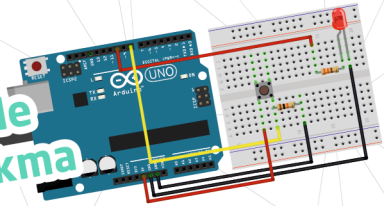
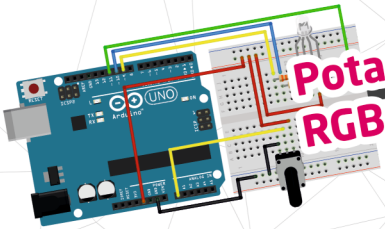
Eğitim Serisi ile Desteklenmektedir
9 Videolu

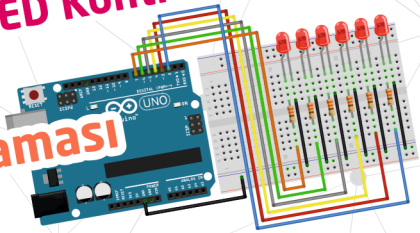
Arduino Başlangıç Kitabı



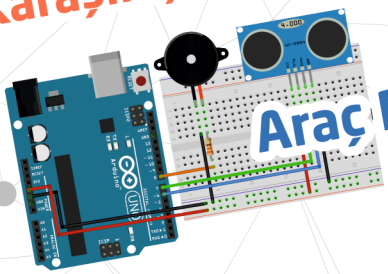
Buton ile LED Yakma



Potansiyometre ile RGB LED Kontrol



Karaşimşek Uygulaması



Araç Park Sensörü

4. Baskı

robotistan.com

Elektronik ve Kodlama dünyasına hoşgeldiniz. Bu kitabı açtığınıza göre siz de merak denizinde yüzüp, yeni şeyler öğrenmeye heveslisiniz demektir. Bu tür konularda yeni şeyler öğrenmek zor gibi düşünülse de adım adım ve doğru uygulamalar ile ilerlerseniz çok basit olduğunu fark edeceksiniz. İlk aşamalarda uygulamaları yaptıkça oturmayan anlamsız gelen yerler olacaktır. Bu sorunu uygulama yaptıkça aşacaksınız. Sadece biraz sabır gerekli...Kolay ve doğru yol haritası ile Arduino programlamayı öğrenebilmeniz için uygulamalar kolaydan başlayarak, daha komplekse doğru ilerlemektedir.

Uygulamaların daha detaylı videolu anlatımlarını izlemek isterseniz kitabın arka kısmındaki QR kodu taratarak YouTube kanalımıza gidebilirsiniz. Uygulamalara dijital ortamda erişmek isterseniz http://maker.robotistan.com blog sayfamızda da bulunmaktadır. Kitapçık içerisinde yazılan kodlara hem ilgili videoların açıklama kısmından hem de blog sayfamızdan ulaşabilirsiniz.

Bu kitap Robotistan Elektronik A.Ş bünyesinde yazılmıştır. Yazılış amacı ise Arduino'ya kolay ve doğru yoldan başlamak isteyenlere rehber olmasıdır. Umudumuz bu içeriklerin herkese faydalı olması ve sizlerin öğrenme sürecini kolaylaştırıp hızlı şekilde proje yapmanızı sağlamaktır.

Set içerikleri, uygulamalar, videolarımız ve aklınıza takılan ter türlü öneri ve sorularınız için info@robotistan.com e-mail adresinden bize iletebilirsiniz.

Robotistan Ekibi

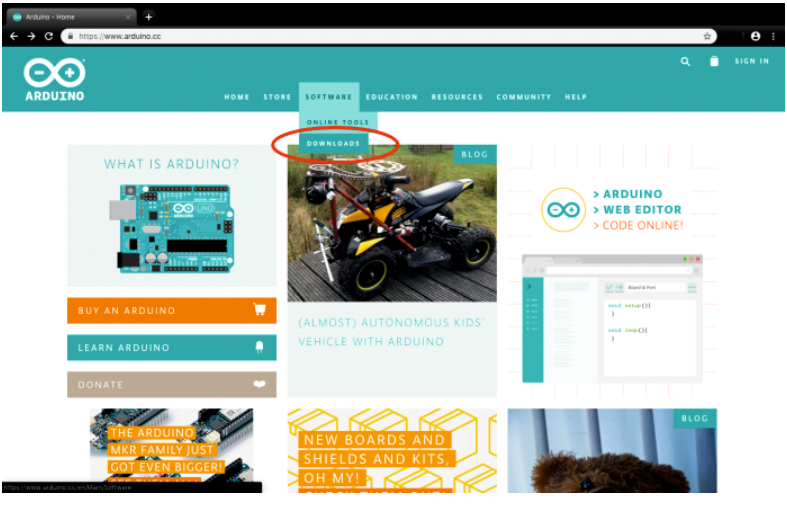
İçindekiler

01 Arduino Yazılım Kurulumu

Bu kitap ile birlikte Arduino dünyasına giriş yapıp, ileri seviye uygulamalara kadar gideceğiz. Uygulamalara başlamadan önce bilgisayarımızda Arduino sürücülerinin ve yazılımının kurulmuş olması gerekiyor. Benim tavsiyem, Arduino kartını bilgisayarımıza USB kablosu ile takmadan önce yazılımı yüklemeniz. Bu sayede, yazılımla birlikte gelen Arduino sürücüleri bilgisayarımıza kurulmuş oluyor ve böylece kartımızı kolaylıkla tanıtıp hemen kullanmaya başlayabiliyoruz.

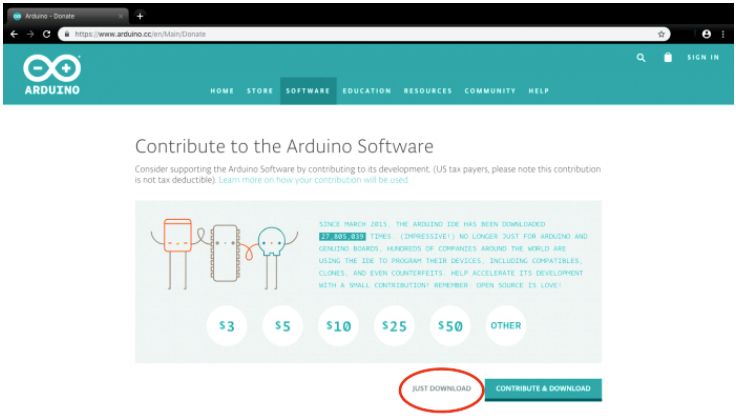
Arduino Yazılımının İndirilmesi

Arduino yazılımını indirmek için www.arduino.cc adresinden **"Downloads"** sekmesine gidiyoruz.



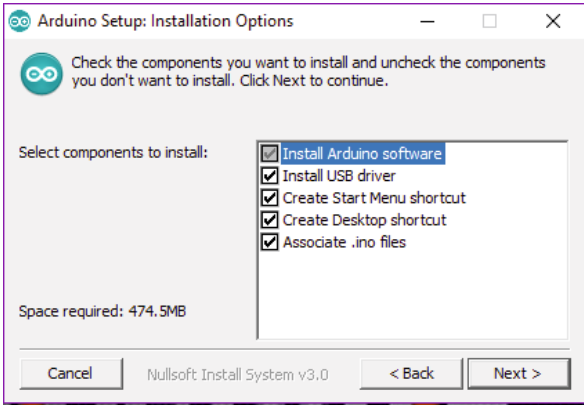
Downloads sekmesini tıkladıktan sonra karşımıza işletim sistemimize göre olan dosyayı indireceğimiz ekran çıkıyor. Bu yazıyı hazırladığım sırada Arduino yazılımının en güncel sürümü 1.8.7 idi. Windows kullananlar **"Windows Installer"** seçeneğini tıklayabilirler. Diğer işletim sistemlerinin de yükleme dosyaları aşağıda bulunmaktadır.

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

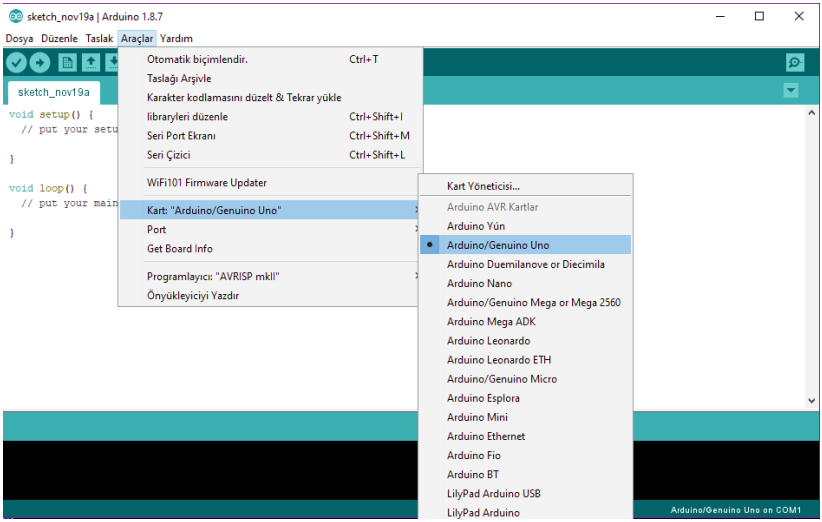
Bundan sonra yazılım kurulum dosyamız inmeye başlıyor. İndirme işlemi bittikten sonra dosyayı açarak kurulum işlemini başlatıyoruz. Kurulum sırasında çıkan "Install USB driver" seçeneğinin seçili olduğundan emin oluyoruz.



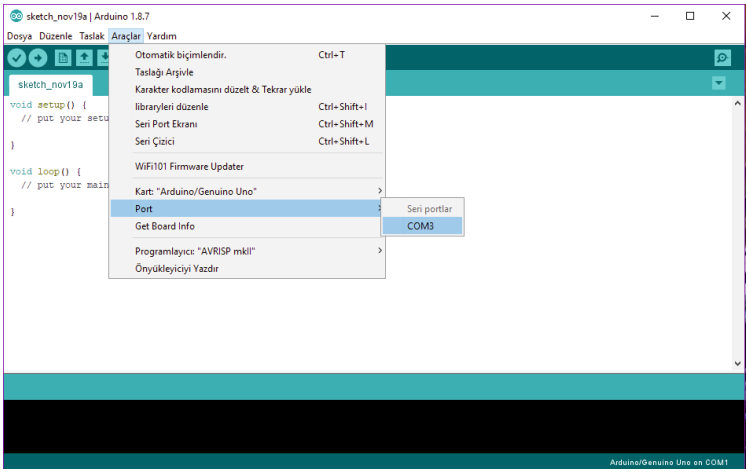
Kurulum işlemi bittikten sonra, kartımızı USB kablomuzla bilgisayarımıza bağlıyoruz. Bilgisayarımızda "Yeni donanım bulundu" penceresi açılıyor. Eğer sürücüler yazılımla birlikte kurulduysa, otomatik yükleme seçeneği Arduino'muzun sürücülerini otomatik olarak yükleyecektir.

Arduino Programının Bilgisayarımızda İlk Çalıştırılması

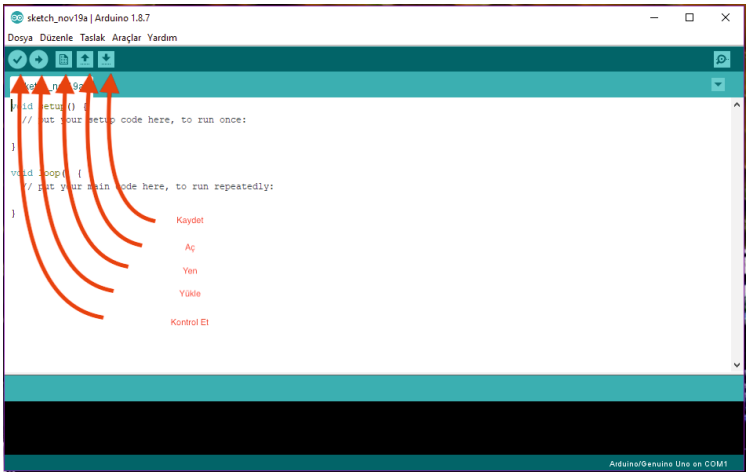
Artık Arduino programımızı açabiliriz. Programımızı açtıktan sonra ilk yapmamız gereken şey, programın Arduino UNO kartımızla çalışacak şekilde ayarlanmasıdır. **Araçlar > Kart** menüsünden Arduino UNO seçeneğini tıklıyoruz.



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

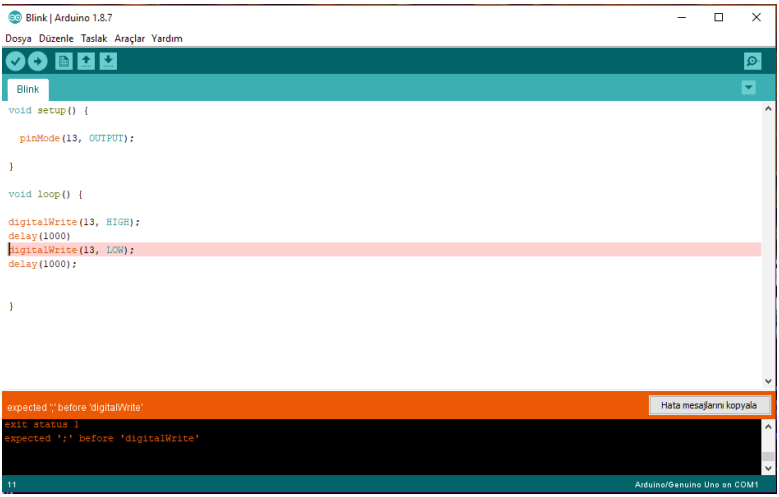


Artık her şeyiyle kullanıma hazır bir Arduino programımız var.



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, "void setup()" fonksiyonundaki komutlar çalıştıktan sonra kartın enerjisi kesilene kadar sürekli çalışacak olan fonksiyonları 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.

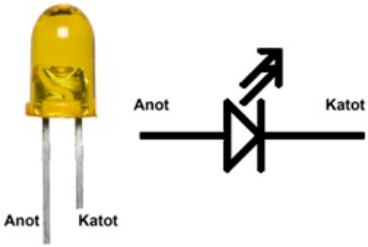
Eğer yazdığımız kodda bir hata yoksa ve Arduino kartımız bilgisayarımıza USB ile bağlıysa, **"Yükle"** seçeceğine tıklayarak kodumuzu kartımıza yükleyebiliyoruz.

Gerekli Malzemeler:

- Arduino Uno
- Breadboard
- Kırmızı LED
- 330 Ohm Direnç (Turuncu - Turuncu - Kahverengi)
- 2 Adet Erkek-Erkek Jumper Kablo

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 15 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 15 mA akıma ihtiyaç duymakta. Sanırım işler biraz karışmaya başladı. Korkmaya gerek yok! Her şeyin bir çözümü var.

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.

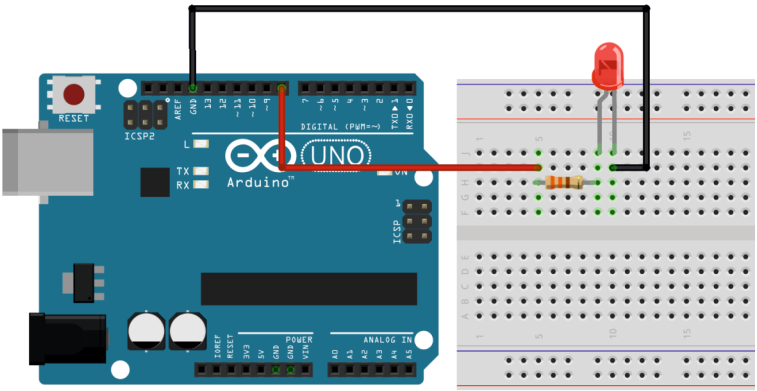
$$V = i \times R$$

Bu denklemde "V" bize gerilimi, "i" akımı ve "R" ise direnci temsil ediyor. Eğer 15 mA akıma ihtiyaç duyan LED'i, Arduino'muzun 5V çıkış sağlayan pinlerinden birine bağlayacak olursak;

$$5V = 0{,}015A \times R$$

Denklemini elde etmiş oluruz. Bu denklemden "R"yi çekecek olursak sonucu 333 buluruz. Bu demek oluyor ki LED'imizi 5V gerilimle kullanmak için 333 Ω (ohm) değerinde bir dirence ihtiyacımız var. Tam değeri doğru tutturmamız çok önemli değil, elimizde mevcut olan 330 Ω'luk direnci kullanabiliriz.

Hemen devremizi kuralım ve sonrasında proje kodumuzu yazmaya başlayalım.



Devre kısmını kurduktan sonra kodu yazmak için Arduino IDE'yi açarak, yukarıdaki sekmelerden "Dosya" sonrada "Yeni" seçeneğini seçerek yeni bir program sayfası açalım. Açılan sayfada "//" ve sonrasında yorum yazan satırları silebilirsiniz.

Bu sayfada ekleyeceğimiz kodları "void setup" ve "void loop" ile başlayan alanlara süslü parantezler"{}" içerisine yazacağız.

Arduino

```
1  void setup() {
2    pinMode(8,OUTPUT);
3  }
4
5  void loop() {
6    digitalWrite(8,HIGH);
7    delay(500);
8    digitalWrite(8,LOW);
9    delay(500);
10 }
```

Setup kısmında 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. Pin için ayar yaptığımıza göre artık LED yakıp söndürme için gerekli olan kodu yazalım.

"loop" kısmında ise öncelikle 8 numaralı pine HIGH lojik seviyesine, yani 5V'a ayarlıyor, 500 milisaniye (yarım 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 yarım saniye hiçbir işlem yapmadan bekliyor.

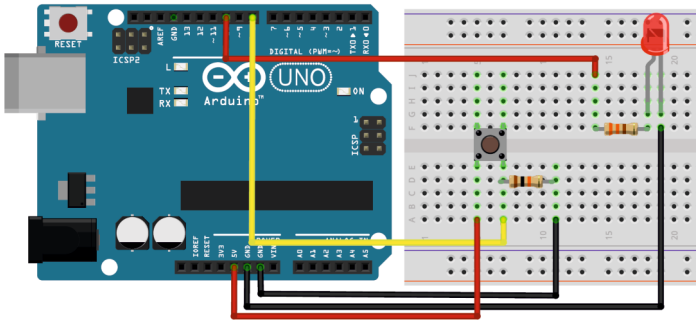
Bu koddaki "delay" komutlarının sürelerini değiştirerek LED'in açık ve kapalı kaldığı süreleri değiştirebiliriz. Eğer başka bir pin kullanmak istersek tek yapmamız gereken "pinMode" ve "digitalWrite" fonksiyonlarında bulunan pin numarasını kullanmak istediğimiz pin numarası ile değiştirmek. LED'imize 330 Ω'luk bir direnci seri bağlamayı unutmuyoruz!

03 Buton ile LED Yakma

Gerekli Malzemeler:

- Arduino Uno
- Breadboard
- Kırmızı LED
- Push Buton
- 330 Ohm Direnç (Turuncu - Turuncu - Kahverengi)
- 10k Ohm Direnç (Kahverengi - Siyah - Turuncu)
- 5 Adet Erkek-Erkek Jumper Kablo

Bu uygulamamızda Arduino üzerindeki pinleri giriş olarak kullanmayı öğreneceğiz. Bu sayede dışarıdan bir butona basıldığında Arduino içerisinde haberimizin olmasını sağlayacağız. LED devremiz bir önceki uygulama ile aynı olabilir. Sadece LED'in bağlı olduğu bacak bu uygulamada 10 numara olacak. Şimdi devremizi kurup daha sonra kod kısmına geçelim.



Butondan sağlıklı veri okuyabilmek için 10k Ohm direnç ile birlikte kullanmamız gerekiyor. Butona basılı değilken pin üzerinde oluşabilecek parazitleri ve bu parazitlerden kaynaklanan yanlış sinyal algılamalarını engellemek için "pull-up" veya "pull-down" direnci kullanmamız gerekiyor. Biz bu uygulamamız da "pull-down" direnci kullanacağız. Bu projemizde buton basılı değilken pinden okunan değer 0V yani lojik LOW seviyesindedir. "Pull-down" direnci, buton basılıp değer HIGH'a çekilmediği sürece bu pindeki gerilimi 0V'ta sabit kalmasını sağlar. Devre kısmındaki mantığı öğrendiğimize göre artık kod kısmına geçelim.

```arduino
#define Buton 8
#define Led 10

int buton_durumu = 0;

void setup() {
  pinMode(Buton, INPUT);
  pinMode(Led, OUTPUT);
}

void loop() {
  buton_durumu = digitalRead(Buton);
  if(buton_durumu == 1){
  digitalWrite(Led,HIGH);
  }
  else{
  digitalWrite(Led,LOW);
  }
}
```

"#define" satırı ile 8 numaralı pine "Buton" ismini veriyoruz bu sayede gerekli yerlerde 8 yazmak yerine "Buton" yazarak çok daha hatırlanabilir ve kolay kod yazabiliriz. LED içinde benzer tanımlamayı 10 numaralı pin için yapıyoruz.Yazılım içerisinde okuduğumuz verileri veya saklamak istediğimiz bilgileri, sonradan tekrar erişebilmek için değişkenleri kullanırız. Değişkenler tiplerine göre içerisinde barındırdıkları veriler farklılık gösterir. En çok karşımıza çıkacak olan ve sıkça kullanacağımız "int" değişkeni "integer"ın kısaltmasıdır. Bu değişken içerisinde -32767'den 32767 ye kadar sayıları tutabilir. Bu sayılar tam sayı olmalıdır. Virgüllü bir sayı içerisine atamak isterseniz yuvarlayarak tam sayıya dönüştürecektir. Bu kodda "buton_durumu" değişkeni tanımlayıp, ilk değerini sıfır olarak atıyoruz. İlk değerin sıfır atanması şart değil ancak "integer" tanımladığınızda değişken içerisinde rasgele bir sayı olabilir. Yazılım kısmında her ihtimale karşı sıkıntı oluşturmaması için ilk değerini sıfır olarak atıyoruz.

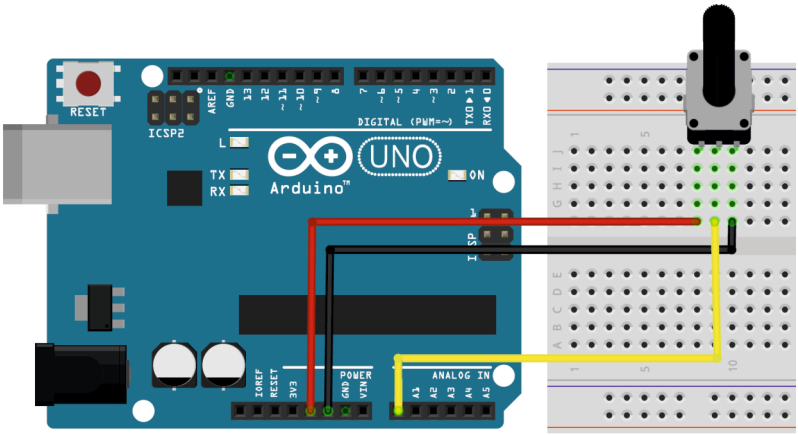
"pinMode" komutu ile "Buton" pinini (8 numara olarak belirledik) giriş olarak ayarlıyoruz. Bir alt satırda ise LED pinini (10 numara olarak belirledik) giriş olarak ayarlıyoruz.

Giriş-çıkış ayarlarken giriş yapmak istediğimiz butonlara "INPUT", çıkış yapmak istediğimiz pinlerede "OUTPUT" yazmamız yeterli. Giriş-çıkış olarak kullanacağınız pinleri tanımlamadığınız taktirde, bu pinler istediğiniz gibi veya stabil çalışmayacaktır."loop" kısmına geçtiğimizde butondan gelen veriyi okuyup, bu veriyi "if-else" komutu ile değerlendireceğiz. Değerlendirme sonucunda gelen verinin "1" veya "0" oluşuna göre karar vererek LED'i yakıp söndüreceğiz.

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 10k Ohm Potansiyometre
- 3 Adet Erkek-Erkek Jumper Kablo

Arduino kartının üzerine baktığınızda "Analog Input" pinlerini göreceksiniz. Bu pinleri kullanarak dijitalden analog sinyale dönüşüm yaparak bu pindeki voltajı okumamız mümkün. Arduino dijital okuma yaparken 0V (sıfır) ve 5V okuyabilmektedir. Bu iki uç değer arasında ara değerler gelirse bunu algılayamamakta ve gelen voltajı eşik değerine göre 0V veya 5V olarak kabul etmektedir. Analog pinler sayesinde 0V'dan 5V'ta kadar ara gerilim değerlerini de algılayarak dijitale çevirebiliyoruz. Ara değerlerdeki sinyalleri elde edebilmek için ayarlı direnç (potansiyometre) kullanacağız. Uygulamamızda analog giriş pininden gelen gerilimin sayısal karışılığını seri porttan okuma işlemini sağlayacağız. Devremizi kurup kod kısmına geçelim.



```arduino
#define potpin A0

int deger=0;

void setup() {
  Serial.begin(9600);
  Serial.println("Pot Değer Okuma");
}

void loop() {
  deger = analogRead(potpin);
  Serial.println(deger);
  delay(300);
}
```

Önceki kodlarımızda da yaptığımız gibi define işlemi ile "A0" pinine "potpin" ismini veriyoruz. Sonraki satırda analog pinden okuduğumuz değerleri saklamak için "integer" türünde ve değer isminde değişken tanımlıyoruz.

"setup" kısmında önceki yazılımlarımızda dijital giriş-çıkış kullandığımızdan dolayı, pinleri ne olarak kullanacağımıza göre ayarlıyorduk. Analog okuma yaparken giriş çıkış tanımlamamıza gerek yok bu sebepten dolayı bu yazılımda "pinMode" komutunu kullanmıyoruz.
Okuduğumuz verileri bilgisayara göndermek için seri haberleşme başlatmamız gerekiyor. Bu seri haberleşme sayesinde Arduino ile bilgisayar USB bağlantı üzerinden haberleşecek ve istediğimiz verileri bilgisayara aktarabileceğiz.

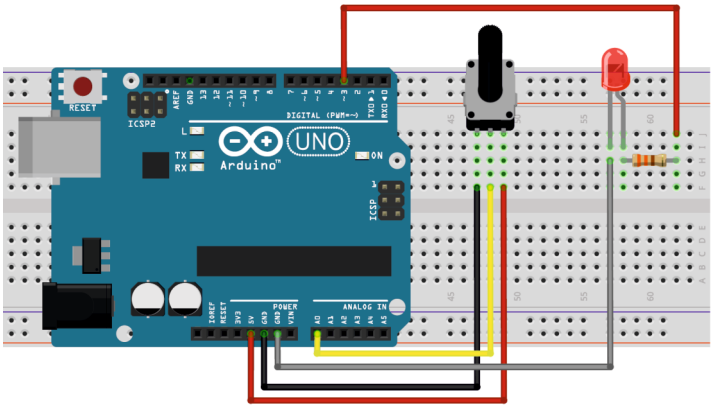
"Serial.begin(9600);" satırı ile bu haberleşmeyi başlatıyoruz. Arduino kodu çalıştırmaya başladığında ilk olarak bilgisayar ile haberleşmeyi başlatacaktır. Haberleşme başladıktan sonra "Serial.println("Pot Deger Okuma");" satırı ile "Pot Deger Okuma" yazısı bilgisayarda seri monitöre yazdırılacaktır. "Serial.print " ve "serial.println" komutlarını aşağıda detaylı açıklayacağız.

05 Potansiyometre ile LED Yakma

Gerekli malzemeler:

- Arduino UNO
- Breadboard
- 10k Ohm potansiyometre
- Kırmızı LED
- 330 Ohm Direnç (Turuncu - Turuncu - Kahverengi)
- 5 Adet Erkek-Erkek Jumper Kablo

Önceki uygulamamızda analog pinden gerilim değerini okumuştuk bu uygulamamızda yine analog pinden gelen değere göre LED'in parlaklığını kontrol edeceğiz. İlk led yakma uygulamamızda dijital çıkış kullandığımızdan dolayı LED'e 0V veya 5V gönderebiliyorduk. Bu yüzden led ya sönüyordu yada yanıyordu. Arduino'nun yeni bir özeliğini kullanarak LED'e 0-5V aralığında ara değerlerde gerilimde gönderebileceğiz. Bu gerilim kontrolü sayesinde LED'in parlaklığını ayarlayabileceğiz. Bu uygulamaya kadar dijital giriş-çıkış ve analog girişi öğrendik. Bu uygulamayla birlikte analog çıkış yani PWM özelliğini öğreneceğiz. Şimdi devremizi kurup hemen kod kısmına geçelim.



PWM (Pulse with Modulation) , sinyal genişlik modülasyonun kısaltmasıdır.Bu özellik Arduino Uno üzerine 6 pinde mevcut yaklaşık işareti (~) bulunan pinlerden (3,5,6,9,10 ve 11. Pinler) PWM ile ilgili detaylı bilgiler için uygulama sonundaki kare kodu taratarak bu uygulamaya ait olan videoyu izleyebilirsiniz. Devremizi kurduktan sonra hemen kod kısmına geçelim.

```
#define led 3
#define pot A0

void setup() {
}

void loop() {
  int deger = analogRead(pot);
  deger = map(deger,0,1023,0,255);
  analogWrite(led,deger);
}
```

Bu uygulamamız da dijital giriş-çıkış kullanmadığımızdan dolayı yine "setup" kısmında bir ayarlama yapmıyoruz.

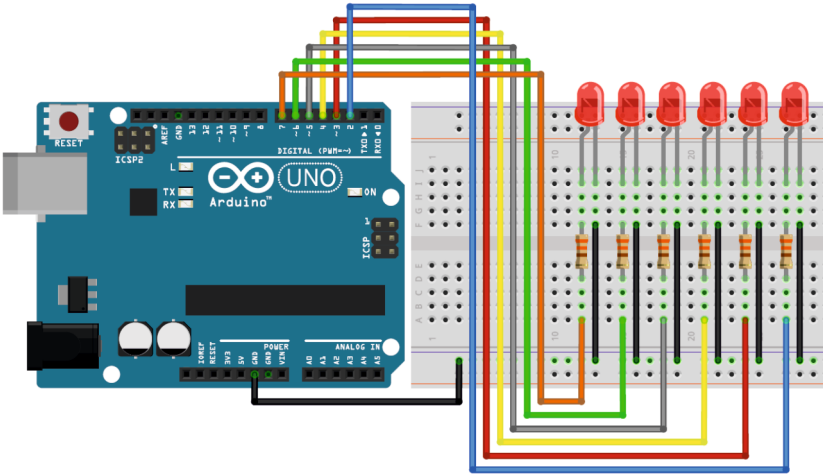
Ana program döngümüzde POT'tan veriyi okuyup bu veriyi LED'e göndermek istiyoruz. İlk olarak "analogRead" komutu ile potansiyometreden veriyi okuyoruz. Okuduğumuz değeri "deger" değişkenine yazıyoruz. İkinci satırda ise "map" komutunu kullanarak 0 ile 1023 arasında gelen değeri 0 ile 255 arasına oranlıyoruz.

Analog okumayı 10 bit (2^10 = 1024) çözünürlükte yaparken, analog yazmayı 8 bit (2^8 = 256) çözünürlükte yapabiliyoruz. Okuduğumuz veriyi, çıkışa göre oranlayıp yazdırmamız gerekiyor. "map" komutu yerine dilerseniz direk olarak 4'e de bölebilirsiniz. Oranlama işleminden sonra "analogWrite" komutu ile PWM pinlerinden çıkış verebiliriz.

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 13 Adet Erkek-Erkek Jumper kablo
- 6 Adet LED
- 6 Adet 330 Ohm Direnç (Turuncu - Turuncu - Kahverengi)

Bu uygulamamızda "for" döngüsünü kullanımını göreceğiz. "for" döngüsü ardı sıra yapılması gereken işlemlerde kullanılabilir. Uygulamamızda 6 adet LED'i yakmak için hepsini çıkış vermemiz ve sırasıyla yakmamız gerekecek. Bunu normal öğrendiğimiz çıkış tanımlama ve LED yakma söndürme komutları ile rahatlıkla uygulayabilmekteyiz. "for" döngüsü kullanmadan yazılan kodda değişiklik yapmak istediğinizde her satırda tekrar tekrar değişiklik yapmanız gerekecek, "for" döngüsünde ise hem kodu anlamak hem yazmak hem de değişiklik yapmak istediğiniz de çok daha hızlı şekilde ilerleyebileceksiniz. Devremizi kurduktan sonra hemen kod kısmına geçelim.



```
int ledler[] = {2,3,4,5,6,7};

void setup() {
  for(int i=0; i<6; i++){
  pinMode(ledler[i], OUTPUT);
  }
}

void loop() {
  for(int i=0; i<6; i++){
    digitalWrite(ledler[i], HIGH);
    delay(80);
    digitalWrite(ledler[i], LOW);
  }

  for(int j=5; j>-1; j--){
    digitalWrite(ledler[j], HIGH);
    delay(80);
    digitalWrite(ledler[j], LOW);
  }

}
```

Dijital pinleri tek tek tanımlarken "int" veya "#define" ile çıkışlara isim tanımlaması yapıyorduk. Bu sefer 6 çıkış birden kullacağımızdan dolayı dizi kullanarak ilerleyeceğiz. Dizileri, değişkenleri barındıran bir küme olarak düşünebilirsiniz. Kodun üst kısmında içerisindeki değişken türleri "int" (tamsayı) olan ve ismi "ledler" olan dizi tanımlıyoruz. Dizi elemanlarını ise içerisine virgüller ile ayırarak yazıyoruz. Dijital çıkışlardan 2 numaranda 7 numaraya kadar kullanacağımız için elamanlarımızı bu şekilde belirledik.

Dizinin elemanlarını çoğaltadabilirsiniz. Dizi tanımlama yaparken eğer istersek "ledler[]" ifadesi içerisine dizinin kaç elemanlı olacağını yazabiliriz. Örneğin "int ledler[6] = {2,3,4,5,6,7};" gibi Diziden eleman çağırırken ilk elemanın numarası 0'dan (sıfırdan) başlar. İstediğiniz elemanı çağırmak içinde "setup" veya "loop" içerisinde "ledler[*dizi elemanı sıra numarası*]" ifadesini kullanabilirsiniz. Yani eğer sıfırıncı dizi elemanını çağırmak için "ledler[0]" yazarsanız bu 2'ye eşit olacaktır. 5. Dizi elemanını çağırmak için "ledler[5]" yazarsanız buda 7'ye eşit olacaktır.

Uygulamamızda 6 adet LED'i yakmak istiyoruz bu durumda 6 adet dijital pinlerden çıkış belirlememiz gerekiyor. "for" döngüsünün parantez içerisinde ilk noktalı virgüle kadarki kısım döngü için kullanılacak koşulun değişkeni olarak tanımlanıyor.

Bu yazılımda sadece "for" döngüsü için kullanılacağından dolayı parantez içerisinde tanımladık. İsterseniz hali hazırda farklı bir değişkeninizi veya değişkeni yazılımın üstünde tanımlayıp sonradan burada kullanabilirsiniz.

"İ<6" ifadesi ise "for" döngüsünün koşulunu belirliyor. "i" değişkeni 6'dan küçük olduğu sürece "for" döngüsü içerisindeki kod satırlarını tekrarlayacak. Eğer "i" değişkeni 6'ya eşit veya büyük olursa artır "for" döngüsüne yapmayıp döngünün bittiği yerden kodu işletmeye devam edecek.

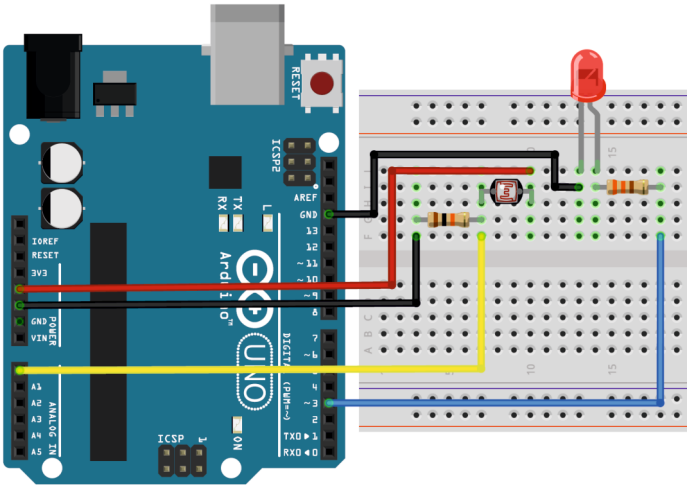
"i++" ifadesi ile "for" döngüsünü her yaptığımızda "i" değişkeninin değerini 1 arttırmasını istiyoruz. Böylelikle "i" değişkeninin ilk değeri 0(sıfır) oluyor. Dijital çıkış verdiğimiz komutlar içerisinde de "i" değişkenini yerine yazdığınızda diziden elemanı çağırıyor. Yani "pinMode(ledler[0], OUTPUT)" yazdığınızda yazılım "ledler" dizisinden sıfırıncı elemanı çağırarak "pinMode(2, OUTPUT)" olarak algılıyor. Bu sayede 2. pini çıkış olarak tanımlıyoruz. "for" döngüsü 0'dan 5'e kadar bunu yapacağı için 6 adet pini çıkış olarak tek satır ile tanımlamış oluyoruz.

"loop" kısmında da "setup" kısmındaki gibi "for" döngüsü kullanımı aynı mantıkla ilerliyor. Bu seferde çıkış tanımlamak yerine "digitalWrite" komutu ile sırasıyla 6 adet LED'i yakıp söndürüyoruz. "for" döngüsü ile ilgili detaylı video anlatımına aşağıdaki kare kodu taratarak izleyebilirsiniz.

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 5 Adet Erkek-Erkek Jumper Kablo
- 330 Ohm Direnç (Turuncu-Turuncu-Kahverengi)
- 10K Ohm Direnç (Kahverengi-Siyah-Turuncu)
- 5mm Kırmızı LED
- 5mm LDR

Bu uygulamamızda ortamdaki ışığı algılayabilen LDR'den veri okuyup, bu veriye göre LED'imizi yakıp söndüreceğiz. LDR (Light Dependent Resistance) yani fotodirenç ortamdaki ışık miktarına göre direncini değiştirir. Bu direnç değişimini Arduino kartı ile algılayabiliriz.Bu sayede ortamdaki ışık miktarını bilebildiğimiz için ortam karanlık olduğunda LED'i yakıp, aydınlık olduğunda LED'i söndürerek otomatik bir lamba yapacağız. Aynı zamanda aldığımız verileri bilgisayara gönderip, serial monitör üzerinde de görüntüleyeceğiz. Hemen devremizi kurarak başlayalım.



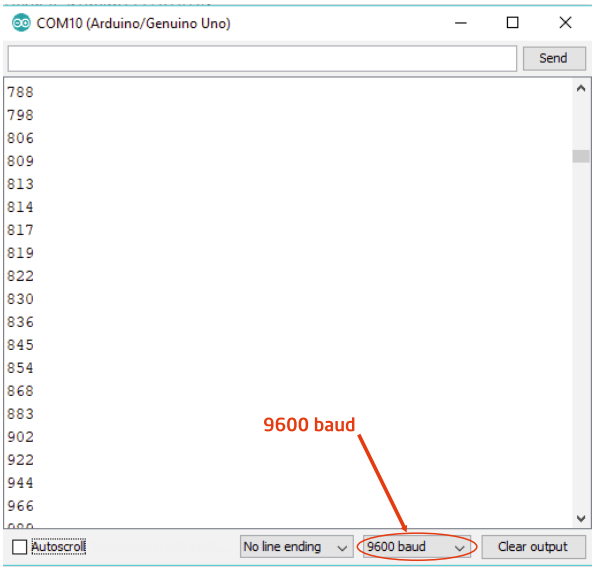
```
Arduino
#define led 3

void setup() {
    pinMode(led,OUTPUT);
    Serial.begin(9600);
}

void loop() {
  int isik = analogRead(A0);
  Serial.println(isik);
  delay(50);       //
  if(isik > 900){
    digitalWrite(led,LOW);
  }
  if(isik < 850){
    digitalWrite(led,HIGH);
  }
}
```

Kod kısmına geçecek olursak ilk satırımızda LED'i bağlayacağımız pine isim veriyoruz. Bu işlemi "#define" komutu ile yapacağız. Bu işlemden sonra artık ihtiyaç halinde 3 yazmak yerine "led" yazarak işlemleri kolaylaştıracağız.

Kodun "setup" kısmında LED'imizi bağladığımız pini çıkış vermemiz ve seri haberleşmeyi başlatmamız gerekiyor. Seri haberleşmeyi 9600 baudrate hızında başlatıyoruz. Bu sayı bilgisayar ve Arduino kartının ne kadar hızlı haberleştiğini belirler. Bu sayıyı rasgele yazamıyoruz. Önceden belirlenmiş hızları kullanmamız gerekmektedir. 300,600,1200,2400,4800,9600,14400,19200,28800,38400 veya 115200 baudrate hızlarını kullanabilirsiniz. Arduino koduna yazdığınız baudrate değeri bilgisayarda açacağını seri monitör'ün sağ alt köşesindeki hız ile aynı olmalıdır.

Ana algoritmamızın döneceği "loop" içerisinde "int" tipinde ve "isik" isminde bir değişken tanımlayıp içerisinde LDR okuduğumu değeri yazdırıyoruz. Okuduğumuz bu değeri seri haberleşme üzerinden bilgisayara gönderiyoruz. 50 ms kadar bekledikten sonra "if" komutu ile gelen değerin istediğimiz değerlerin altında veya üstünde olup olmadığına göre değerlendirip karar veriyoruz. Ortamdaki ışık az ise LDR üzerinden gelen değer küçülecektir. Bizim ortamımızda 850 değerinden sonra LED'in yanmasını istiyoruz. Ortam aydınlanmaya başlayınca da değer artacağından dolayı 900 değerinden sonra sönmesini istiyoruz.

Kodu kartımıza attıktan sonra Arduino IDE'si içerisinde "Serial Monitor" butonuna tıklayıp gelen verileri görebiliriz. LDR üzerinde elinizi getirdiğinizde ışık şiddeti değişeceği için okuduğunuz değerlerde değişecektir.

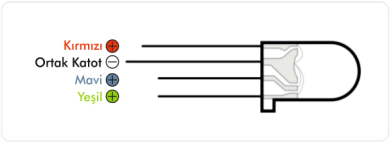


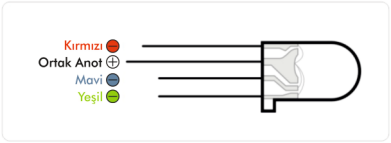
08 RGB LED Uygulaması

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 3 Adet 330 Ohm Direnç (Turuncu,Turuncu, Kahverengi)
- RGB LED
- 10K Potansiyometre
- 9 Adet Erkek-Erkek Jumper Kablo

RGB LED içerisinde kirmizi, yeşil ve mavi renkleri içeren 3 adet led yapısı bulunur. İçinde bulundurduğu renklerin baş harflerinin birleşmesi RGB (Red, Green, Blue) ismi oluşmuştur. 3 adet LED düşündüğümüzde her birinde bir artı bir eksi olacak şekilde toplam 6 bacak olması gerekir. Kullandığımız RGB LED'de 4 bacak bulunur. İçerideki 3 renk artı bacağını ortak olarak kullanır. Artı bacaktan enerji verildiğinde her renk ait olan eksi bacaktan eksiye bağlantı yapıldığında ilgili LED yanacaktır. RGB LED'lerin ortak artı yerine ortak eksi bacağı olanları da mevcut. Bu durumda ilgili LED'de artı sinyalini verdiğinizde yanacaktır. LED'leri tarif ederken ortak anot(artı) ve ortak katot(eksi) terimlerini kullanabilirsiniz. Bu durumda bizim kullanacağımız LED ortak anot olacaktır.

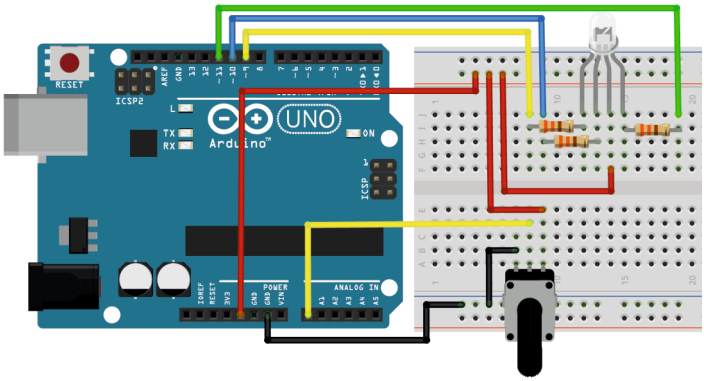




Bu uygulamamızda renkleri tam parlaklık dışında ara renklerde de yakmak istiyoruz. LED'i istediğimiz parlaklık seviyesinde yakabilmek için PWM özelliğini kullanmamız gerekiyor. PWM özelliği belirli bacaklarda (3,5,6,9,10,11) bulunduğu için bağlantılarımız yaparken bu bacakları kullanmaya dikkat edeceğiz. Hemen devremizi kurarak projemize başlayalım.

- Eğer RGB Ledimizin uzun bacağı anot(+) ise devrede 5V'a bağlanmalıdır.
- Eğer çalışmıyorsa RGB Ledimizin uzun bacağı GND hattına bağlanmalıdır.

Bu uygulamamızda renkleri tam parlaklık dışında ara renklerde de yakmak istiyoruz. LED'i istediğimiz parlaklık seviyesinde yakabilmek için PWM özelliğini kullanmamız gerekiyor. PWM özelliği belirli bacaklarda (3,5,6,9,10,11) bulunduğu için bağlantılarımız yaparken bu bacakları kullanmaya dikkat edeceğiz. Hemen devremizi kurarak projemize başlayalım.



Kod kısmında diğer yazılımlarda yaptığımız gibi isim atamalarını yapıp değişkenlerimizi oluşturacağız. İsim ataması yaparken istersek burada yaptığımız gibi değişken ataması yapabilir. İnteger türünde "potPin" değişkeni oluşturup içerisine 3 yazabiliriz. Bu durumda "potPin" yazdığımız her yere 3 sayısı yazılmış gibi olacaktır. Bu tanımlamayı yaparken dilerseniz "define" komutunu da kullanabilirsiniz. Bu komutu kullanırken "#define potPin 3" şeklinde kullanılmalıdır. Eşittir ifadesi ve satır sonunda noktalı virgül koymaya gerek yoktur.

Arduino

```
int potPin =A0;      //
int potDeger = 0;    //


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


int kirmiziDeger = 0;
int yesilDeger = 0;
int maviDeger = 0;
```

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

void loop()
{
  potDeger = analogRead(potPin);

  if (potDeger < 341)
  {
    potDeger = (potDeger * 3) / 4;

    kirmiziDeger = 255 - potDeger;
    yesilDeger = potDeger;
    maviDeger = 0;
  }
  else if (potDeger < 682)
  {
    potDeger = ( (potDeger-341) * 3) / 4;

    kirmiziDeger = 0;
    yesilDeger = 255 - potDeger;
    maviDeger = potDeger;
  }
  else
  {
    potDeger = ( (potDeger-683) * 3) / 4;

    kirmiziDeger = potDeger;
    yesilDeger = 0;
    maviDeger = 255 - potDeger;
  }
  analogWrite(kirmiziPin, 255-kirmiziDeger);
  analogWrite(yesilPin, 255-yesilDeger);
  analogWrite(maviPin, 255-maviDeger);
  }
```

Projenin "setup" kısmında ise çıkış vereceğimiz pinleri belirlememiz yeterli. Kırmızı, yeşil ve mavi LED'in eksi bacakları için 3 adet çıkış ihtiyacımız olacak.

Ana program döngüsüne geçtiğimizde ise "potPin" pininden okuduğumuz değeri değerlendirerek devam edeceğiz. Okumayı yaptıktan sonra hem "if", "ifelse" ve "else" komutlarını kullanarak karar verip gerekli LED'leri gelen değerlere göre yakacağız. İlk "if" komutunda gelen değer 341'den küçük ise "if" parantezi altındaki işlemleri yapmasını istiyoruz.

. "if" içerisinde "potDeger" içerisindeki değeri 0-255 arasında (PWM çıkış verebildiği değerler) oranlamak için 4'e bölüp, 3 ile çarpıyoruz. Bu sayede 340 değeri geldiğinde bu hesaplama sonucunda 255 değerini elde edeceğiz.

Analog pin üzerinden 0 ile 1023 arasında okuma yapabiliyoruz. Bu değeri 3 adet LED olduğu için 3 farklı bölgeye böldük. 0-1023 arasında bu bölgeler 0-341, 342-681, 682-1023 olarak belirledik. Gelen değerin bu bölgelerden hangisinde olduğunu belirlemek için "if-else" yapısını kullanıyoruz. Gelen değer "if" komutu ile değerlendirilir eğer istenen değerse "if" içerisi yapılır ve diğer koşullar (if else, else) atlanır. Eğer "if" koşulu sağlanamaz ise "if else" yani ikinci bölge değerlendirilir. Burası sağlanıp sağlanmadığına göre ya içerisindeki komutlar uygulanır yada "else" satırına geçilir. "if else" satırlarını çoğaltarak 3 basamak yerine birçok basamak belirleyebilirsiniz.

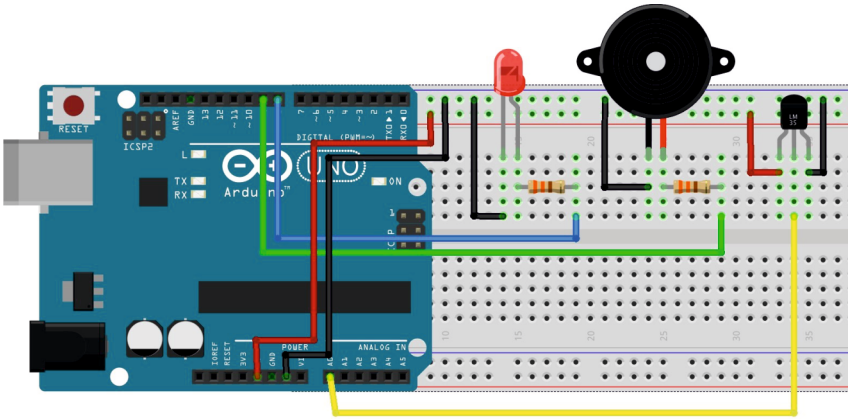
Her basamak içerisinde benzer komutlar uygulanıyor. Sadece atanan değer farklı renklerde oluyor. Örneğin "if" içerisini incelersek, gelen değer 0 ile 255 arasına oranlanıyor. Burada bir açıklama yapmamız gerekli bir LED'i PWM ile kontrol ederken 255 verdiğimizde tam parlaklıkta yanar. Ancak buradaki devrede LED'in artı bacağını değil eksi bacağını PWM pin'ine bağladığımız için ters ekti olacaktır. PWM çıkışından 0(sıfır) verdiğimizde LED tam parlaklıkta yanacak, 255 verdiğimizde sönecektir. Bu problemi ters durumu aşmak için çıkan değerleri LED'lere yollamadan önce 255'ten çıkararak yollayacağız. Bu durumda "if" koşulları içerisinde sanki normal LED bağlamış gibi kodumuzu yazabileceğiz. İlk "if" içerisinde kırmızı LED'e göndermek üzere 255'ten "potDeger"i çıkararak gönderiyoruz. Yeşil LED'e ise direk olarak potDeger'ini gönderiyoruz. Mavi LED'i söndürmek için 0(sıfır) değerini atıyoruz. Renkler arasında geçiş için 3 basamağın her birinde bir LED'i tamamen söndürüp diğer LED'lere gönderilen değerlerin toplamının 255 olmasını sağlıyoruz. Bu yöntem ile potansiyometreyi çevirdiğimizde bir rengin parlaklığı artarken diğeri azalıyor ve renk geçişleri meydana geliyor.

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 9 Adet Erkek-Erkek Jumper Kablo
- LM35 Sıcaklık Sensörü
- 5mm Kırmızı LED
- 2 Adet 330 Ohm (Turuncu-Turuncu-Kahverengi)
- Buzzer

LM35 sensörü sıcaklığı okuyarak analog veriye dönüştürür. LDR uygulamamızda yaptığımız gibi analog veriyi okuyup yorumladığımızda ordamdaki veya LM35 bağlı olduğu cismin sıcaklığını ölçmüş oluruz. LM35 ile -55°C ile +150°C derece arasında ölçüm yapabilirsiniz. Lineer çalışan, yani sıcaklık arttıkça aynı oranda çıkıştaki gerilimi arttırarak 10mV/°C hassasiyetinde çalışır. LM35'in üç bacağı bulunmaktadır.

Sensörün üzerindeki kesim kısım aynı zamanda yazılı kısıma önden baktığınızda sağda kalan bacak negatif(-), solda kalan bacak pozitif(+) ve ortadaki bacak ise sinyal çıkışıdır. Sensörün pozitif bacağına +5V ve negatif bacağınıda GND'ye bağlıyoruz. Buzzer'ı kullanırken, LED kullanırkenki gibi direnç koyarak buzzer'ın zarar görmesini engelliyoruz. Buzzer'ın pozitif bacağını ise direnç üzerinden Arduino kartımızın 9 numaralı pinine bağlıyoruz. Buzzer'ın negatif bacağını ise GND hattına bağlıyoruz. LED'i ise pozitif bacağını direnç üzerinden Arduino kartınızın 8 numaralı pinine negatif bacağını ise GND hattına bağlıyoruz. Devremiz tamamlanmış oldu.



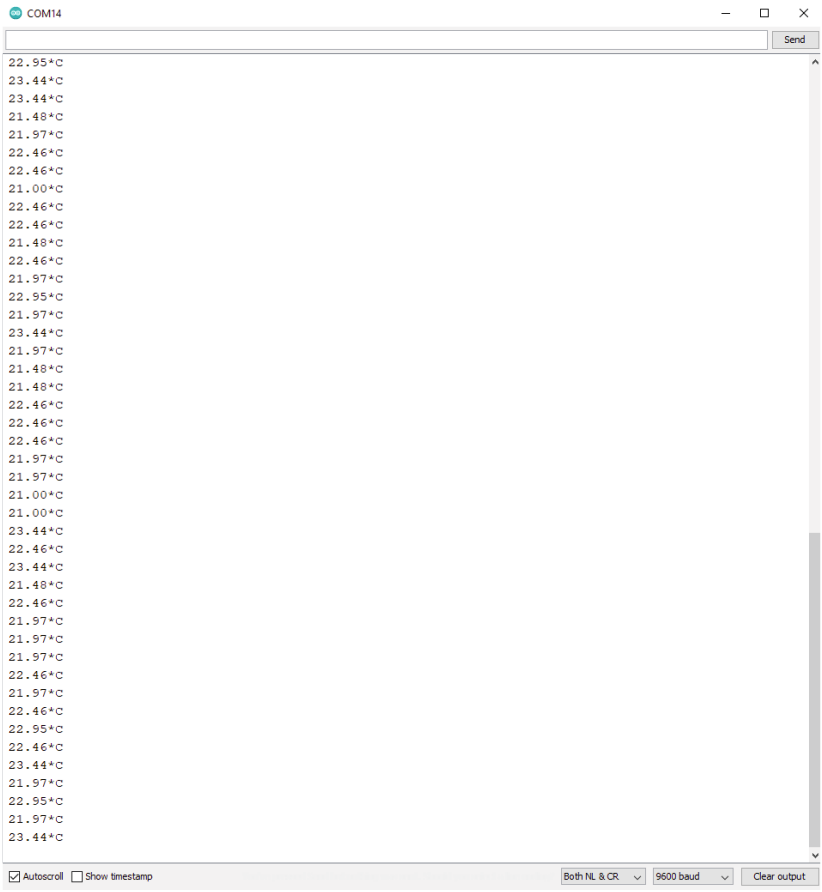
```arduino
int lm35Pin = A0;
int led = 8;
#define buzzer 9
int zaman = 50;
int okunan_deger = 0;
float sicaklik_gerilim = 0;
float sicaklik = 0;

void setup()
{
  pinMode(led,OUTPUT);
  pinMode(buzzer,OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  okunan_deger = analogRead(lm35Pin);
  sicaklik_gerilim = (okunan_deger / 1023.0)*5000;
  sicaklik = sicaklik_gerilim /10.0;
  Serial.println(sicaklik);
  if(sicaklik >= 30){
    digitalWrite(led,HIGH);
    digitalWrite(buzzer,HIGH);
    delay(zaman);
    digitalWrite(led,LOW);
    digitalWrite(buzzer,LOW);
    delay(zaman);
  }
  else{
    digitalWrite(led,LOW);
    digitalWrite(buzzer,LOW);
  }
}
```

Kod kısmına geçtiğimizde "int lm35Pin = A0;" satırı ile A0 pinine "LM35Pin" ismini veriyoruz. İkinci satırda yine 8 numaralı bacağa "led" ismini veriyoruz. Üçüncü satırda ilse isimlendirme işlemini "#define" komutunu kullanarak yaptık. Bu kodda kullanım bakımından iki tanımlamanın bir farkı bulunmamaktadır. Daha ileri seviye uygulamalarda sizde zamanla "int" ve "#define" komutlarının farkını anlayabileceksiniz. Kodumuzun içerisinde kullanacağımız "zaman" , "okunan değer" değişkenlerini "int" değişkeni kullanarak tam sayı olarak belirledik. "sicaklik_gerilim" ve "sicaklik" değişkenlerini "float" değişken tipi kullanarak noktalı sayı değerleri alabilecek şekilde tanımladık.

Setup kısmında 2 adet çıkış bacağımızı ve Seri haberleşme başlatmayı ayarladıktan sonra asıl kodumuzun çalışacağı loop kısmına geçebiliriz.

Loop kısmında ilk yapmamız gereken A0 pininden okuma yapıp okudumuz gerilim değerini Arduino içerisinde yorumlayabildiğimiz bir sayıya çevirmek. Loop içerisindeki ilk üç satırda bu işlemi gerçekleştirip "Serial.println(sicaklik);" komutu ile "sicaklik" değişkeni içerisine yazdığımız sıcaklık değerini seri monitörde gözlemliyoruz. Seri mönitörün nasıl kullanıldığını önceki projelerimizde görmüştük.

Sıcaklık değeri 30°C dereceyi geçtiğinde hem buzzer hem de LED ile uyarı vermesini istiyoruz. Bu sebeple 22.satırdaki if komutu ile sıcaklığın istediğimiz değerden büyük olup olmadığını kontrol ediyoruz. Eğer 30°C dereceden büyük ise Buzzer ve LED kesik aralıklar ile bize uyarı verecek. Eğer 22.satırdaki 30°C dereceden büyüktür koşulu sağlanamaz ise ozaman 30.satırdaki "else" ifadesinin altındaki Buzzer ve LED'i söndürme komutları çalışacaktır.

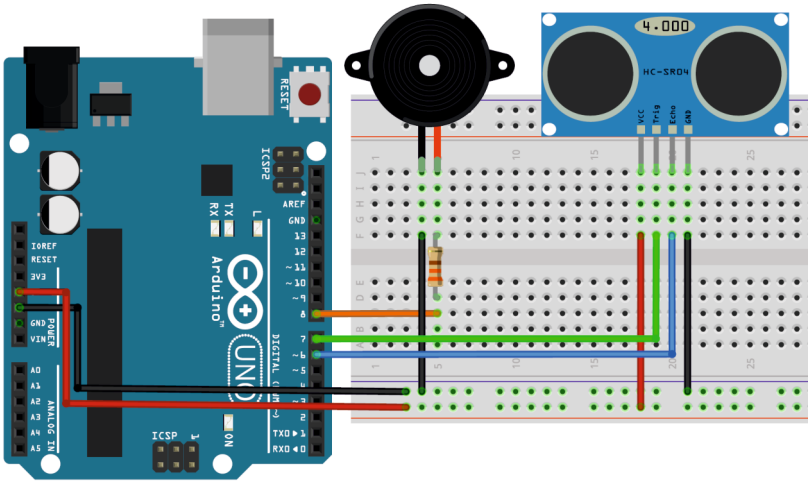


Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 8 Adet Erkek-Erkek Jumper Kablo
- Buzzer
- 330 Ohm Direnç (Turuncu-Turuncu-Kahverengi)
- HC-SR04 Ultrasonik Sensör

Ultrasonik sensör uygulamamızda yeni bir kullanım örneği göreceğiz. HC-SR04 ultrasonik sensörü üzerinde bir tane ses gönderebilen, bir tane de ses algılayabilen metal kısımlar bulunuyor. Sensörden ses gönderildikten sonra eğer önünde cisim veya engelden yansıyarak tekrar sensöre geliyor. Alıcı kısım yansıyan sinyali algılayarak ölçüm yapabiliyor.

Mesafeyi ölçmek için sinyal gönderdikten itibaren geri gelene kadarki süreyi ölçüyoruz. Sesin havadaki hızını bildiğimiz için süre ve hızdan toplam mesafeyi hesaplayabiliyoruz. Hemen devremizi kurarak devam edelim.



```arduino
#define echoPin 6
#define trigPin 7
#define buzzerPin 8

int maximumRange = 50;
int minimumRange = 0;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  int olcum = mesafe(maximumRange, minimumRange);
  melodi(olcum*10);
}

int mesafe(int maxrange, int minrange)
{
  long duration, distance;

  digitalWrite(trigPin,LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration / 58.2;
  delay(50);     //50 ms bekliyoruz.

  if(distance >= maxrange || distance <= minrange)
  return 0;
  return distance;
}

int melodi(int dly)
{
  tone(buzzerPin, 440);
  delay(dly);
  noTone(buzzerPin);
  delay(dly);
}
```

Yazılım kısmında"#define" komutları ile kullanacağımız pinlere isimler veriyoruz. "maximumRange" ve "minimumRange" isminde "integer"(tamsayı) tipinde değişkenler tanımlıyoruz. "setup" kısmında giriş ve çıkış olacak pinleri ayarlıyoruz.

Ana program döngümüz çok kısa görünüyor. Bu kısımda ilk olarak mesafe fonksiyonuna gidiyoruz."long" türünde "duration" ve "distance" değişkenleri tanımlanıyor."long" Önceden kullandığımız "integer" gibi bir değişken. İçerisinde "integer" değişkenine göre çok daha büyük sayılar tutabilir. +2,147,483,647'den, -2,147,483,647'ye kadar içerisine atanabilmektedir. tutabilmektedir. Tutabildiği sayı hacminden dolayı tanımlandığında "integer" değişkenine göre 2 kat fazla hafıza kullanır. Değişken tanımladıktan sonra sensörün "trig" pinini yüksek ve alçak yaparak sensörün fiziksel ortama ses dalgası yollamasını sağlıyoruz. Ses dalgası yollandıktan sonra "pulseIn(echoPin,HIGH)" komutu ile yolladığımız ses dalgasının cisimden yansıyıp geri gelmesini bekliyoruz. Bu beklediğimiz zamanıda "pulseIn" komutu ile ölçebiliyoruz. Ölçtüğümüz bu değer "duration" değişkenine yazdırıyoruz. Süre ölçüldüğüne göre şimdi mesafeyi hesaplamaya geldi. Ölçtüğümüz süreyi sesin hızına göre mesafeye çevirmek için "58.2"ye bölüyoruz. Mesafe değerine ulaşınca, bu değer sensörün ölçebildiği minimum (2 cm) ve maksimum (400 cm) arasında değilse 0(sıfır) değeri ile dönüş yap diyoruz. İstediğimiz araralıkta ise tekrar ana fonksiyona dönerek "olcum" değişkeni içerisine veri yazılıyor.

Ana fonksiyonumuzda "melodi" fonksiyonuna olcum değişkeninin içindeki değer 10 ile çarpılıp gönderiliyor. Bu değer "melodi" içerisindeki bekleme sürelerinde kullanılarak 2 dıt sesi arasındaki süreyi belirleyecek. Eğer sensör az mesafe ölçüyor ise kısa aralıklar ile, eğer sensör uzun mesafe algılıyor ise uzun aralıklar ile ötecek.

robotistan BLOG

Uygulamaların blog yazısına
aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinodersleri

YouTube

Uygulamaların videosuna
aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

robotistan.com

Robotistan Elektronik Ticaret A.Ş.

Hazırlayanlar: İlge İPEK (İçerik - Editör) - Yasir ÇİÇEK (Editör) - Mehmet Nasır KARAER (Grafik)
info@robotistan.com - www.robotistan.com
Tel: 0850 766 0 425