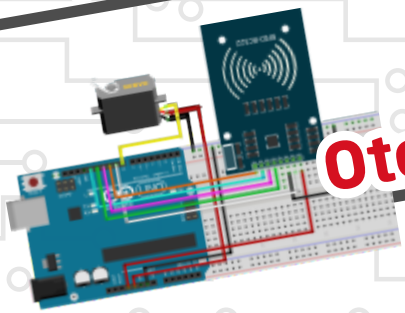
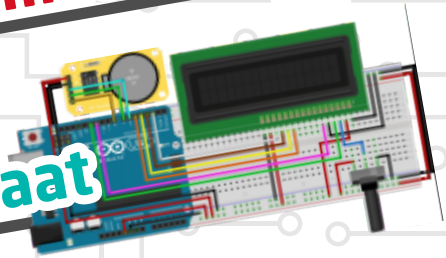
Eğitim Serisi ile Desteklenmektedir
24 Videolu

Arduino Uygulama Kitabı

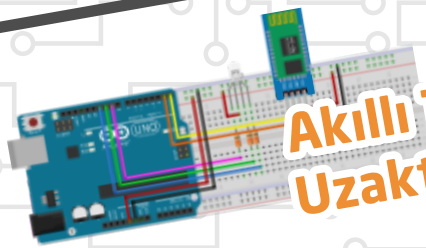
Araç Park Sensörü



Otomatik Kapı



Dijital Saat



Akıllı Telefon ile Uzaktan Kontrol

ve daha fazlası bu kitapta...

2. Baskı

robotistan.com

Elektronik ve Kodlama dünyasına hoşgeldiniz. Bu kitabı açtığınıza göre sizde 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 uygulamaların giriş 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 hemde 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 yapmalarını sağlamaktır.

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

Robotistan Ekibi

İçindekiler

01

Arduino Nedir? Nasıl Kurulur ve Neler Yapılabilir?

robotistan BLOG

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

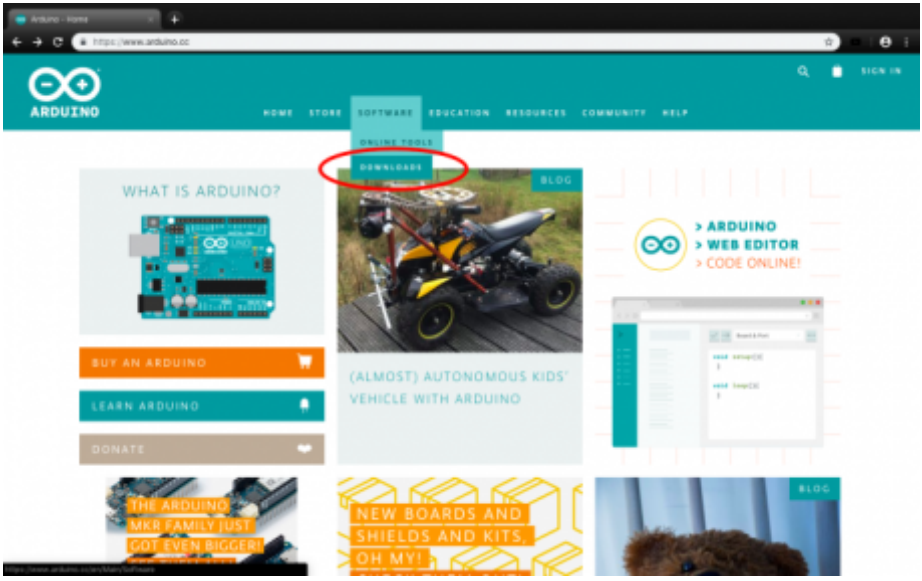
YouTube

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

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.

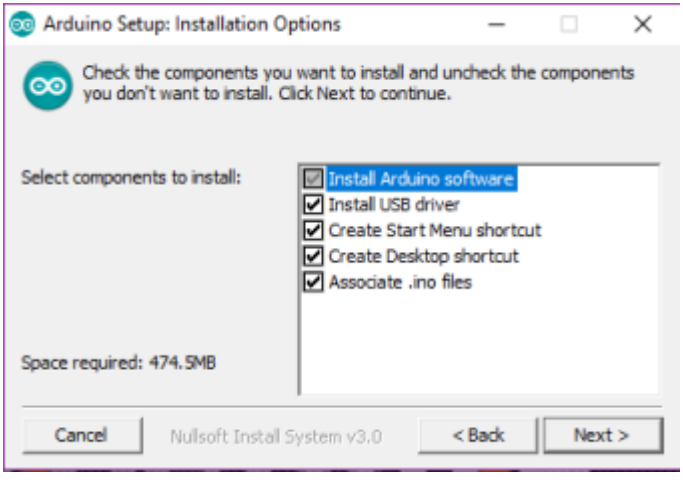


Downloads sekmesini tıkladıktan sonra karşımıza işletim sistemimize göre olan dosyayı indireceğimiz ekran çıkıyor. Bu yazıyı hazırladığım sırada Arduino yazılımının en güncel sürümü 1.8.7 idi. Windows kullananlar **"Windows Installer"** seçeneğini tıklayabilirler. Diğer işletim sistemlerinin de yükleme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

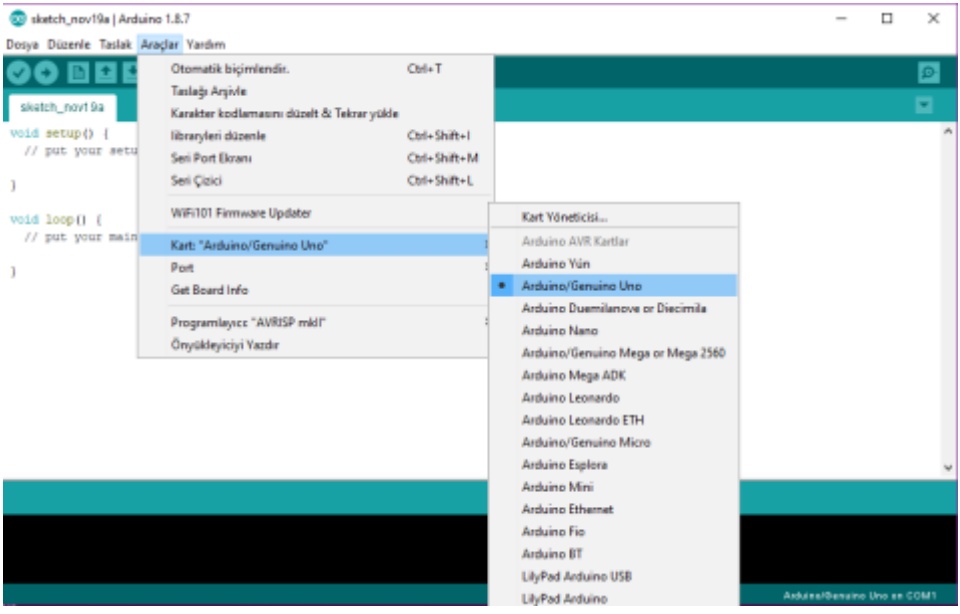
Bundan sonra yazılım kurulum dosyamız inmeye başlıyor. İndirme işlemi bittikten sonra dosyayı açarak kurulum işlemini başlatıyoruz. Kurulum sırasında çıkan "Install USB driver" seçeneğinin seçili olduğundan emin oluyoruz.



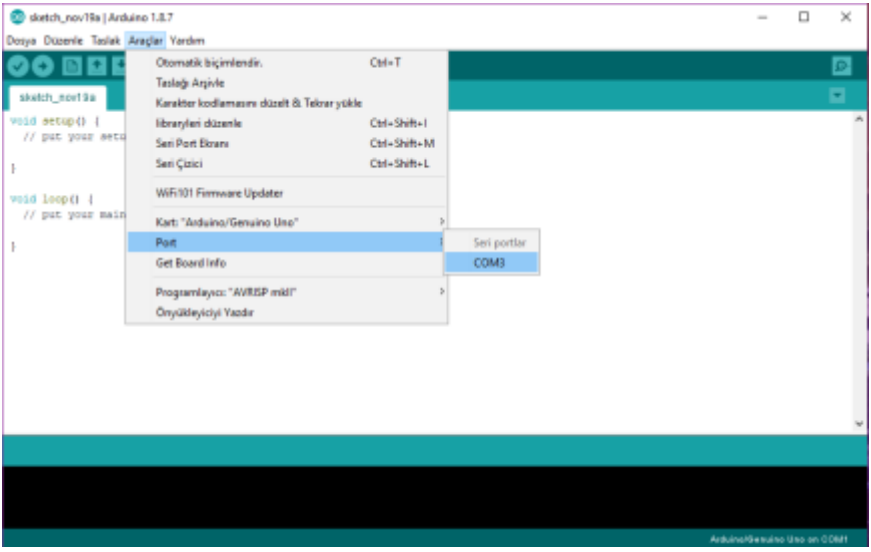
Kurulum işlemi bittikten sonra, kartımızı USB kablomuzla bilgisayarımıza bağlıyoruz. Bilgisayarımızda "Yeni donanım bulundu" penceresi açılıyor. Eğer sürücüler yazılımla birlikte kurulduysa, otomatik yükleme seçeneği Arduino'muzun sürücülerini otomatik olarak yükleyecektir.

Arduino Programının Bilgisayarımızda İlk Çalıştırılması

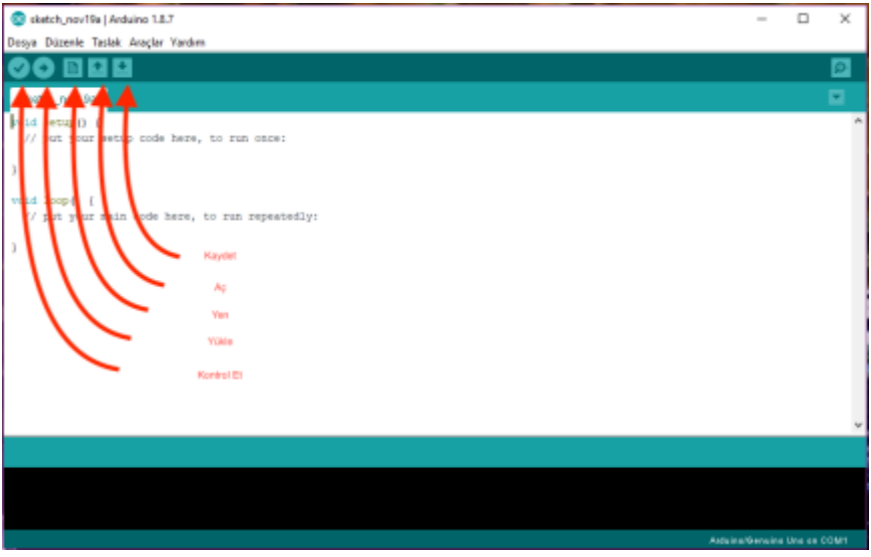
Artık Arduino programımızı açabiliriz. Programımızı açtıktan sonra ilk yapmamız gereken şey, programın Arduino UNO kartımızla çalışacak şekilde ayarlanmasıdır. **Araçlar > Kart** menüsünden Arduino UNO seçeneğini tıklıyoruz.



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

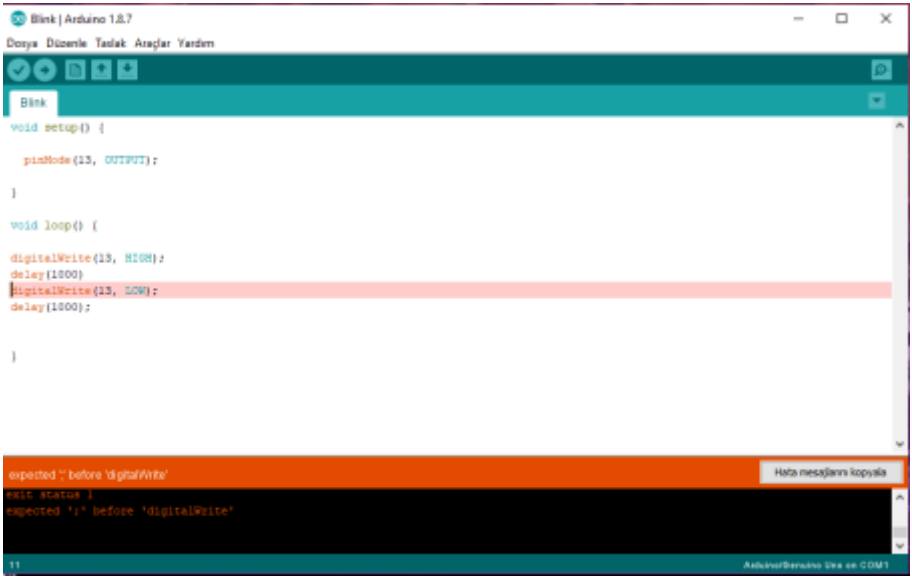


Artık her şeyiyle kullanıma hazır bir Arduino programımız var.



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.

02

Arduino ile LED Yakma
Blink Uygulaması

robotistan BLOG

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

YouTube

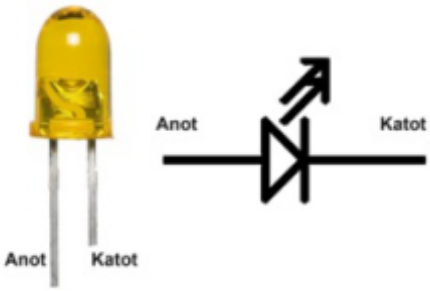
Uygulamanın videosuna
aşağıdaki linkten
ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

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.

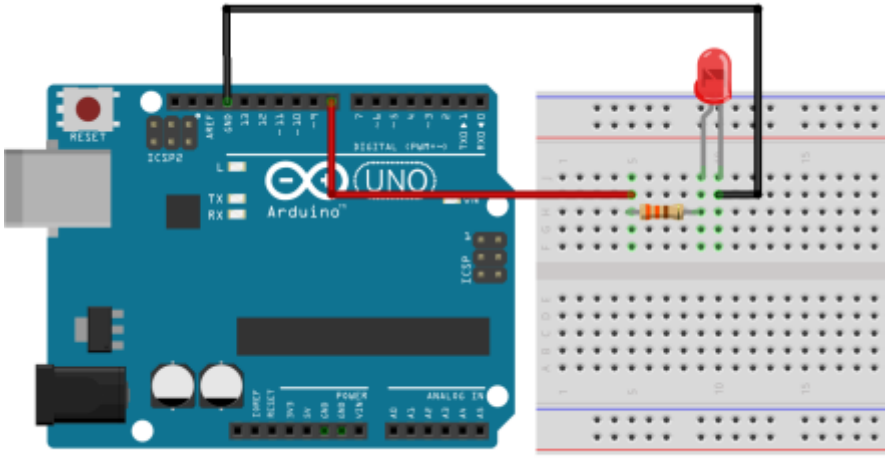
$$V = i \times R$$

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
Blink Uygulaması

robotistan BLOG

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

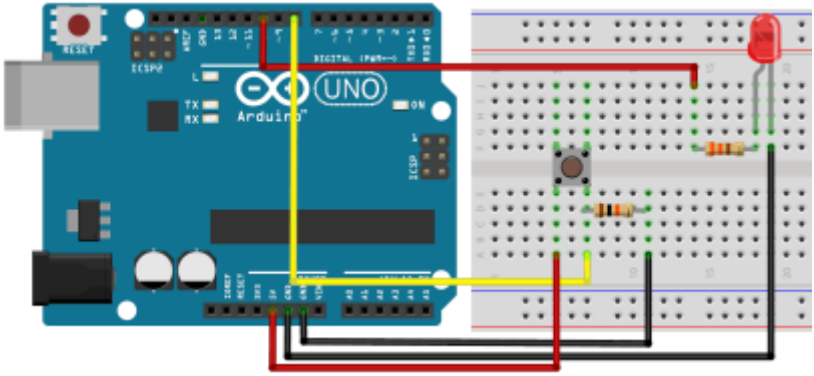
YouTube

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

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.

```
#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.

04

Arduino ile Analog Okuma ve Seri Haberleşme

robotistan BLOG

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

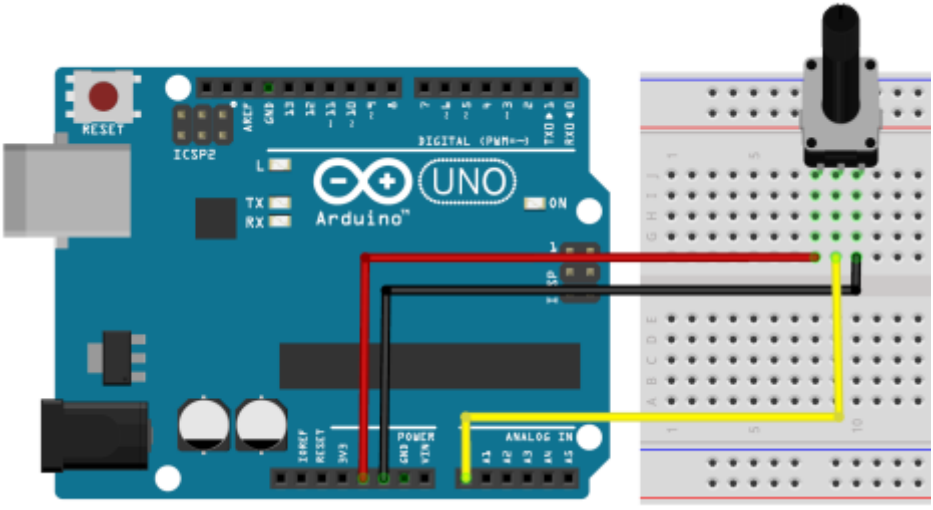
YouTube

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

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

robotistan BLOG

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

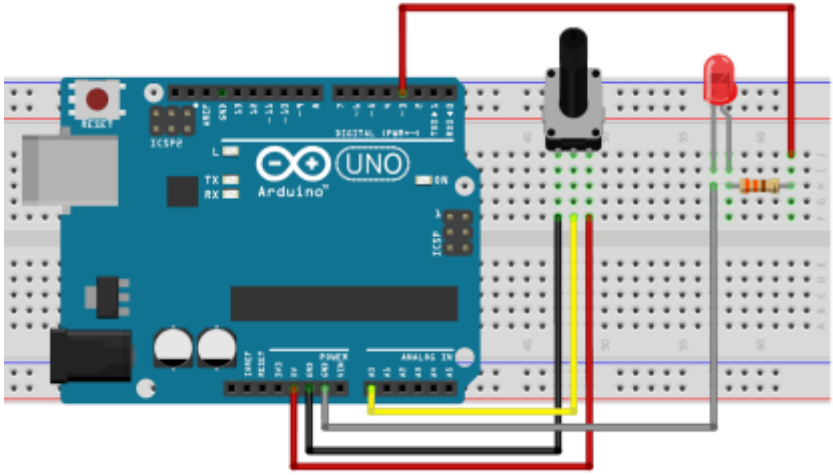
YouTube

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

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.

06

Arduino ile Karaşimşek Uygulaması

robotistan BLOG

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

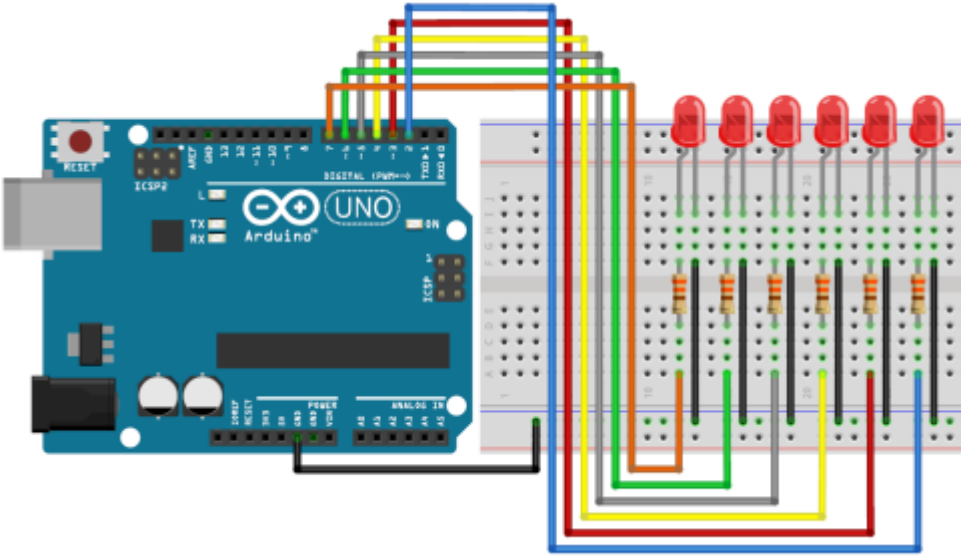
YouTube

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

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.

07

LDR ile Otomatik Lamba Uygulaması

robotistan BLOG

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

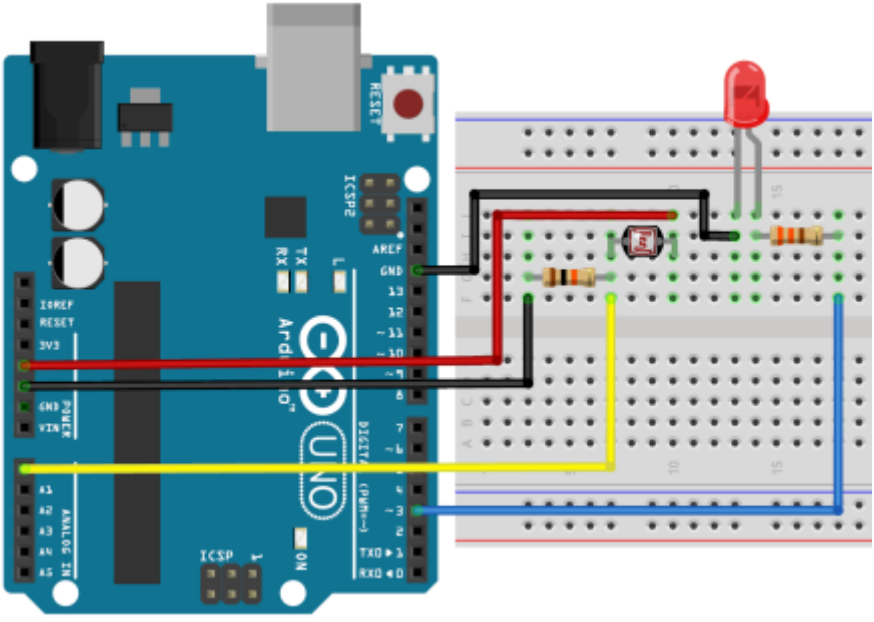
YouTube

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

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.



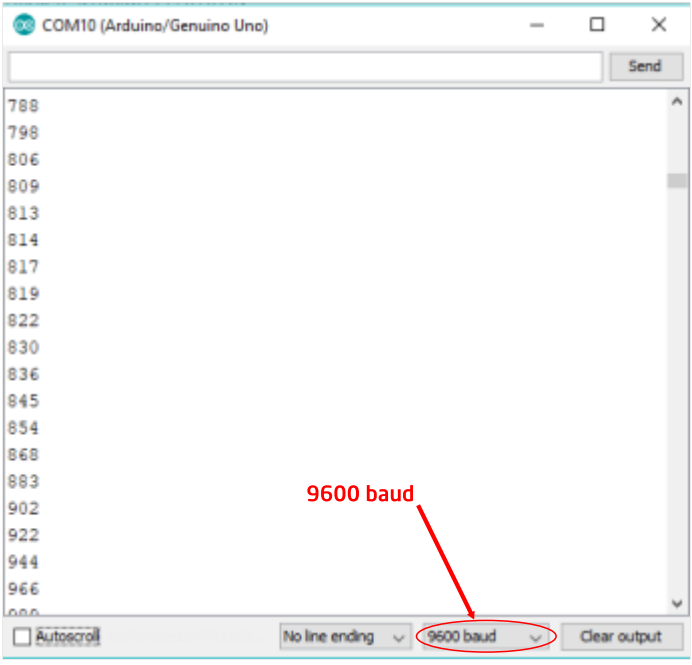
```arduino
#define led 3

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

robotistan BLOG

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

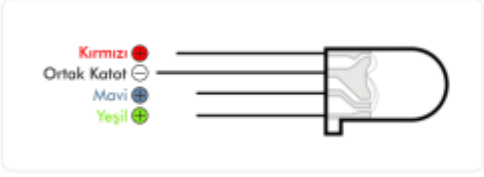
YouTube

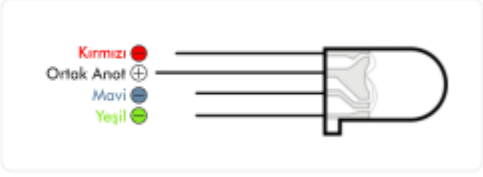
Uygulamanın videosuna aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

Gerekli malzemeler:

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

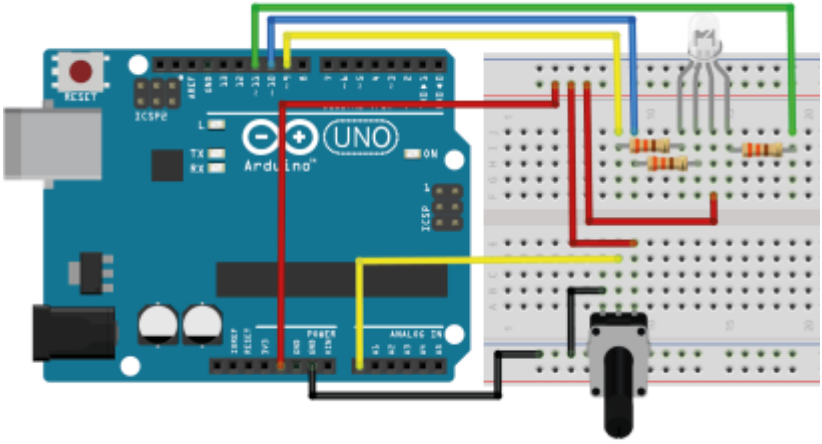
RGB LED içerisinde kirmizi, yeşil ve mavi renkleri içeren 3 adet led yapısı bulunur. İçinde bulundurduğu renklerin baş harflerinin birleşmesi RGB (Red, Green, Blue) ismi oluşmuştur. 3 adet LED düşündüğümüzde her birinde bir artı bir eksi olacak şekilde toplam 6 bacak olması gerekir. Kullandığımız RGB 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.

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
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.

09

NTC ile Sıcaklık Ölçümü

robotistan BLOG

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

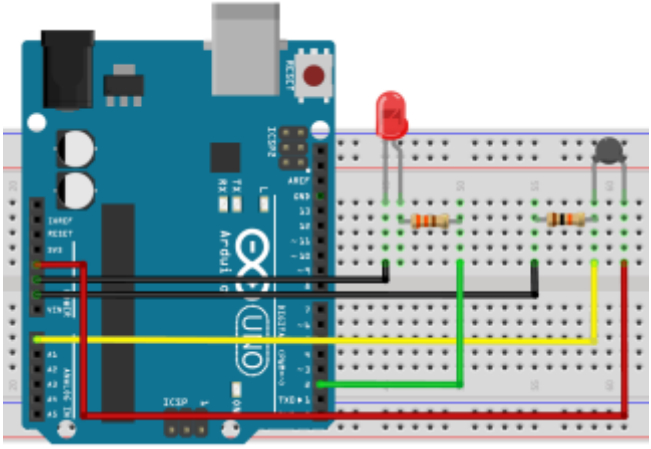
YouTube

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

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);
}
```

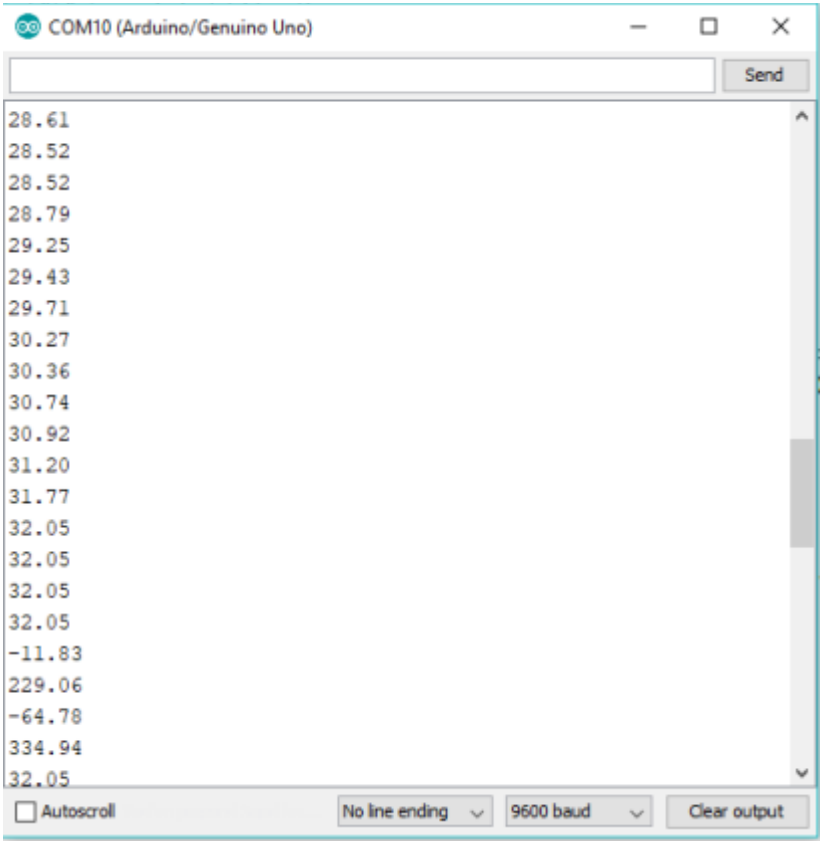
Kod kısmında logaritmik fonksiyonlar kullanacağımız için matematik kütüphanesini dahil etmemiz gerekiyor. "#include <math.h>" satırı ile matematik kütüphanesini dahil ediyoruz. Alt kısmında "#define led 2" satırı ile 2. pine led ismini veriyoruz.

"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.



10

Ultrasonik Sensör ile Park Sensörü Yapımı

robotistan BLOG

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

YouTube

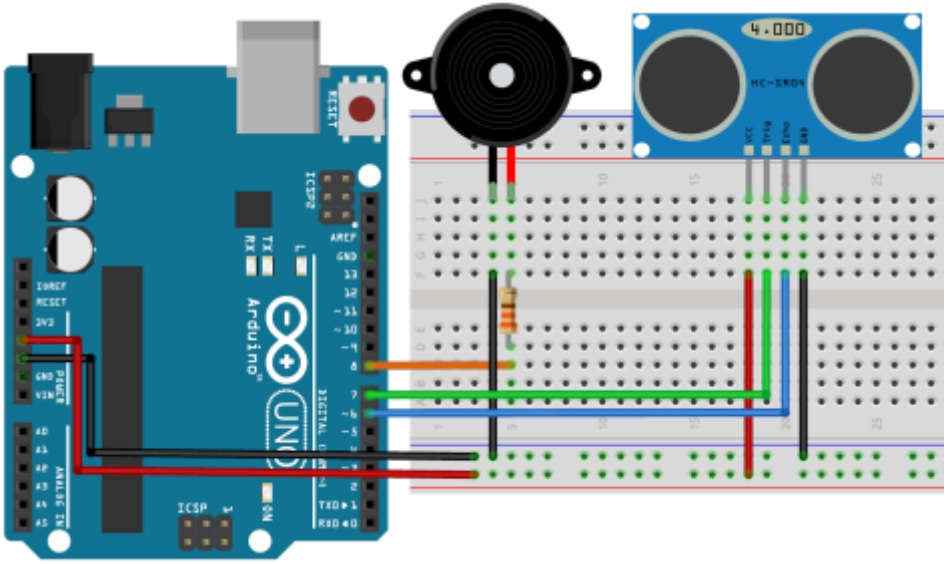
Uygulamanın videosuna aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

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.



```
#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.

11

Ses ile Motor Kontrolü

robotistan BLOG

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

YouTube

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

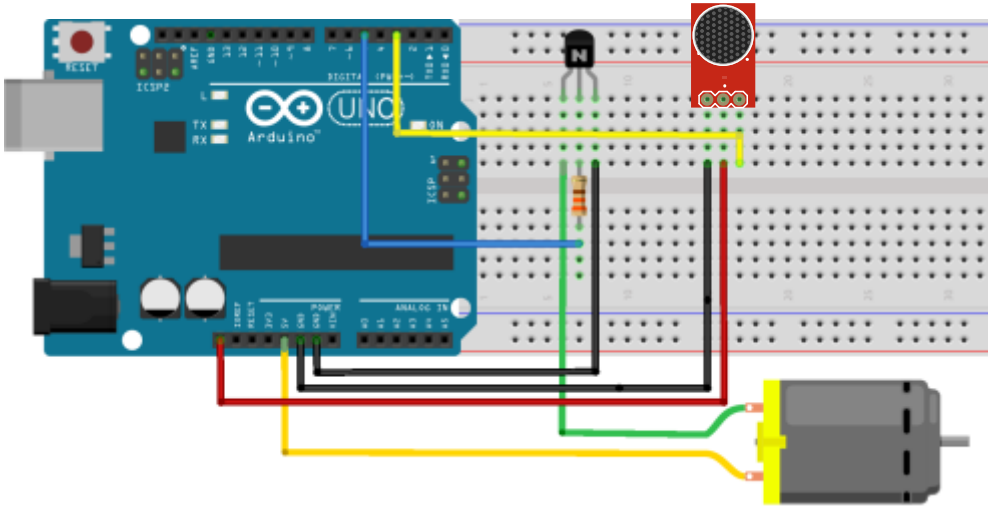
Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 5 Adet Erkek-Erkek Jumper Kablo
- DC Motor
- 330 Ohm Direnç (Turuncu-Turuncu-Kahverengi)
- BC547 NPN Transistör
- Ses Sensör Kartı

Bu örnekte ses sensörü ile okuduğumuz değere göre motor hareketi sağlayacağız. Ses sensörü, üzerinde bulunan mikrofon aracılığıyla ortamdaki ses seviyesini ölçerek dijital bir çıkış vermektedir. Sensör devresi, mikrofondan alınan ses sinyalini yükseltir ve analog ses sinyalini eşik seviyesine göre dijital sinyale çevirir.

Motor, fazla akım çektiği için bu tür devrelerde motor sürücü kartı kullanırız. Motor sürücü, Arduino'dan aldığı sinyale göre motora güç verir. Bu sayede Arduino'muza zarar vermeden güvenli bir şekilde motoru kontrol edebiliriz.

Ses sensörü üzerinde bulunan potansiyometre ile ses seviyesinin eşik değeri ayarlanabilmektedir. Bir tornavida yardımıyla bu ayar yapılabilmektedir.



```arduino
#define SensorPin 3
#define MotorPin 5
int MotorDurum = LOW;

void setup() {
  pinMode(SensorPin, INPUT);
  pinMode(MotorPin, OUTPUT);
}
void loop() {
  if( digitalRead(SensorPin) ){
    if(MotorDurum == LOW){
      MotorDurum = HIGH;
    }
    else{
      MotorDurum = LOW;
    }
    digitalWrite(MotorPin, MotorDurum);
  }
  delay(50);
}
```

İlk olarak ses isimli değişkenimizi tanımlıyoruz. Bu değişken ile sensörden okunan değeri hafızada tutacağız.

"setup()" kısmında, ses sensöründen gelen sinyal pinini giriş olarak ayarlıyoruz. Daha sonra motor sürücüsüne giden 5. pini çıkış olarak ayarlıyoruz.

"loop()" kısmında, "digitalRead()" fonksiyonu ile ses sensöründen gelen veriyi okuyup "if" içerisinde değerlendiriyoruz. Eğer ses gelirse "MotorDurum" değişkeni içerisine "HIGH" yazıyoruz. Değilse "LOW" yazıyoruz. "if-else" döngüsü bittiğinde "digitalWrite(MotorPin,MotorDurum)" satırında dijital çıkış olarak "MotorDurum" değişkenini gönderiyoruz. Motoru değişken kullanmadan "if" ve "else" içerisinde de "digitalWrite(MotorPin, HIGH)" veya "digitalWrite(MotorPin, LOW)" komutları ile de kullanabilirsiniz. Ana döngünün çok hızlı dönmemesi için kodun sonuna 50 ms'lik bir bekleme koyuyoruz.

Ses seviyesi eşik değerini aştığında, ses değişkeni HIGH değerini alır. Bu durumda, 7. pine bağlı motor sürücüsüne HIGH değeri gönderilir ve motor 300 milisaniye boyunca döner.

Ses seviyesi eşik değerinin altında kaldığında, ses değişkeni LOW değerini alır. Bu durumda, 7. pine bağlı motor sürücüsüne LOW değeri gönderilir ve motorun durması sağlanır.

12

Joystick ile Servo Motor Kontrolü

robotistan BLOG

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

YouTube

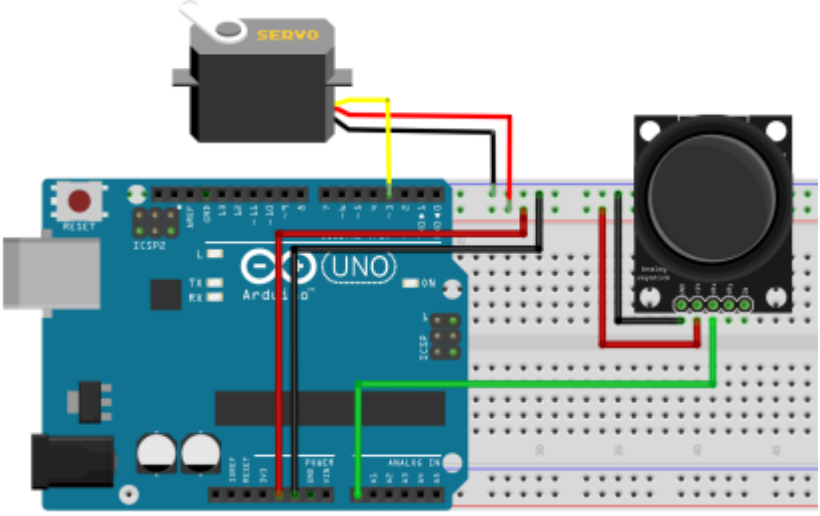
Uygulamanın videosuna aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

Gerekli malzemeler:

- Arduino UNO
- Breadboard
- 1 Adet Servo Motor
- 1 Adet Joystick
- 8 Adet Erkek-Erkek Jumper Kablo

Joystick, üzerindeki kolun ileri-geri veya sağ-sol hareketi yaptırılmasına göre analog sinyal olarak pozisyon çıkışı verir.

Servo, data pininden aldığı sinyale göre belli bir açıda kendini konumlandıran bir motordur.



Joystick üzerindeki kolu ileri-geri hareket ettirdiğimizde VRX pinindeki, sağa-sola hareket ettirdiğimizde VRY pinindeki gerilim değerleri değişir. Joystick üzerine tıkladığımızda ise SW pini 5V çıkış verir. Bu örnekte sadece bir adet servo kullanacağımızdan VRX pinini kullanacağız. VRX pininden alacağımız veri 0 ile 5V arasında analog bir veri olduğundan bu pini Arduino üzerindeki A0 pinine bağlıyoruz. Servo motorun data pinini ise analog çıkış verebilen 3 numaralı pine bağlıyoruz. Örnek kodumuz joystickten alınan veri ile servo motorun 0 ile 180 derece arasında dönmesini sağlar.

```arduino
#include <Servo.h>

Servo motor;
int okuma;
int derece;

void setup() {
  motor.attach(3);
}
void loop() {
  okuma = analogRead(A0);
  derece = map(okuma, 0, 1023, 0, 180);
  motor.write(derece);
}
```

İlk olarak "Servo.h" kütüphanemizi kodumuza ekliyoruz. Daha sonra "motor" adında bir servo motor oluşturuyoruz. "setup()" kısmında, "motor"u 3 numaralı pine bağladığımızı yazılıma belirtiyoruz.

"loop()" kısmında ise ilk olarak A0 pinine bağlı olan joystickten "analogRead()" fonksiyonu ile okuma yapıyoruz ve "okuma" adlı değişkeni "analogRead()" fonksiyonu ile okunan değere eşitliyoruz.

Arduino Uno, analog okuma pinleri üzerinden 0 ile 1023 arasında veri vermektedir. Ancak servo motorumuz 0 ile 180 derece arasında hareket edebilmektedir. Bu yüzden, A0 pininden okuduğumuz değer ile servo motorumuzu kontrol edebilmek için "map()" fonksiyonunu kullanıyoruz. "map()" fonksiyonu girdi olarak verilen değişkenin istenilen aralığa oranlanması sağlar.

"map()" fonksiyonu ile oranlanan okuma değeri, derece değerine eşitlenir. Son olarak derece değeri servo motora yazdırılarak servonun istenilen dereceye gelmesi sağlanır.

13

IR Kumanda ile LED Kontrol

robotistan BLOG

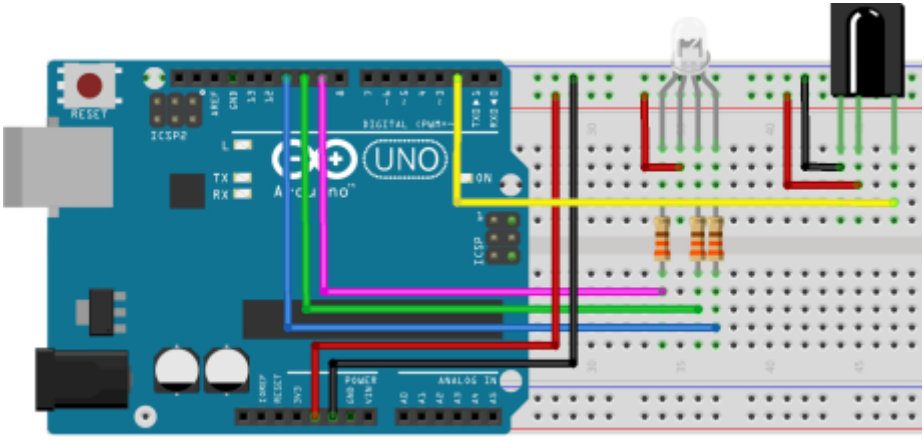
Uygulamanın blog yazısına aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinodersleri

YouTube

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

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 9 Adet Erkek-Erkek Jumper Kablo
- RGB Led
- IR Kumanda ve Modülü
- 3 Adet 330 Ohm direnç (Turuncu-Turuncu-Kahverengi)



Uygulamamızda IR kumandasından aldığımız kızılötesi sinyallere göre LED'in istediğimiz renkte yanmasını sağlamaktadır. 1'e bastığımızda kırmızı, 2'ye bastığımızda yeşil, 3'e bastığımızda mavi LED'ler yanmaktadır. 4'e bastığımızda bütün LED'ler aynı anda yanıp beyaz ışık üretmektedirler. 0'a bastığımızda ise bütün "LED"ler sönmektedir. Diğer butonlara bir atama yapılmamıştır.

IR kumandanın üzerinde bir adet kızılötesi led bulunur. Kızılötesi LED bastığımız butonun adresine göre belli bir frekansta yanıp söner ve IR alıcıya sinyal gönderir. Genellikle bu kızılötesi LED'in yanıp sönme frekansı 38kHz'dir. Bu örnekteki kullandığımız kumandanın frekansı da 38kHz'dir.

Arduino

```
#include <IRremote.h>
int RECV_PIN = 2;
int kirmiziLed = 9;
int yesilLed = 10;
int maviLed = 11;
decode_results results;
IRrecv irrecv(RECV_PIN);

#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

void setup(){
  pinMode(kirmiziLed, OUTPUT);
  pinMode(yesilLed, OUTPUT);
  pinMode(maviLed, OUTPUT);
  Serial.begin(9600);
  irrecv.enableIRIn();
}
void loop() {
  if (irrecv.decode(&results)){
    if (results.value == BUTON1){
      digitalWrite(kirmiziLed, !digitalRead(kirmiziLed));
      if (digitalRead(kirmiziLed) == HIGH){
        Serial.println("Kirmizi yandi");
      }
      else{
        Serial.println("Kirmizi sondu");
      }
    }
    if (results.value == BUTON2){
      digitalWrite(yesilLed, !digitalRead(yesilLed));
      if (digitalRead(yesilLed) == HIGH){
        Serial.println("Yesil yandi");
      }
```

```
Arduino
54        else{
55          Serial.println("Yesil sondu");
56          }
57      }
58      if (results.value == BUTON3){
59        digitalWrite(maviLed, !digitalRead(maviLed));
60        if (digitalRead(maviLed) == HIGH){
61          Serial.println("Mavi yandi");
62        }
63        else{
64          Serial.println("Mavi sondu");
65        }
66      }
67      if (results.value == BUTON4){
68        digitalWrite(kirmiziLed, LOW);
69        digitalWrite(yesilLed, LOW);
70        digitalWrite(maviLed, LOW);
71        Serial.println("Tum LED'ler yandi");
72      }
73      if (results.value == BUTON0){
74        digitalWrite(kirmiziLed, HIGH);
75        digitalWrite(yesilLed, HIGH);
76        digitalWrite(maviLed, HIGH);
77        Serial.println("Tum LED'ler sondu");
78      }
79      irrecv.resume();
80    }
81  }
```

İlk olarak IR modülünü kullanmamız için gerekli olan fonksiyonları içeren "IRremote.h" kütüphanesini kodumuza ekliyoruz. Ardından gerekli olan değişkenlerimizi tanımlıyoruz. Daha sonra "receive" pinimizi kütüphanemize tanıtıyoruz.

Bu kısımda tanımladığımız değişkenler kumanda üzerinde bulunan butonların adresleridir. Bu örnekte bütün butonları kullanmayacağız. Sadece 0, 1, 2, 3, 4 yazan butonları kullanacağız.

"setup" kısmında LED pinlerimizi çıkış olarak ayarlıyoruz. Seri haberleşmeyi 9600 haberleşme hızıyla başlatıyoruz ve IR haberleşmesini başlatıyoruz.

"loop" kısmında, "irrecv.decode(&results)" fonksiyonu ile kumandadan gelen verileri okuyoruz ve "results" değişkenine atıyoruz. "results" değişkenini tanımladığımız adreslerle karşılaştırıyoruz. "results" değişkeni "BUTTON1" e eşit ise kırmızı, "BUTTON2"ye eşit ise yeşil, "BUTTON3"e eşik ise mavi LED'i yakıyoruz. "BUTTON0"a eşit ise tüm LED'leri söndürüyoruz ve "BUTTON4"e eşit ise bütün LED'leri yakıp RGB LED'in beyaz yanmasını sağlıyoruz.

14

Arduino ile Dijital Metre Yapımı

robotistan BLOG

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

YouTube

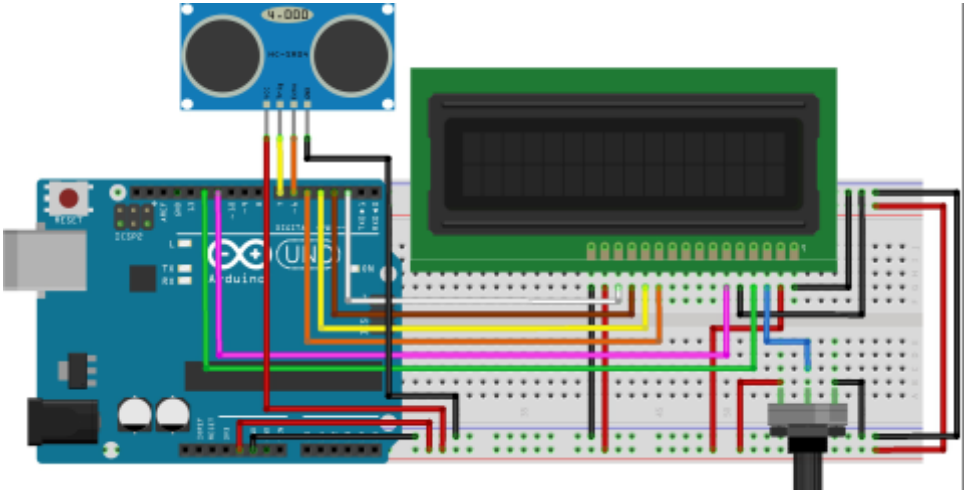
Uygulamanın videosuna aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

Gerekli malzemeler:

- Arduino UNO
- Breadboard
- 16x2 Karakter LCD
- HC-SR04 Ultrasonik Sensör
- 10K Potansiyometre
- 4 Adet Erkek-Dişi Jumper Kablo
- 16 Adet Erkek-Erkek Jumper Kablo

Ultrasonik mesafe sensörü, bir ses dalgası gönderebilen ve yansıyan ses dalgasını tespit edebilen bir cihazdır. LCD, verdiğimiz verilere göre ekrana karakter yazan bir elemandır. LCD ekran, 2 satırdan oluşmaktadır ve her satıra 16 adet karakter yazılabilmektedir. Bir karakter 5x7 tane pixelden oluşmaktadır.

Potansiyometre, ayarlanabilir bir dirençtir. Bu devrede potansiyometreyi gerilim bölücü olarak kullandık. Seri bağlı iki farklı dirence gerilim uyguladığımızda, dirençler üzerinde direnç değerleri ile doğru orantılı gerilimler elde ederiz. Potansiyometre de gerilim bölücü olarak kullanılabilmektedir. Potansiyometre döndürüldüğünde orta pininin gerilimi değişmektedir. Bu değişen gerilimde LCD ekranın Kontrastını ayarlamamızı sağlayacak. Devremizi kurarak devam edelim.



```arduino
#include <LiquidCrystal.h>
int trigPin = 7;
int echoPin = 6;
int sure;
int uzaklik;
int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin,INPUT);
  lcd.begin(16, 2);
}
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  sure = pulseIn(echoPin, HIGH, 11600);
  uzaklik= sure*0.0345/2;
  delay(250);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Uzaklik:");
  lcd.setCursor(0, 1);
  lcd.print(uzaklik);
  lcd.print("cm");
```

Kod kısmına ultrasonik sensörden mesafe bilgisi alıp, LCD ekran üzerinde görüntüleyeceğiz. LCD'yi kullanmak için "LiquidCrystal.h" kütüphanesi kullanılmaktadır. Bu kütüphane, LCD'yi kullanmak için gerekli olan fonksiyonları içinde barındırır. LCD kütüphanemizi kodumuza ekliyoruz. Daha sonra gerekli değişkenler tanımlanır. Ardından LCD pinleri tanımlanır ve gerekli pin ayarları yapılır.

"setup" kısmında, echo pini giriş ve trig pini çıkış olarak tanımlanır. Daha sonra "lcd.begin()" fonksiyonu ile LCD satır-sütun uzunluk ayarı yapılır.

"loop" kısmında, önce trig pini LOW seviyesine getirilerek ultrasonik sensör ölçüm için hazır duruma getirilir. Daha sonra trig pini önce HIGH daha sonra LOW seviyesine çekilerek ses dalgası gönderilir.

"pulseln()" fonksiyonu ile ses dalgasının toplam gidiş-geliş süresi ölçülür. Uzaklığı hesaplamak için gidiş-geliş süresi 0.0345 sayısı ile çarpılır. 0.0345 sayısı, ses dalgasının 1 mikrosaniyede aldığı mesafedir. Ses dalgası gidip geldiği için hesaplanan uzaklık değerinin ikiye bölünmesi gerekir. Park sensörü yapımı uygulamamızda ölçütüğümüz değeri "58.2" sayısına bölmüştük. Bu iki işlem arasında bir fark bulunmuyor birisi böle diğeri çarpma işlemi olarak uygulanmıştır.

"lcd.clear()" fonksiyonu ile daha önceden kalan yazılar ekrandan silinir. "lcd.setCursor()" fonksiyonu ile LCD ekrana yazacağımız yazının hangi satır ve sütuna yazdırılacağı ayarlanır. Daha sonra uzaklık değeri LCD ekrana "lcd.print()" fonksiyonu ile yazdırılır.

15

Hareket Sensörü (PIR) ile Servo Motor Kontrolü

robotistan BLOG

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

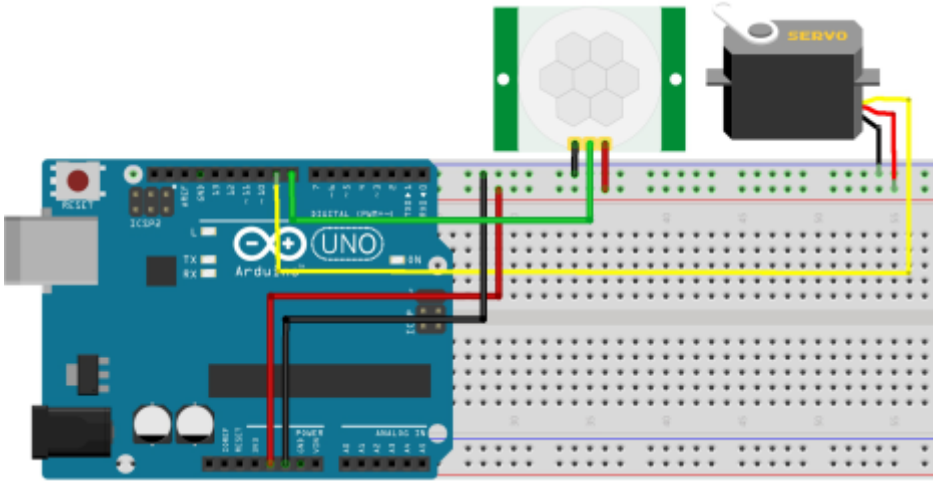
YouTube

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

Gerekli malzemeler:

- Arduino UNO
- Breadboard
- Servo Motor
- Hareket Sensörü
- 5 Adet Erkek-Erkek Jumper Kablo
- 3 Erkek-Dişi Jumper Kablo

Bu örnekte hareket detektörü kullanarak ortamda hareket algılandığında motor hareket ettirebileceğimiz bir sistem kuracağız.



Hareket sensörü, ortamda bulunan kızılötesi ışık dalgalarını algılayarak çalışır. Maddelerin doğası gereği etrafa sahip oldukları ısıdan dolayı kızılötesi dalga saçar. Hareket sensörü maddelerin yaymış olduğu kızılötesi dalgalarındaki değişimi algılar ve sinyal çıktısı verir.

Örnek kodumuzda hareket detektöründen veri okuyarak ortamda bir hareket algılandığında servo motorumuzun hareketini sağlayacağız.

```arduino
#include <Servo.h>

int pirPin = 8;
int servoPin = 9;
int hareket;
Servo motor;


void setup() {
  motor.attach(servoPin);
  pinMode(pirPin, INPUT);

}

void loop() {
  hareket = digitalRead(pirPin);

  if(hareket == HIGH){
    motor.write(150);
    delay(250);
    motor.write(30);
    delay(250);
    motor.write(150);
    delay(250);
    motor.write(30);
    delay(250);
    motor.write(150);
    delay(250);
    motor.write(30);
    delay(250);
    motor.write(90);
  }
  else{
    motor.write(90);

  }
}
```

İlk olarak servo motor kütüphanesi olan "Servo.h"ı kodumuza ekliyoruz. Daha sonra gerekli pin tanımlamalarını yapıyoruz. Ardından sensörden aldığımız veriyi hafızada tutmak için hareket adında bir değişken tanımlıyoruz. Daha sonra "motor" adında bir servo motor tanımlıyoruz. "motor" değişkeni, servo motorumuzu kontrol etmemizi sağlayacak değişkendir.

"setup" kısmında, servo motorumuzun takılı olduğu pini motor değişkenimizle ilişkilendiriyoruz. Daha sonra sensörümüzün takılı olduğu pini giriş olarak ayarlıyoruz.

"loop" kısmında, "digitalRead()" fonksiyonu ile sensörümüzden hareket verisini okuyoruz. Daha sonra "if-else" yapısı ile hareket pinimizin değerine göre servo motorumuza gerekli açı komutunu gönderiyoruz.

16

Bluetooth ile RGB LED Kontrollü

robotistan BLOG

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

YouTube

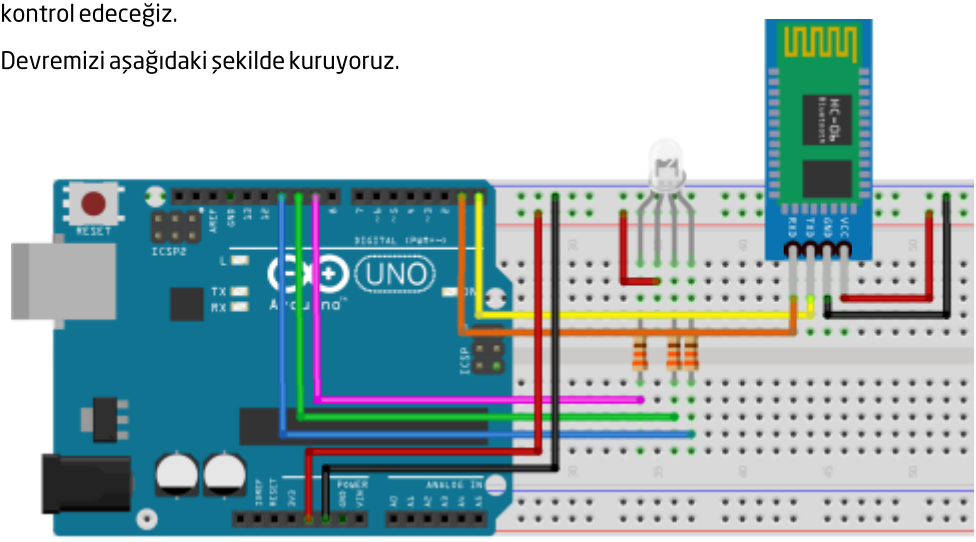
Uygulamanın videosuna aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 9 Adet Erkek-Erkek Jumper Kablo
- Bluetooth modülü
- RGB LED
- 3 Adet 330 Ohm Direnç (Turuncu-Turuncu-Kahverengi)

Bu örnekte, telefon üzerinden gönderdiğimiz veriyi Bluetooth modülü ile alıp RGB LED'in rengini kontrol edeceğiz.

Devremizi aşağıdaki şekilde kuruyoruz.



RGB; Red (kırmızı), Green (yeşil) ve Blue (mavi) renklerin baş harfleri birleştirilerek oluşmuş bir terimdir. RGB led, 3 ana renkteki LED'lerin birleşiminden oluşan bir komponenttir. İçerisindeki farklı renkteki LED'lere farklı gerilimler vererek renk kombinasyonları oluşturulmasına izin verir. İki çeşit RGB LED türü vardır. Bunlar, ortak anot ve ortak katot. Ortak anot RGB LED, içerisindeki 3 adet LED'in pozitif pinlerinin birleştirilmesiyle oluşmuştur. Ortak katot RGB LED ise içerisindeki 3 adet LEDin negatif pinlerinin birleştirilmesiyle oluşmuştur. Bu örneğimizde ortak anot RGB LED kullandık.

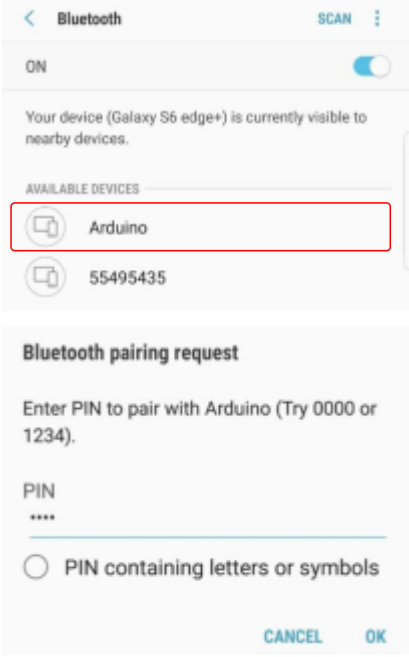
Arduino, Bluetooth modülü ile haberleşmek için UART(Universal Asynchronous Receiver Transmitter) protokolünü kullanır. Bu bir seri haberleşme protokolüdür. UART protokolünü kullanabilmemiz için iki taraftaki Baud Rate(Haberleşme hızları) 'nın aynı olması gerekmektedir. 4800, 9600, 57600, 115200 en çok kullanılan haberleşme hızlarıdır. Bu örnekte 9600 Baud Rate kullanacağız. Bluetooth modülü öntanımlı olarak bu haberleşme hızını kullanmaktadır.

```arduino
int veri;
int kirmiziPin = 9;
int yesilPin = 10;
int maviPin = 11;
void setup() {
  Serial.begin(9600);
  pinMode(kirmiziPin,OUTPUT);
  pinMode(yesilPin,OUTPUT);
  pinMode(maviPin,OUTPUT);
}
void loop() {
  if(Serial.available()>0){
    veri = Serial.read();
  }
  if(veri == 'k'){
    digitalWrite(kirmiziPin,LOW);
    digitalWrite(yesilPin,HIGH);
    digitalWrite(maviPin,HIGH);
  }
  else if(veri == 'y'){
    digitalWrite(kirmiziPin,HIGH);
    digitalWrite(yesilPin,LOW);
    digitalWrite(maviPin,HIGH);
  }
  else if(veri == 'm'){
    digitalWrite(kirmiziPin,HIGH);
    digitalWrite(yesilPin,HIGH);
    digitalWrite(maviPin,LOW);
  }
  else{
    digitalWrite(kirmiziPin,HIGH);
    digitalWrite(yesilPin,HIGH);
    digitalWrite(maviPin,HIGH);
  }
}
```

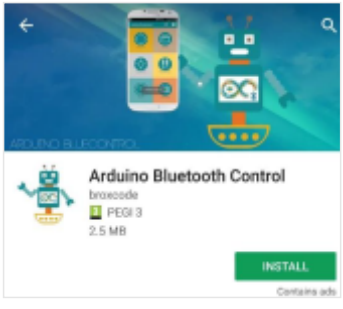
İlk olarak kullanacağımız değişkenleri tanımlıyoruz.

"setup" kısmında seri haberleşmeyi başlatıyoruz. Daha sonra RGB LED için kullanacağımız çıkış pinlerimizi tanımlıyoruz.

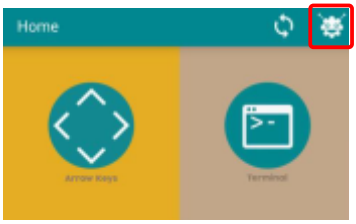
"loop" kısmında, seri haberleşme üzerinden aldığımız veriyi okuyoruz ve veri isimli değişkene eşitliyoruz. "İf", "else" ve "if-else" yapıları ile veri değişkeninin değerine göre RGB LED'imizi yakıyoruz. Kodumuzu Arduino'ya yükledikten sonra telefonumuza yükleyeceğimiz bir Arduino Bluetooth terminal uygulaması ile RGB LED kontrolünü yapabiliriz.



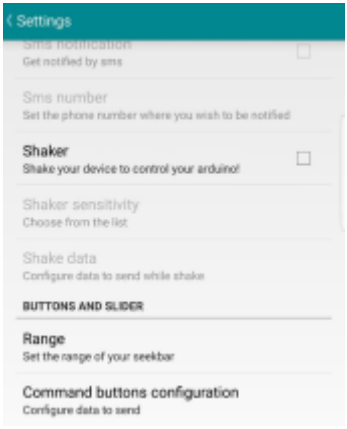
Telefonumuzdan Bluetooth'a bağlanmak için öncelikle Bluetooth'u aktif etmemiz gerekiyor. Ardından Bluetooth ayarlarında giriyoruz. Sol taraftaki resme benzer bir ekranın karşımıza çıkması gerekmektedir. Bu ekran farklı telefonlarda farklı şekilde gözükebilir. Ekranda Bluetooth ayarını yaparken tanımladığımız isimde bir aygıt gözükmektedir. Aygıtın üzerine tıklıyoruz ve eşleştir diyoruz. Arduino kodunda ayarlar kısmında tanımladığımız şifremizi giriyoruz ve bağlantı kuruyoruz.



Arduino Bluetooth Control isimli uygulamayı Google Play üzerinden telefonumuza indiriyoruz. Bu uygulama ile Bluetooth modülümüze veri göndereceğiz.

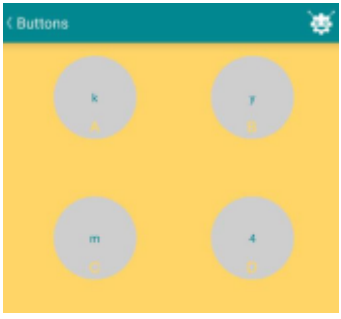


Ardından telefonumuza yüklediğimiz Arduino Bluetooth Control uygulamamıza giriyoruz ve sağ üstte bulunan ayarlar düğmesine tıklıyoruz.



Ayarlardan "Command buttons configuration" sekmesine tıklıyoruz ve butonlara tıkladığımızda Bluetooth ile göndereceği verileri tanımlıyoruz. Button A'ya tıklıyoruz ve çıkan ekrana "k" harfini yazıyoruz. Button B için "y" harfini, Button C için "m" harfini yazıyoruz.

Daha sonra ayarlar bölümünden çıkarak ana ekrana geliyoruz. Ana ekranda bulunan Buttons&Slider sekmesine tıklayarak RGB ledi kontrol edeceğimiz butonları açıyoruz.



Burada bulunan A, B ve C butonlarına tıklayarak Bluetooth üzerinden Arduino'ya gerekli komutları gönderiyoruz.

RGB ledin, "k" harfine bastığımızda kırmızı; "y" harfine bastığımızda yeşil ve "m" harfine bastığımızda mavi renkte yanmasını sağlıyoruz.

17

Arduino ile Dijital Saat Yapımı

robotistan BLOG

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

YouTube

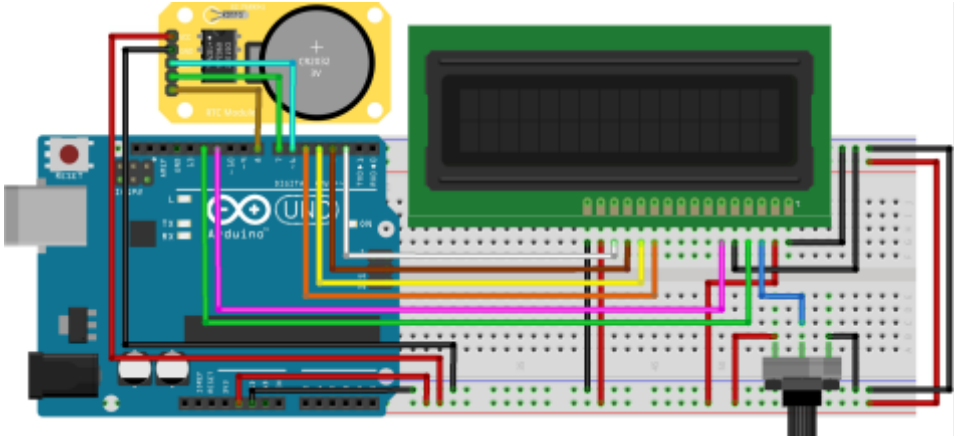
Uygulamanın videosuna aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- DS1302 RTC Modülü
- 16x2 LCD Ekran
- 10K Potansiyometre
- 17 Adet Erkek-Erkek Jumper Kablo
- 5 Adet Erkek-Dişi Jumper Kablo

Arduino gibi cihazlarda saat örneği yapmaya çalıştığımızda karşılaşacağımız en büyük sorun, herhangi bir güç kesintisinde Arduino'nun süreyi saymayı bırakması olacaktır. Bu da zamanda sapmalara neden olur ve doğru zamanı öğrenmemizi engeller. RTC modülleri zamanı sürekli senkron tutması için üretilmiştir. Çok az güçle bile çalışabilen bu modül üzerinde bulunan pil ile uzun yıllar boyunca zamanda sapmaya yol açmadan üzerinde bulunan kristal sayesinde sayım yapar. Bu kristal saniyede 32000 sinyal üretmektedir. RTC bu sinyalleri okur ve her 32000 adımda bir saniye ileri saymaktadır.

Saati kullanabilmemiz için öncelikle zamanı şu anki zamana göre ayarlamamız lazım. Onun için ilk önce ayar kodumuzu yüklememiz gerekmektedir.



```
#include <virtuabotixRTC.h>
int CLK_PIN = 6;
int DAT_PIN = 7;
int RST_PIN = 8;
virtuabotixRTC myRTC(CLK_PIN, DAT_PIN, RST_PIN);
void setup() {
 Serial.begin(9600);
 myRTC.setDS1302Time(10, 10, 14, 4, 13, 9, 2018);
}
void loop() {
 myRTC.updateTime();
 Serial.print("Tarih / Saat: ");
 Serial.print(myRTC.dayofmonth);
 Serial.print("/");
 Serial.print(myRTC.month);
 Serial.print("/");
 Serial.print(myRTC.year);
 Serial.print(" ");
 Serial.print(myRTC.hours);
 Serial.print(":");
 Serial.print(myRTC.minutes);
 Serial.print(":");
 Serial.println(myRTC.seconds);
 delay(1000);
```

Gerekli değişkenleri tanımlıyoruz. Kullanacağımız Arduino pinlerine, hangi RTC pinlerini atayacağımızı belirliyoruz. "setup" kısmında seri haberleşmeyi 9600 haberleşme hızında başlatıyoruz. Ardından zamanı; saniye, dakika, saat, haftanın günü, ayın günü, ay ve yıl olacak şekilde ayarlıyoruz."loop" kısmında RTC'den zamanı okuyoruz ve seri port ekranına yazdırıyoruz. Seri port ekranında okuduğumuz değerleri kontrol ediyoruz. Bir hata olmadığından emin olduktan sonra LCD üzerinde gösterme kodumuza başlıyoruz.

```
#include <LiquidCrystal.h>
#include <virtuabotixRTC.h>
int CLK_PIN = 6;
int DAT_PIN = 7;
int RST_PIN = 8;
virtuabotixRTC myRTC(CLK_PIN, DAT_PIN, RST_PIN);
int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
void setup() {
  lcd.begin(16, 2);
}
void loop() {
  myRTC.updateTime();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(myRTC.dayofmonth);
  lcd.print("/");
  lcd.print(myRTC.month);
  lcd.print("/");
  lcd.print(myRTC.year);
  lcd.setCursor(0, 1);
  lcd.print(myRTC.hours);
  lcd.print(":");
  lcd.print(myRTC.minutes);
  lcd.print(":");
  lcd.print(myRTC.seconds);
  delay(1000);
```

İlk önce gerekli kütüphanelerimizi kodumuza dahil ediyoruz. Ardından kullanacağımız pinleri değişkenlere atıyoruz. Atadığımız değişkenleri kütüphanemize tanıtıyoruz.

setup() kısmında LCD'nin en boy oranını ayarlıyoruz. Bu örnekte 16x2 LCD kullanacağımızdan en-boy oranını 16'ya 2 olacak şekilde ayarlıyoruz.

loop() kısmında ilk önce RTC den zaman verisini okuyoruz. Ardında LCD'nin ekranındaki karakterleri temizliyoruz. Bunu yapmaz isek kullandığımız karakterler üst üste binebilir ve yanlış veri elde etmemize yol açar. İlk satır ilk sütundan itibaren; gün, ay ve yıl olacak şekilde tarihi bastırıyoruz. İkinci satır ilk sütundan itibaren ise saat, dakika, saniye olacak şekilde zamanı bastırıyoruz ve 1 saniyelik bir bekleme koyuyoruz.

18

Arduino ile Toprak Nem Sensörü Kullanımı

robotistan BLOG

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

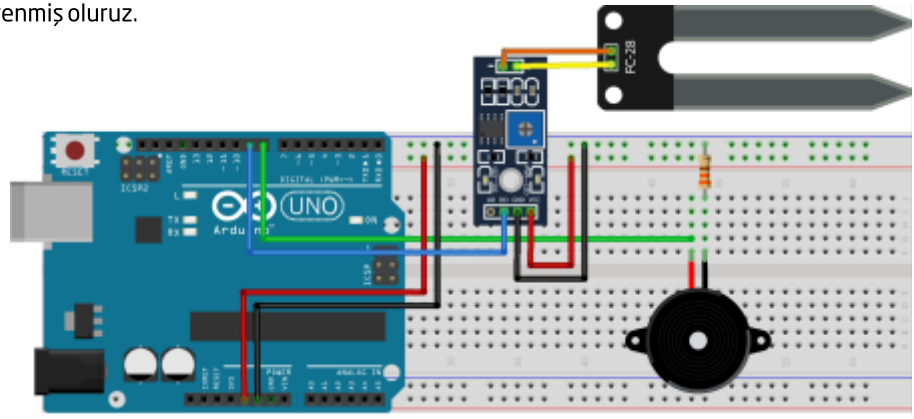
YouTube

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

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- Toprak Nem Sensörü
- 3 Adet Erkek-Erkek Jumper Kablo
- 3 Adet Erkek-Dişi Jumper Kablo
- Buzzer
- 1 Adet 330 Ohm Direnç (Turuncu - Turuncu - Kahverengi)

Toprak nem sensörü iki elektrottan oluşur. Bu elektrotlar arasındaki iletkenliği ölçerek bize toprağın ne kadar nemli olduğuna dair bilgi verir. Toprak nemli ise elektrotlar arasındaki iletkenlik artar. İletkenlik arttığında direnç azalacağından sensörün içindeki gerilim bölücüden daha az voltaj gelmeye başlar. Toprak kuru ise elektrotlar arasındaki direnç artacağından daha yüksek bir voltaj elde etmiş oluruz ve bu veriyi analog olarak işleyerek topraktaki nem miktarını öğrenmiş oluruz.



Bu örnekte, topraktaki nem miktarı belli bir eşik değerini geçtiğinde bize bir sinyal gönderecek ve devre üzerindeki buzzer ses çıkarmaya başlayacak. Bu devrede eşik değerini sensörün üzerinde bulunan potansiyometre ile ayarladık. Yani eşik değer ayarladığımız değerin üzerine çıkarsa D0 pininden bize 0V'luk bir çıkış verecektir. Aynı şekide A0 pininden 0V-5V aralığında analog olarak aldığımız veriyi kod içinde belli bir eşik değeri geçtiğinde çalışacak şekilde hazırlayabiliriz.

Kodumuzda toprak nem sensöründen veri okuyarak toprak neminin azalması durumunda Arduino'nun buzzer ile alarm vermesini sağlayacağız.

```
Arduino
int sensorPin = 9;
int buzzerPin = 8;
int veri;

void setup() {
  pinMode(sensorPin, INPUT);
  pinMode(buzzerPin, OUTPUT);
}
void loop() {
  veri = digitalRead(sensorPin);
  if(veri == true){
    tone(buzzerPin, 440);
    delay(100);
    noTone(buzzerPin);
    delay(100);
  }
  else{
    noTone(buzzerPin);
  }
}
```

İlk olarak gerekli değişkenlerimiz tanımlıyoruz. "setup" kısmında giriş-çıkış ayarlamalarını yapıyoruz. "loop" kısmında, "digitalRead()" fonksiyonu ile sensörden alınan değer veri değişkenine eşitlenir. Daha sonra veri değişkeninin durumu "if-else" yapısı ile kontrol edilir.

Toprak nem sensörümüz su algıladığında dijital çıkış pininden 0V sinyal vermektedir. veri değişkeninin LOW değerinde olması, toprak nem sensörü tarafından suyun algılandığını göstermektedir. Bu durumda veri değişkeni HIGH değerini alır ve else yapısının içindeki komutlar çalıştırılır.

Toprak nem sensörü tarafından su algılanmadığında, sensörün dijital çıkış pininden 5V çıkış verilmektedir. Bu durumda buzzer ötmeye başlar.

19

Arduino ile Yağmur Sensörü Kullanımı

robotistan BLOG

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

YouTube

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

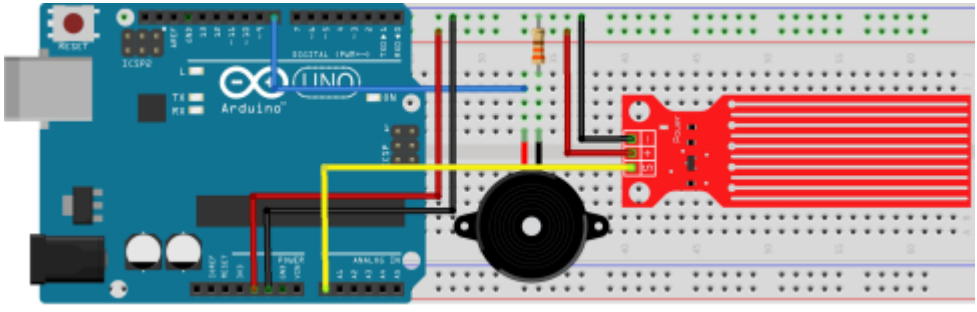
Gerekli malzemeler:

- Arduino Uno
- Breadboard
- Yağmur Sensörü
- 3 Adet Erkek-Erkek Jumper Kablo
- 3 Adet Erkek-Dişi Jumper Kablo
- Buzzer
- 1 adet 330 Ohm Direnç (Turuncu - Turuncu - Kahverengi)

Hava yağmurlu ise yağmur sensörünün üzerindeki su miktarı belli bir eşik değerini geçtiğinde bize bir sinyal gönderecek ve devre üzerindeki buzzer ses çıkarmaya başlayacak. Bu devreyi analog pin üzerinden aldığımız sinyale göre alarm verecek şekilde hazırladık.

Yağmur sensörü iki elektrottan oluşur. Bu elektrotlar arasındaki iletkenliği ölçerek bize bilgi verir. Sensör üzerindeki su damları elektrotlar arasındaki iletkenliği arttırır. Sensör, bu veriyi analog ve dijital çıktı olarak verir.

Kodumuzda yağmur sensöründen veri okuyarak yağmur yağdığında buzzer ile alarm vermesini sağlayacağız.



```
Arduino
int sensorPin = A0;
int esikDegeri = 100;
int buzzerPin = 8;
int veri;

void setup() {
  pinMode(buzzerPin, OUTPUT);
}
void loop() {
  veri = analogRead(sensorPin);
  if(veri > esikDegeri){
    tone(buzzerPin, 440);
    delay(100);
    noTone(buzzerPin);
    delay(100);
  }
  else{
    noTone(buzzerPin);
  }
}
```

İlk olarak gerekli değişkenlerimiz tanımlıyoruz. "setup" kısmında buzzer bağlayacağımız pini çıkış olarak tanımlıyoruz. "loop" kısmında, "analogRead()" fonksiyonu ile sensörden alınan değer veri değişkenine eşitlenir. Daha sonra veri değişkeninin durumu "if-else" yapısı ile kontrol edilir.

Yağmur sensöründen okunan veri eşik değerinden büyük olması, havanın yağmurlu olduğunu gösterir. Sensörün üzerine su varsa "if" içerisindeki komutlar çalıştırılır ve alarm çalar.

Yağmur sensöründen okunan verinin eşik değerinin altında olması, havanın kuru olduğu gösterir. Hava kuru ise else içerisindeki komutlar çalıştırılır ve alarm çalmaz.

29

Arduino ile Gaz Sensörü Kullanımı

robotistan BLOG

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

YouTube

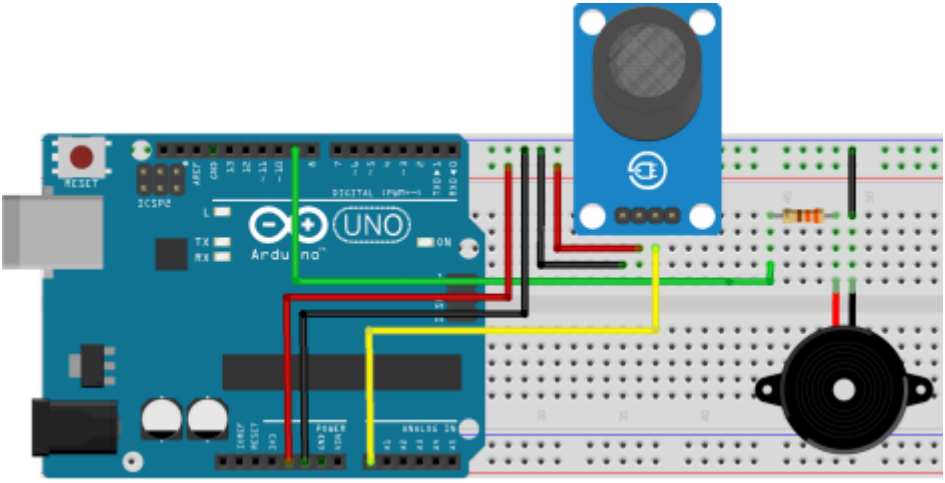
Uygulamanın videosuna aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- Gaz sensörü
- Buzzer
- 7 Adet Erkek-Erkek Jumper Kablo
- 1 Adet 330 Ohm Direnç (Turuncu - Turuncu - Kahverengi)

Bu devrede gaz sensörü ile çalışan alarm yaptık. Ortamdaki doğalgaz miktarı belli bir eşik değerini geçtiğinde bize bir sinyal gönderecek ve devre üzerindeki buzzer ses çıkarmaya başlayacak. Aynı şekilde sensörün üzerinde bulunan potansiyometre ile eşik değerini belirleyip DO pininden dijital veri alarak da yapılabilmektedir.

Gaz sensörünün içinde bir dizi direnç bulunmaktadır. Ortamdaki gaz, sensörün içindeki dirençlerle etkileşime girerek direnç değerlerini değiştirir. Sensör, içinde bulunan bir gerilim bölücü sayesinde gaz yoğunluğuna göre farklı gerilim değerleri verir ve bu gerilim değerlerinin analog bir şekilde okunmasıyla ortamdaki gaz miktarı ölçülür.



Gaz sensörü, analog ve dijital çıkış verebilmektedir. Bu uygulamamızda analog çıkışı kullanarak örnek devremizi kuracağız.

Örnek kodumuzda gaz sensörünün analog pininden değer okuyarak eşik değerimiz ile karşılaştırma yapacağız. Okuduğumuz değer, kod içinde belirlediğimiz eşik değerinden büyük olduğunda buzzer yardımıyla alarm sesi çıkaracağız.

Arduino

```
int esikDegeri = 400;
int buzzerPin = 9;
int deger;

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

void loop() {
  deger = analogRead(A0);
  if(deger > esikDegeri){
    tone(buzzerPin, 440);
    delay(100);
    noTone(buzzerPin);
    delay(100);
  }
  else{
    noTone(buzzerPin);
  }
}
```

İlk olarak gerekli olan değişkenlerimizi tanımlıyoruz. Sensörün eşik değeri, "esikDegeri" adlı değişkenin değeri değiştirilerek ayarlanabilir. "setup" kısmında, buzzer bağladığımız pini çıkış olarak ayarlıyoruz. "loop" kısmında, "analogRead()" fonksiyonu ile sensörden analog değer okuyarak "deger" isimli değişkenimize eşitliyoruz. Daha sonra "if-else" yapısını kullanarak "deger" isimli değişkenimizi "esikDegeri" isimli değişkenimiz ile karşılaştırıyoruz. Sensörden okuduğumuz analog veri, eşik değişkeninden büyük olduğu zaman buzzer pinini aç kapa yaparak buzzer ile alarm sesi oluşturuyoruz. Sensörden okuduğumuz veri, eşik değerinin altında kaldığında buzzer pinine LOW değerini vererek buzzerdan ses çıkmamasını sağlıyoruz.

21

Arduino ile RFID Sensörü Kullanımı

robotistan BLOG

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

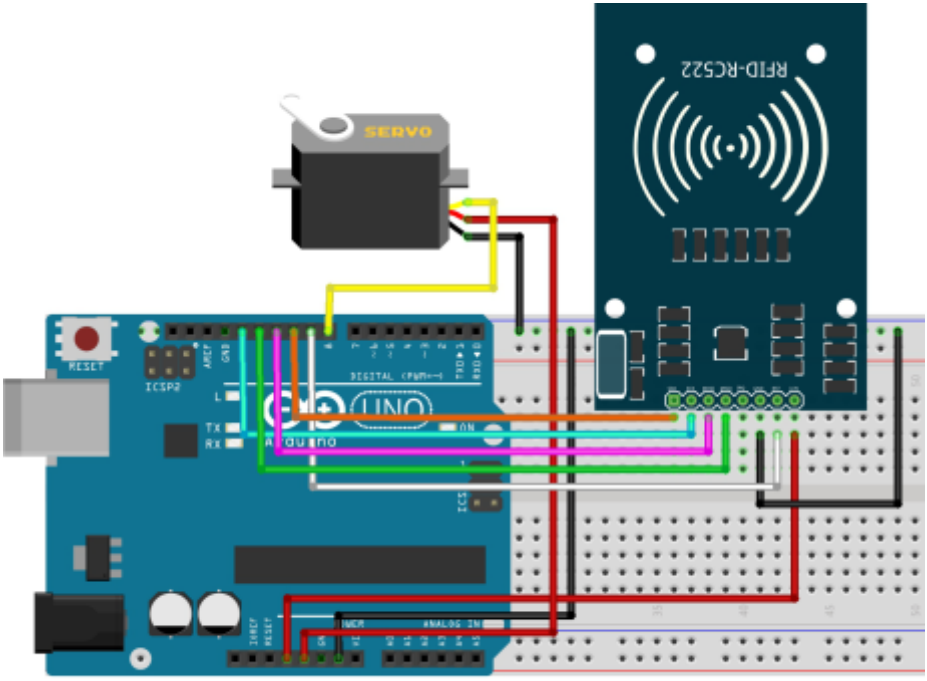
YouTube

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

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- Servo Motor
- RC522 RFID modül
- RFID Kart
- 12 Adet Erkek-Erkek Jumper Kablo

Uygulamamızda RFID kart okuyucuya yaklaştırılan kartın ID numarası okunup SPI(Serial Perhiperal Interface) protokolü ile Arduino'ya gönderir. Eğer okunan ID sistemde kayıtlı ise servoyu harekete geçirir ve kapı açılır. Kayıtlı değil ise kapı kapalı kalmaya devam eder.



SPI Master-Slave mantığına dayanan bir seri haberleşme protokolüdür. Yani birbirleriyle senkronize olarak çalışması için aralarında bir clock(saat) sinyaline ihtiyaç duyarlar. Bu şekilde UART gibi asenkron protokollere göre daha güvenilir bir haberleşme sağlamış oluruz. SPI için en az 4 adet pine ihtiyaç duyarız. SCK clock sinyalini için kullanılır. MOSI(Master Out Slave In), master olan cihazdan slave olan cihaza veri göndermek için kullanılır. MISO (Master In Slave Out), slave olan cihazdan master olan cihaza veri göndermek için kullanılır. SS(Slave Select) pini, master cihazın hangi cihaz ile haberleşeceğini belirler. Bu uygulamada kullandığımız RC522 RFID modülü de SPI protokolü ile haberleşmektedir.

Şimdi kodumuzu yazalım.

Arduino

```
#include <SPI.h>
#include <MFRC522.h>
#include <Servo.h>

int RST_PIN = 9;
int SS_PIN = 10;
int servoPin = 8;

Servo motor;
MFRC522 rfid(SS_PIN, RST_PIN);
byte ID[4] = {97, 76, 67, 9};

void setup() {
  motor.attach(servoPin);
  Serial.begin(9600);
  SPI.begin();
  rfid.PCD_Init();
}

void loop() {

  if ( ! rfid.PICC_IsNewCardPresent())
    return;

  if ( ! rfid.PICC_ReadCardSerial())
    return;
```

```
Arduino
27
28  if (rfid.uid.uidByte[0] == ID[0] &&
29    rfid.uid.uidByte[1] == ID[1] &&
30    rfid.uid.uidByte[2] == ID[2] &&
31    rfid.uid.uidByte[3] == ID[3] ) {
32        Serial.println("Kapi acildi");
33        ekranaYazdir();
34        motor.write(180);
35        delay(3000);
36        motor.write(0);
37        delay(1000);
38    }
39    else{
40      Serial.println("Yetkisiz Kart");
41      ekranaYazdir();
42    }
43  rfid.PICC_HaltA();
44 }
45 void ekranaYazdir(){
46   Serial.print("ID Numarasi: ");
47   for(int sayac = 0; sayac < 4; sayac++){
48     Serial.print(rfid.uid.uidByte[sayac]);
49     Serial.print(" ");
50   }
51   Serial.println("");
```

İlk olarak kodumuzda kullanacağımız kütüphanelerimizi tanımlıyoruz. MFRC522 kütüphanesi RC522 modülünü kullanmak için gerekli olan fonksiyonları içermektedir.

RST_PIN, SS_PIN değişkenleri RC522 modülünün kullandığı pinlerdir. servoPin değişkeni, servo motorumuzu bağladığımız pindir.

Daha sonra motor isimli "Servo" değişkenimizi tanımlıyoruz. RFID modülümüz için gerekli ayarları yaptıktan sonra "ID" isimli bir dizi oluşturuyoruz. Bu dizinin elemanları yetki vermek istediğimiz RFID kartının ID numarasıdır. Tüm kodu yazdıktan sonra seri port üzerinden kendi kartımızı okutarak "ID" değişkenimizi değiştirebiliriz. Bu sayede istediğimiz karta giriş yetkisi verebiliriz.

"setup" kısmında, servo motorun bağlı olduğu pin motor değişkeni ile ilişkilendirilir. Daha sonra seri haberleşme ve SPI başlatılır. "rfid.PCD_Init()" fonksiyonu ile RC522 modülü başlatılır.

"loop" kısmında, RC522 modülünden yeni bir kart okunana kadar beklenir. Yeni kart okunduğunda "if-else" yapısı ile kayıtlı olan yetkili kart numarası ile okunan kart numarası karşılaştırılır. Okunan kart yetkili ise servo motor hareketi sağlanır ve "ekranaYazdir()" fonksiyonu ile seri porta kart numarası yazdırılır. Bu fonksiyon bir sonraki bölümde anlatılacaktır. Okunan kart numarası yetkili değil ise seri porta yetkisiz kart uyarısı yazdırılır.

Kodlarda çok fazla tekrarlanan kısımlar fonksiyonlara dönüştürülür. Bu sayede koddaki karmaşıklık en aza indirgenerek işlemcinin hafızasının gereksiz yere kullanılmaması sağlanır.

"ekranaYazdir()"" fonksiyonu bizim oluşturduğumuz bir fonksiyondur. Bu fonksiyonun kullanıldığı yerlerde fonksiyon içerisindeki komutlar çalıştırılmış olur. Bu fonksiyon, okunan RFID kart bilgisinin seri porta yazılmasını sağlar.

27

ESP8266 ile Sıcaklık ve Nem Ölçümü

robotistan BLOG

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

YouTube

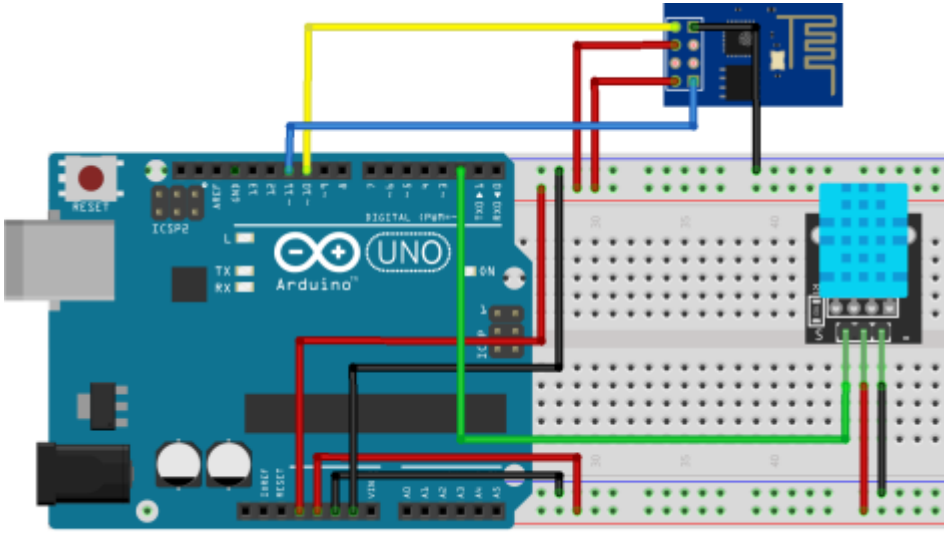
Uygulamanın videosuna aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- ESP8266 WiFi Modülü
- DHT11 Nem ve Sıcaksık Sensörü
- 8 Adet Erkek-Dişi Jumper Kablo
- 3 Adet Erkek-Erkek Jumper Kablo

Uygulamamızda DHT11 üzerinden aldığımız sıcaklık ve nem verilerini ESP8266 WiFi modülü kullanarak Thingspeak platformuna göndereceğiz. ESP8266 ile haberleşirken UART protokolünü kullanıyoruz. Bu örnekte 115200 Baud rate(Haberleşme hızını) kullanacağız.

Thingspeak, açık kaynaklı bir IoT(Internet of Things) uygulamasıdır. Kullanıcılar HTTP üzerinden siteye veri gönderip kendi oluşturdukları uygulamaları site üzerinde bulunan grafik arayüzleri sayesinde daha görsel ve anlaşılması kolay bir hale getirir. Bu örnekte, DHT11 sensöründen aldığımız veriler ile Thingspeak'te zaman-nem ve zaman-sıcaklık grafikleri oluşturacağız.



ESP8266, üzerinde bulunan kablosuz haberleşme devresi sayesinde ethernet protokolü ile kablosuz internete bağlanmamızı sağlar. Arduino gibi cihazların içinde ethernet protokolünü kullanabilmesini sağlayan donanım yoktur. Bu yüzden ethernet protokolünü kullanabilmek için modüller kullanırız. Bu modüller ethernet protokolünü bizim daha kolay iletişim kurabileceğimiz haberleşme protokollerine dönüştürürler. ESP8266'da bu modüllere bir örnektir. Ethernet protokolünü basitleştirip UART protokolüne dönüştüren bir tercüman görevi görmektedir.

DHT11, nem ve sıcaklık verilerini okumamıza yarayan bir sensördür. Daha önce kullandığımız sensörlerdeki gibi bu sensör de iletkenlikleri ölçerek havanın nemini ve sıcaklığını ölçmemize yarar. Sensörün içinde bir NTC(Negative Temperature Coefficient) bulunmaktadır. NTC,bir çeşit dirençtir ve ortamın ısısı arttıkça iletkenliği artar ve direnç değeri düşer. Sensörün içinde 2 adet elektrot ve elektrotların arasında ise havadaki nemi tutan bir yüzey bulunmaktadır. Bu yüzey havadaki nem miktarı arttıkça elektrotlar arasındaki iletkenliği değiştirir. Bu şekilde havadaki nem düzeyini ölçmüş oluruz.

```
#include <SoftwareSerial.h>
#include <dht11.h>
String agAdi = "Robotistan";
String agSifresi = "fortinet";
int rxPin = 10;
int txPin = 11;
int dht11Pin = 2;
String ip = "184.106.153.149";
float sicaklik, nem;
dht11 DHT11;

SoftwareSerial esp(rxPin, txPin);
void setup() {
  Serial.begin(9600);
  esp.begin(115200);
  esp.println("AT");
  while(!esp.find("OK")){
    esp.println("AT");
    Serial.println("ESP8266 Bulunamadı.");
  }
```

```
Arduino
21  esp.println("AT+CWMODE=1");
22  while(!esp.find("OK"));
23  Serial.println("Aga Baglaniliyor...");
24  esp.println("AT+CWJAP=\""+agAdi+"\",\""+agSifresi+"\"");
25  while(!esp.find("OK"));
26  Serial.println("Aga Baglandi.");
27  delay(1000);
28 }
29 void loop() {
30  esp.println("AT+CIPSTART=\"TCP\",\""+ip+"\",80");
31  if(esp.find("Error")){
32    Serial.println("AT+CIPSTART Error");
33  }
34  DHT11.read(dht11Pin);
35  sicaklik = (float)DHT11.temperature;
36  nem = (float)DHT11.humidity;
37  String veri = "GET /update?key=11NOMDXGH6DIVEXB&field1=";
38  veri += String(sicaklik);
39  veri += "&field2=";
40  veri += String(nem);
41  veri += "\r\n\r\n";
42  esp.print("AT+CIPSEND=");
43  esp.println(veri.length()+2);
44  delay(2000);
45  if(esp.find(">")){
46    esp.print(veri);
47    esp.print("\r\n\r\n");
48    Serial.println(veri);
49    Serial.println("Veri gonderildi.");
50    delay(1000);
51  }
52  Serial.println("Baglanti Kapatildi.");
53  esp.println("AT+CIPCLOSE");
54  delay(36000);
55 }
```

ESP8266 modülü, Arduino ile seri haberleşmeyle iletişim kurar. Bu iletişim Arduino Uno'nun 0. ve 1. pinleriyle donanımsal olarak yapılabilmektedir ancak SoftwareSerial kütüphanesi ile diğer pinlerden yazılımsal olarak yapılabilmektedir. Bu örneğimizde SoftwareSerial kütüphanesini kullanarak ESP8266 modülü ile iletişim kuracağız. İlk olarak kodumuza SoftwareSerial kütüphanesini ekliyoruz. Ardından kullanacağımız sıcaklık ve nem sensörünün "dht11.h" kütüphanesini ekliyoruz.

Bağlanacağımız ağın ismini ve şifresini değişken olarak hafızada tutuyoruz. Buradaki "agAdi" ve "agSifresi" değişkenlerini kendi ağ adımız ve şifremize göre değiştiriyoruz.

ESP8266 modülünü bağlayacağımız RX ve TX pinleri için değişken oluşturuyoruz. Ardından DHT11 sensörünü bağlayacağımız pin için değişken oluşturuyoruz.

Uygulamamızda kullanacağımız Thingspeak platformunun IP numarasını tutacağımız, "ip" isimli değişkeni oluşturuyoruz. DHT11 ile okuyacağımız sıcaklık ve nem verilerini hafızada tutmak için "sicaklik" ve "nem değişkenlerini" oluşturuyoruz. Bu veriler virgüllü sayılar olacağı için float tipinde oluşturuyoruz. "float" değişkeni küsürlü sayıları saklayabileceğimiz bir değişkendir. ESP8266 modülünü bağlayacağımız pinlerin ayarını yapıyoruz. "setup" kısmında, bilgisayar ile haberleşmeyi başlatıyoruz. Bilgisayar ile haberleşme için 9600 hızını kullanıyoruz. Daha sonra "esp.begin" fonksiyonu ile ESP8266 modülü ile iletişimi başlatıyoruz. Seri haberleşmenin hızı, ESP8266 modülümüzün iletişim hızı ile aynı olmalıdır. Bu yüzden 115200 hızını kullanıyoruz. Eğer örnek kod çalışmazsa ESP8266 modülünün firmware güncelleştirmesini yaparak bu sorunu giderebilirsiniz. Firmware güncelleştirme ile ilgili detaylı bilgileri
https://maker.robotistan.com/esp8266-ile-iot-dersleri-1-esp8266-modulunu-guncelleme/
adresinde bulabilirsiniz.

Seri haberleşme ile AT komutu göndererek ESP8266 modülünün hazır olup olmadığını öğreniyoruz. ESP8266 modülü hazır olana kadar bekliyoruz. "AT+CWMODE=1" komutunu göndererek modülümüzü "client" olarak ayarlıyoruz. Ardından "AT+CWJAP" komutu ile ağımıza bağlanıyoruz.

"

"loop" kısmında "AT+CIPSTART" komutu ile Thingspeak ile bağlantı kuruyoruz. Bağlantı kurulumunda hata alınıp alınmadığını kontrol ediyoruz. Daha sonra DHT11 sensörü ile sıcaklık ve nem değerlerini okuyoruz. Okuduğumuz değerleri "sicaklik" ve "nem" isimli değişkenlere yüklüyoruz.

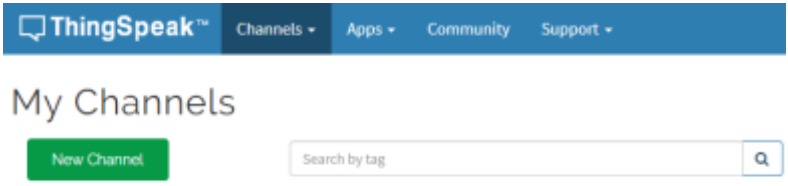
Thingspeak'e göndereceğimiz veri isimli komutu oluşturuyoruz. Buradaki "key" kısmına kendi keyimizi yazacağız. Kendi keyimizi nasıl öğreneceğimizi daha sonra anlatacağız.

Göndereceğimiz verileri oluşturuyoruz ve "AT+CIPSEND" komutu ile veri uzunluğunu ESP8266 modülüne gönderiyoruz.

ESP8266 modülü, hazır olduğunda ">" simgesini göndermektedir. Bu simgenin gelmesini bekliyoruz. Simge geldiğinde bağlantı verisini modülümüze gönderiyoruz. Veri gönderimi tamamlandığında "AT+CIPCLOSE" komutu ile bağlantıyı sonlandırıyoruz.

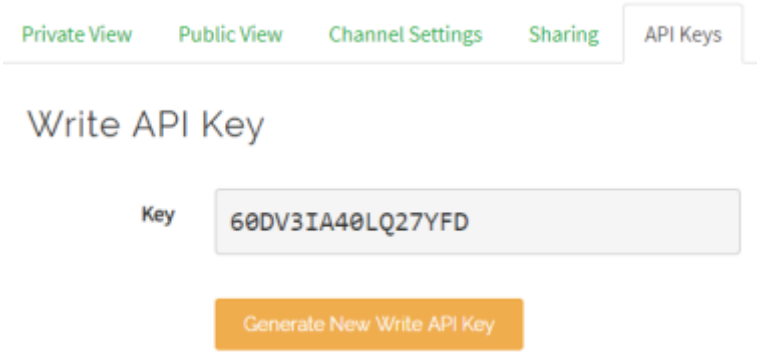
Örnek uygulamamızda, verilerin internet üzerinde görüntülenmesi için Thingspeak platformunu kullanıyoruz.

Thingspeak platfomuna https://www.thingspeak.com adresinden üye oluyoruz.



Üye olduktan sonra "Channels" sekmesinden "My Channels" bölümüne giriyoruz. Ardından "New Channel" butonuna tıklıyoruz.

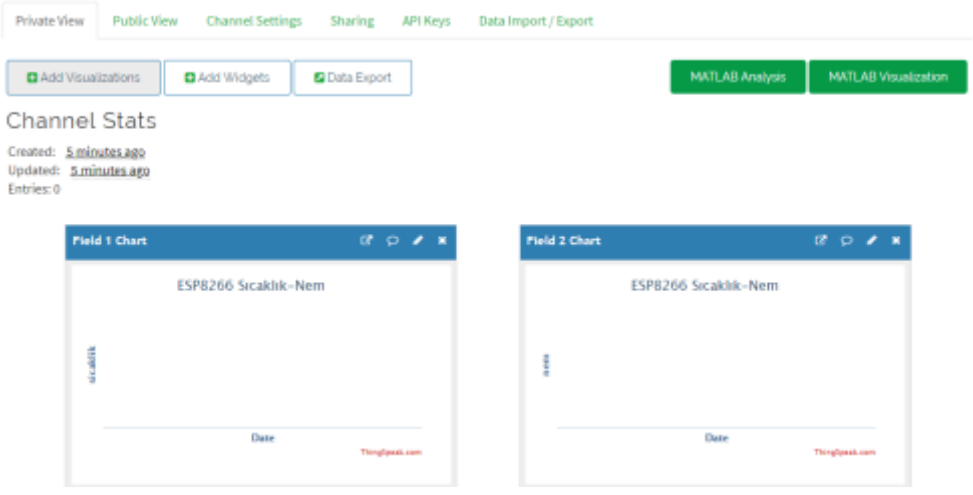
Gelen sayfada gerekli ayarları yukarıdaki şekilde yapıyoruz. Daha sonra sayfanın en altında bulunan "Save Channel" butonuna tıklayarak yeni kanalımızı oluşturuyoruz.



Arduino'nun Thingspeak ile haberleşmesi için "api key"e ihtiyaç vardır. Arduino, "api key" ile Thingspeak hesabınıza bağlanarak verileri kanalınıza kaydeder. "Api key"e ulaşmak için "Api Keys" sekmesine tıklıyoruz. Gelen sayfadaki "Write API Key" bölmesindeki "Key"i kopyalayarak örnek Arduino kodumuzdaki gerekli yere yapıştırıyoruz.

Kodumuzu Arduino'ya yükledikten sonra Arduino IDE üzerinden seri portu açıyoruz.

Seri port hızını 9600 olarak ayarladıktan sonra Arduino'dan gelen sonuçları görüntülüyoruz. Seri portta veri gönderildi yazısını gördükten sonra thingspeak üzerinden verilerimizi görüntülüyoruz.



"Private View" sekmesine tıklayarak Arduino tarafından gönderilen verileri grafikler üzerinde görüntüleyebiliyoruz.

23

ESP8266 ile Step Motor Kontrolü

robotistan BLOG

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

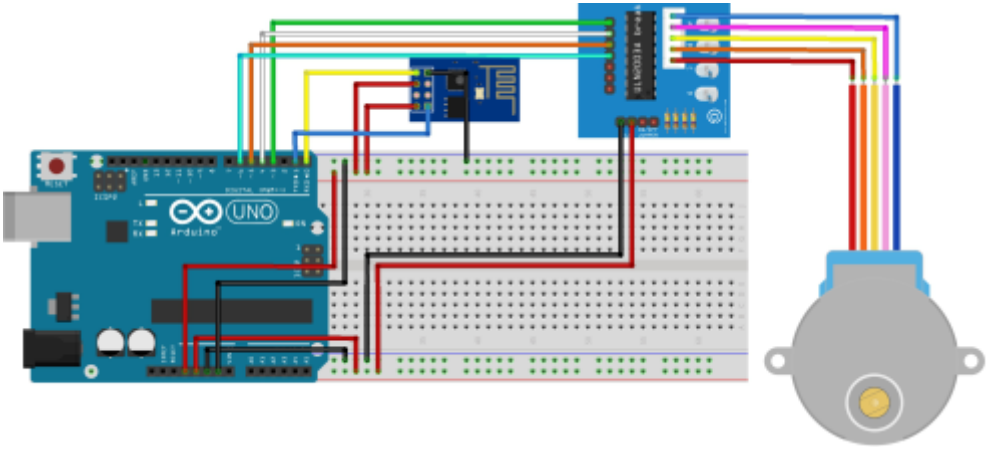
YouTube

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

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- ESP8266 WiFi Modülü
- Step Motor
- ULN2003A Motor Sürücü
- 11 Adet Erkek-Dişi Jumper Kablo
- 2 Adet Erkek-Erkek Jumper Kablo

Uygulamamızda ESP8266 kullanarak açtığımız WEB Server üzerinden step motorun kontrolünü sağlayacağız. Step motorlar, elektrik enerjisini dönme hareketi ile fiziksel enerjiye çeviren elektromekanik aygıtlardır. İsimlerinden de anlaşıldığı üzere adım adım hareket eden motorlardır. İçinde bulunan bobinlerin sırayla açılıp kapanması ile hareketlerini sürdürürler. Bu örnekte kullandığımız step motor 4 fazlı olup içinde 4 adet bobin bulunmaktadır. Step motorlar, çok yüksek hızlı anahtarlayabilen motor sürücülerle ve motor kontrol kartları ile kontrol edilirler. Kullandığımız step motor sürücüsü ise Arduino'dan aldığı sinyallere göre motoru sürmektedir.



İlk olarak değişkenlerimizi tanımlıyoruz. "agAdi" ve "agSifresi" değişkenlerini kendi Wi-Fi adımıza ve şifremize göre yazıyoruz. Motor pinlerimizi tanımlıyoruz. Bu değişkenler, step motorumuzun bağlanacağı pinlerdir.

```
String agAdi = "Robotistan";
String agSifresi = "fortinet";
int motorPin1 = 3, motorPin2 = 4, motorPin3 = 5, motorPin4 = 6;

void setup(){
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(motorPin3, OUTPUT);
  pinMode(motorPin4, OUTPUT);
  Serial.begin(115200);
  Serial.println("AT");
  while(!Serial.find("OK")){
    Serial.println("AT");
  }
  delay(1000);
  Serial.println("AT+RST");
  delay(1000);
  while(!Serial.find("ready"))
  delay(1000);
  Serial.println("AT+CWMODE=1");
  while(!Serial.find("OK"));
  Serial.println("AT+CWJAP=\""+agAdi+"\",\""+agSifresi+"\"");
  while(!Serial.find("OK"));
  Serial.print("AT+CIFSR\r\n");
  Serial.print(espOkuma(1000));
  serialTemizle(2000);
  Serial.print("AT+CIPMUX=1\r\n");
  serialTemizle(2000);
  Serial.print("AT+CIPSERVER=1,80\r\n");
  serialTemizle(2000);
}
```

"setup" kısmında, motor pinlerimizi çıkış olarak ayarlıyoruz. Ardından seri haberleşmeyi başlatıyoruz. ESP8266 modülü ile seri haberleşme üzerinden bağlantı kurulmaktadır. Seri haberleşmenin hızı, ESP8266 modülümüzün iletişim hızı ile aynı olmalıdır. Eğer örnek kod çalışmazsa ESP8266 modülünün firmware güncelleştirmesini yaparak bu sorunu giderebilirsiniz. Firmware güncelleştirme ile ilgili detaylı bilgileri https://maker.robotistan.com/esp8266-ile-iot-dersleri-1-esp8266-modulunu-guncelleme/ adresinde bulabilirsiniz.

Seri haberleşme ile AT komutu göndererek ESP8266 modülünün hazır olup olmadığını öğreniyoruz. ESP8266 modülü hazır olana kadar bekliyoruz. Daha sonra "AT+RST" komutu ile modülümüzü resetliyoruz. Resetleme işlemi bitene kadar bekliyoruz.

"AT+CWMODE=1" komutunu göndererek modülümüzü client olarak ayarlıyoruz. Ardundan "AT+CWJAP" komutu ile ağımıza bağlanıyoruz. Ağa bağlanmayı bekledikten sonra "AT+CIFSR" komutu ile IP ve MAC adreslerini okuyoruz. Daha sonra bu verileri seri porta bastırıyoruz. Arduino'nun bilgisayarla iletişim pinleri ile esp8266 modülü ile iletişimi aynı pinlerde olduğu için seri port'a veri yazdırdıktan sonra "serialTemizle" fonksiyonu ile iletişimin bozulmasını engelliyoruz. Bu fonksiyon, seri haberleşmede kullanılmayan verilerin silinmesini sağlar. Bu fonksiyonu daha sonra açıklayacağız.

Arduino

```
void loop(){
  if(Serial.available()){
    if(Serial.find("+IPD,")){
      delay(200);
      int connectionId = Serial.read() - 48;
      String komut = espOkuma(1000);
      if(komut.indexOf("step=ileri") != -1){
        for(int adim = 0; adim < 5; adim++){
          stepIleri(50);
        }
      }
      else if(komut.indexOf("step=geri") != -1){
        for(int adim = 0; adim < 5; adim++){
          stepGeri(50);
        }
      }
      String sayfa = "<h1>Step Motor Kontrol</h1><br>";
      sayfa+="<br><a href=\"?step=ileri\"><button><h1>Ileri</h1></buttor
      sayfa+="<br><br><a href=\"?step=geri\"><button><h1>Geri</h1></butt
      komut = "AT+CIPSEND=";
      komut += connectionId;
      komut += ",";
      komut +=sayfa.length();
      komut +="\r\n";
      Serial.print(komut);
      delay(1000);
      Serial.print(sayfa);
      delay(1000);
      komut = "AT+CIPCLOSE=";
      komut+=connectionId;
      komut+="\r\n";
      Serial.print(komut);
    }
  }
}
```

"AT+CIPMUX=1" komutu ile modülümüzü çoklu bağlantıya izin verecek şekilde ayarlıyoruz. Daha sonra "AT+CIPSERVER=1",80 komutu ile sunucu oluşturuyoruz ve 80. porttan dinleme yapmaya başlıyoruz. ESP8266 modülü, gelen bağlantı isteklerini seri haberleşme ile iletir. "loop" kısmında, seri haberleşmeden veri gelip gelmediğini kontrol ediyoruz. Ardından IPD yazısının gelmesini bekliyoruz. Bu yazı geldikten sonraki veri, iletişim numarasıdır. Bu numarayı "connectionId" değişkenine eşitliyoruz. Bu numarayı sayfa verilerini gönderirken ve bağlantıyı sonlandırırken kullanacağız.

ESP modülünden gelen bağlantı verilerini "espOkuma" fonksiyonu ile alıyoruz. Daha sonra "if-else if" yapısı ile bu veride step motor kontrol komutu olup olmadığına bakıyoruz. Bu kontrolü yapmak için "if" içerisinde "indexOf" fonksiyonunu kullanıyoruz. Bu foksiyon, verilen değişkenin konumunu bir yazı dizisinde arar. Eğer bu dizide verilen değişken bulunmuyorsa -1 değerini döndürür.

Komut işleme kısımları tamamlandıktan sonra site verilerini istemciye iletmeye başlıyoruz. İlk olarak istemci ekranında görüntülenecek site kodlarını oluşturuyoruz. Bu kodlar html dilinde yazılmıştır. Bu kodların uzunluğunu ve hangi istemciye gideceğini "AT+CIPSEND" komutu ile ESP8266 modülüne gönderiyoruz. Daha sonra site komutlarını "AT+CIPSEND" komutu ile modüle gönderiyoruz. Son olarak "AT+CIPCLOSE" komutu ile bağlantımızı sonlandırıyoruz.

Arduino

```
String espOkuma(long int zamanAsimi){
  long int baslangic = millis();
  String gelen;
  while(millis() - baslangic < zamanAsimi){
    if(Serial.available()>0){
        char c = Serial.read();
        gelen += c;
    }
  }
  gelen.replace("AT+","");
  return gelen;
}
void serialTemizle(long int zamanAsimi){
  long int baslangic = millis();
  while(millis() - baslangic < zamanAsimi){
    if(Serial.available()>0){
        Serial.read();
    }
  }
}
```

"espOkuma" fonksiyonu, seri haberleşme ile esp'den gelen komutların okunmasını sağlar. "zamanAsimi" isimli değişkeni girdi olarak alır. "zamanAsimi" değişkenine girilen değer kadar milisaniye cinsinden zaman süresince veri okuması yapılır. İlk olarak fonskiyonun başlama değeri "baslangic" değişkeni ile hafızada tutulur. Daha sonra "while()" yapısı ile toplam fonksiyonun çalışma süresi, "zamanAsimi" değişkeninden küçük olduğu sürece seri haberleşme ile veri okuması yapılır. Okunan bu veri foksiyon çıktısı olarak verilir.

"serialTemizle" fonksiyonu, tıpkı "espOkuma" fonksiyonu gibi çalışır ancak çıktı olarak herhangi bir değer vermez. Bu sayede kullanılmayan seri haberleşme ile gelen byte'lar arduinonun hafızasından silinmiş olur.

Arduino

```
void stepIleri(int beklemeSuresi){
  digitalWrite(motorPin1, HIGH);
  digitalWrite(motorPin2, LOW);
  digitalWrite(motorPin3, LOW);
  digitalWrite(motorPin4, LOW);
  delay(beklemeSuresi);
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, HIGH);
  digitalWrite(motorPin3, LOW);
  digitalWrite(motorPin4, LOW);
  delay(beklemeSuresi);
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, LOW);
  digitalWrite(motorPin3, HIGH);
  digitalWrite(motorPin4, LOW);
  delay(beklemeSuresi);
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, LOW);
  digitalWrite(motorPin3, LOW);
  digitalWrite(motorPin4, HIGH);
  delay(beklemeSuresi);
}
```

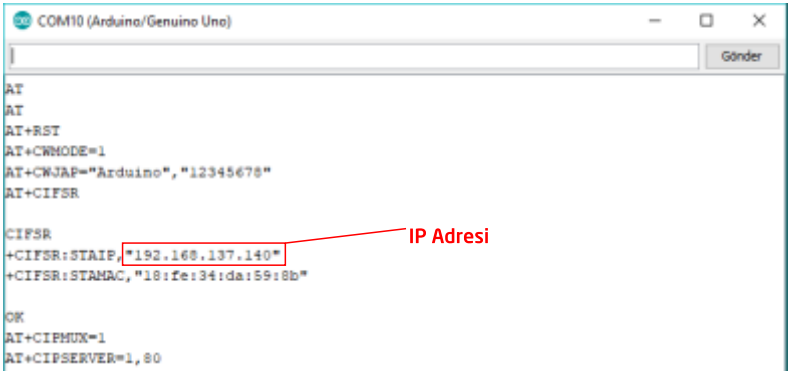
stepIleri() fonksiyonu girdi olarak beklemeSuresi değişkenini alır. Step motorun pinlerine sırasıyla HIGH değeri vererek step motorun ileri yönde hareket etmesini sağlar. 2 adım arasındaki süre beklemeSuresi değişkeni ile ayarlanır.

```
Arduino
109 void stepGeri(int beklemeSuresi){
110   digitalWrite(motorPin1, LOW);
111   digitalWrite(motorPin2, LOW);
112   digitalWrite(motorPin3, LOW);
113   digitalWrite(motorPin4, HIGH);
114   delay(beklemeSuresi);
115   digitalWrite(motorPin1, LOW);
116   digitalWrite(motorPin2, LOW);
117   digitalWrite(motorPin3, HIGH);
118   digitalWrite(motorPin4, LOW);
119   delay(beklemeSuresi);
120   digitalWrite(motorPin1, LOW);
121   digitalWrite(motorPin2, HIGH);
122   digitalWrite(motorPin3, LOW);
123   digitalWrite(motorPin4, LOW);
124   delay(beklemeSuresi);
125   digitalWrite(motorPin1, HIGH);
126   digitalWrite(motorPin2, LOW);
127   digitalWrite(motorPin3, LOW);
128   digitalWrite(motorPin4, LOW);
129   delay(beklemeSuresi);
```

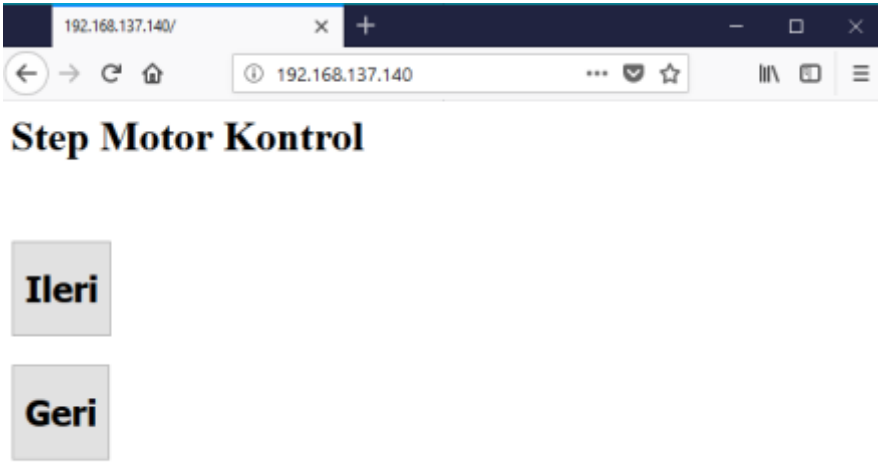
"stepGeri" fonksiyonu, tıpkı "steplleri" fonksiyonu gibi çalışır ancak motor pinlerine verilen HIGH değerleri "steplleri" fonksiyonunun tersi bir sırayla verilir. Bu sayede step motorun ters bir şekilde dönmesi sağlanır.

Kodlarımızı Arduino'ya atmadan önce ESP8266 modülünün TX ve RX pinlerinin Arduino'dan sökülmesi gerekmektedir. Arduino programlanırken ESP8266 modülünün TX ve RX pinlerinin bağlı olması, Arduino ile Bilgisayar arasındaki iletişimi engelleyerek Arduino'nun programlanmasını engellemektedir.

Örnek kodumuzu yazdıktan ve örnek devremizi kurduktan sonra sistemi çalıştıralım.



Arduino IDE üzerinden seri portumuzu açarak veri hızını 115200 olarak ayarlıyoruz. ESP8266 modülü ile iletişimi buradan görebiliriz. Yukarıdaki ayarlarlar yapıldığında seri porttaki IP adresine bir web tarayıcı ile bağlanıyoruz.



Web tarayıcısında karşımıza çıkan arayüzdeki ileri-geri butonlarına tıklayarak step motorumuzu hareket ettiriyoruz.

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