GİKAL TEKNOLOJİ

OCAK 2023
Sayı 2

DEVRE TASARIMLARI
PROJELERLE ARDUINO

YAYIN EKİBİ
RAMAZAN ÇOBAN
ÖZLEM MENEKŞE
ETEM DURSUN

05

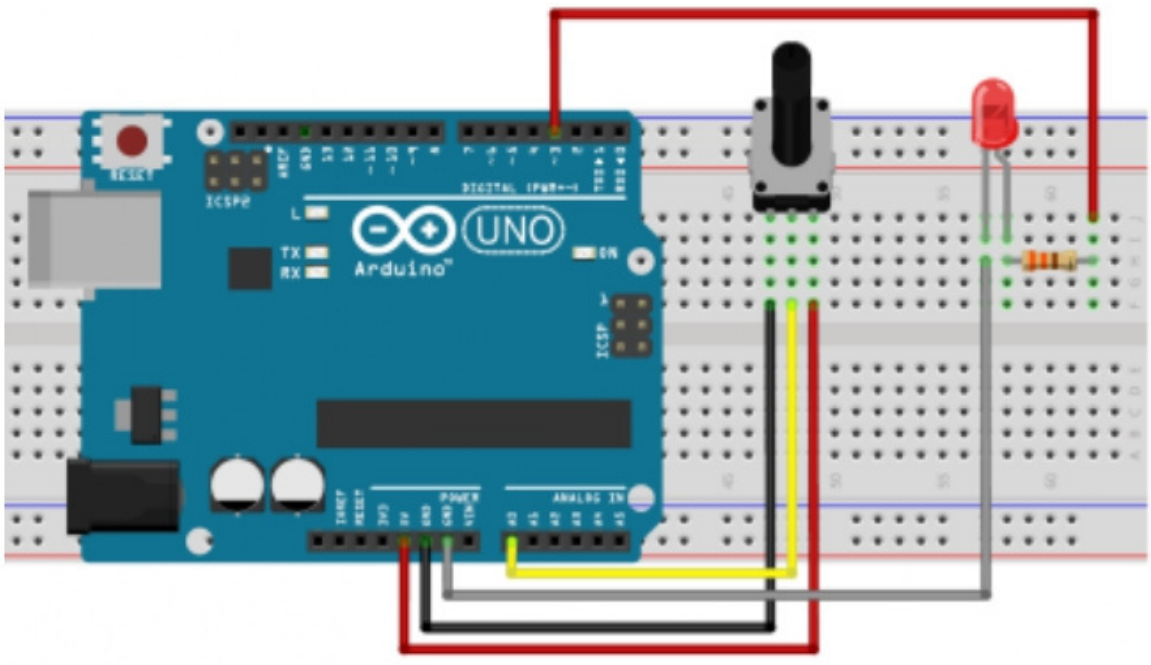
Potansiyometre ile LED Yakma

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.



Potansiyometre ile LED Yakma

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.

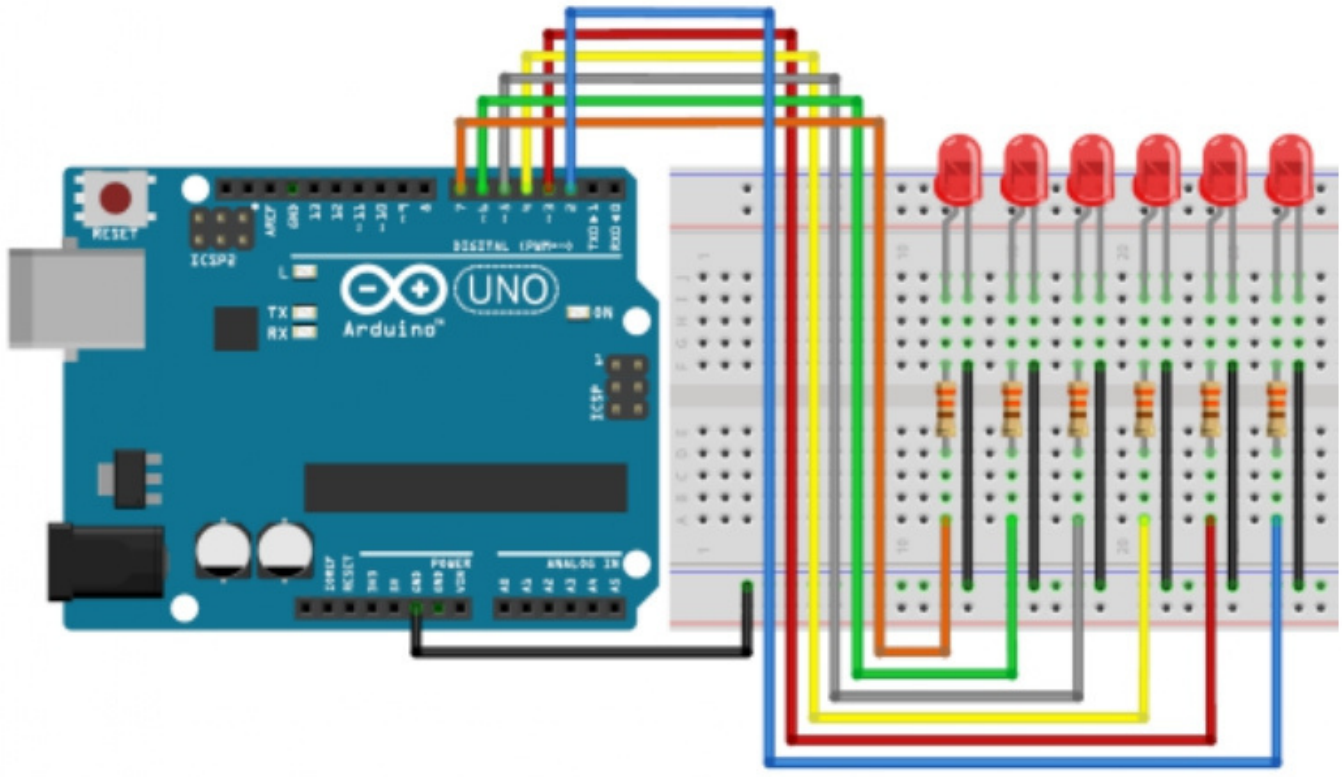
06

Arduino ile Karaşimşek Uygulaması

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.



```
Arduino
1  int ledler[] = {2,3,4,5,6,7};
2
3  void setup() {
4    for(int i=0; i<6; i++){
5    pinMode(ledler[i], OUTPUT);
6    }
7  }
8
9  void loop() {
10   for(int i=0; i<6; i++){
11     digitalWrite(ledler[i], HIGH);
12     delay(80);
13     digitalWrite(ledler[i], LOW);
14   }
15
16   for(int j=5; j>-1; j--){
17     digitalWrite(ledler[j], HIGH);
18     delay(80);
19     digitalWrite(ledler[j], LOW);
20   }
21
22 }
```

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.

07

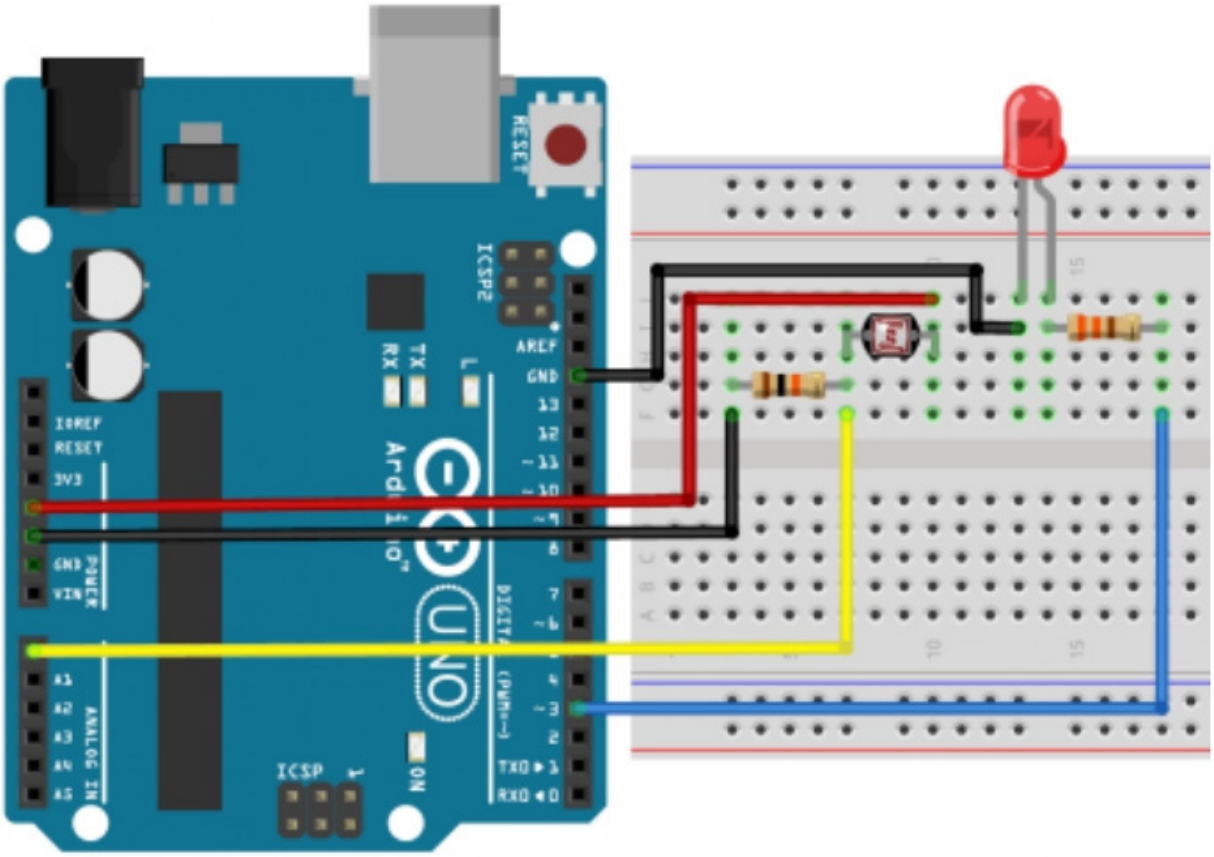
LDR ile Otomatik Lamba Uygulaması

LDR ile Otomatik Lamba Uygulaması

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.



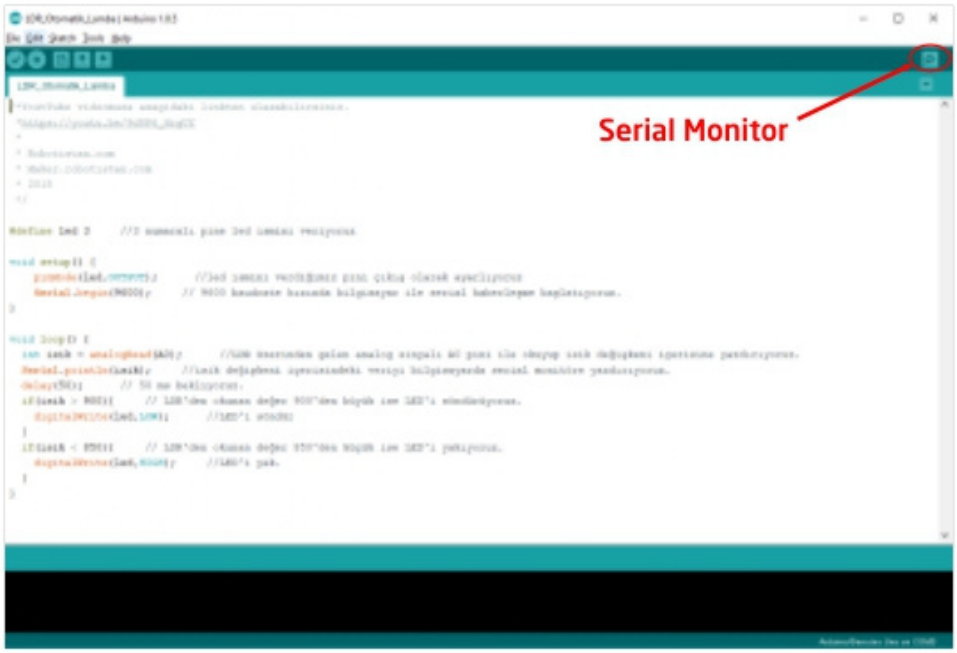
```arduino
#define led 3

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);
  }
}
```

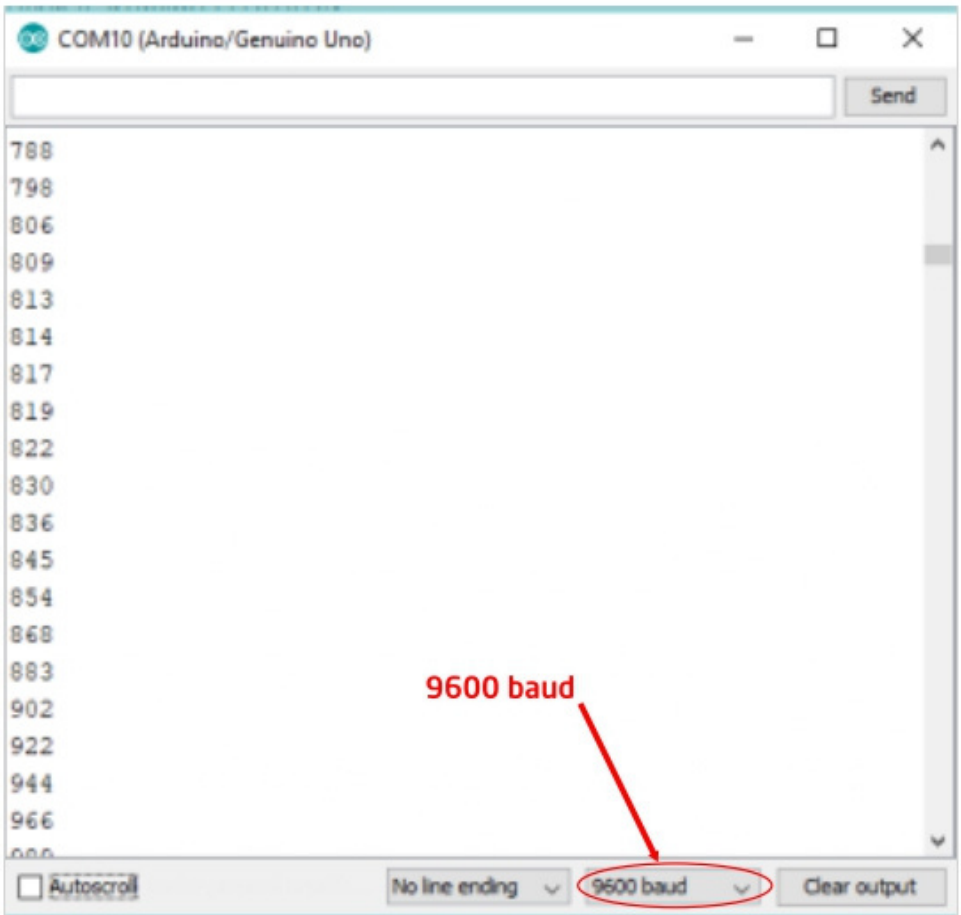
Kod kısmına geçecek olursak ilk satırımızda LED'i bağlayacağımız pine isim veriyoruz. Bu işlemi "#define" komutu ile yapacağız. Bu işlemden sonra artık ihtiyaç halinde 3 yazmak yerine "led" yazarak işlemleri kolaylaştıracağız.

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 okudğunuz değerlerde değişecektir.





08

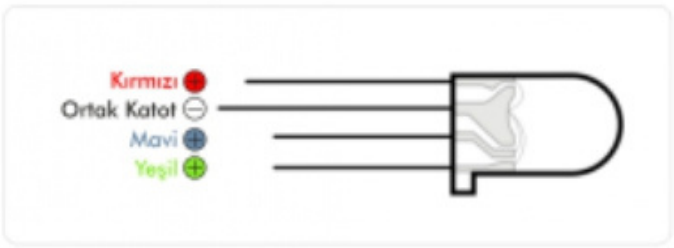
Arduino ile RGB LED Uygulaması

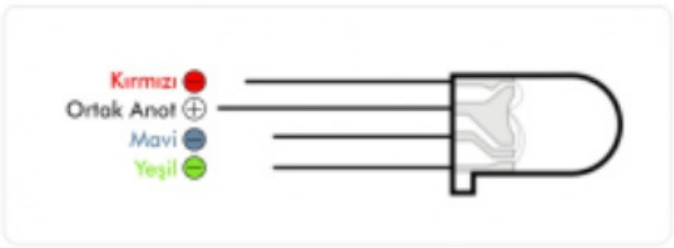
RGB LED Uygulaması

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.

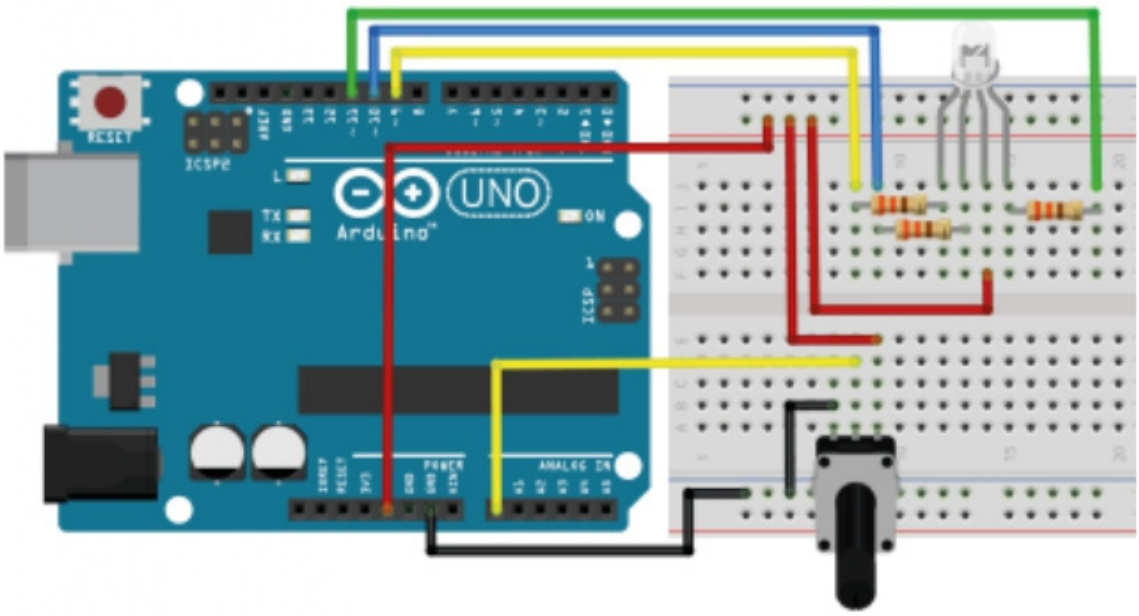




Bu uygulamamızda renkleri tam parlaklık dışında ara renklerde de yakmak istiyoruz. LED'i istediğimiz parlaklık seviyesinde yakabilmek için PWM özelliğini kullanmamız gerekiyor. PWM özelliği belirli bacaklarda (3,5,6,9,10,11) bulunduğu için bağlantılarımız yaparken bu bacakları kullanmaya dikkat edeceğiz.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

RGB LED Uygulaması

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 = 3;      //
int potDeger = 0;    //


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


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

```

```
Arduino
14 void setup()
15 {
16   pinMode(kirmiziPin, OUTPUT);
17   pinMode(yesilPin, OUTPUT);
18   pinMode(maviPin, OUTPUT);
19 }
20
21 void loop()
22 {
23   potDeger = analogRead(potPin);
24
25   if (potDeger < 341)
26   {
27     potDeger = (potDeger * 3) / 4;
28
29     kirmiziDeger = 255 - potDeger;
30     yesilDeger = potDeger;
31     maviDeger = 0;
32   }
33   else if (potDeger < 682)
34   {
35     potDeger = ( (potDeger-341) * 3) / 4;
36
37     kirmiziDeger = 0;
38     yesilDeger = 255 - potDeger;
39     maviDeger = potDeger;
40   }
41   else
42   {
43     potDeger = ( (potDeger-683) * 3) / 4;
44
45     kirmiziDeger = potDeger;
46     yesilDeger = 0;
47     maviDeger = 255 - potDeger;
48   }
49   analogWrite(kirmiziPin, 255-kirmiziDeger);
50   analogWrite(yesilPin, 255-yesilDeger);
51   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.

09

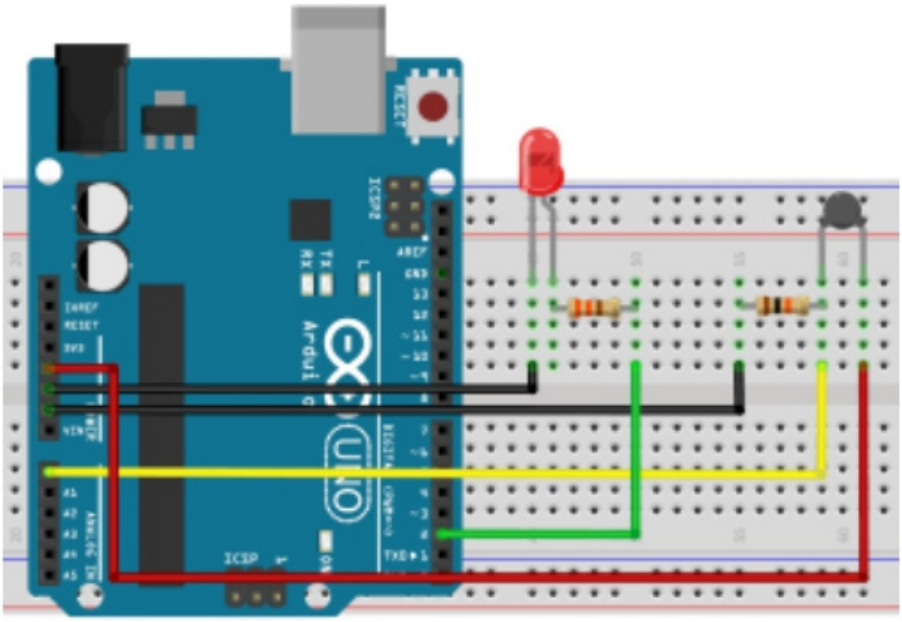
NTC ile Sıcaklık Ölçümü

NTC ile Sıcaklık Ölçümü

Gerekli malzemeler:

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

LDR uygulamamızda yaptığımız gibi sıcaklık okurken de analog sinyali okuyup yorumlayarak istediğimiz şeyleri yaptıracağız. NTC (Negative Temperature Coefficient), sıcaklık artışı karşısında iç direncini düşüren bir elemandır. NTC yerine genellikle kullanılan elemanlardan bir tanesi de PTC'dir. PTC ise sıcaklık artışı karşısında iç direncini arttırarak tepki verir. NTC'den okuduğumuz veriyi sıcaklık birimine dönüştürmek için bir takım işlemlerden geçirmemiz gerekiyor. Aynı zamanda NTC sensörünün sıcaklık artısına göre değişken direnç değeri sabit olmadığı için logaritmik fonksiyonlardan geçirmek gerekiyor. Bu fonksiyonları detaylı olarak incelememiz şuan için gerekmiyor işlemlerin bu şekilde yapılması gerektiğini bilmemiz yeterli. NTC sensörü kullanım mantığını öğrendikten sonra bu kısmında detaylı inceleyebiliriz. İlk olarak devremizi kurarak başlayalım. Bu uygulamada farklı olarak fonksiyon oluşturup, fonksiyona giderek bazı işlemleri yapacağız.



Arduino

```
#include <math.h>

#define led 2

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

double Termistor(int analogOkuma){
 double sicaklik;
 sicaklik = log(((10240000 / analogOkuma) - 10000));
 sicaklik = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * sicaklik * sicaklik)) * sicaklik);
 sicaklik = sicaklik - 273.15;
 return sicaklik;
}

void loop() {
  int deger = analogRead(A0);
  double sicaklik = Termistor(deger);
  Serial.println(sicaklik);
  if(sicaklik > 30){
    digitalWrite(led,HIGH);
  }
  else{
    digitalWrite(led,LOW);
  }
  delay(250);
}
```

Kod kısmında logaritmik fonksiyonlar kullanacağımız için matematik kütüphanesini dahil etmemiz gerekiyor. "#include <math.h>" satırı ile matematik kütüphanesini dahil ediyoruz. Alt kısmında "#define led 2" satırı ile 2. pine led ismini veriyoruz.

"setup" kısmında verileri bilgisayara göndereceğimiz için seri haberleşme başlatıp, "led" olarak isimlendirdiğimiz pini çıkış olarak tanımlıyoruz.
Ana döngümüzde "A0" pinine bağlı olan NTC üzerinden analog veriyi okuyoruz. Bu veriyi "deger" değişkeni içerisine yazdırıyoruz. Okuduğumuz bu veriyi "Termistor" fonksiyonuna gönderip sıcaklık değerine dönüştüreceğiz. Fonksiyon tanımlamadan da ana döngü içerisine, "Termistor" fonksiyonu içerisindeki kodları yazarak kullanabiliriz. Fonksiyon tanımladığınızda kodun karmaşık yapısından kurtulup bölümlere ayırmış olursunuz. Bu sayede ana kodumuz daha yalın olup hem yazması hem de sonradan okuyunca anlaşılması daha kolay olacaktır.

Matemetiksel işlemleri "setup" ile "loop" fonksiyonları arasında yer alan "double Termistor(int analogOkuma)" fonksiyonu içerisine yazdık. Fonksiyon ismini biz belirlediğimiz için istediğiniz başka bir isimde verebilirsiniz.

Okuduğumuz değeri fonksiyona gönderdiğimizde fonksiyonun geri dönüş değeri sıcaklık olacak, bizde bu değişkeni seri haberleşme ile bilgisayarda seri monitörde görüntüleyeceğiz.

Sıcaklik değerini "if" koşulu ile değerlendirip 30 derecenin üzerinde ise LED'i yakıyoruz. 30 derecenin altında ise LED'i söndürüyoruz. Bu sayede kendimize sıcaklık değerine göre sıcaklık alarm sistemi yapmış olduk. Yazılımın çok hızlı şekilde kendini tekrarlamaması içinde en sona 250 ms'lik bir bekleme koyuyoruz. Arduino IDE üzerinden serial monitörü açıp sıcaklık değerlerini gözlemleyebiliriz.