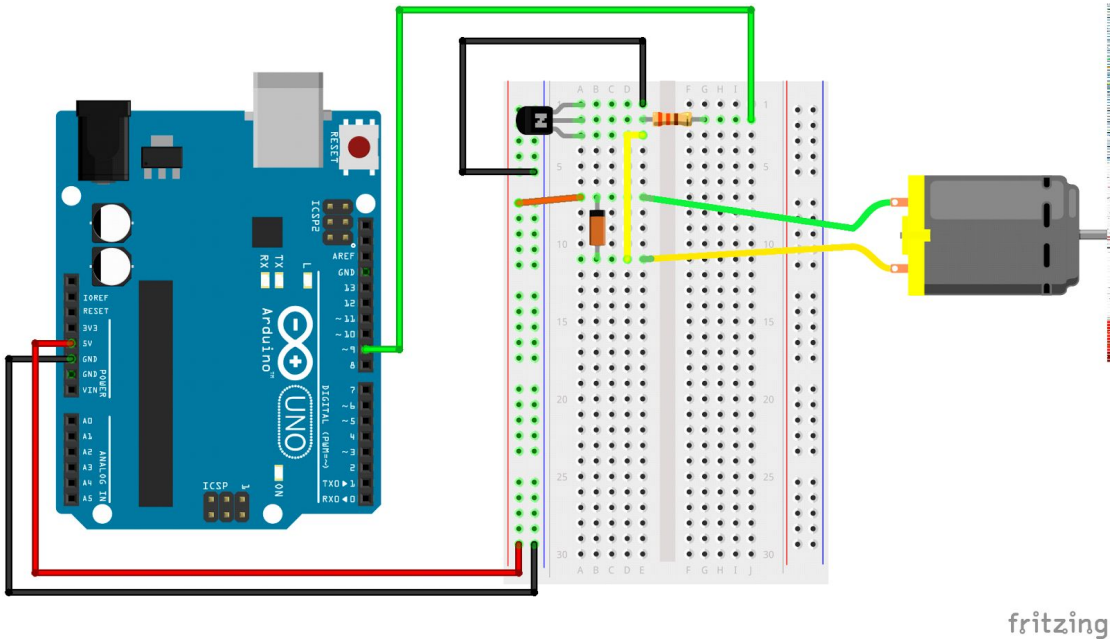


Transistörü çok basit bir şekilde elektrik akımı veya gerilimi ile kontrol edebileceğimiz bir switch olarak düşünebiliriz. Bunu anlamak için elimizdeki NPN tipi transistörü bir musluk olarak düşünelim: transistörün kolektör (collector) bacağından emitör (emitter) bacağına doğru bir boru içerisinden su aktığını hayal edelim. Biz, ortada bulunan baz bacağından vereceğimiz akım ile tıpkı bir musluğun suyuk akışını kısıp arttırması gibi kolektörden emitöre akan akım miktarını kontrol edebiliriz:



Bu sayede çok küçük akımlar kullanarak, büyük akım çeken cihazları kontrol edebilmemiz mümkün olur.

Devre şeması:



Devredeki diyotun görevi, motorun durur vaziyetten harekete geçtiği anda oluşan gerilim

sıçramalarından transistörü korumak içindir.

NOT: Bağlantıları yapmadan önce Arduino'ya kodu yükleyin. Bağlantıları yaptıktan sonra ise USB bağlantısı yerine Arduino üzerinde bulunan güç girişinden 9V adaptör veya 9V pil ile kartımızı besleyin. Aksi takdirde bilgisayarımızın USB portuna veya Arduino'ya zarar verebilirsiniz!

Bu kod ile motorun hızı önce kademeli olarak artacak; en yüksek değere ulaştıktan sonra ise aynı şekilde azalacak:

```
int motorPin = 9;
int hiz = 0;

void setup()
{
  pinMode(motorPin, OUTPUT);
}

void loop()
{
  for(hiz = 0; hiz <= 255; hiz++)
  {
    analogWrite(motorPin,hiz);
    delay(20);
  }
  for(hiz = 255; hiz>=0; hiz--)
  {
    analogWrite(motorPin,hiz);
    delay(20);
  }
}
```

Örnek Kod:

```
int motorPin = 9;

void setup()
{
  pinMode(motorPin, OUTPUT);
  Serial.begin(9600);
  while (! Serial);
  Serial.println("0 ile 255 arasi hiz giriniz");
}

void loop()
{
  if (Serial.available())
  {
    int hiz = Serial.parseInt();
    if (hiz >= 0 && hiz <= 255)
    {
      analogWrite(motorPin, hiz);
```

```
      }
    }
}
```

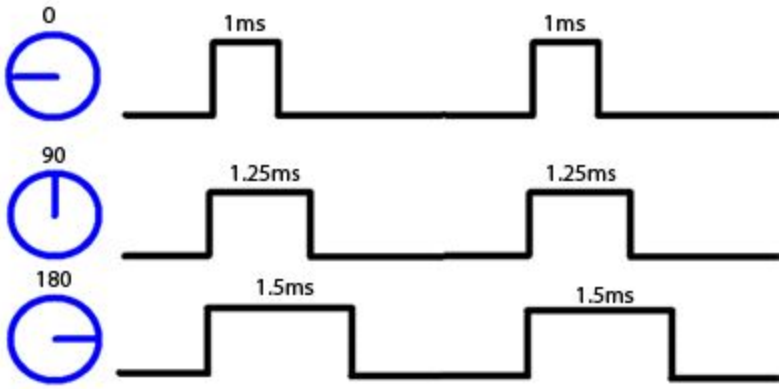
Bu kod sayesinde, motorumuzun hızını bilgisayardaki Arduino programının seri port ekranından gireceğimiz değerle kontrol edebilirsiniz. **Motor gibi fazla akım çekebilen cihazlarla çalışırken 9V adaptör veya 9V pil ile Arduino'yu güç girişinden beslemeyi unutmayın.**

14.DERS

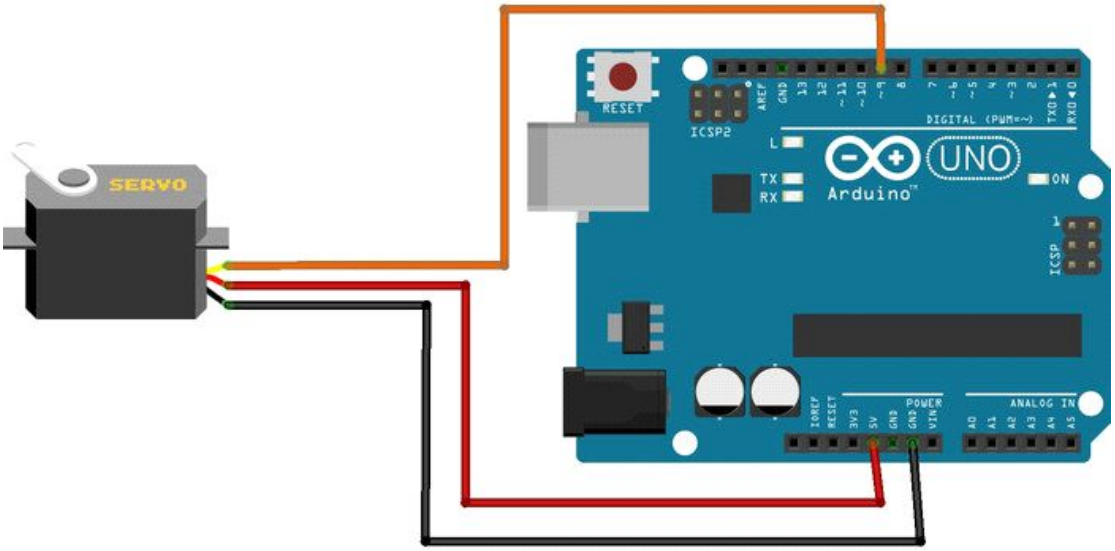
SERVO MOTOR KONTROLÜ

Servo motorlar, RC (radio controlled, uzaktan kumandalı) araba, helikopter ve uçak gibi araçlarda kullanılır. Bu motorlar, DC motorlardan farklı olmak üzere istediğimiz pozisyonda sabit kalacak şekilde tasarlanmıştır. Çoğunlukla 0-180 derece arası açılarda çalışırlar. RC arabamızın direksiyonunda, helikopterlerin pervanelerine açı vermede ve uçakların kontrol yüzeylerini hareket ettirmede kullanılırlar. PWM sinyal ile çalışırlar.

Arduino'da analogWrite() komutu ile aldığımız PWM sinyalinin sadece 5V seviyesinde kaldığı süreyi (duty cycle) değiştirerek farklı analog sinyaller elde edilir. Servo motorlarda ise yine benzer şekilde 20 ms'lik sinyalin açık kaldığı periyodu 1 ms ile 2 ms arasında değiştirerek, servo motoru 0 dereceden 180 derece arasında istediğimiz konuma ayarlayabilirsiniz.



Bağlantı Şeması:



Uygun koda Arduino programından

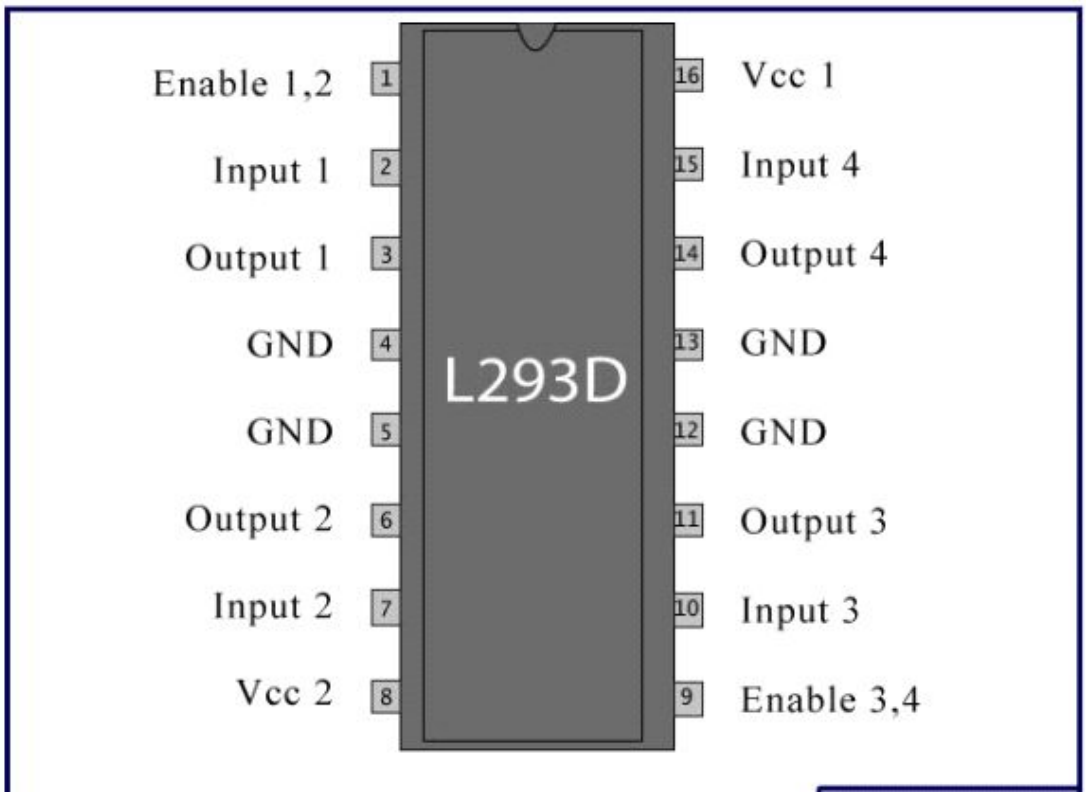
Dosya > Örnekler > Servo > Sweep

adımlarını takip ederek ulaşabilirsiniz. Bu kod, servoyu önce 0 dereceden 180 dereceye adım adım ilerletecek; 180 dereceye ulaşınca da tekrardan 0 dereceye adım adım geri döndürecektir.

15.DERS

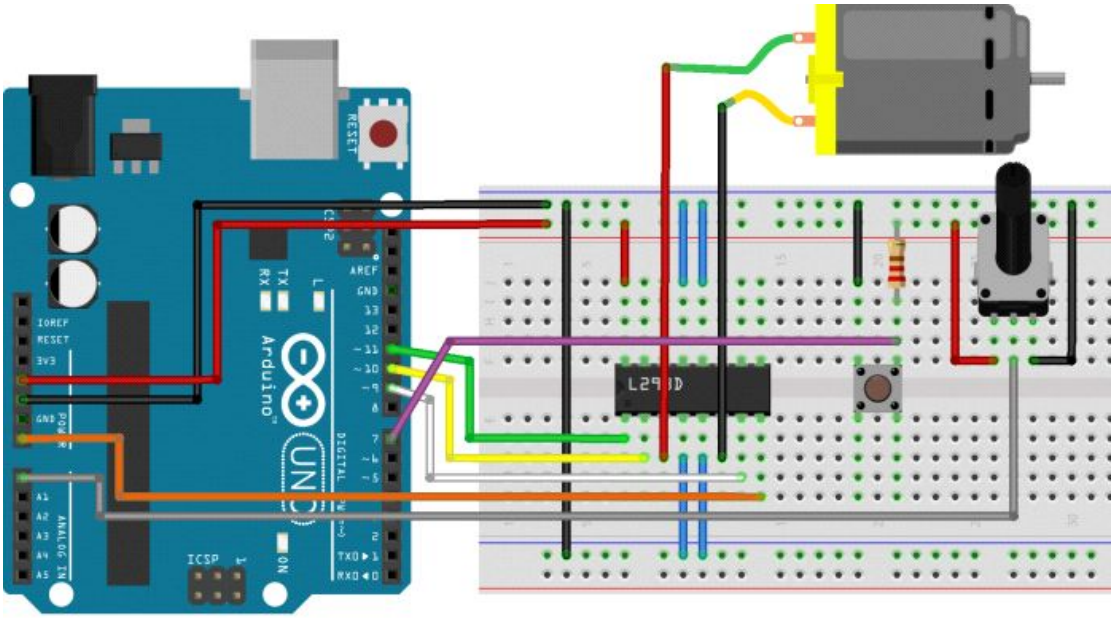
DC MOTOR HIZ VE YÖN KONTROLÜ

Yön kontrolü yapabilmemiz için bir motor sürücü entegresine ihtiyacımız var. L293D entegresi, 2 adet DC motor veya 1 adet step motor sürmek için kullanılan oldukça popüler bir entegredir.



Bu entegrenin input 1 ve input 2 girişleri, motorun döneceği yönü; enable pini ise hangi çıkışların aktif olacağını kontrol ediyor. Enable pinine uygulayacağınız PWM sinyal, motorların hızını değiştirmemize olanak sağlıyor.

Devre şeması:



Örnek Kod:

```
int enablePin = 11;
int in1Pin = 10;
int in2Pin = 9;
int butonPin = 7;
int potPin = 0;

void setup()
{
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  pinMode(enablePin, OUTPUT);
  pinMode(butonPin, INPUT);
}



void loop()
{
  int hiz = analogRead(potPin) / 4;
  boolean ters = digitalRead(butonPin);
  motorCalistir(hiz, ters);
}

void motorCalistir(int hiz, boolean ters)
{
  analogWrite(enablePin, hiz);
  digitalWrite(in1Pin, ! ters);
  digitalWrite(in2Pin, ters);
}
```

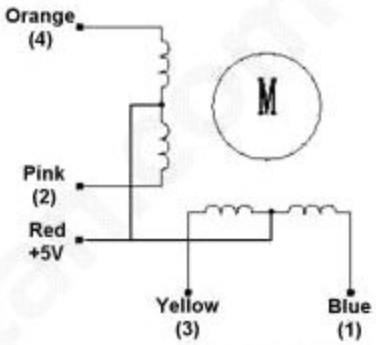
Entegremizin enable bacağına analogWrite() fonksiyonu sayesinde PWM sinyal göndererek hız kontrolü yapmış olursunuz.. Butona bastığımızda input 1 ve input 2 pinlerine giden sinyaller tersine çevrilir. Böylece motorun döndüğü yön ters çevrilmiş olur.

16.DERS

STEP MOTOR KONTROLÜ

Step motorlar, hassas hareket gerektiren uygulamalarda kullanılan motorlardır. Fonksiyon olarak DC motorlar ile servo motorların her ikisinin de karakterini gösterir: istenildiğinde belirli bir konuma adım adım ilerleyebilir veya sürekli olarak istenilen yönde döndürülebilir. Normal yazıcılar ve 3B yazıcıların vazgeçilmez parçalarındandırlar.

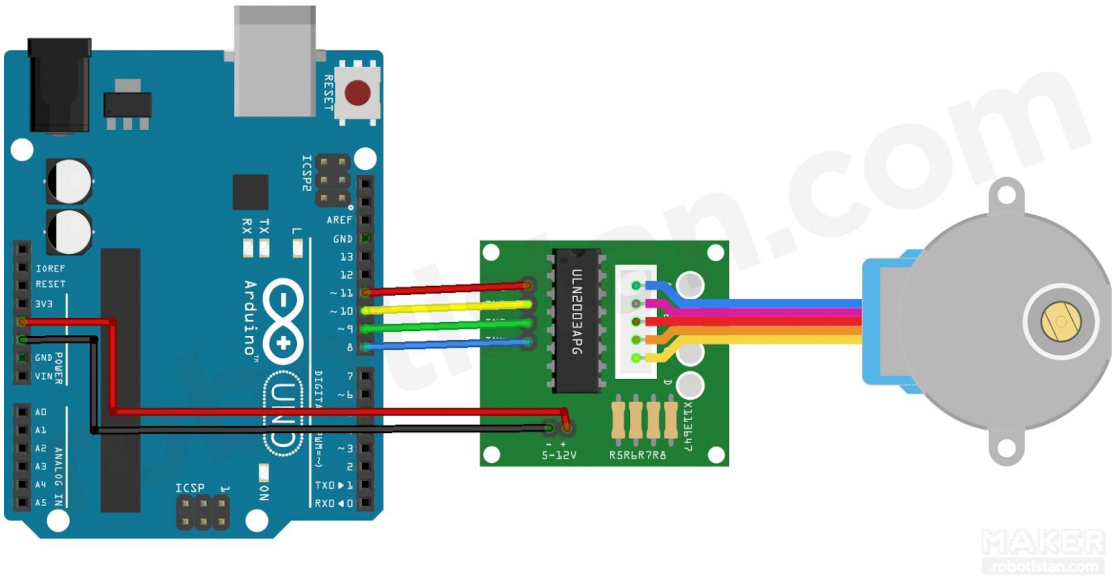
Step motorlar, bipolar ve unipolar olmak üzere iki çeşittir: bipolar step motorlar 4 kablolu, unipolar step motorlar ise 4,5,6 ya da 8 kablolu olabilirler.



Half-Step Switching Sequence

Lead Wire Color	---> CW Direction (1-2 Phase)							
	1	2	3	4	5	6	7	8
4 Orange	-	-						-
3 Yellow		-	-	-				
2 Pink				-	-	-		
1 Blue						-	-	-

Step motor genellikle sürücü kartıyla birlikte kullanılmaktadır. Bu sayede breadboard'a ihtiyaç duymadan kolaylıkla bağlantı yapmanız mümkündür.



Bu uygulama için hâlihazırda bulunan bir kütüphane mevcuttur. Bu kütüphaneyi indirmek için şu linki tıklayınız: https://github.com/tardate/X113647Stepper/archive/master.zip

İndirme işlemi tamamlandığında, zip dosyasını açıyor ve **X113647Stepper-master** isimli klasörü **C:\Program Files (x86)\Arduino\libraries** klasörünün altına kopyalıyoruz. (Eğer bilgisayarınızda 32-bit Windows yüklü ise **C:\Program Files\Arduino\libraries** klasörünü kullanın.)

Arduino programını açıp **Dosya > Örnekler > X113647Stepper-master > FullSweep** adımlarını takip edin ve bu kodu karta yükleyin. Kod, step motoru önce tam tur bir yönde döndürecek, daha sonra zıt yönde tekrardan tam tur döndürecektir. Koddaki **myStepper.setSpeed()** fonksiyonunun değerini değiştirerek motorun dönme hızını ayarlayabilirsiniz.

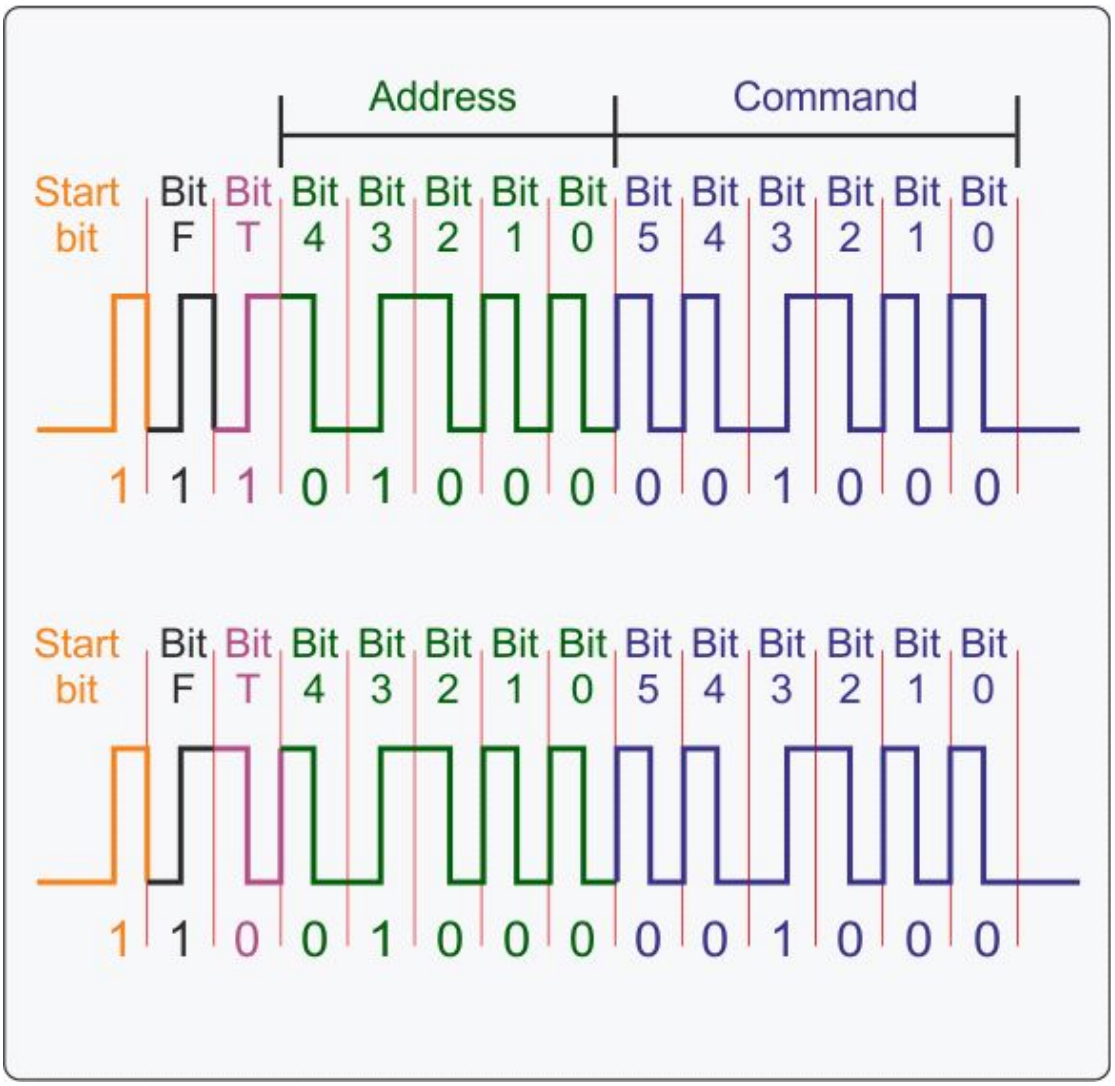
17.DERS

KIZILÖTESİ KUMANDA KULLANIMI

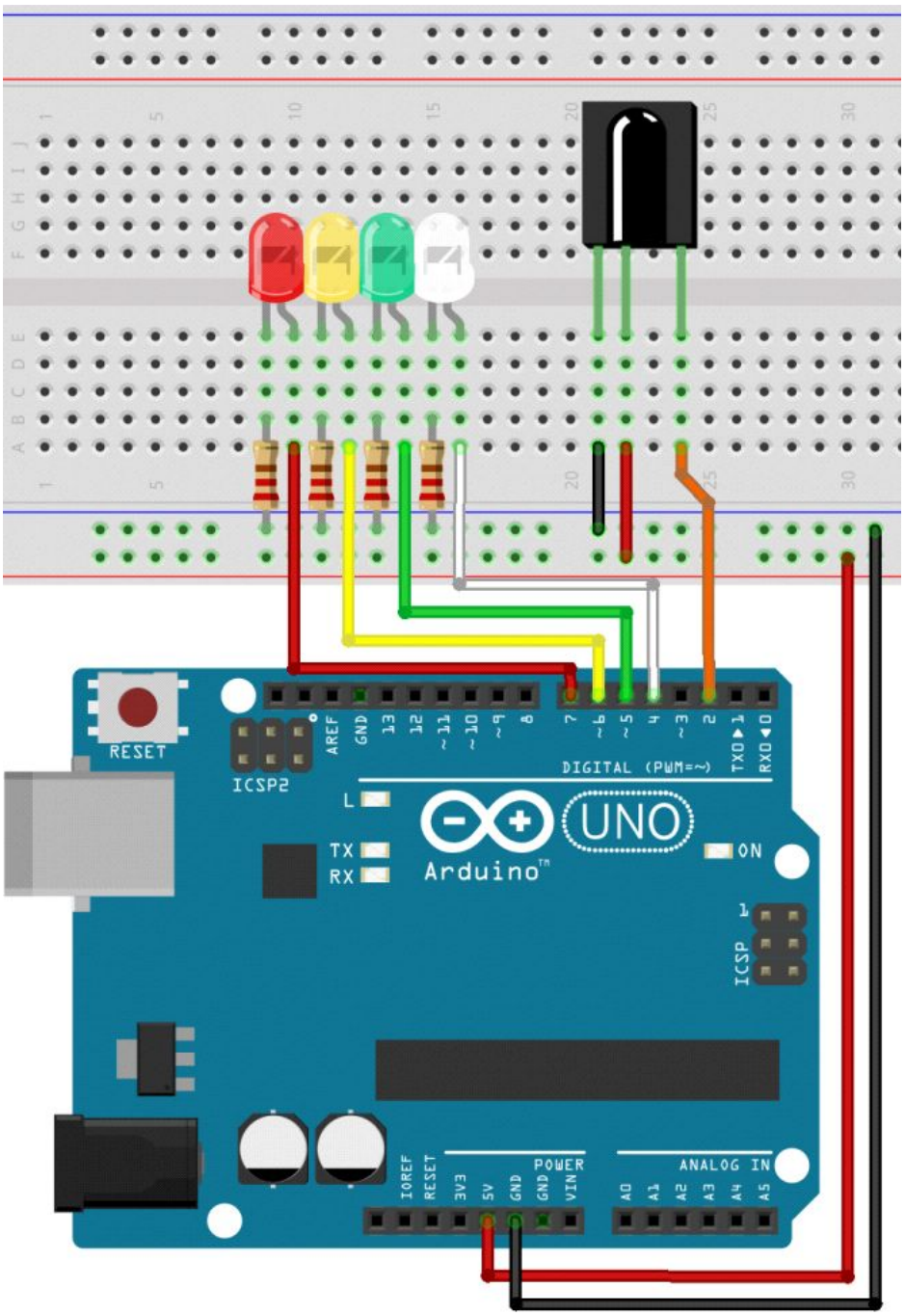
Kızılötesi kumandalar, günlük kullandığımız çoğu elektronik alette sık sık kullandığımız cihazlardır.

Çalışma prensibi şu şekildedir: kumanda üzerinde bir kızılötesi (infrared, ir) LED bulunur. Bu LED, kumanda üzerindeki herhangi bir tuşa bastığımızda önceden belirlenmiş bir kod verecek şekilde belirli bir frekansta yanıp söner. Çoğu kumanda için bu frekans 38 kHz'tir. 38 kHz'lik taşıyıcı sinyale her bir tuş için farklı bir kod oluşturacak şekilde modülasyon uygulanır. Her marka için farklı modülasyon ve kodlama teknikleri mevcuttur. Projede kullandığımız 38 kHz

kızılötesi alıcı, aldığı sinyali demodüle ederek Arduino'ya doğrudan basılan buton ile ilgili kodu göndermektedir. Bu sayede farklı marka ve model kumandaları 38 kHz taşıyıcı sinyale sahip olduğu sürece bu alıcı ile kullanabilmekteyiz.



Devre Şeması:



Kızılötesi alıcıyı kullanabilmemiz için, Arduino yazılımına Arduino-IRremote kütüphanesini yüklememiz gereklidir. Bu kütüphaneyi indirdikten sonra C:\Program Files (x86)\Arduino\libraries veya C:\Program Files\Arduino\libraries klasörünün altına “IRremote” ismiyle kaydetmemiz gereklidir.

Örnek Kod:

```
#include <IRremote.h>
int RECV_PIN = 2;
IRrecv irrecv(RECV_PIN);
 decode_results results;
 #define CH1 0xFFA25D
 #define CH 0xFF629D
#define CH2 0xFFE21D
 #define PREV 0xFF22DD
 #define NEXT 0xFF02FD
 #define PLAYPAUSE 0xFFC23D
 #define VOL1 0xFFE01F
 #define VOL2 0xFFA857
 #define EQ 0xFF906F
 #define BUTON0 0xFF6897
 #define BUTON100 0xFF9867
 #define BUTON200 0xFFB04F
 #define BUTON1 0xFF30CF
 #define BUTON2 0xFF18E7
#define BUTON3 0xFF7A85
 #define BUTON4 0xFF10EF
#define BUTON5 0xFF38C7
 #define BUTON6 0xFF5AA5
 #define BUTON7 0xFF42BD
 #define BUTON8 0xFF4AB5
#define BUTON9 0xFF52AD
int led1 = 7;
 int led2 = 6;
 int led3 = 5;
int led4 = 4;
void setup() {
 pinMode(led1, OUTPUT);
 pinMode(led2, OUTPUT);
 pinMode(led3, OUTPUT);
```

```
 pinMode(led4, OUTPUT);
 Serial.begin(9600);
irrecv.enableIRIn();
 }
void loop() {
if (irrecv.decode(&results))
{
if (results.value == BUTON1)
 {
digitalWrite(led1, !digitalRead(led1));
 if (digitalRead(led1) == HIGH)
{ Serial.println("LED 1 yandi");
 }
else
{
Serial.println("LED 1 sondu");
}
 }
 if (results.value == BUTON2)
{
digitalWrite(led2, !digitalRead(led2));
 if (digitalRead(led2) == HIGH)
 {
Serial.println("LED 2 yandi");
}
 else
 {
Serial.println("LED 2 sondu");
 }
 }
 if (results.value == BUTON3)
 {
digitalWrite(led3, !digitalRead(led3));
```

```
 if (digitalRead(led3) == HIGH)
{
Serial.println("LED 3 yandi");
 }
else
{
Serial.println("LED 3 sondu");
 }
 }
 if (results.value == BUTON4)
{
 digitalWrite(led4, !digitalRead(led4));
 if (digitalRead(led4) == HIGH)
 {
Serial.println("LED 4 yandi");
 }
 else
{
 Serial.println("LED 4 sondu");
 }
 }
 if (results.value == BUTON0)
{
 digitalWrite(led1, LOW);
 digitalWrite(led2, LOW);
 digitalWrite(led3, LOW);
 digitalWrite(led4, LOW);
 Serial.println("Tum LED'ler sondu");
}
if (results.value == BUTON5)
{
digitalWrite(led1, HIGH);
 digitalWrite(led2, HIGH);
```

```
 digitalWrite(led3, HIGH);
 digitalWrite(led4, HIGH);
 Serial.println("Tum LED'ler yandi");
}
irrecv.resume();
 }
}
```

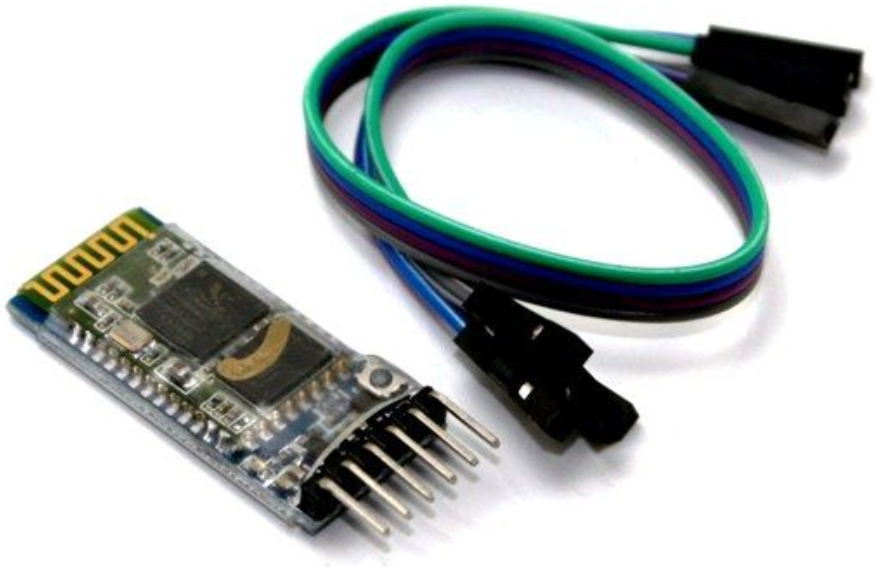
Bu koda göre, kumandamız üzerindeki 1, 2, 3 ve 4 numaralı butonlar sırasıyla 7, 6, 5 ve 4 numaralı dijital pinlere bağlı olan LED'leri yakıp söndürecektir. 0 tümünü söndürür ve 5 tümünü yakar. Ayrıca seri port ekranını açarsak, butona bastığımızda LED'in değişimini bu ekrandaki ifadelerden de görebiliriz. Sizler de kodun baş kısmındaki #define ile tanımlanan diğer buton kodlarını kullanarak farklı işlevler gerçekleştirecek şekilde programlama yapabilirsiniz.

Kullanılan kumandalar farklılaştıkça Arduino programından **Dosya>Örnekler>IRremote>IRrecvDump** örnek kodunu kullanarak mevcut kumandanızın tuşlarına ait hex kodları tespit edebilir ve bu kodları kullanabilirsiniz.

18.DERS

HC-05 BLUETOOTH MODÜLÜ KULLANIMI

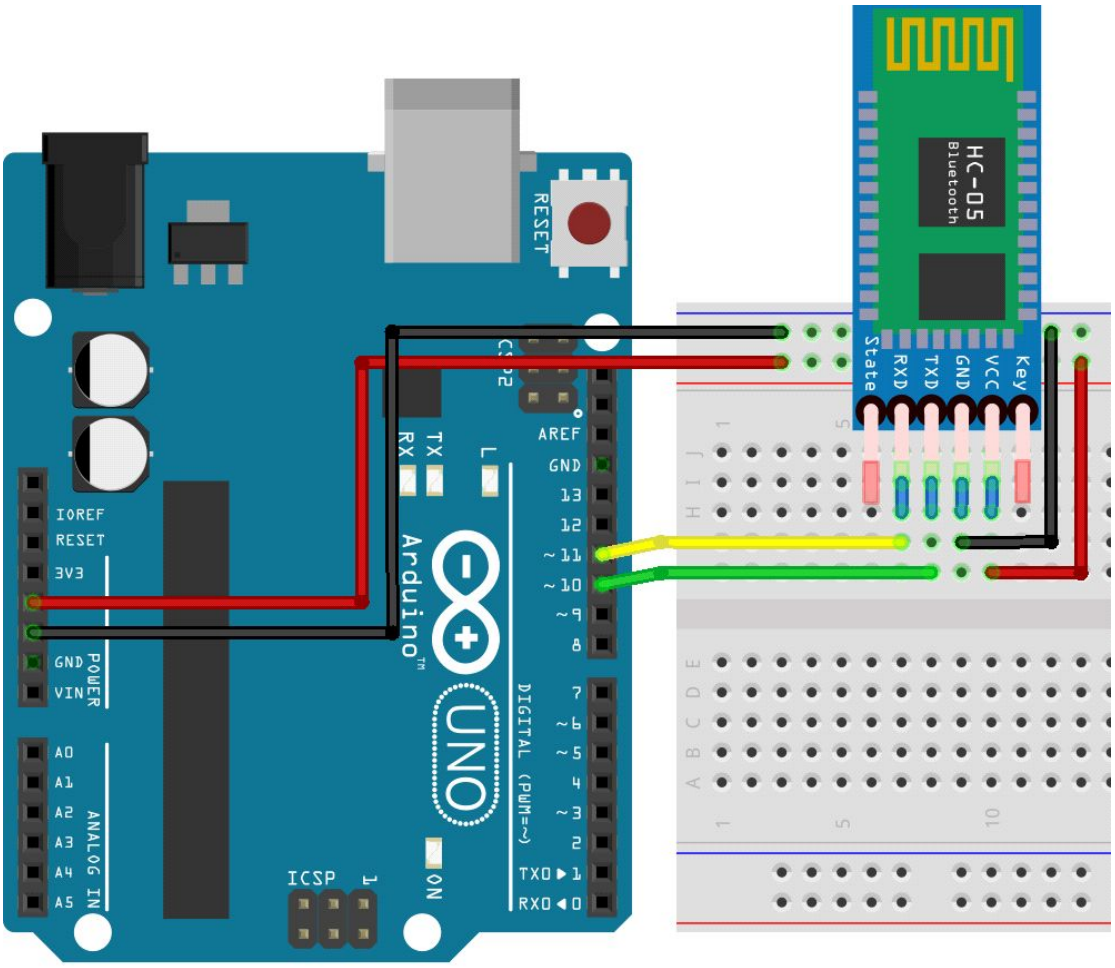
Bluetooth, cep telefonlarımızdan kulaklıklarımıza kadar neredeyse kablosuz iletişim yeteneğine sahip tüm cihazlarda var olan bir teknolojidir. Arduino projelerinde de bluetooth ekleyebilmemiz için piyasada çeşitli modüller bulunmaktadır. Bunlardan en uygun fiyatlı ve en kullanışlı olanı HC-05 modelidir.



HC-05 görünüm olarak HC-06 modülü ile neredeyse tamamen aynıdır. Temelde aynı işlevi görseler de HC-05 modelini kullanarak iki adet HC-05 veya HC-06 modelleri arasında doğrudan iletişim kurulabilmektedir. HC-05'i ayırt etmek için basit bir yöntem vardır: çoğu HC-05 modülün üzerinde ufak bir buton bulunmaktadır. HC-06'da ise bu buton mevcut değildir.

Bluetooth modülünü Arduino'muza bağladığımızda ilk modül ismi, baud rate ve şifre ayarlarını yapmamız bizim için büyük bir kolaylık olacaktır. HC-05 bluetooth modülünü konfigürasyon moduna geçirebilmek için 5V bağlantısını yaptığımız sırada modül üzerindeki butonu basılı tutmamız gereklidir. Konfigürasyon moduna girdiğimizi, modül üzerinde yanan LED'in sıklığından anlayabiliriz. Eğer 3'er saniyelik aralıklarla yanıp sönüyorsa, modül konfigurasyon modundadır. LED'in yanıp sönmesi sık ise bu bize modülün iletişim modunda olduğunu gösterir. Modül iletişim modundayken diğer bluetooth cihazlar tarafından yapılan taramalarda listelenir. İletişim modunda bir cihaz modüle bağlandığında ise LED, 3 saniyede bir kere kısa yanıp sönme yapar.

Modülümüzün konfigürasyonunu kolay bir şekilde yapabilmek için aşağıdaki şemaya göre bağlantı yapmamız ve 5V ve GND pinlerini Arduino'ya takarken modül üzerindeki butonu basılı tutmamız gerekir.

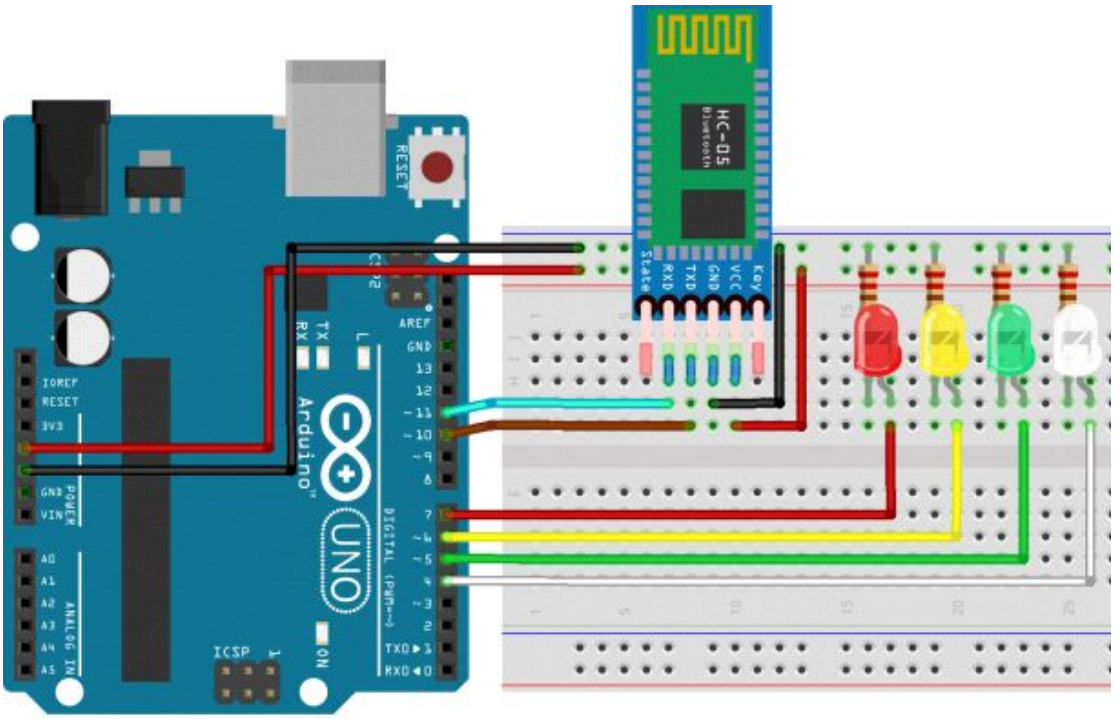


Konfigürasyon için kod:

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX
String isim = "Arduino UNO";
int sifre = 1234;
String uart = "9600,0,0";

void setup() {
  Serial.begin(9600);
  Serial.println("HC-05 Modul Ayarlaniyor...");
  Serial.println("Lutfen 5 sn icinde HC-05 modulun uzerindeki butona basili
tutarak baglanti yapiniz.");
  mySerial.begin(38400);
  delay(5000);
  mySerial.print("AT+NAME=");
  mySerial.println(isim);
  Serial.print("Isim ayarlandi: ");
```

```
  Serial.println(isim);
  delay(1000);
  mySerial.print("AT+PSWD=");
  mySerial.println(sifre);
  Serial.print("Sifre ayarlandi: ");
  Serial.println(sifre);
  delay(1000);
  mySerial.print("AT+UART=");
  mySerial.println(uart);
  Serial.print("Baud rate ayarlandi: ");
  Serial.println(uart);
  delay(2000);
  Serial.println("Islem tamamlandi.");
}

void loop()

{

}
```

Kodda bulunan isim, sifre ve uart değişkenleri ile modülü istediğiniz şekilde ayarlayabilirsiniz. Sırada, asıl devremiz var:



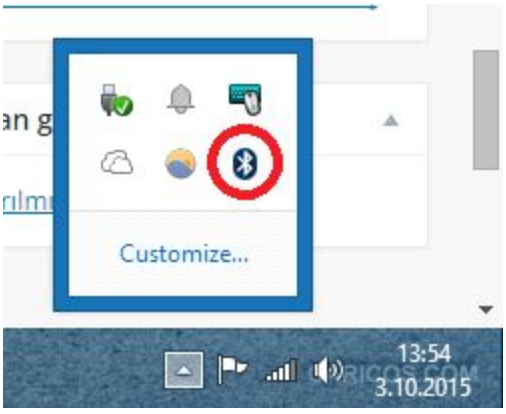
Kodumuz:

```
#include <SoftwareSerial.h>
```

```
SoftwareSerial mySerial(10, 11); // RX, TX
int led1 = 7;
int led2 = 6;
int led3 = 5;
int led4 = 4;

void setup()
{
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  mySerial.begin(9600);
  mySerial.println("LED uygulamasi");
}

void loop()
{
  char ch = mySerial.read();
  if (ch == 'q')
  {
    digitalWrite(led1, !digitalRead(led1));
    if (digitalRead(led1) == HIGH)
    {
      mySerial.println("LED 1 yandi");
    }
    else
    {
      mySerial.println("LED 1 sondu");
    }
  }
  if (ch == 'w')
  {
    digitalWrite(led2, !digitalRead(led2));
    if (digitalRead(led2) == HIGH)
    {
      mySerial.println("LED 2 yandi");
    }
    else
    {
      mySerial.println("LED 2 sondu");
    }
  }
  if (ch == 'e')
  {
    digitalWrite(led3, !digitalRead(led3));
    if (digitalRead(led3) == HIGH)
    {
      mySerial.println("LED 3 yandi");
    }
    else
    {
      mySerial.println("LED 3 sondu");
    }
```

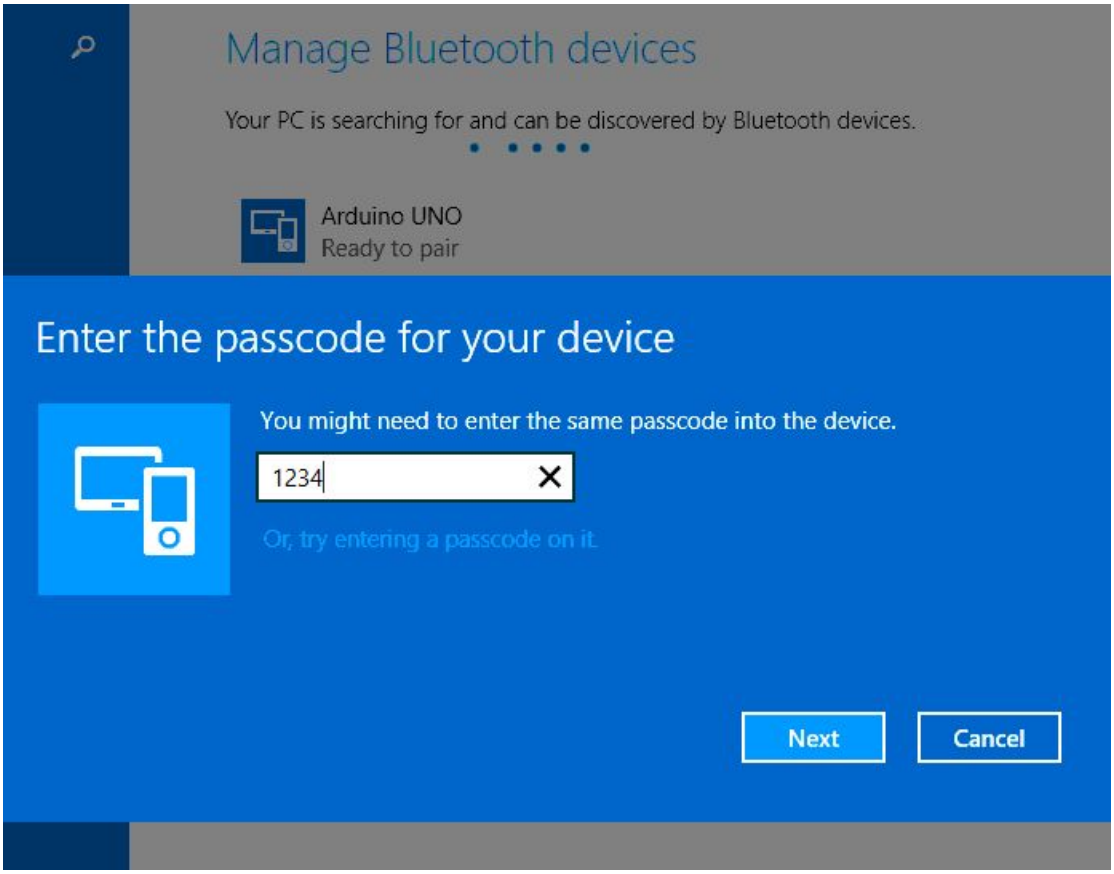
```
  }
  if (ch == 'r')
  {
    digitalWrite(led4, !digitalRead(led4));
    if (digitalRead(led4) == HIGH)
    {
      mySerial.println("LED 4 yandi");
    }
    else
    {
      mySerial.println("LED 4 sondu");
    }
  }
  if (ch == 'z')
  {
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
    mySerial.println("Tum LED'ler sondu");
  }
  if (ch == 'x')
  {
    digitalWrite(led1, HIGH);
    digitalWrite(led2, HIGH);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, HIGH);
    mySerial.println("Tum LED'ler yandi");
  }
}
```

Sıra geldi bluetooth modülümüzü Windows bilgisayarımıza tanıtmaya. Sistem tepsisinde bulunan "Bluetooth Aygıtları" simgesine çift tıklayarak açıyoruz.

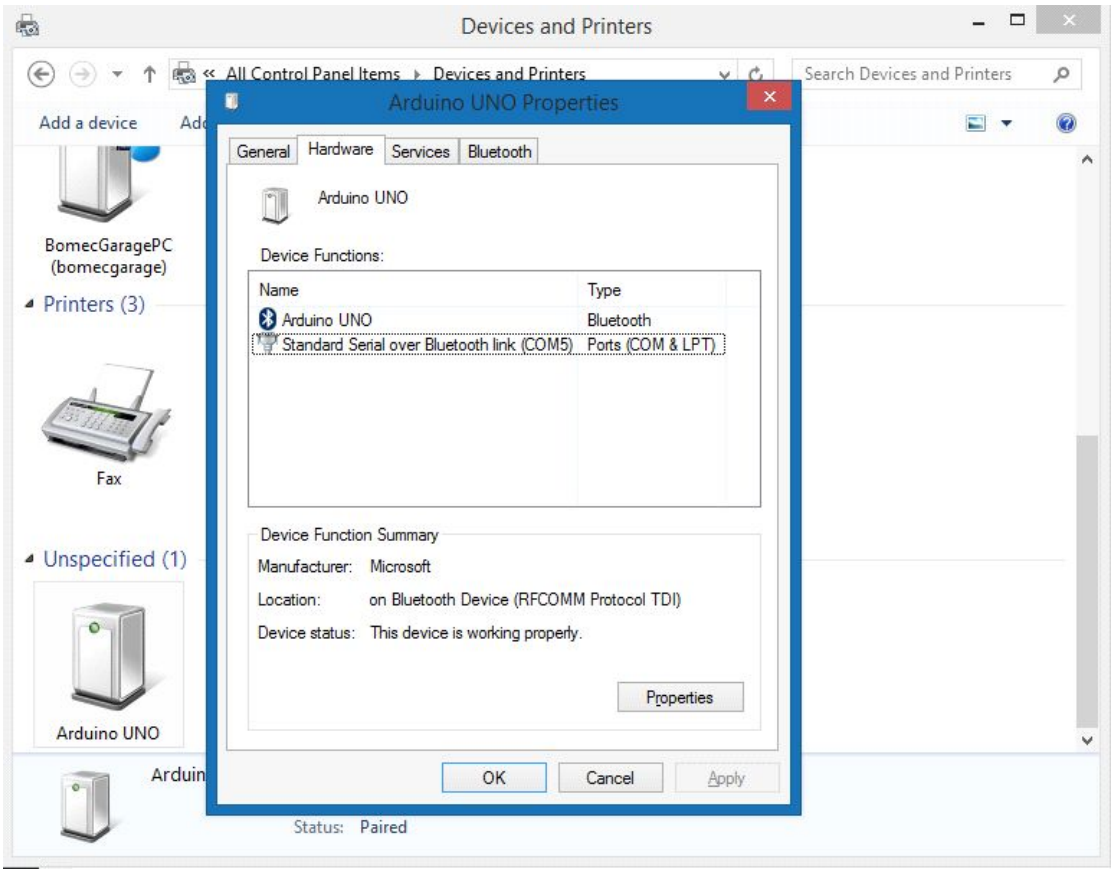


Arduino'muza bluetooth modülünün takılı olduğundan ve modülün iletişim modunda olduğundan emin oluyoruz (LED'in hızlıca yanıp sönüyor olması gerek). Bilgisayarımız tarama yaparak bluetooth modülümüzü buluyor. Burada, isim olarak konfigürasyon programında yazdığımızı ismi görmeliyiz.

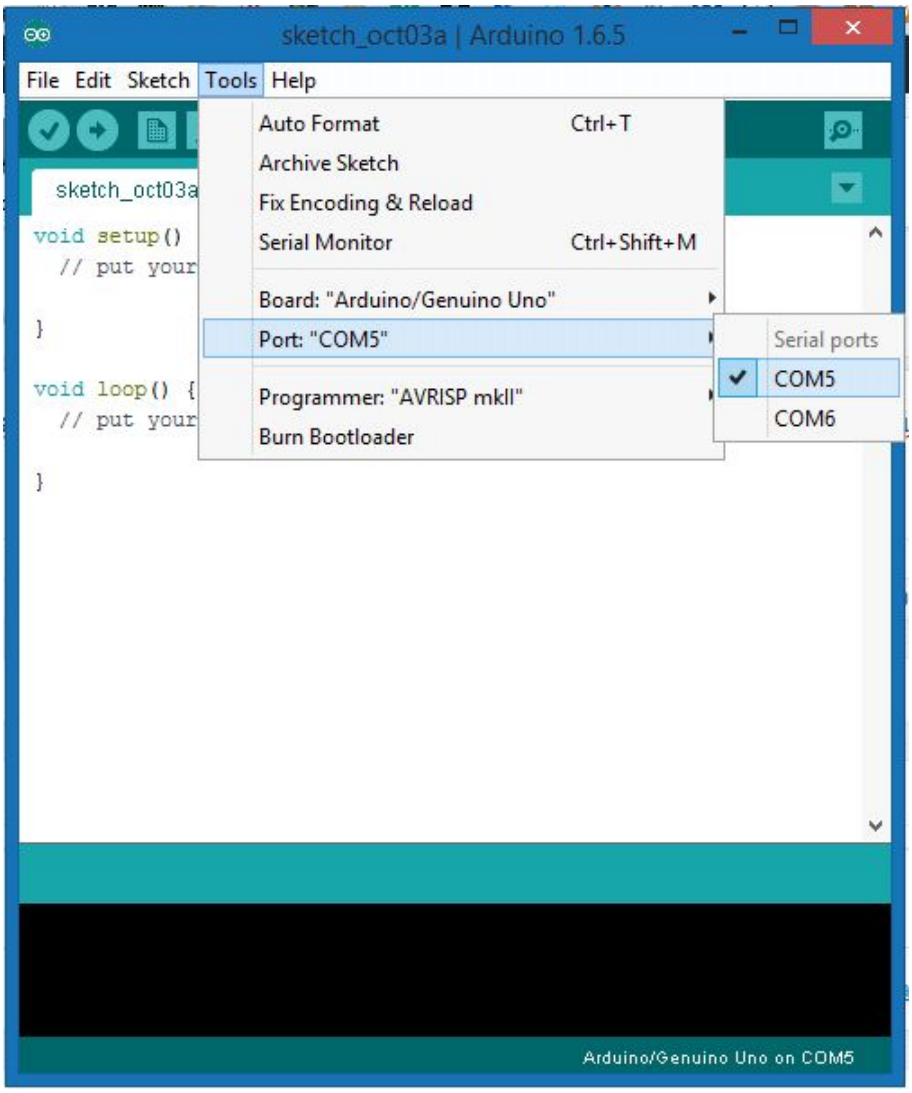
Eşleştirme işlemini başlatınca, bu sefer bilgisayarımız modüle verdiğimiz şifreyi soracak.



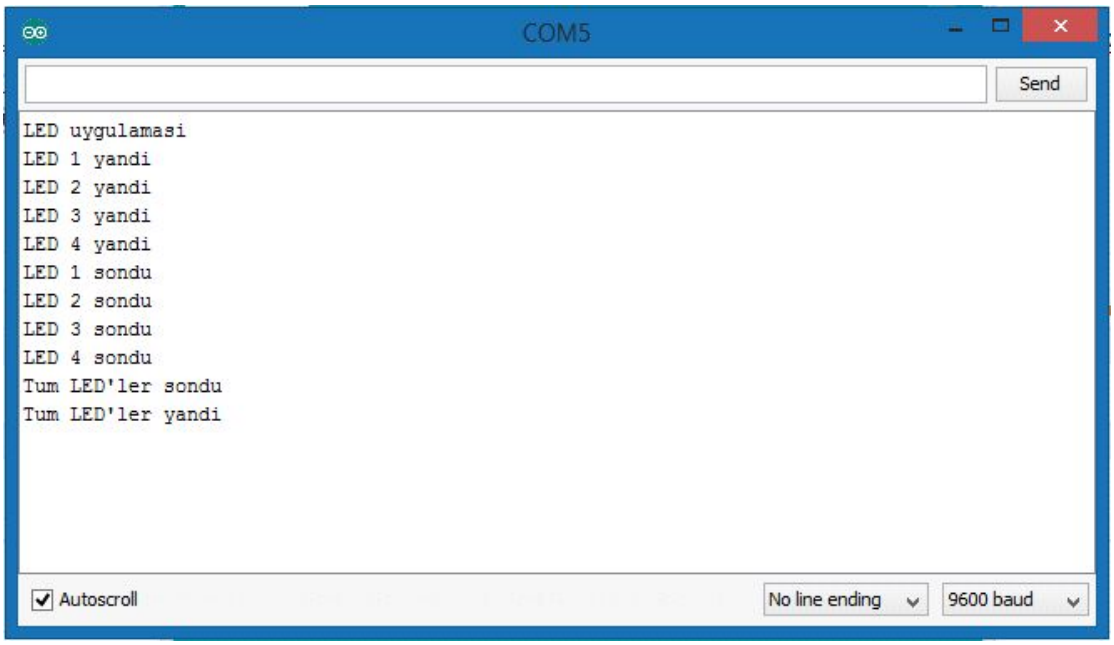
Bilgisayarımız aygıt yükleme işlemini tamamladıktan sonra “Denetim Masası”ndan “Aygıtlar ve Yazıcılar” ı seçerek listenin en altında bulunan bluetooth modülümüze sağ tıklıyor ve “Özellikler”i seçiyoruz. Buradaki pencereden “Donanım”a gelerek burada yazılı olan COM portunun numarasını öğreniyoruz.



Arduino programını açarak “Ayarlar” menüsünden portu değiştirerek bluetooth modülümüze bağlanacak şekilde ayarlıyoruz.



Arduino programından “Seri Port Ekranı”nı açarak baud rate’i 9600 olarak seçiyor ve istediğimiz komutu giriyoruz. q, w, e ve r tuşları sırasıyla 7, 6, 5 ve 4 numaralı pinlere bağlı LED’leri yakar veya söndürür. Z tuşu tüm LED’leri söndürür ve X tuşu tüm LED’leri yakar.



19.DERS

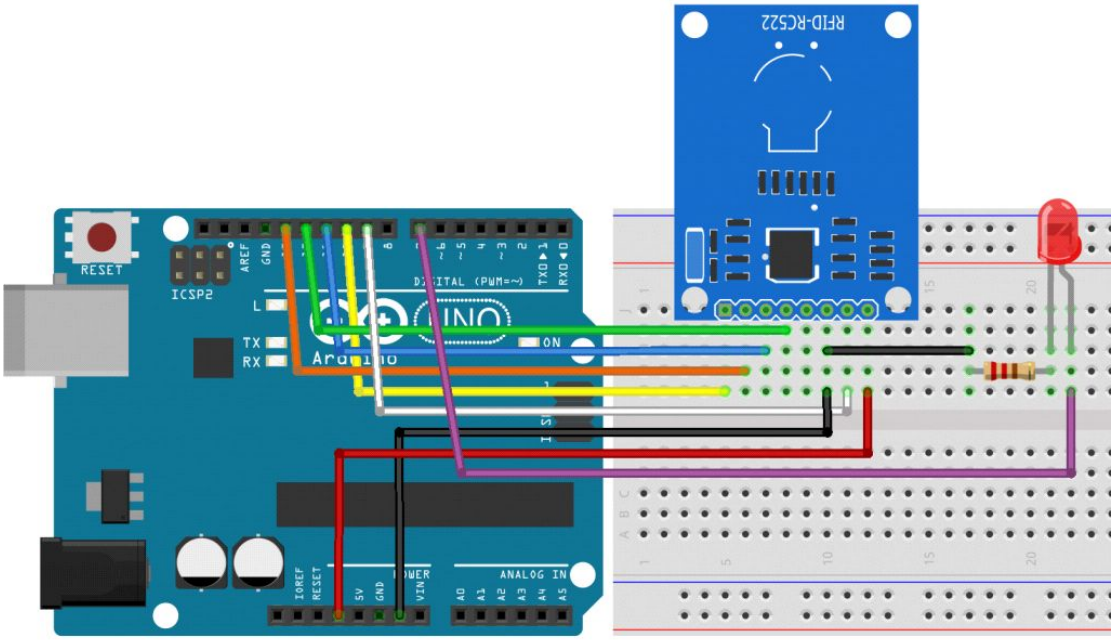
RFID MODÜL KULLANIMI

- 220 Ω direnç

RFID genel anlamıyla nesnelerin radyo dalgaları kullanılarak tanınması için kullanılan teknolojidir. Günlük hayatımızda toplu taşıma biletlerinde, işyeri ve okul girişlerindeki turnikelerde karşımıza sıklıkla çıkmaktadır. Kablosuz haberleşme teknolojileri ile ilgili daha fazla bilgi edinmek isterseniz, Kablosuz Haberleşme Teknolojileri yazımızı okuyabilirsiniz.

Kullandığımız kartların kendilerine ait UID isimli bir numarası vardır. Bu numara, her kart için farklıdır. Okuyucumuza kartımızı veya anahtarlığımızı yaklaştırdığımızda bu numara okunarak işlem yapılır. Bu uygulamada, öncelikle elimizdeki kartların UID'lerini Arduino'muzun dahili EEPROM'una kaydederek ve daha sonra okuttuğumuz kartın UID'sini, bellekteki UID değerleriyle karşılaştırarak işlem yapacağız.

Devre şemamız şu şekilde:



Bu adresten kartımızı çalıştırmamızda gerekli olan kütüphaneyi indirerek Arduino programının kurulu olduğu klasör altındaki **libraries** klasörüne **MFRC522** ismiyle kaydediyoruz.

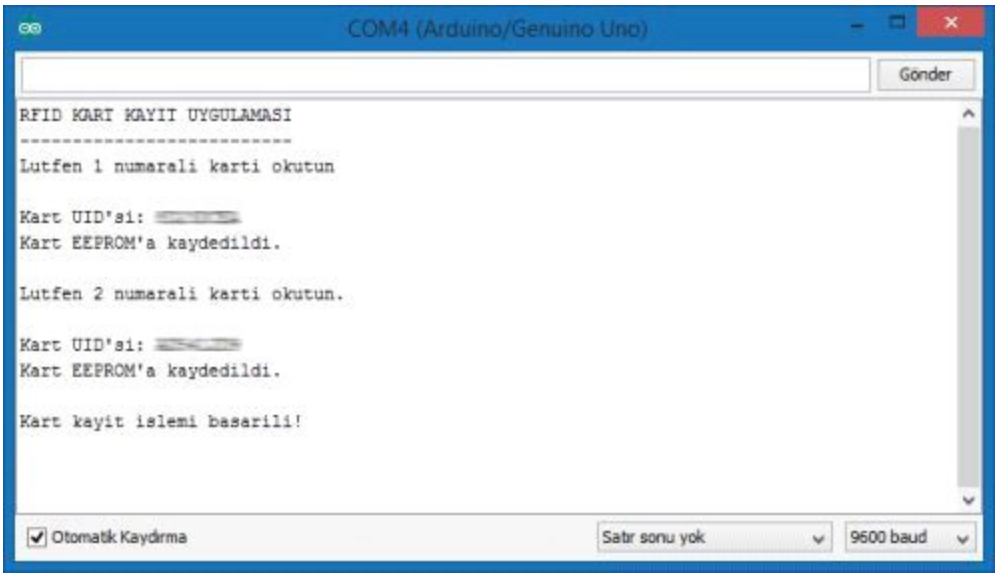
Öncelikle, elimizdeki kartları Arduino'nun EEPROM'una kaydedecek kodu yüklüyoruz:

```
#include <SPI.h>
 #include <MFRC522.h>
 #include <EEPROM.h>
 #define RST_PIN 9
 #define SS_PIN 10 byte readCard[4];
int successRead;
 MFRC522 mfrc522(SS_PIN, RST_PIN);
 MFRC522::MIFARE_Key key;
 void setup()
 {
Serial.begin(9600);
SPI.begin();
 mfrc522.PCD_Init();
 Serial.println("RFID KART KAYIT UYGULAMASI");
Serial.println("-------------------------");
Serial.println("Lutfen 1 numarali karti okutun");
```

```
 Serial.println();
 do
{
 //okuma başarılı olana kadar getID fonksiyonunu çağır successRead = getID();
} while (!successRead);
for ( int i = 0; i < mfrc522.uid.size; i++ )
{
//kartın UID'sini EEPROM'a kaydet EEPROM.write(i, readCard[i] );
 }
Serial.println("Kart EEPROM'a kaydedildi.");
 Serial.println();
Serial.println("Lutfen 2 numarali karti okutun.");
Serial.println();
 do {
 successRead = getID();
}
 while (!successRead);
 for ( int i = 0; i < mfrc522.uid.size; i++ )
 {
 EEPROM.write(i + 4, readCard[i] );
 }
Serial.println("Kart EEPROM'a kaydedildi.");
 Serial.println();
 Serial.println("Kart kayit islemi basarili!");
}
void loop() { }
int getID() {
//yeni bir kart okunmadıysa 0 döndür if ( ! mfrc522.PICC_IsNewCardPresent())
{
 return 0;
}
if ( ! mfrc522.PICC_ReadCardSerial())
 {
```

```
return 0;
}
 Serial.print("Kart UID'si: ");
 //kartın UID'sini byte byte oku ve seri monitöre yaz
for (int i = 0; i < mfrc522.uid.size; i++)
 {
// readCard[i] = mfrc522.uid.uidByte[i];
 Serial.print(readCard[i], HEX);
}
Serial.println("");
//kart okumayı durdur ve 1 döndür (okuma başarılı) mfrc522.PICC_HaltA();
 return 1;
}
```

Bu kodu yükledikten sonra, Arduino programından “Seri Port Ekranı”nı açarak kart okuma işlemini yapıyoruz.



Burada dikkat etmemiz gereken nokta, 1 ve 2 numaralı kartların birbirinden farklı olması. Eğer okuma işlemi sırasında 1 ve 2 numaralı kart olarak aynı kartı seçersek, bir sonraki adımda kullanacağımız kod çalışmayacaktır. Kart okuma işlemi yapıldıktan sonra, aşağıdaki kodu Arduino’muza yükleyerek bir sonraki adıma geçiyoruz:

```
#include <SPI.h>
```

```
#include <MFRC522.h>
#include <EEPROM.h>
#define RST_PIN 9
 #define SS_PIN 10
 #define ledPin 7
MFRC522 mfrc522(SS_PIN, RST_PIN);
 String lastRfid = "";
 String kart1 = "";
String kart2 = "";
MFRC522::MIFARE_Key key;
 void setup()
{
Serial.begin(9600);
 SPI.begin();
 mfrc522.PCD_Init();
 pinMode(ledPin, OUTPUT);
 Serial.println("RFID KART OKUMA UYGULAMASI");
Serial.println("--------------------------");
Serial.println();
 //EEPROM'dan kart bilgisini oku readEEPROM();
 }
void loop() {
 //yeni kart okunmadıkça devam etme if ( ! mfrc522.PICC_IsNewCardPresent())
{
return;
}
 if ( ! mfrc522.PICC_ReadCardSerial())
{
return;
 }
 //kartın UID'sini oku, rfid isimli string'e kaydet String rfid = "";
 for (byte i = 0; i < mfrc522.uid.size; i++)
{
```

```
rfid += mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ";
 rfid += String(mfrc522.uid.uidByte[i], HEX);
}
//string'in boyutunu ayarla ve tamamını büyük harfe çevir rfid.trim();
 rfid.toUpperCase();
 if (rfid == lastRfid) return;
 lastRfid = rfid;
Serial.print("Kart 1: ");
 Serial.println(kart1);
Serial.print("Kart 2: ");
 Serial.println(kart2);
 Serial.print("Okunan: ");
 Serial.println(rfid);
Serial.println();
//1 nolu kart okunduysa LED'i yak, 2 nolu kart okunduysa LED'i söndür if (rfid == kart1) {
digitalWrite(ledPin, HIGH);
 Serial.println("LED yandi.");
 }
if (rfid == kart2)
{
digitalWrite(ledPin, LOW);
 Serial.println("LED sondu.");
 }
Serial.println();
delay(200);
}
 void readEEPROM() {
 //EEPROM'dan ilk kartın UID'sini oku (ilk 4 byte) for (int i = 0 ; i < 4 ; i++)
{
 kart1 += EEPROM.read(i) < 0x10 ? " 0" : " ";
kart1 += String(EEPROM.read(i), HEX);
}
 //EEPROM'dan ikinci kartın UID'sini oku for (int i = 4 ; i < 8 ; i++)
```

```
{
kart2 += EEPROM.read(i) < 0x10 ? " 0" : " ";
kart2 += String(EEPROM.read(i), HEX);
 }
//Okunan stringleri düzenle
kart1.trim();
kart1.toUpperCase();
kart2.trim();
kart2.toUpperCase();
}
```

Bu kod sayesinde, 1 numaralı kartımızı okuttuğumuzda Arduino, RC522 modülün okuduğu UID değerini EEPROM'da kayıtlı olan 1 numaralı kart değeriyle karşılaştırıyor, eğer iki değer birbirinin aynısı ise 7 numaralı pine bağladığımız LED'i yakıyor. Arduino'muz, yeni bir kart okumak üzere hazır bekler haldeyken 2 numaralı kartı okutur isek, bu sefer yanmakta olan LED söndürülüyor. Eğer EEPROM'da UID'si kayıtlı olmayan bir kart okutursak LED'in durumunda herhangi bir değişiklik olmuyor.

20.DERS

HC-SR04 ULTRASONİK MESAFE SENSÖRÜ

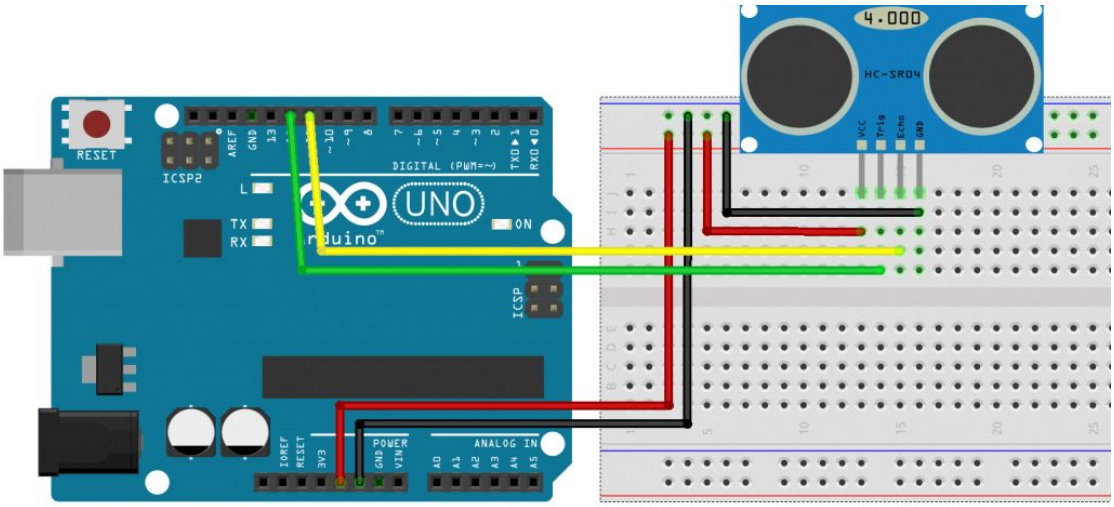
KULLANIMI

HC-SR04:

Bu sensör, robotik projelerde Arduino ile kullanılan en popüler sensörlerden birisidir. Kullanımı oldukça kolaydır ve program kısmı doğru olduğu sürece 2cm – 400cm arası uzaklıkları düzgün bir şekilde ölçebilmektedir. Çalışma prensibi ise şu şekildedir: Sensörün Trig pininden uygulanan sinyal 40 kHz frekansında ultrasonik bir ses yayılmasını sağlar. Bu ses dalgası herhangi bir cisme çarpıp sensöre geri döndüğünde, Echo pini aktif hale gelir. Biz ise bu iki sinyal arasındaki süreyi ölçerek -yani sesin yankısını algılayarak- cismin sensörden uzaklığını tespit edebiliriz.

Bu uygulamadaki gibi zamana duyarlı işlemlerde, Arduino'nun timer interrupt'larını kullanan bir koda ihtiyacımız var. Bir önceki paragrafta programın düzgün çalışmasından bahsetmiştim. Eğer biz timer interrupt gibi bir metod kullanmadan süre ölçümü yaparsak, alacağımız sonuç çok verimli olmayacaktır. Bu kısım size karışık geldiyse endişe etmenize gerek yok, Arduino ile HC-SR04 sensörü düzgün bir şekilde kullanmamızı sağlayacak bir kütüphane mevcut. https://bitbucket.org/teckel12/arduino-new-ping/downloads adresinden NewPing isimli kütüphanenin en güncel halini indirip Arduino yazılımına ekliyoruz.

Devre şeması:



Bağlantımızı yaptıktan sonra, Arduino programımızı açıp Dosya>Örnekler>NewPing>NewPingExample adımlarını takip ederek örnek kodu açıyoruz. Bu koddaki **#define TRIGGER_PIN 12** ve **#define ECHO_PIN 11** satırları, HC-SR04 sensörümüzün Trig ve Echo pinlerinin bağlanacağı Arduino pinlerini ayarlamamızı sağlıyor. **#define MAX_DISTANCE 200** satırı da sensörümüzün ölçeceği maksimum mesafayi 200 cm olarak ayarlamamızı sağlıyor. **loop** fonksiyonundaki **delay(50)** komutu, her bir ultrasonik ses göndermenin arasında 50 ms bekleme yapılmasını sağlıyor. İki ölçüm arasındaki süreyi bu komutla değiştirmemiz mümkün, dikkat etmemiz gereken ise bu iki ölçüm arasındaki minimum süre 29 ms'den daha az olmaması. Aksi takdirde bir önceki ölçümden gelen yankı ile çakışma meydana gelir ve ölçümümüz doğru bir sonuç vermez.