Uygulamalı Projeler İle Ardunio

2 Ekim 2020 Cuma 19:30

- https://www.robotistan.com/arduino-proje-seti?utm_source=youtube&utm_medium=aciklama

- ☑ 01 LED Yakma
- ☑ 02 Buton İle Led Yakma
- ☑ 03 Analog Değer Okuma
- ☑ 04 Potansiyometre İle Led Yakma
- ☑ 05 Karaşimşek
- ☑ 06 LDR İle Otomatik Lamba
- ☑ 07 RGB LED
- ☑ 08 NTC İle Sıcaklık Ölçümü
- ☑ 09 LM35 ile Sıcaklık Ölçümü
- ☑ 10 Ultrasonik İle Park Sensörü
- ☑ 11 Joystick İle Servo Kontrolü
- ☑ 12 Dijital Metre Yapımı
- ☑ 13 PIR İle Servo Kontrolü
- ☑ 14 Bluetooth İle RGB LED Kontrolü
- ☑ 15 Dijital Saat Yapımı
- ☑ 16 RFID İle Servo Kontrolü
- ☑ 17 ESP8266 İle Sıcaklık ve Nem Ölçümü
- ☑ 18 ESP8266 İle Step Motor Kontrolü

CENGİZHAN TOPÇU

MEKATRONİK MÜHENDİSİ

@cengizhantopcu53

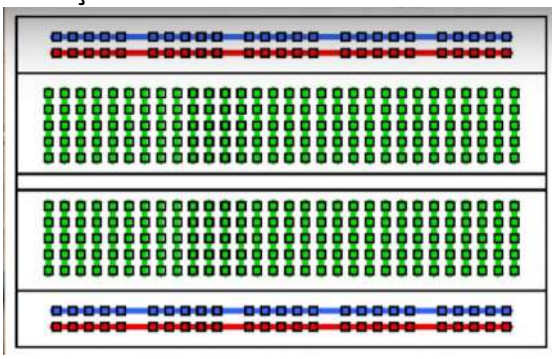
01 LED Yakma

28 Aralık 2020 Pazartesi 21:03

01 LED Yakma

Teorik Bilgi

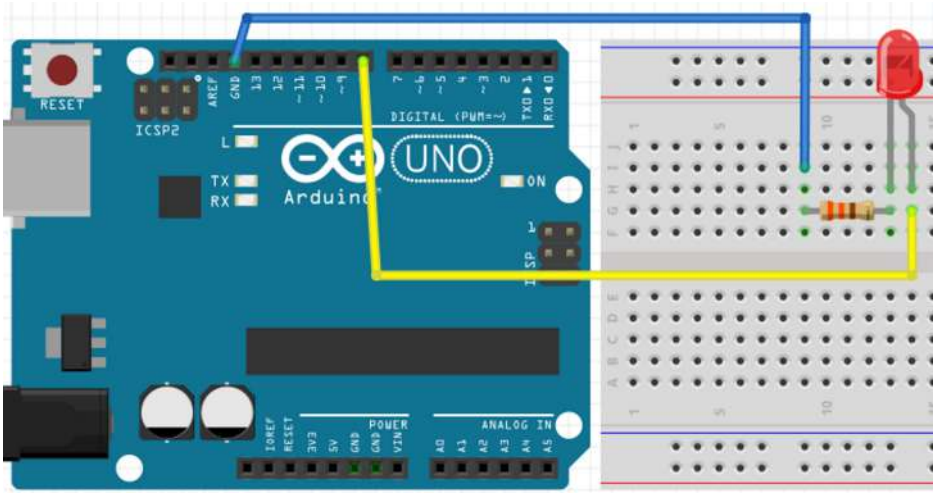
- 0.25W değerine kadar enerjiye dayanıklı dirençlerdir.
- Direnç değerini hesaplamak için; https://devreokulu.com/DirencHesaplama.html |
- Ohm çevirme işlemleri yapmak için; https://www.convertworld.com/tr/elektrik-direnci-direnc/ohm.html
- Direnç hakkında ayrıntı bilgiler için; https://maker.robotistan.com/direnc/
- 1.5-3V arası gerilimde çalışır. 5V ve üstü voltaj değerleri için gerekli dirençlerle kullanılması gerekir.
- Anot ve katot olmak üzere iki farklı bacağı vardır. Bunlardan anodu uzun olan bacağı pozitif gerilime yani + uca, katot kısa olan bacak ise negatif gerilime yani – uca ya da toprak hattına (GND, Ground) bağlanmalıdır.
- İşaretli olan kısımlar kendilerince kısa devredir.



- Arduino kartımız 5V gerilimle çalışmaktadır. LED'in üzerinden geçecek maksimum akımın 15 mA değerini geçmemesi gereklidir. Bunun için akım sınırlayıcı bir direnci LED'imize seri olarak bağlamamız gerekmekte.

5V = 0,015A x R --> R=333,33

Devre Kısmı



- Devrede LED'e seri olarak bir direnç bağlanır. Böylelikle LED üzerinden yüksek akım geçmesi ve LED'in zarar görmesi engellenir. LED'in **(+)** bacağı Arduino'nun 8.pinine bağlıdır. LED'in **(-)** bacağını dirence seri bağlanır, direncin diğer bacağına da Arduino'nun **GND** pinine bağlantı yapılmıştır.

Kod Kısmı

- Kullanacağımız pin çıkış veya giriş olarak belirlenmez ise programın devamında yazacağımız giriş veya çıkış fonksiyonları o pini kullanamaz.
- Loop fonksiyonu ile öncelikle 8 numaralı pine HIGH lojik seviyesine, yani 5V'a ayarlıyor, 500 milisaniye (1 saniyenin yarısına 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 500 milisaniye hiçbir işlem yapmadan bekliyor.

02 Buton İle Led Yakma

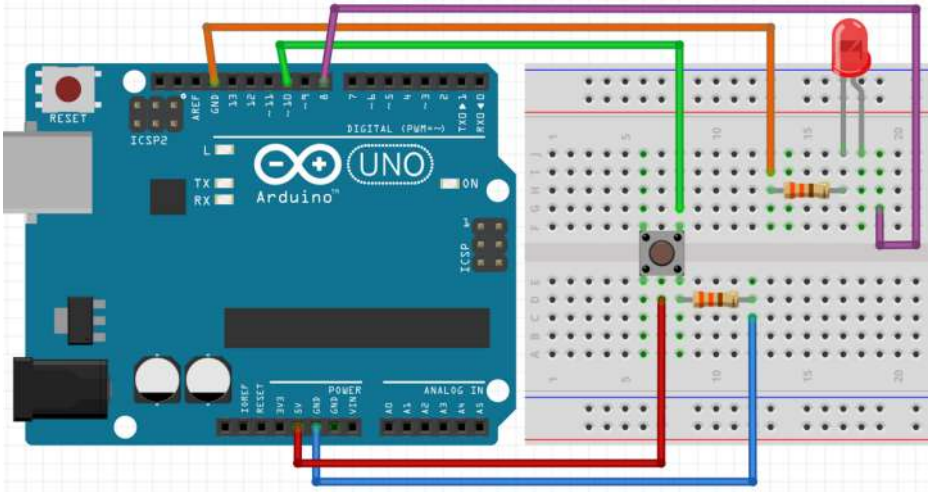
28 Aralık 2020 Pazartesi 21:03

02 Buton İle Led Yakma

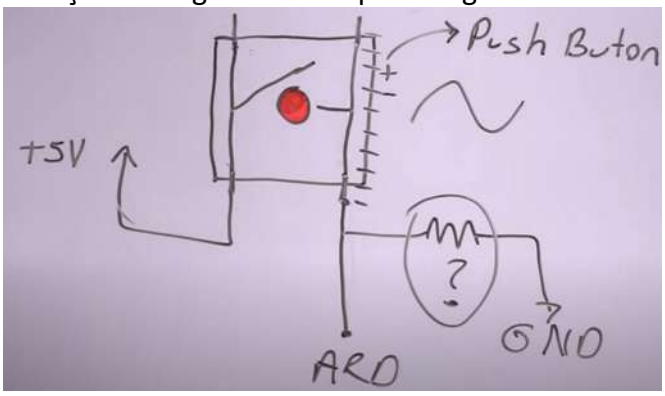
Teorik Bilgi

- Push butonlar sadece basıldığında on olan bırakıldığında off olan devre elemanlarıdır.
- Bir push buton iki anahtarın işini tek başına görebilir.
- Pinlerin olduğu yüzlerin kısa devre olması sadece butonun basılmasına bağlıdır.
- 4 pinli push butonların ikişer bacakları paralel bağlıdır. Yani buton 4 pinli de olsa 2 pinli mantığı ile bağlanmaktadır Butona basıldığında 1 ve 3 nolu kırmızı işaretli pinler ile 2 ve 4 nolu mavi işaretli pinler birbirine parele bağlıdır.

Devre Kısmı



- Butonu bağlarken 10kΩ direnç kullandık. Bu direncin ismi **pull-down direncidir**. Pull-down direnci, dijital pinleri giriş olarak kullandığımızda sinyalin bozulmamasını sağlar. Buton basılı değilken dijital pinden okunan değer 0V yani lojik LOW seviyesidir. Pull-down direnci, buton basılıp değer HIGH'a çekilmediği sürece bu pindeki gerilimin 0V'ta sabit kalmasını sağlar.



Kod Kısmı

- Giriş-çıkış ayarlarken giriş yapmak istediğimiz butonlara "INPUT", çıkış yapmak isteğimiz pinlerede "OUTPUT" yazarız.

```ino
#define Buton 10
#define Led 8

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 Buton 10
#define Led 8

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

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

```

03 Analog Değer Okuma

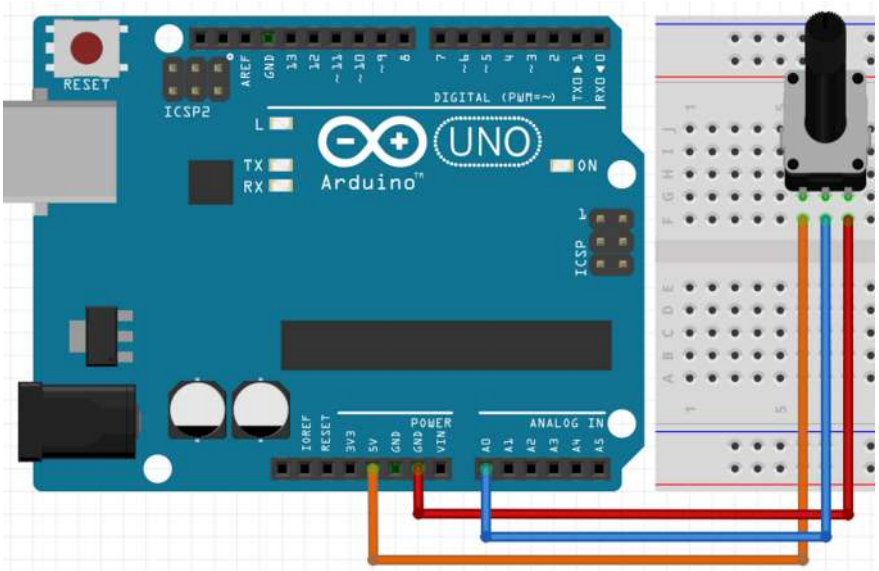
28 Aralık 2020 Pazartesi 21:03

03 Analog Değer Okuma

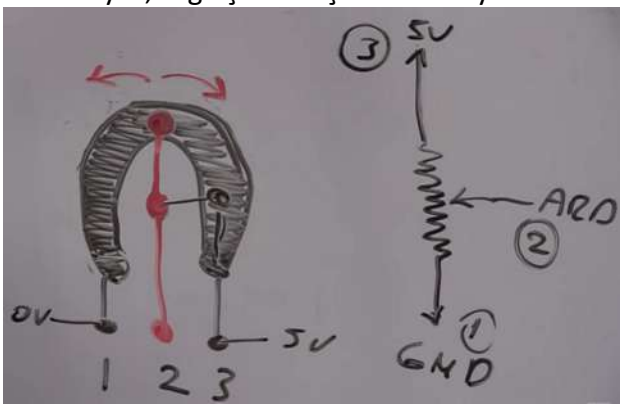
Teorik Bilgi

- Potu çevirdikçe direnci değişir.
- Ara direnç değerlerine ihtiyaç duyulduğu zaman veya Analogdan Dijitale Dönüştürücü (ADC - analog to digital converter) işlemlerinde kullanılabilir.
- Potansiyometre 3 bacaklı devre elemanıdır. Orta bacak sinyali alacağımız kısımdır. Sağ ve sol bacaklar + ve - voltaj bağlayacağımız kısımdır. Dirençlerin voltaja göre yönü olmadığından sağ ve sol bacaklar için kaç volt olduğun bir önemi yoktur.Mikrokontrolcü uygulamalarında ise genellikle gerilim bölücü olarak kullanılır.
- Arduino UNO kartımızdaki işlemcide, 10-bit çözünürlüğe sahip analogdan dijitale dönüştürücü (ADC – analog to digital converter) mecvuttur. Arduino mikrokontrolcüsü 5V gerilimle çalışmakta. Bu mikrokontrolcüde sahip olduğunu söylediğimiz 10-bit ADC, 0V ile 5V arası gerilimleri **2^10=1024** adım hassasiyet ile okuyabilir. Yani analog input pinlerinden birine vereceğimiz 0V gerilim bize 0 değerini; aynı şekilde 5V gerilim ise 1023 değerine denk düşüyor. Bu pine herhangi bir gerilim uygulamadığımız takdirde ADC'miz 0-5V arasında çalışacaktır.

Devre Kısmı



- Potansiyometre, analog kontrol sunduğundan dolayı Arduino'da **Analog INPUT** olarak bağlanır. Böylelikle potansiyometreden gelen analog giriş sinyalleri ile çıkıştaki uyarıcılar kontrol edilebilir veya giriş sinyallerinin takibi yapılabilir. Potansiyometrenin orta bacağı **Data** olarak kullanılmıştır. Giriş sinyalleri bu bacaktan kontrol edilmektedir. Soldaki bacak **GND**, sağdaki bacak da **5V** pinine bağlanmıştır. Böylelikle potansiyometre sağa doğru çevrildikçe Arduino'ya giden veri artacak, sola doğru çevrildikçe veri azalacaktır. Sağdaki ve soldaki bacakların yerleri değiştirilseydi bu sefer sola çevrildikçe veri artıyor, sağa çevirdikçe veri azalıyor olurdu.



Kod Kısmı

- "setup" kısmında önceki kod kısımlarında dijital giriş-çıkış kullandığımızdan dolayı, pinleri ne olarak kullanacağımıza göre ayarlıyorduk. Analog okuma yaparken giriş çıkış tanımlamamıza gerek yok bu sebepten dolayı bu kodda "pinMode" komutunu kullanmıyoruz.
- 6.satır ile 9600 Baund bir seri haberleşme başlatıyoruz. Arduino kodu çalışmaya başladığında ilk olarak bilgisayar ile haberleşmeyi başlatacaktır. Haberleşme başladıktan sonra 7.satırı okuyarak Pot Değer Okuma yazısı bilgisayarda seri mönitöre yazdırılacaktır. Aynı şekilde 14.satırda okunan değeri yazdırır.
- 12.satırda 5V gerilim, 1024 bite bölünür ve potansiyometreden gelen değer ile çarpılır; değerin eşiti olan gerilim hesaplanır.

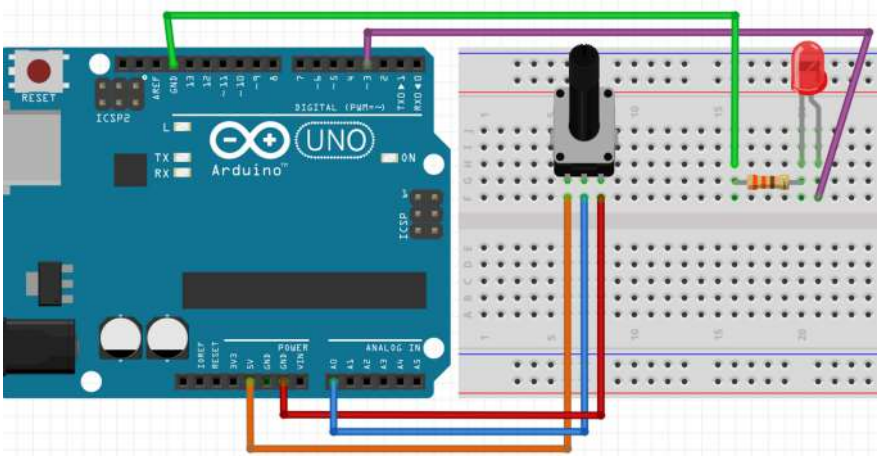
```
1  #define potpin A0
2
3  int deger=0;
4
5  void setup(){
6    Serial.begin(9600);
7    Serial.println("Pot Değer Okuma");
8  }
9
10 void loop(){
11   deger = analogRead(potpin);
12   //float gerilim = (5.00/1024.00)*deger;
13   //Serial.println(gerilim);
14   Serial.println(deger);
15   delay(300);
16 }
```

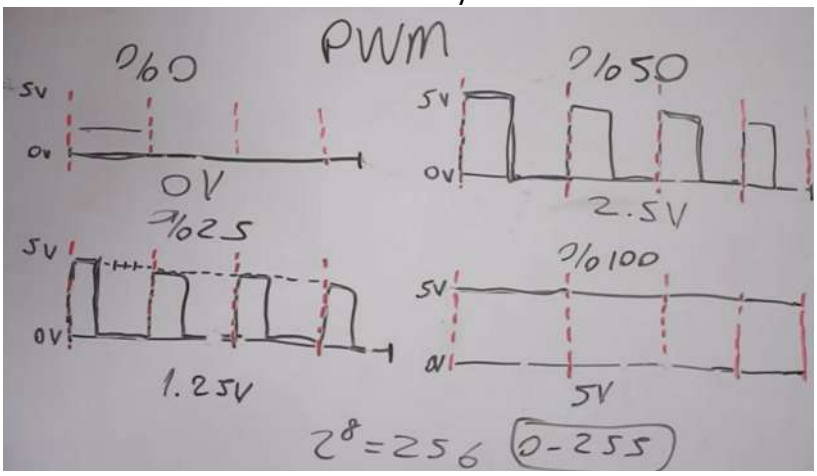
04 Potansiyometre İle Led Yakma

28 Aralık 2020 Pazartesi 21:03

04 Potansiyometre İle Led Yakma

Devre Kısmı



- Bu uygulamaya kadar dijital giriş-çıkış ve analog girişi gördük. Bu uygulamayla analog çıkış yani PWM özelliğini öğreneceğiz.
- PWM (Pulse with Modulation), sinyal genişlik modülasyonun kısaltmasıdır. Bu özellik Arduino Uno üzerinde 6 pinde mevcut yaklaşık işareti bulunan 3,5,6,9,10 ve 11.pinlerdir.
- PWM özelliği ile istediğimiz voltajı vererek ledin parlaklığını ayarlayabiliriz. Biz bu özelliği genellikle robotlarda motor hızlarında kullanıyoruz.
- 

Kod Kısmı

- Dijital giriş-çıkış kullanmadığımızdan "setup" kısmı boş bırakıyoruz.
- Döngümüzde Pot'dan veriyi Led'e göndermek için "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 arasında oranlıyoruz.
- Map kullanımında 4 parametre var. İlk olarak değişkenimiz yazıyoruz. Oranlanacak sayımızın min ve max değerleri yazılır sonra oranlanacak sayıının min ve max değeri yazılarak kullanılır.
- Analog okumayı 10 bit (2^10=1024) çözünürlükte yaparken, analog yazmayı 8 bit (2^8=256) çözünürlükte yapabiliyoruz. Okuduğumuz veriyi, çıkışa göre oranlamamız gerekiyor.
- "map" komutu yerine direkt olarak 4'e bölebiliriz.
- Oranlama işleminden sonra "AnalogWrite" komutu ile PWM pinlerinden çıkış verebiliriz.

```
1 #define led 3
2 #define pot A0
3
4 void setup(){
5 }
6
7 void loop(){
```

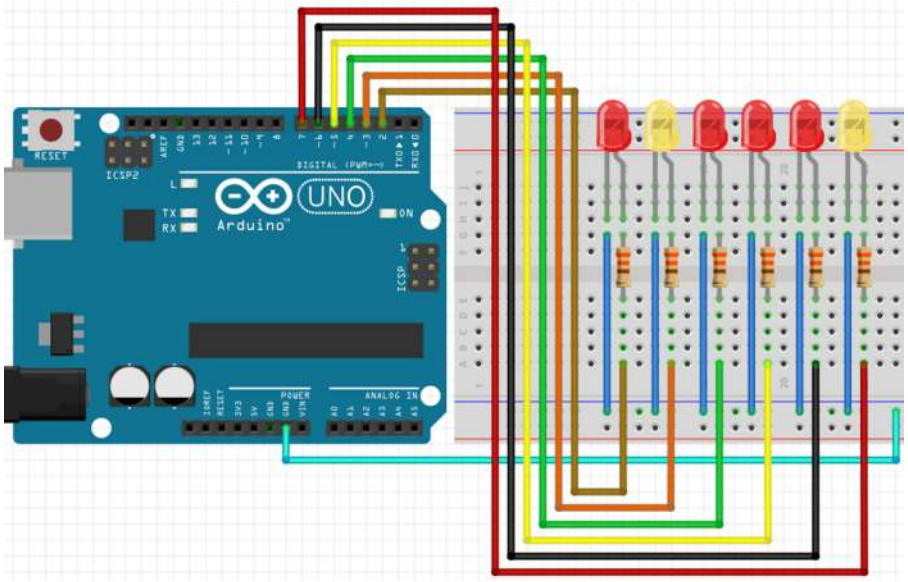
```
#define led 3
#define pot A0

void setup(){
}

void loop(){
  int deger = analogRead(pot);
  //deger=map(deger,0,1023,0,255);
  deger=deger/4;
  analogWrite(led,deger);
```

05 Karaşimşek

28 Aralık 2020 Pazartesi 21:09

05 Karaşimşek

Devre Kısmı



- Diğer Led uygularımızda direnci Led'in Katot yani (-) uca seri bağlamıştık. Bu uygulamada Anato yani (+) uca seri bağladık. Yani direnci Led'in hangi bacağına bağladığımızın bir önemi yok.

Kod Kısmı

- Dizi tanımlarken "ledler[] ifasinin içerisine dizinin elaman sayısını yazabiliriz.

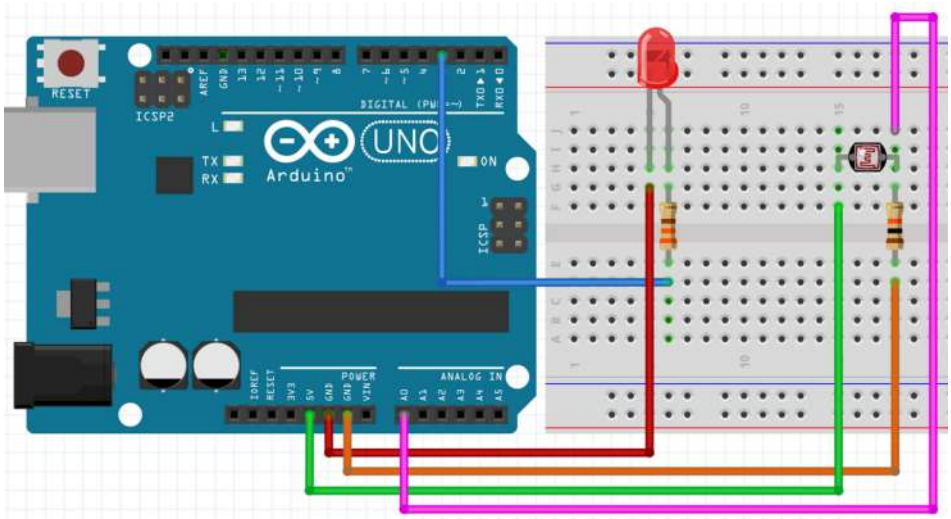
```
int ledler[6] = {2,3,4,5,6,7};

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(20);
      digitalWrite(ledler[i], LOW);
  }
  for(int j=5; j>=0; j--){
      digitalWrite(ledler[j], HIGH);
      delay(20);
      digitalWrite(ledler[j], LOW);
  }
}
```

06 LDR İle Otomatik Lamba

28 Aralık 2020 Pazartesi 21:09

06 LDR İle Otomatik Lamba

Devre Kısmı



- Üzerine düşen ışığa bağlı olarak iki ucu arasındaki direnç değeri değişir.
- LDR'nin bir bacağından (-) hattına diğer bacağından (+) hattına gideriz. Direnci LDR'nin (-) hattına bağladık ve direncin diğer bacağından GND'ye bağladık. LDR'nin (+) bacağından 5V'a gittik. LDR ile direncin birbirleriyle ortak bacağından Analog Giriş olan A0'gideriz.

Kod Kısmı

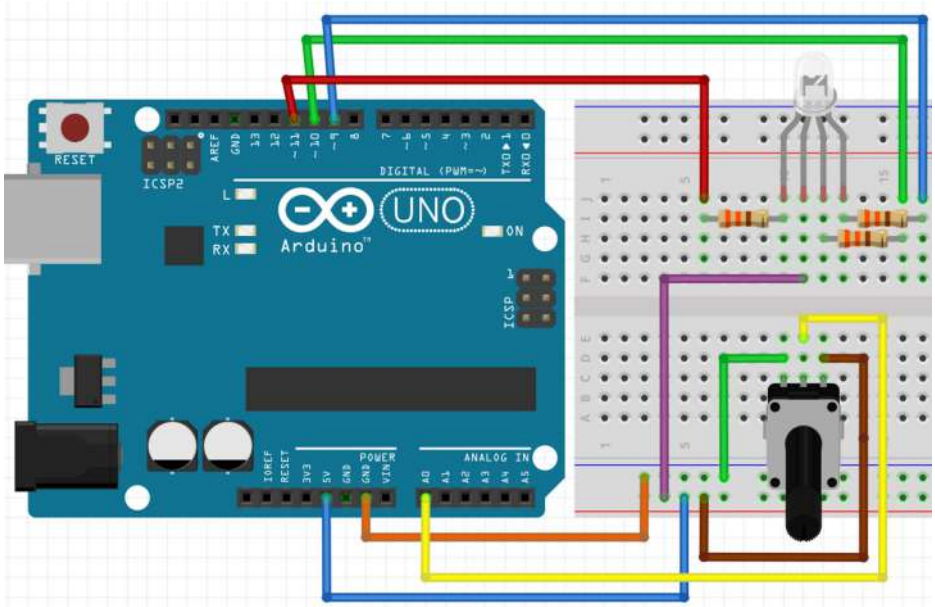
```
#define led 3

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

void loop(){
  int isik=analogRead(A0);
  Serial.println(isik);
  delay(50);
  if(isik>580){
    digitalWrite(led,LOW);
  }
  if(isik<530){
    digitalWrite(led,HIGH);
  }
}
```

07 RGB LED

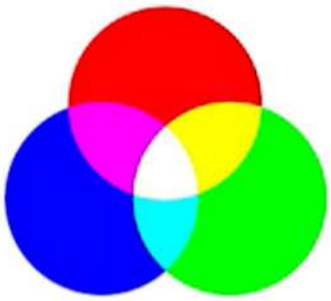
28 Aralık 2020 Pazartesi 21:09

07 RGB LED

Devre Kısmı



- İçerisinde Kırmızı, Yeşil ve Mavi olmak üzere üç farklı renkte LED barındırmaktadır.
- Uzun ucu artı uçtur, uzun uca artı voltajı verdikten sonra toprağa çektiğiniz bacağın rengi yanacaktır.
- 1.5-3V arası gerilimle çalışmaktadır.
- 5V ve üstü voltaj değerleri için gerekli dirençlerle kullanılması gerekir.
- RGB LED'imizn kırmızı, mavi ve yeşil bacakların hepsine aynı anda elekrtik verdiğimizde sadece kırmızı ve yeşil yanacak fakat mavi yanmayacaktır çünkü iç dirençleri farklıdır. Akımda en düşük dirençli olan yoldan geçiceğinden önce kırmızıyı sonra yeşili tercih edecektir. Mavi ışık bu yüzden yanmayabilir. Bu yüzden RGB LED'imizi daha sağlıklı kullanabilmek için Mavi, Kırmızı ve Yeşil bacakların hepsine ayrı ayrı 330 Ohm direnç bağlıyoruz.



Kod Kısmı

- Loop içerisine yazdıklarımızı deger_oku ve deger_yaz olarak fonksiyon olarak ayrı şekilde belirttik.
- Loop kısmı olmadan setup kısmını çalıştırırsak beyaz ışık yanar. Yorum kısmını kaldırıp çalıştırırsak mavi yanar. Bu rengi digitalWrite kısmındaki HIGH ve LOW ile değiştirirsek başka renklerde elde ederiz.
- LED'i istediğimiz parlaklık seviyesinde yakabilmek için PWM özelliğini kullanıyoruz. PWM bacaklarda (3,5,6,9,10,1 1) bulunduğu için bağlantılarımız bunlarla yapıyoruz.
- 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 kullandık.
- Her basamak içerisinde benzer komutlar uygulanıyor. Sadece atanan değer farklı renklerde oluyor. “if” içerisinde öncelikle gelen değer O ile 255 arasına oranlanıyor. 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 O (sıfır) verdiğimizde LED tam parlaklıkta yanacak, 255 verdiğimizde sönecektir. Bu problemi ters durumu aşmak için çıkan değerleri LED'lere yollamadan örce 255'ten çıkararak yollayacağız. Bu durumda “if” koşulları içerisinde sanki narmal LED bağlamış gibi kodumuzu yazabileceğiz. İlk “if” içerisinde kırmızı LED'e göndermek Üzere 2595'ten “potDeger"i çıkararak gönderiyoruz. Yeşil LED'e ise direk olarak potDeger'ini gönderiyoruz. Mavi LED'i söndürmek için Ofsıfır) 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.

```
#define redpin 11
#define greenpin 10
#define bluepin 9

#define potpin A0

int potdeger=0;

int reddeger=0;
int greendeger=0;
int bluedeger=0;

void deger_oku() {
  potdeger = analogRead(potpin);

  if (potdeger < 341)
  {
    potdeger=(potdeger*3)/4;
    reddeger=255-potdeger;
    greendeger=potdeger;
    bluedeger=0;
  }
  else if (potdeger < 682)
  {
    potdeger=((potdeger-341)*3)/4;
    reddeger=0;
    greendeger=255 - potdeger;
    bluedeger=potdeger;
  }
  else
  {
    potdeger=((potdeger-683)*3)/4;
    reddeger=potdeger;
    greendeger=0;
    bluedeger=255 - potdeger;
  }
}

void deger_yaz() {
  analogWrite(redpin,255 - reddeger);
  analogWrite(greenpin,255 - greendeger);
  analogWrite(bluepin,255 - bluedeger);
}

void setup()
{
  pinMode(redpin,OUTPUT);
```

```
45 void setup()
46 {
47   pinMode(redpin,OUTPUT);
48   pinMode(greenpin,OUTPUT);
49   pinMode(bluepin,OUTPUT);
50
51   //digitalWrite(redpin,HIGH);
52   //digitalWrite(greenpin,LOW);
53   //digitalWrite(bluepin,HIGH);
54 }
55
56 void loop()
57 {
58   deger_oku();
59   deger_yaz();
60 }
```

08 NTC İle Sıcaklık Ölçümü

28 Aralık 2020 Pazartesi 21:09

08 NTC İle Sıcaklık Ölçümü

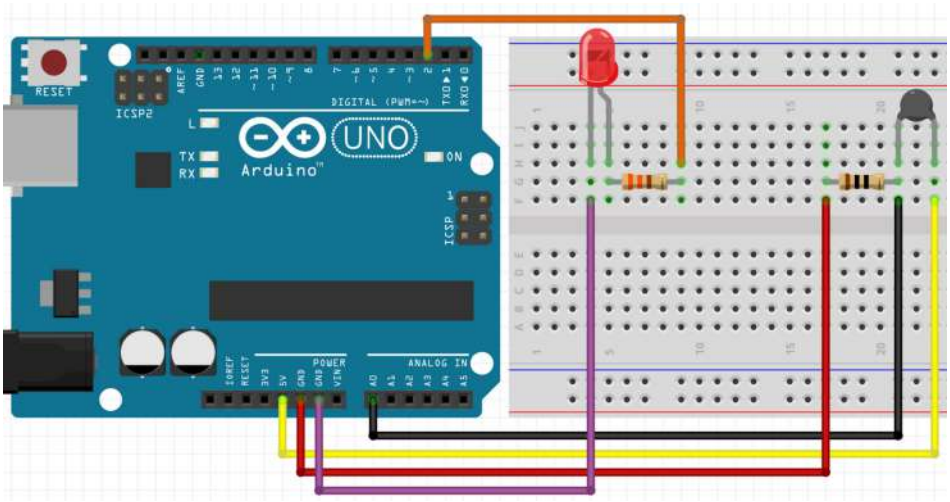
Teorik Bilgi

- NTC, bulunduğu ortamın veya temas ettiği yüzeyin sıcaklığı arttıkça elektriksel direnci azalan devre elemanıdır.

– 300 C° ile +50 C° arasındaki sıcaklıklarda kararlı bir şekilde çalışabilirler.

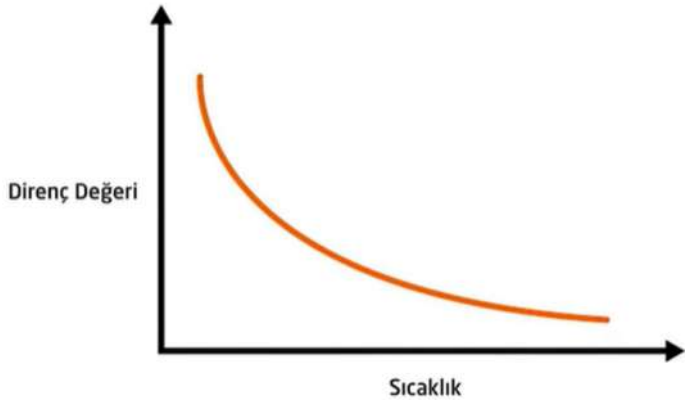
- LDR uygulamamızda yaptığımız gibi sıcaklık okurken de analog sinyali okuyup yorumlayarak istediğimiz şeyleri yaptıracağız. NTC (Negative Temperature Coefficient), sıcaklık artışı karşısında iç direncini düşüren bir elemandır. NTC yerine genellikle kullanılan elemanlardan bir tanesi de PTC'dir. PTC 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.

Devre Kısmı



- NTC'nin bir bacağından (-) hattına diğer bacağından (+) hattına gideriz. Direnci NTC'nin (-) hattına bağladık ve direncin diğer bacağından GND'ye bağladık. LDR'nin (+) bacağından 5V'a gittik. NTC ile direncin birbirleriyle ortak bacağından Analog Giriş olan A0'gideriz.

Kod Kısmı



```
#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);
  }
}
```

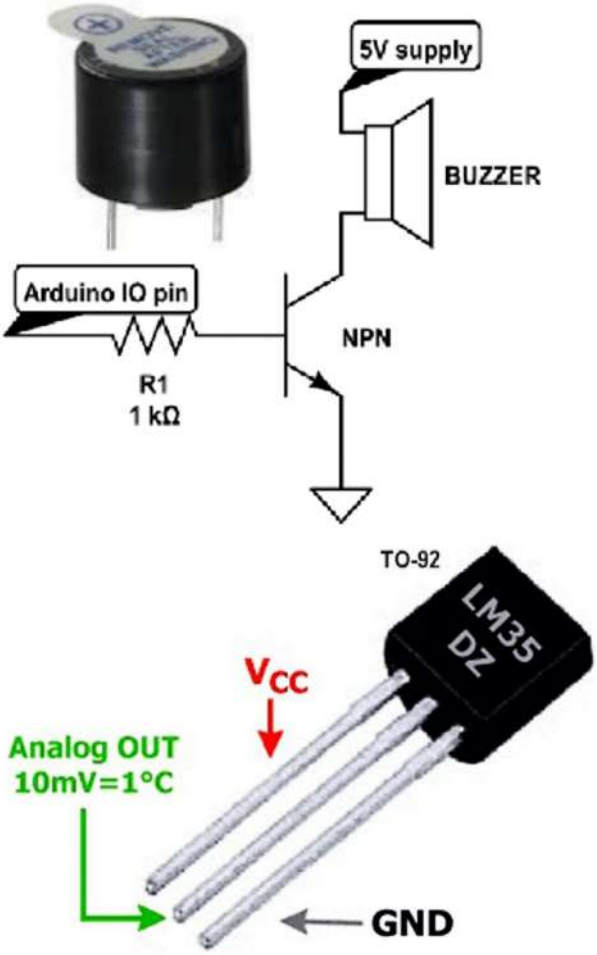
09 LM35 ile Sıcaklık Ölçümü

28 Aralık 2020 Pazartesi 21:09

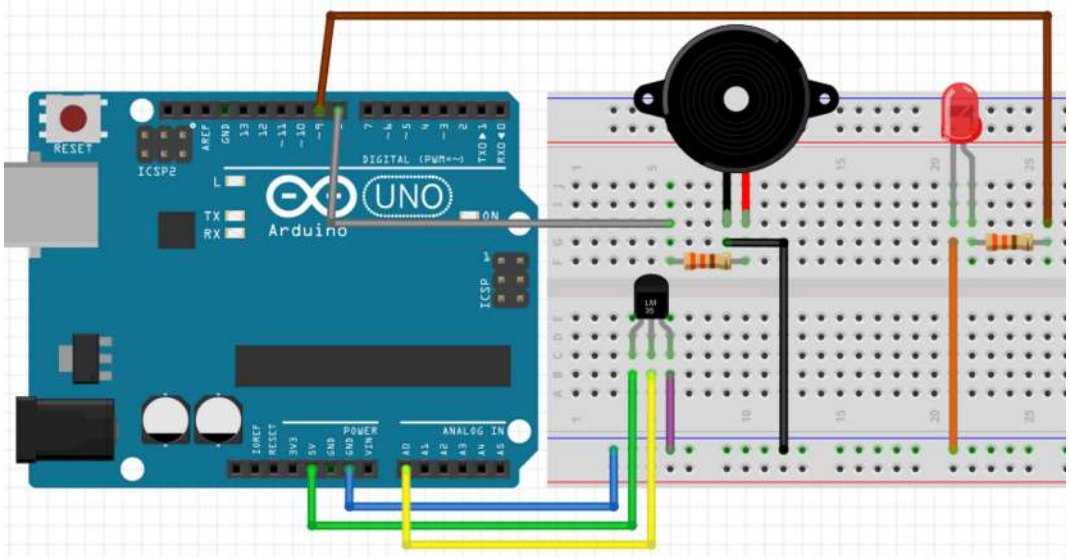
09 LM35 ile Sıcaklık Ölçümü

Teorik Bilgi

- LM35 sensörü sıcaklığı okuyarak analog veriye dönüştürür. LM35 ile -55°C ile +150°C derece arasında ölçüm yapabilirsiniz. Lineer çalışan, yani sıcaklık arttıkça aynı oranda çıkıştaki gerilimi arttırarak 10mV/°C hassasiyetinde çalışır.



Devre Kısmı



- LM35'in üç bacağı bulunmaktadır. Sensörün üzerindeki kesim kısım aynı zamanda yazılı kısıma önden baktığınızda sağda kalan bacak negatif(-), solda kalan bacak pozitif(+) ve ortadaki bacak ise sinyal çıkışıdır. Sensörün pozitif bacağına +5V ve negatif bacağınıda GND'ye bağlıyoruz. Buzzer'ı kullanırken, LED kullanırkenki gibi direnç koyarak buzzer'ın zarar görmesini engelliyoruz. Buzzer'ın pozitif bacağını ise direnç üzerinden Arduino kartımızın 8 numaralı pinine bağlıyoruz. Buzzer'ın negatif bacağını ise GND hattına bağlıyoruz. LED'i ise pozitif bacağını direnç üzerinden Arduino kartınızın 9 numaralı pinine negatif bacağını ise GND hattına bağlıyoruz.

Kod Kısmı

```
#define led 9
#define buzzer 8

#define lm A0

int zaman=100;
int okunan_deger=0;
float sicaklik_gerilim=0;
float sicaklik=0;

void setup()
{
  pinMode(led, OUTPUT);
  pinMode(buzzer,OUTPUT);
}

void loop()
{
  okunan_deger=analogRead(lm);
  sicaklik_gerilim=(5000.0/1023.0) * okunan_deger;
  sicaklik=sicaklik_gerilim/10.0;

  if (sicaklik>=30){
    digitalWrite(led,HIGH);
    digitalWrite(buzzer,HIGH);
    delay(zaman);
    digitalWrite(led,LOW);
    digitalWrite(buzzer,LOW);
  }
  else{
  digitalWrite(led,LOW);
  digitalWrite(buzzer,LOW);
  }
}
```

10 Ultrasonik İle Park Sensörü

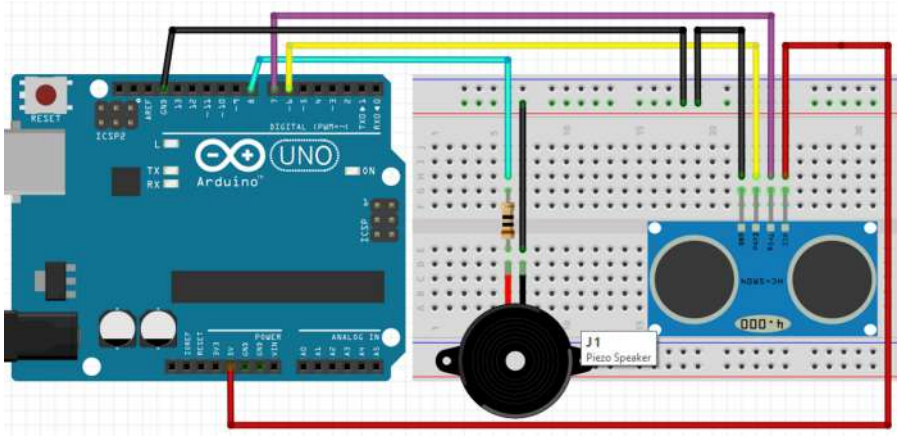
28 Aralık 2020 Pazartesi 21:14

10 Ultrasonik İle Park Sensörü

Teorik Bilgi

- 2cm'den 400cm'ye kadar 3mm hassasiyetle ölçüm yapabilen bu ultrasonik sensör çeşididir.
- HC-SR04 Sensörün Trig pininden uygulanan sinyal 40 kHz frekansında ultrasonik bir ses yayılmasını sağlar. Bu ses dalgası herhangi bir cisme çarpıp sensöre geri döndüğünde, Echo pini aktif hale gelir. Biz ise bu iki sinyal arasındaki süreyi ölçerek yani sesin yankısını algılayarak cismin sensörden uzaklığını tespit edebiliriz.

Devre Kısmı



Kod Kısmı

- Long, int değişkenine göre 2 kat fazla hafıza kullanır.
- Trig pinini yüksek ve alçak yaparak sensörün fiziksel ortama ses dalgası yollamasını sağlıyoruz. Ses dalgası yollandıktan sonra “pulseln(echoPin,HIGH)” komutu ile yolladığımız ses dalgasının cisimden yansıyıp geri gelmesini bekliyoruz. Bu beklediğimiz zamanıda “pulseln” komutu ile ölçebiliyoruz. Ölçtüğümüz bu değer “duration” değişkenine yazdırıyoruz. 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 O(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 mesafez algılıyor ise uzun aralıklar ile ötecek.

```
#define echopin 6
#define trigpin 7
#define buzzerpin 8

int maxRange=50;
int minRange=0;

void setup()
{
  pinMode(echopin, INPUT);
  pinMode(trigpin, OUTPUT);
  pinMode(buzzerpin, OUTPUT);
}

void loop()
{
  int olcum=mesafe(maxRange,minRange);
  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);

  if (distance>=maxRange || distance <=minRange)
  return 0;
  return distance;
}

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

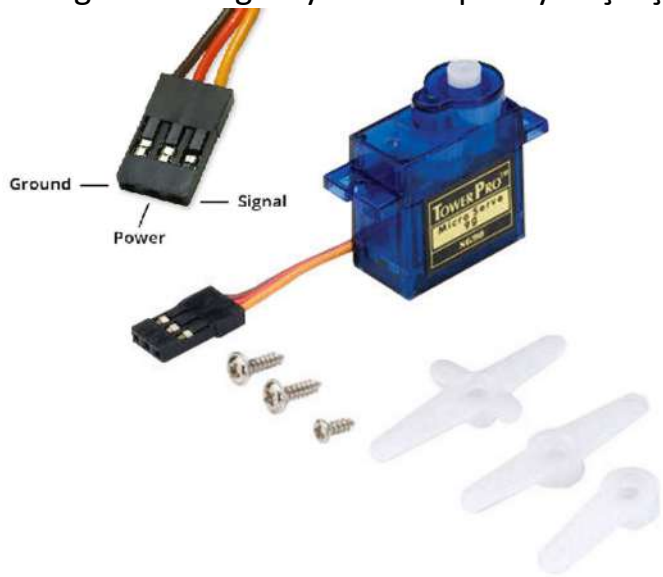
11 Joystick İle Servo Kontrolü

28 Aralık 2020 Pazartesi 21:14

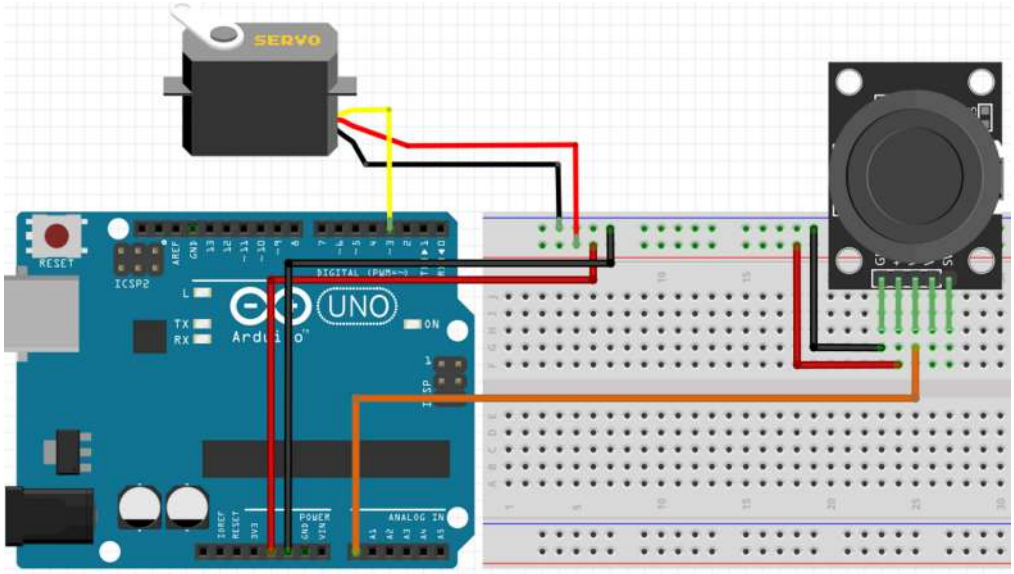
11 Joystick İle Servo Kontrolü

Teorik Bilgi

- X ve Y ekseni olmak üzere iki eksende analog çıkış verir. Bununla birlikte joystick'in ortasında bir adet de buton bulunmaktadır.
- Joystick, üzerindeki kolun ileri-geri veya sağ-sol hareketi yaptırılmasına göre analog sinyal olarak pozisyon çıkışı verir.



Devre Kısmı



- 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 O 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 O ile

180 derece arasında dönmesini sağlar.

Kod Kısmı

- Arduino Uno, analog okuma pinleri üzerinden 0 ile 1023 arasında veri vermektedir. Ancak servo motorumuz O 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.

```
1  #include <Servo.h>
2
3  Servo motor;
4  int deger;
5  int derece;
6
7  void setup() {
8    motor.attach(3);
9  }
10
11 void loop() {
12   deger = analogRead(A0);
13   derece = map(deger, 0,1023,0,180);
14   motor.write(derece);
15 }
```

12 Dijital Metre Yapımı

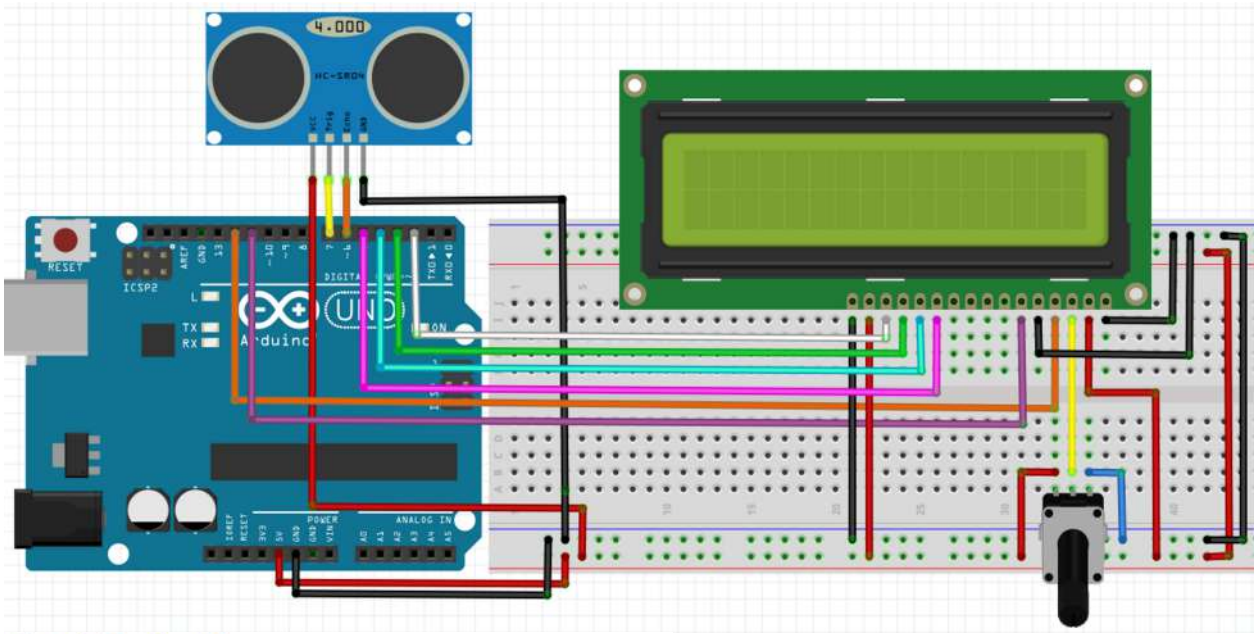
28 Aralık 2020 Pazartesi 21:22

12 Dijital Metre Yapımı

Teorik Bilgi

- 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 döndürüldüğünde orta pininin gerilimi değişmektedir. Bu değişen gerilimde LCD ekranın Kontrastını ayarlamamızı sağlayacak.

Devre Kısmı



Bacak Bağlantıları:

LCD Pin	Sembol	Bağlantı
1	Vss	GND
2	Vdd	+5V
3	Vo	Kontrast ayarı
4	RS	Register seçme pini
5	R/W	Data read/write pini.
6	E	Enable pini
7	DB0	Data Bus 0
8	DB1	Data Bus 1
9	DB2	Data Bus 2
10	DB3	Data Bus 3
11	DB4	Data Bus 4
12	DB5	Data Bus 5
13	DB6	Data Bus 6
14	DB7	Data Bus 7
15	A	LED arka ışık anot (+5V)
16	K	LED arka ışık katot (GND)

Kod Kısmı

- 7. ve 8.satırda LCD'nin pin bağlantılarını ayarlıyoruz.
- 13.satırda “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.
- “lcd.clear()” fonksiyonu ile daha önceden kalan yazılar ekrandan silinir.
- “İcd.setCursor()”fonksiyonu ile LCD ekrana yazacağımız yazının hangi satır ve sütuna yazdırılacağı ayarlanır.
- uzaklık değeri LCD ekrana “lcd.print()” fonksiyonu ile yazdırılır.

```
1  #include  <LiquidCrystal.h>
2
3  int trigPin = 7;
4  int echoPin = 6;
5  int sure;
6  int uzaklik;
7  int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
8  LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
9
10  void setup() {
11    pinMode(trigPin, OUTPUT);
12    pinMode(echoPin,INPUT);
13    lcd.begin(16, 2);
14  }
15
16  void loop() {
17    digitalWrite(trigPin, LOW);
18    delayMicroseconds(5);
19    digitalWrite(trigPin, HIGH);
20    delayMicroseconds(10);
21    digitalWrite(trigPin, LOW);
22
23    sure = pulseIn(echoPin, HIGH, 11600);
24    uzaklik= sure*0.0345/2;
25    delay(250);
26
27    lcd.clear();
28    lcd.setCursor(0, 0);
29    lcd.print("Uzaklik:");
30    lcd.setCursor(0, 1);
31    lcd.print(uzaklik);
32    lcd.print("cm");
33  }
```

13 PIR İle Servo Kontrolü

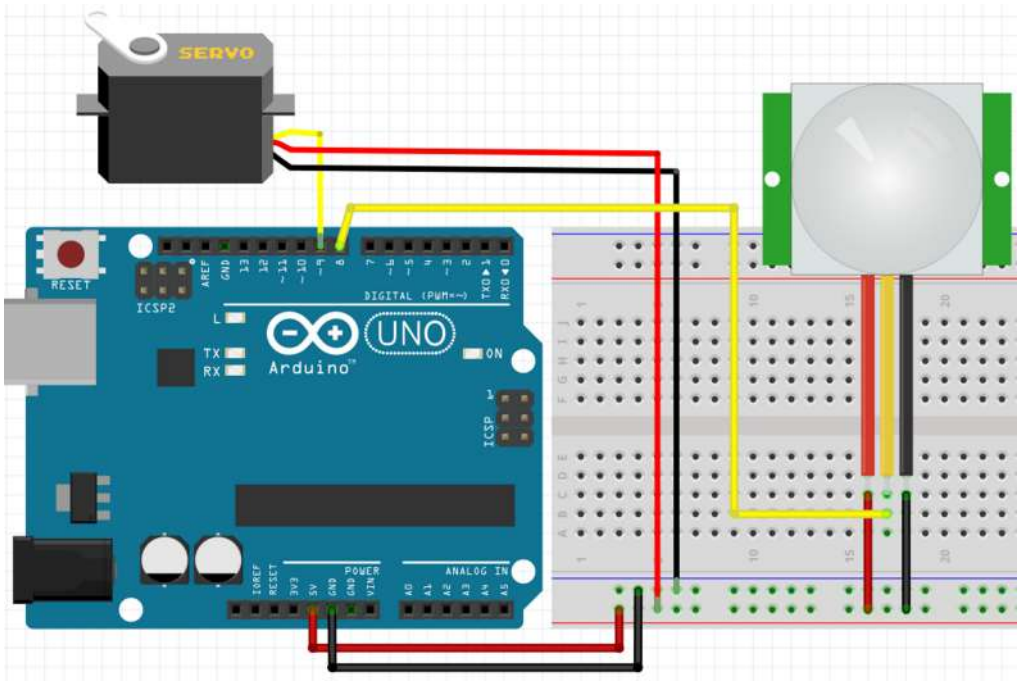
28 Aralık 2020 Pazartesi 21:22

13 PIR İle Servo Kontrolü

Teorik Bilgi

- PIR ismi Passive Infra-Red kelimelerinin baş harflerinden gelmektedir. Bu da bu sensörün kızılötesi dalgalarla çalıştığı anlamına gelir. Hareket sensörü ortamdaki sıcaklık ve kızılötesi dalga değişimlerini algılamaya yarayan sensörlerdir. Yapılarında bir fresnel lens bulunur. Bu lens sayesinde ortamdaki nesnelerden gelen ışınlar sensörün odaklanmasını sağlar. Ortamda bir dalga değişimi olduğunda sensör algılama işlemi gerçekleştirir.
  Dijital çıkışlı olan bu modül, ortamda hareket algılamadığı zaman lojik 0, hareket algıladığı zaman ise lojik 1 çıkışı vermektedir.
  Sensör üzerinde Sx ve Tx olmak üzere iki adet potansiyometre bulunmaktadır. Sx potansiyometresi sensörün görme mesafesini 3 ile 5 metre arasında değiştirmektedir. Tx potu ise sensör gördükten sonra ne kadar süre daha çıkış pininden lojik 1(3.3V) çıkışını vereceğini ayarlamaktadır.

Devre Kısmı



Kod Kısmı

- 9.satır ile servomuzu 9. pin ile ilişkilendiriyoruz.

```
#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);
  }
}
```

14 Bluetooth İle RGB LED Kontrolü

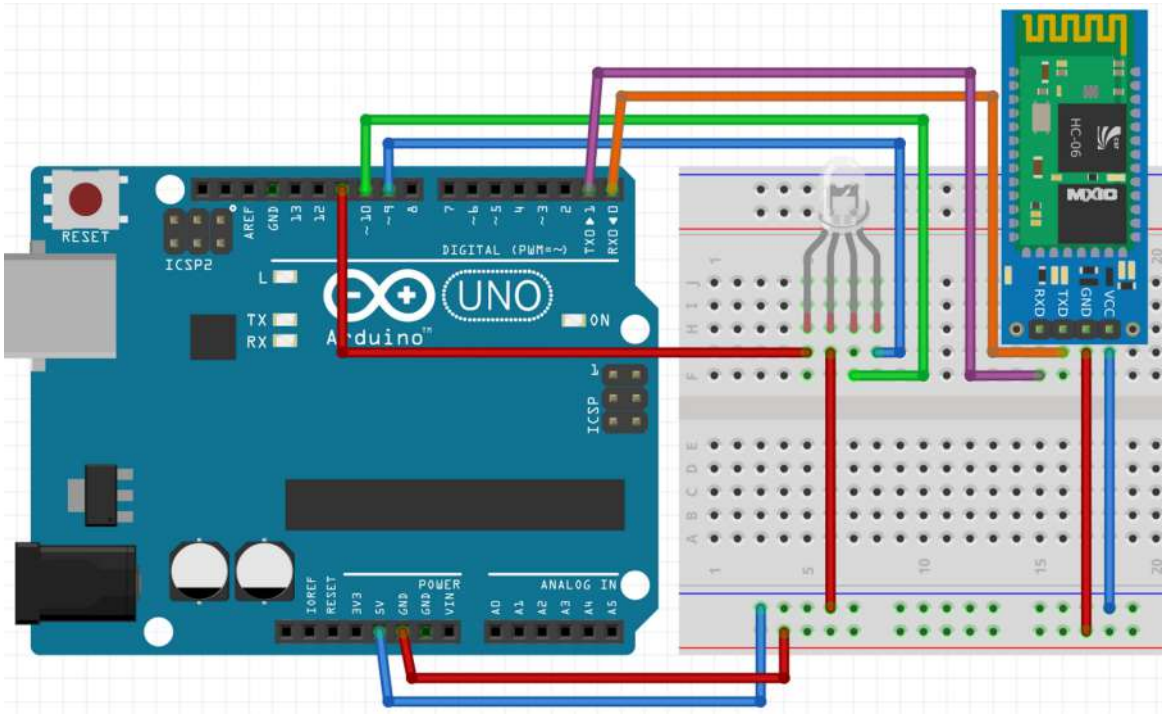
28 Aralık 2020 Pazartesi 21:22

14 Bluetooth İle RGB LED Kontrolü

Teorik Bilgi

- Bluetooth modülü Arduino ile TX/RX protokolü ile birbirine bağlanır. RX “Receive” yani almak, TX ise “Transmit” yani vermek kelimelerinden gelmektedir. Yani kısaca bir alıcı/verici protokolüdür.
- 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. Bluetooth modülü öntanımlı olarak bu haberleşme hızını kullanmaktadır.
- HC06 Bluetooth-Serial Modül Kartı, Bluetooth SSP(Serial Port Standart) kullanımı ve kablosuz seri haberleşme uygulamaları için tasarlanmıştır.
- Bluetooth 2.0'ı destekleyen bu kart, 2.4GHz frekansında haberleşme yapılmasına imkan sağlayıp açık alanda yaklaşık 10 metrelik bir haberleşme mesafesine sahiptir.
- RX ve TX pinleri lojik 3.3V seviyesindedir.

Devre Kısmı



- Bağlantıda Arduino’nun TX pini Bluetooth modülünün RX pinine bağlanmalı, Arduino’nun RX pini de Bluetooth modülünün TX pinine çapraz şekilde bağlanmalıdır.

Kod Kısmı

- Bilgisayardan Ardunio'ya yükleme yaparken 0 nolu pindeki jumper kablo çıkarılması gerekir.
- 12.satır Seri haberleşmeden veri gelmesini bekler.
- 13.satır Seri haberleşmeden gelen veriyi okur.

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

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);
  }
}
```

15 Dijital Saat Yapımı

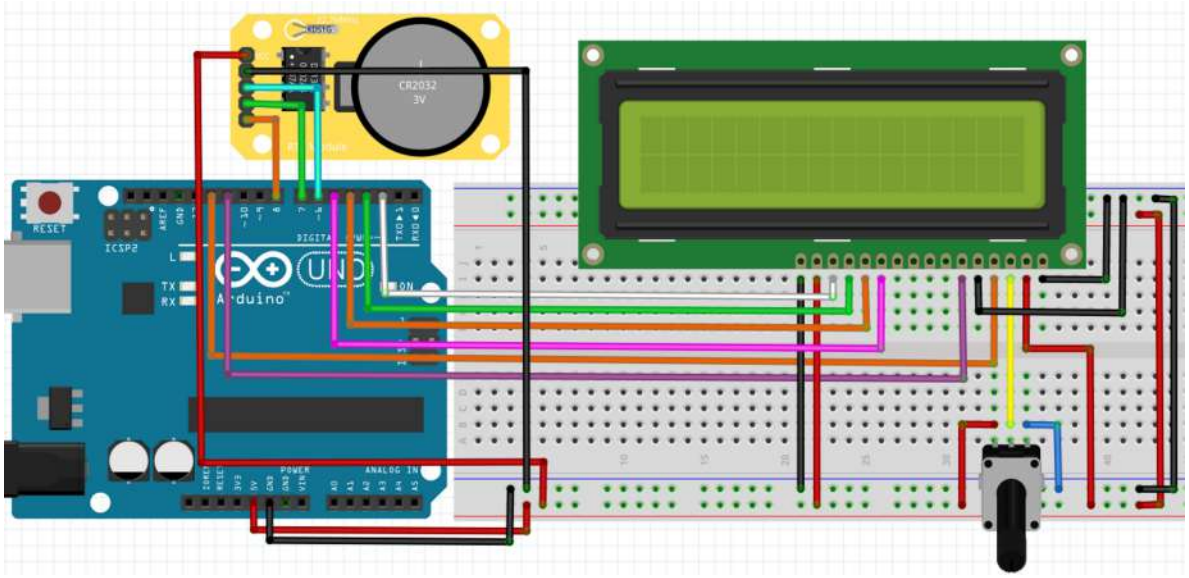
28 Aralık 2020 Pazartesi 21:22

15 Dijital Saat Yapımı

Teorik Bilgi

- Kart üzerinde CR2032 pil konnektörü bulunur.
- DS1302 entegresi saniye, dakika, saat, gün, ay ve yıl bilgisini sürekli olarak içinde tutabilen bir RTC entegresidir. Seri olarak SCLK pini üzerinden sürekli olarak çıkış verebilen karttır.
- 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.

Devre Kısmı



Kod Kısmı

- 6. pini clock, 7. pini data, 8. pini reset pini olarak tanımladık.
- 14.satırda saati saniye, dakika, saat, haftanın günü, ayın günü, ay, yıl olarak ayarlıyoruz. Bu işlem zaman hatalı ise düzeltmek için kullanıyoruz yoksa her yüklemede bu belirli tarihlerden saymaya başlar
- 19.satırda RTC'den zamanı okuyoruz
- 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.

```
#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() {
   Serial.begin(9600);
   //myRTC.setDS1302Time(10, 11, 01, 4, 31, 12, 2020);
   lcd.begin(16, 2);
}

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);

  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);
}

```

16 RFID İle Servo Kontrolü

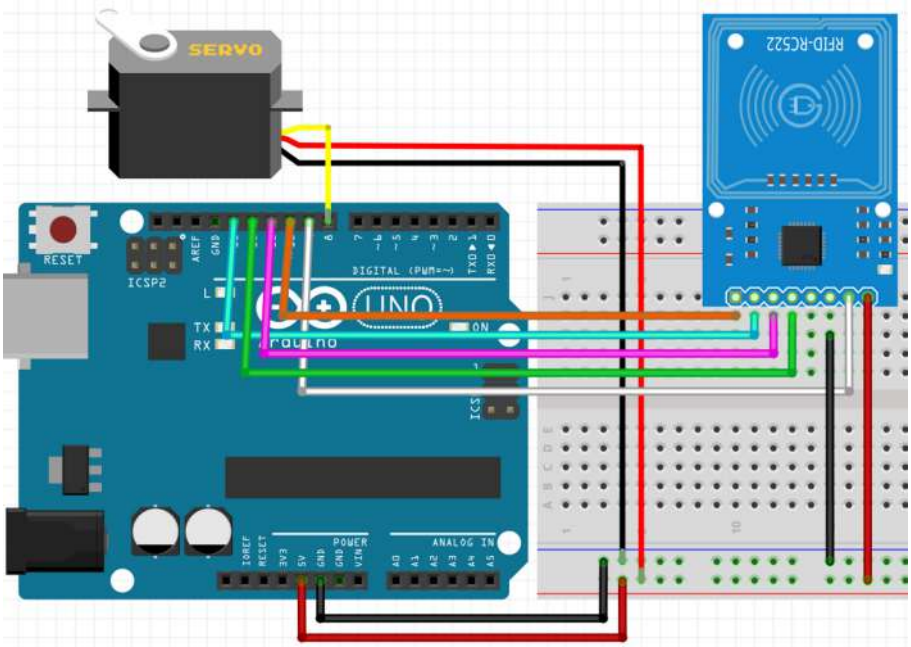
28 Aralık 2020 Pazartesi 21:26

16 RFID ile Servo Kontrolü

Teorik Bilgi

- RC522 RFID kartı, NFC frekansı olan 13,56 MHz frekansında çalışan tagler üzerinde okuma ve yazma işlemeni yapabilen, düşük güç tüketimli, ufak boyutlu bir karttır.
- 424 kbit/s haberleşme hızına sahiptir. RFID üzerinde farklı şifreleme türlerini desteklemektedir. Desteklediği kart türleri mifare1 S50, mifare1 S70 mifare ultralight, mifare pro ve mifare desfire'dir.
- 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.
- RFID'nin açılımı **Radio Frequency Identification** yani radyo frekansı ile tanımlamadır. RFID teknolojisi nesnelerin radyo dalgaları kullanılarak tanınması için kullanılan teknolojidir.
- Kullandığımız kartların kendilerine ait UID isimli bir numarası vardır. Bu numara, her kart için farklıdır. Okuyucumuza kartımızı veya anahtarlığımızı yaklaştırdığımızda bu numara okunarak işlem yapılır.

Devre Kısmı



- 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. 55(Slave Select) pini, master cihazın hangi cihaz ile haberleşeceğini belirler

Kod Kısmı

- 5.satırda RC522 modülü reset pinini, 6.satırda RC522 modülü chip select pinini tanımlıyoruz.
- 9.satırda Servo motor için değişken oluşturuyoruz.
- 10.satırda RC522 modülü ayarlarını yapıyoruz.
- 11.satırda Yetkili kart ID'sini tanımlıyoruz.
- 14.satırda Servo motor pinini motor değişkeni ile ilişkilendiriyoruz.

- 16.satırda SPI iletişimini başlatıyoruz.
- 17.satırda RC522 modülünü başlatıyoruz.
- Ve 23.satırda Yeni kartın okunmasını bekliyoruz.
- Ve 26.satırda Kart okunmadığı zaman bekliyoruz.
- 28.satırdan 31.satıra kadar Okunan kart ID'si ile ID değişkenini karşılaştırıyoruz.

```
1  #include <MFRC522.h>
2  #include <Servo.h>
3  #include <SPI.h>
4
5  int RST_PIN = 9;
6  int SS_PIN = 10;
7  int servoPin = 8;
8
9  Servo motor;
10 MFRC522 rfid(SS_PIN, RST_PIN);
11 byte ID[4] = {218, 5, 185, 37};
12
13 void setup() {
14   motor.attach(servoPin);
15   Serial.begin(9600);
16   SPI.begin();
17   rfid.PCD_Init();
18 }
19
20 void loop() {
21
22   if ( ! rfid.PICC_IsNewCardPresent())
23     return;
24
25   if ( ! rfid.PICC_ReadCardSerial())
26     return;
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(90);
35         delay(1000);
36         motor.write(0);
37         delay(500);
38     }
39     else{
40       Serial.println("Yetkisiz Kart");
41       ekranaYazdir();
42     }
43   rfid.PICC_HaltA();
44 }
45 void ekranaYazdir(){
```

```
  rfid.PICC_HaltA();
}
void ekranaYazdir(){
  Serial.print("ID Numarasi: ");
  for(int sayac = 0; sayac<4; sayac++){
    Serial.print(rfid.uid.uidByte[sayac]);
    Serial.print(" ");
  }
  Serial.println(" ");
}

```

17 ESP8266 İle Sıcaklık ve Nem Ölçümü

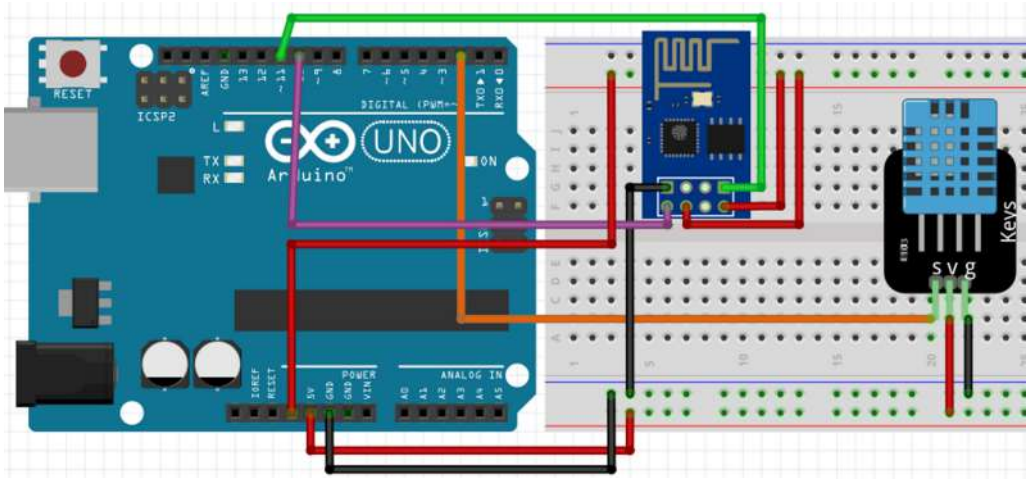
28 Aralık 2020 Pazartesi 21:26

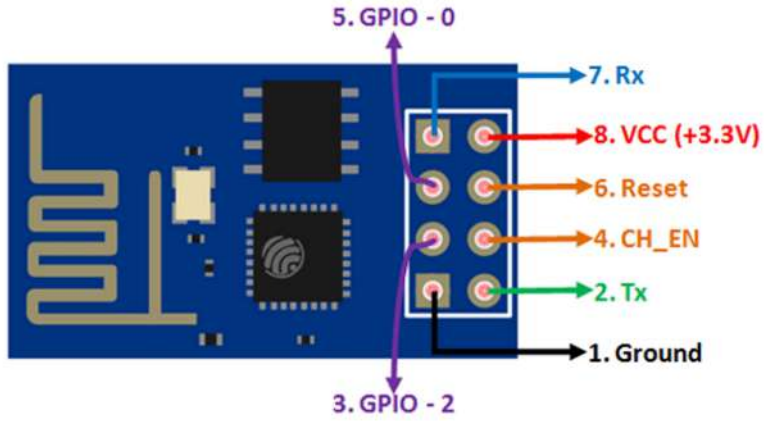
17 ESP8266 İle Sıcaklık ve Nem Ölçümü

Teorik Bilgi

- ESP8266 üzerinde dahili anten bulunmaktadır. Bu sayede ortamdaki Wifi ağına rahatlıkla bağlanabilmekte, veri paketleri alıp gönderebilmektedir.
- DHT11 sıcaklık ve nem algılayıcı kalibre edilmiş dijital sinyal çıkışı veren gelişmiş bir algılayıcı birimidir. Yüksek güvenilirliktedir ve uzun dönem çalışmalarda dengelidir. 8 bit mikroişlemci içerir, hızlı ve kaliteli tepki verir. 0 ile 50°C arasında 2°C hata payı ile sıcaklık ölçen birim, 20-90% RH arasında 5% RH hata payı ile nem ölçer.
- En sağdaki g pini GND, s pini dijital sinyal çıkış ve ortadaki v pini ise 5V gerilim pinidir.
- Thingspeak, açık kaynaklı bir loT(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.
- ESP8266, üzerinde bulunan kablosuz haberleşme devresi sayesinde ethernet protokolü ile kablosuz internete bağlanmamızı sağlar. Ethernet protokolünü basitleştirip UART protokolüne dönüştüren bir tercüman görevi görmektedir.
- DHT11, nem ve sıcaklık verilerini okumamıza yarayan bir sensördür. Sensörün içinde bir NTC(Negative Temperature Coefficient) bulunmaktadır. NTC,bir çeşitndirençtir ve ortamın Isısı 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.

Devre Kısmı



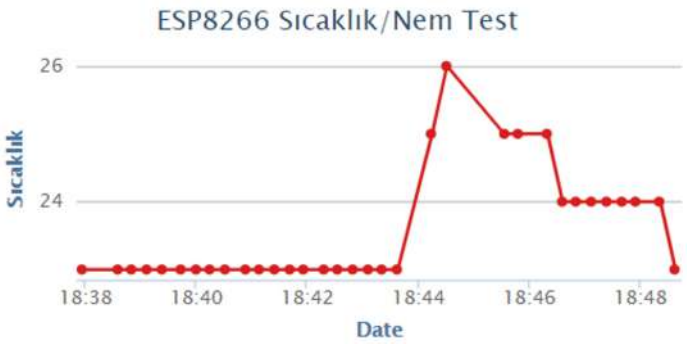


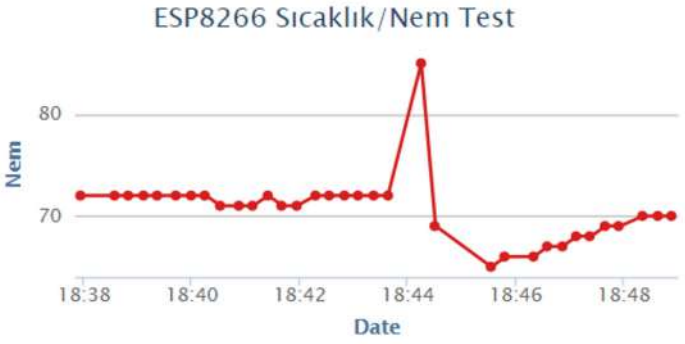
- ESP8266 modülü, Arduino ile seri haberleşmeyle iletişim kurar. Bu iletişim Arduino Uno'nun 0. ve 1. pinleriyle donanımsal olarak yapılabilmektedir ancak SoftwareSerial kütüphanesi ile diğer pinlerden yazılımsal olarak yapılabilmektedir.

- ESP8266 bazı durumlarda arduino'dan enerji aldığında eğer voltaj yetersiz geldiğinde sensörde bazı bağlantı ya da haberleşme problemleri olabilir. Bu problemleri engellemek için sensörün artı ve eksi bacakları arasına 100uF kondansatör bağlanarak voltaj dalgalanmaların önüne geçebilir.

Kod Kısmı

- 11.satırda Thingspeak ip adresini yazıyoruz.
- 15.satırda Seri haberleşme pin ayarlarını yapıyoruz.
- 21.satırda ESP8266 ile seri haberleşmeyi başlatıyoruz.
- 22.satırda AT komutu ile modül kontrolünü yapıyoruz.
- 24.satır ile Modül hazır olana kadar bekliyoruz.
- 29.satırda ESP8266 modülünü client olarak ayarlıyoruz.
- 30.satırda Ayar yapılana kadar bekliyoruz.
- 36.satır ile Ağımıza bağlanıyoruz.
- 43.satır ile Thingspeak'e bağlanıyoruz.
- 50.satırda Key kısmına kendi api keyimizi yazıyoruz.
- 52. ve 54.satırda Göndereceğimiz sıcaklık ve nem değişkeni
- 56.satırda ESP'ye göndereceğimiz veri uzunluğunu veriyoruz.
- 59.satırda ESP8266 hazır olduğunda içindeki komutlar çalışıyor.
- 66.satırda Bağlantıyı kapatıyoruz
- Verilerin internet üzerinde görüntülenmesi için Thingspeak platformunu kullanıyoruz.Thingspeak platfomuna https://www.thingspcak.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 Arduino kodumuzdaki gerekli yere yapıştırıyoruz. Kodumuzu Arduino'ya yükledikten sonra Arduino IDE üzerinden seri portu açıyoruz. Seri portta veri gönderildi yazısını gördükten sonra thingspeak üzerinden verilerimizi görüntülüyoruz. “Private View" sekmesine tıklayarak Arduino tarafından gönderilen verileri grafikler üzerinde
görüntüleyebiliyoruz.






```
1  #include <dht11.h>
2  #include <SoftwareSerial.h>
3
4  String agAdi = "BAYBARS-2019";
5  String agSifresi = "        ";
6
7  int rxPin = 10;
8  int txPin = 11;
9  int dht11Pin = 2;
10
11 String ip = "184.106.153.149";
12 float sicaklik, nem;
13
14 dht11 DHT11;
15 SoftwareSerial esp(rxPin, txPin);
```

```
13
14 dht11 DHT11;
15 SoftwareSerial esp(rxPin, txPin);
16
17 void setup() {
18
19   Serial.begin(9600);
20   Serial.println("Started");
21   esp.begin(115200);
22   esp.println("AT");
23   Serial.println("AT Yollandı");
24   while(!esp.find("OK")){
25     esp.println("AT");
26     Serial.println("ESP8266 Bulunamadı.");
27   }
28   Serial.println("OK Komutu Alındı");
29   esp.println("AT+CWMODE=1");
30   while(!esp.find("OK")){
31     esp.println("AT+CWMODE=1");
32     Serial.println("Ayar Yapılıyor....");
33   }
34   Serial.println("Client olarak ayarlandı");
35   Serial.println("Aga Baglaniliyor...");
36   esp.println("AT+CWJAP=\""+agAdi+"\",\""+agSifresi+"\"");
37   while(!esp.find("OK"));
38   Serial.println("Aga Baglandi.");
39   delay(1000);
40 }
41
42 void loop() {
43   esp.println("AT+CIPSTART=\"TCP\",\""+ip+"\",80");
44   if(esp.find("Error")){
45     Serial.println("AT+CIPSTART Error");
46   }
47   DHT11.read(dht11Pin);
48   sicaklik = (float)DHT11.temperature;
49   nem = (float)DHT11.humidity;
50   String veri = "GET https://api.thingspeak.com/update?api_key=UPWBDRGHSOZ34QHP";
51   veri += "&field1=";
52   veri += String(sicaklik);
53   veri += "&field2=";
54   veri += String(nem);
55   veri += "\r\n\r\n";
56   esp.print("AT+CIPSEND=");
57   esp.println(veri.length()+2);
58   delay(2000);
59   if(esp.find(">")){
60     esp.print(veri);
61     Serial.println(veri);
62     Serial.println("Veri gonderildi.");
63     delay(1000);
64   }
65   Serial.println("Baglantı Kapatildi.");
66   esp.println("AT+CIPCLOSE");
67   delay(1000);
68 }
```

18 ESP8266 İle Step Motor Kontrolü

28 Aralık 2020 Pazartesi 21:26

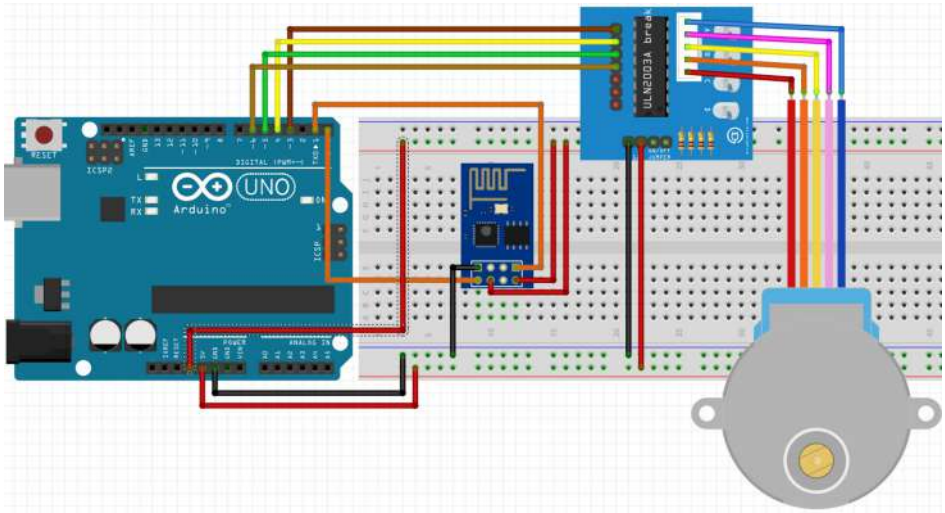
18 ESP8266 İle Step Motor Kontrolü

Teorik Bilgi

- Kart üzerinde 4 led ve 4 kontrol pini bulunmaktadır. Bu pinler sayesinde step motorun sürülmesini, Arduino veya herhangi bir mikrodenetleyici ile gerçekleştirebilirsiniz.
- Step motorlar, elektrik enerjisini dönme hareketi ile fiziksel enerjiye çeviren elektromekanik aygıtlardır. Adım adım hareket eden motorlardır. İçinde bulunan bobinlerin sırayla açılıp kapanması ile hareketlerini sürdürürler. 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.

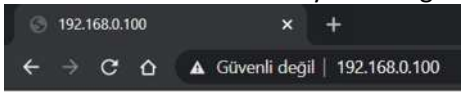
ULN2003A Step Motor Sürücü Kartı

Devre Kısmı



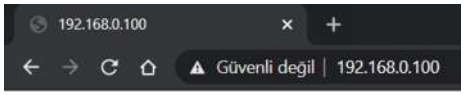
Kod Kısmı

- 17.satırda ESP8266'yı resetliyoruz.
- 30.satırda Server oluşturuyoruz ve 80. porttan dinlemeye başlıyoruz.
- 70. ve 81.satırlar arası ESP'den gelen dönütleri okur.
- 83. ve 90.satırlar arası Seri haberleşmede kullanmadığımız byteları temizler.
- 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.
- Arduino IDE üzerinden seri portumuzu açarak veri hızını 115200 olarak ayarlıyoruz. ESP8266 modülü ile iletişimi buradan görebiliriz. Ayarlarlar yapıldığında seri porttaki IP adresine bir web tarayıcı ile bağlanıyoruz.



Step Motor Kontrol

İleri



Step Motor Kontrol

Ileri

Geri

```
String agAdi = "BAYBARS-2019";
String agSifresi = "         ";

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("OK"))
  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);
}

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 < 50; adim++){
          stepIleri(50);
        }
      }
      else if(komut.indexOf("step=geri") != -1){
        for(int adim = 0; adim < 50; adim++){
          stepGeri(50);
        }
      }
      String sayfa = "<h1>Step Motor Kontrol</h1><br>";
      sayfa+="<br><a href=\"?step=ileri\"><button><h1>Ileri</h1></button></a>";
      sayfa+="<br><br><a href=\"?step=geri\"><button><h1>Geri</h1></button></a>";
      komut = "AT+CIPSEND=";
      komut += connectionId;
      komut += ",";
```

```
      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);
    }
  }
}

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();
    }
  }
}

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);
}

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

```
  digitalWrite(motorPin4, HIGH);
  delay(beklemeSuresi);
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, LOW);
  digitalWrite(motorPin3, HIGH);
  digitalWrite(motorPin4, LOW);
  delay(beklemeSuresi);
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, HIGH);
  digitalWrite(motorPin3, LOW);
  digitalWrite(motorPin4, LOW);
  delay(beklemeSuresi);
  digitalWrite(motorPin1, HIGH);
  digitalWrite(motorPin2, LOW);
  digitalWrite(motorPin3, LOW);
  digitalWrite(motorPin4, LOW);
  delay(beklemeSuresi);
}
```