T.C.

ERCİYES ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

ELEKTRONİK SİSTEMLER LABORATUVARI

ARDUINO UYGULAMALARI

DENEY SORUMLUSU

Arş. Gör. Burak ULU

ŞUBAT 2023

KAYSERİ

ARDUINO DİJİTAL GİRİŞ-ÇIKIŞ KONTROLÜ

1. GİRİŞ

Arduino, açık kaynaklı bir geliştirme platformudur. Arduino kartları üzerinde Atmega firmasının 8 ve 32 bit mikrodenetleyicileri bulunur. Arduino kütüphaneleri ile mikrodenetleyicileri kolaylıkla programlayabilirsiniz. Klasik bir mikrodenetleyiciden farkı ise kolay kullanım imkanı sağlamasıdır. Ama tabi ki buna paralel olarak sağladığı kolaylığa karşı üzerindeki mikrodenetleyiciyi tam performansda kullanamamak gibi eksileri de mevcuttur.

Bu deneylerde, daha önceden edinilen programlama bilgisinin pekiştirilmesi ve gömülü sistemlere giriş yapılması amaçlanmıştır. Arduino gibi bir geliştirme platformunu kullanabilme becerisi kazanmak, çeşitli yelpazede proje yapılabilmesine katkı sağlayacaktır.

2. DENEYİN AMACI

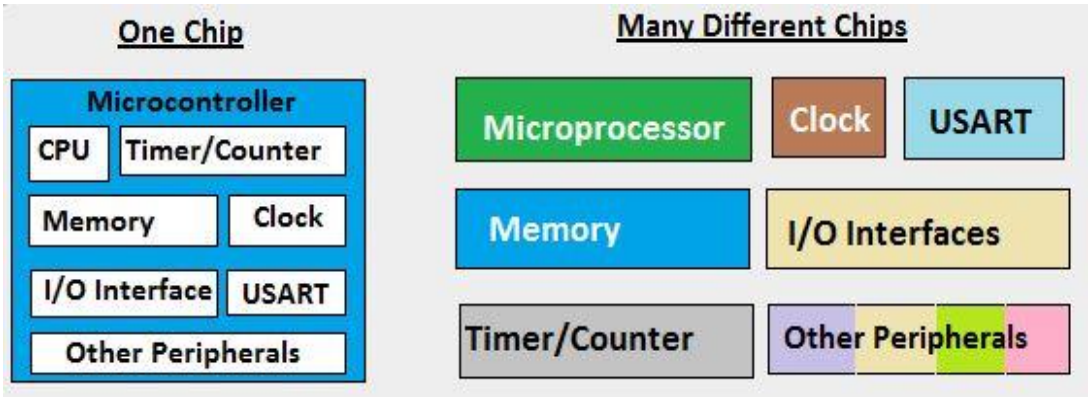
Arduino dijital giriş-çıkışlarının kontrol edilebilmesi ve motor sürücü devresi kurularak Arduino üzerinden motor kontrolünün gerçekleştirilmesi.

3. ÖN BİLGİ

3.1. Gömülü Sistemler

Belirli bir işi gerçekleştirmek için tasarlanmış bir takım bilgisayar donanım ve yazılım kombinasyonları gömülü sistemler olarak adlandırılmaktadır. Bilinen genel amaçlı bilgisayarlardan farklı olarak, kişisel bilgisayar gibi, gömülü bir sistem kendisi için önceden özel olarak tanımlanmış görevleri yerine getirir. Sistem belirli bir amaca yönelik olduğu için tasarımların her geçen gün boyutu ve maliyeti düşmektedir.

Gömülü bir sistemin çekirdeğini, belirli bir sayıda görevi yerine getirmek için programlanan mikroişlemciler ya da mikrodenetleyiciler oluşturmaktadır. Kullanıcıların üzerinde istediği yazılımları çalıştırabildiği genel maksatlı bilgisayarlardan farklı olarak, gömülü sistemlerdeki yazılımlar yarı kalıcıdırlar ve firmware ismiyle anılırlar.



Şekil 3.1. Mikrodenetleyici ve Mikroişlemci yapıları.

Gömülü sistemlere örnekler:

-atm cihazları (automatic teller machines)

-cep telefonları

-network ekipmanları (router, firewall vs.)

-bilgisayar yazıcıları

-disk sürücüleri

-motor denetleyicileri ve abs sistemleri

-ev otomasyon ürünleri (termostat, klima, güvenlik sistemleri)

-elektronik ev eşyaları (mikrodalga fırın, çamaşır makinesi, tv, dvd player)

-savunma sistemleri, uçaklardaki ve füzelerdeki uçuş kontrol sistemleri

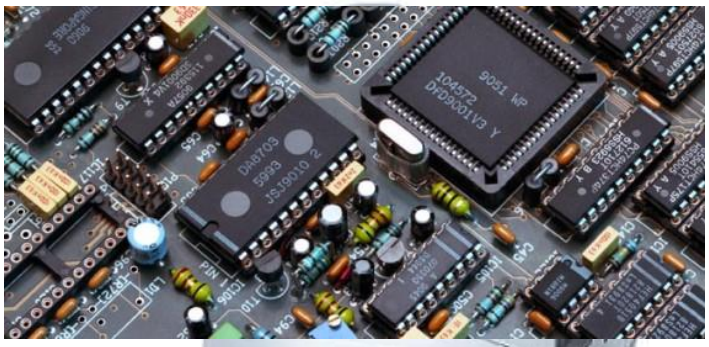
-medikal ekipmanlar

-ölçüm sistemleri (osiloskop, spektrum anizörü, enerji analizörü)

-kişisel sayısal asistanlar (pda)

-endüstriyel otomasyon ve izleme sistemleri

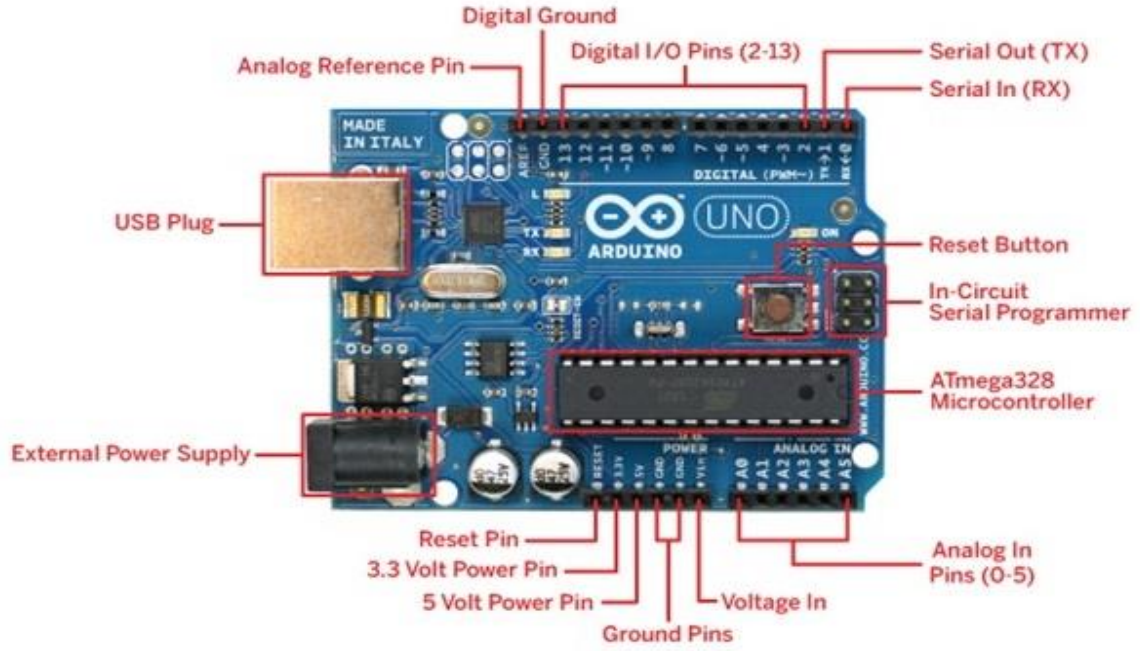
-oyun konsolları



Şekil 3.1. Gömülü sistem örnekleri.

3.2. Arduino Uno R3

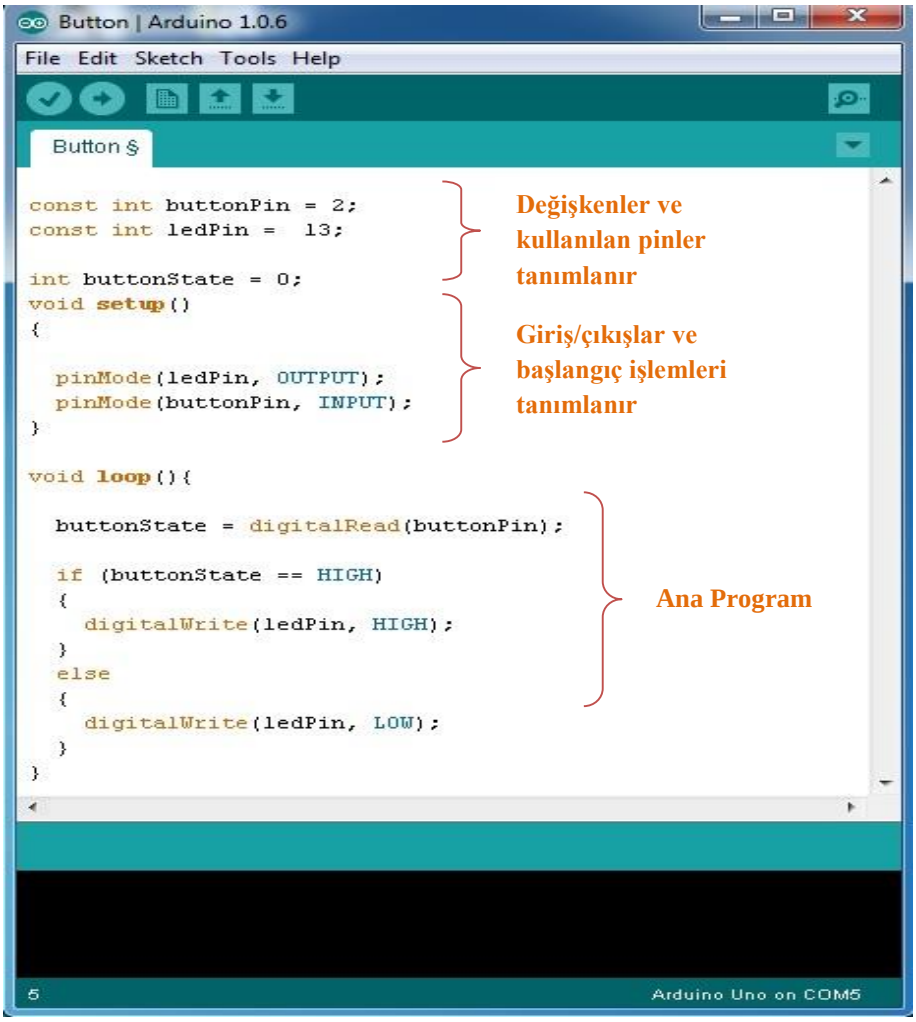
Donanım: Arduino Uno, üzerinde Atmega328 mikrodenetleyicisi bulunan bir geliştirme kartıdır.



Şekil 3.2. Arduino Uno üzerinde birimlerin görünüşü.

- 14 tane giriş/çıkış pini (6' sı pwm için)
- 6 analog giriş
- 16 Mhz seramik rezonatör
- USB bağlantı
- Power jack
- ICSP Header
- Reset buton mevcuttur.
- Çalışma gerilimi 5V

Yazılım: Geliştirme ortamı ile Arduino programları yazılıp, derlenip kart içerisine yüklenebilmektedir. Arduino programlamada C/C++/Java temelli bir dil kullanılmaktadır. Kütüphaneler sayesinde donanım seviyesine inmeye gerek kalmamaktadır. Zaten Arduino platformunun tercih edilmesindeki en büyük sebeplerden bir tanesi de geniş kütüphane desteği sağlamasıdır. Arduino web sitesinde programlama, donanımlar ve kütüphaneler hakkında bilgiler yer almaktadır. Ayrıca internet üzerinde oldukça canlı bir Arduino topluluğu bulunmakta ve aklınıza gelebilecek hemen her konuda yapılmış proje bulunmaktadır.



Şekil 3.3. Arduino program arayüzü görünüşü.

Önemli Yapılar

void setup(): setup() fonksiyonu, program(sketch) başladığında çağırılır. Başlangıç değerleri, pin durum tanımlamaları, kullanılan kütüphanelerin başlatılması gibi işlemler bu fonksiyon içinde gerçekleştirilir. setup() fonksiyonu dışarıdan herhangi bir müdahale olmadığı sürece programda bir kez çalışır.

void loop(): Arduino' yu kontrol eden ana program bu döngü içerisine yazılır ve dışarıdan müdahale olmadığı sürece sürekli olarak çalışır.

if: If karşılaştırma operatörünün Arduino' da kullanımına aşağıda örnek verilmiştir.

```
if (x > 120) digitalWrite(LEDpin, HIGH);

if (x == 120 && y <=100)
digitalWrite(LEDpin, HIGH);

if (x != 120){ digitalWrite(LEDpin, HIGH); }
```

```
if (x < 120 ||  ){
  digitalWrite(LEDpin1, HIGH);
  digitalWrite(LEDpin2, HIGH);
}
```

for döngüsü**:** Temel olarak bir kod bloğunu belirli bir sayıda ve üst üste çalıştırmak için kullanılan döngüdür.

for (başlangıç ; koşul ; artış yada azalış)

while döngüsü**:** İçerisindeki şart doğru olduğu sürece devam eder. Dolayısıyla döngü içerisindeki şartı değiştiren bir ifade bulunmadığı takdirde sonsuz döngüye girer. İstenilen şart sağlanmadığı zaman döngüden çıkılır.

while(şart){…

…}

do.. while döngüsü**:** do.. while döngüsünün while döngüsünden farkı şartın çevrim sonucunda kontrol edilmesidir.

do{

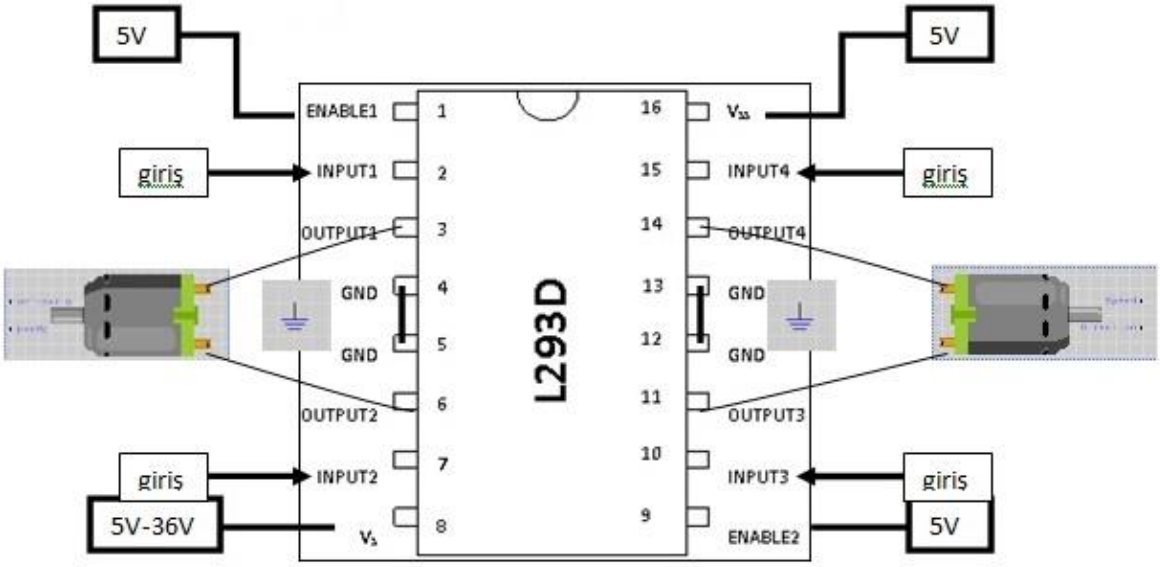
}while(şart);

#include: Önişlemci direktifleri # işareti ile başlar ve program derlenmeden önce C önişlemcisi tarafından işletilir. #include direktifi program içerisinde kullanılan fonksiyonlar için gerekli kodları programa dahil etmek için kullanılır.

#define: #define komutu bir sembol tanımlamak için kullanılır.

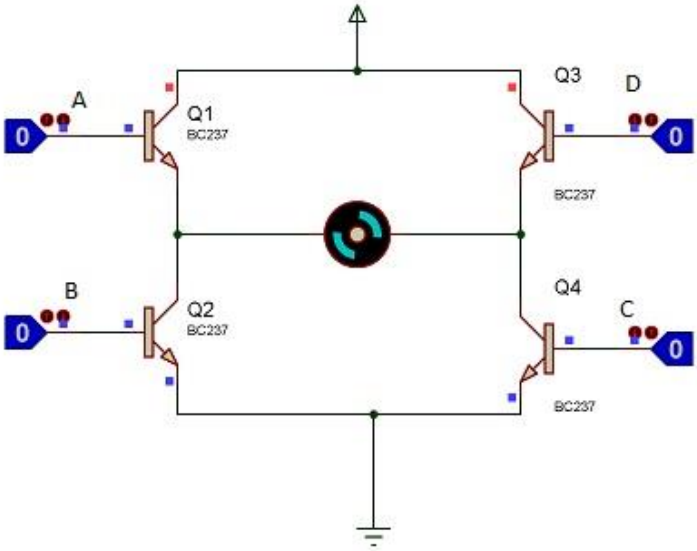
3.3. Motor Sürücü Devresi

L293D: L293D motor sürücü entegreleri içerisinde en basit yapılı entegredir diyebiliriz. 16 bacaklı kılıf yapısındadır ve 2 motoru birbirlerinden ayrı şekilde ileri- geri kontrol edebilir. Enable bacağı da kullanılarak motor PWM ile de sürülebilir. 4.5V – 36V aralığında motorlar L293D ile kontrol edebilmektedir. L293D' nin maksimum akım sınırı 600mA' dir. L293B' nin akım sınırı ise 1A' dir. H Köprü mantığıyla çalışır.



Şekil 3.4. L293D bağlantı şeması.

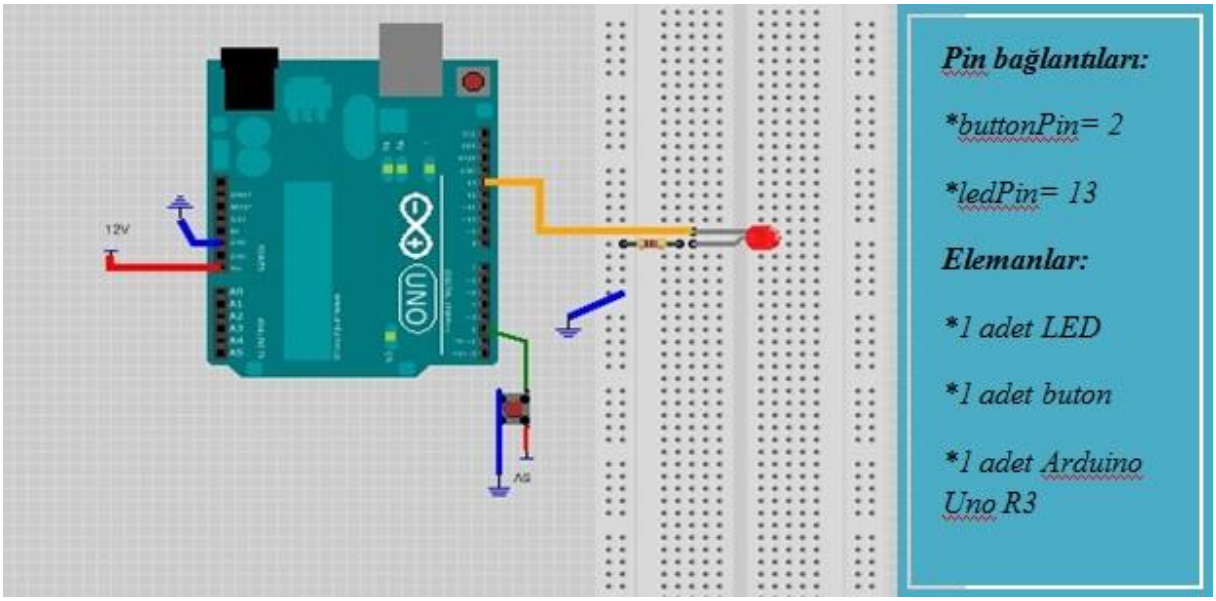
H Köprü akım yönünü dijital olarak kontrol edebileceğiniz bir yapıdır. Bu yapıda transistor, tristör ve triyak gibi elemanlar kullanıldığı için aynı zamanda düşük akımlarla yüksek akımlar kontrol edilebilmektedir.



Şekil 3.5. H Köprü devre şeması

4. DENEYİN YAPILIŞI

4.1. Arduino İle Led Yakma



Şekil 4.1. Arduino led yakma uygulama devresi.

Deneyin Yapılısı:

1- Şekil 4.1' te görülen bağlantıları Arduino ve board üzerinde gerçekleştiriniz.
2- Arduino Vin pininden 12v besleme ile Arduino' ya güç veriyoruz.
3- Giriş/çıkış pinlerinin de bağlantılarını yapıp program aşamasına geçiniz.

Program:

```
const int buttonPin = 2;
const int ledPin =  13;

int buttonState = 0;
```

// Butonun Arduino ile bağlantısı 2 nolu pin ile gerçekleştirildi ve bu pine buttonPin ismi verildi.
// LED' in Arduino ile bağlantısı 13 nolu pin ile gerçekleştirildi ve bu pine ledPin ismi verildi.
// Butonun değerini kontol eden buttonState adında bir değişken atandı.

```
void setup()
{

  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}
```

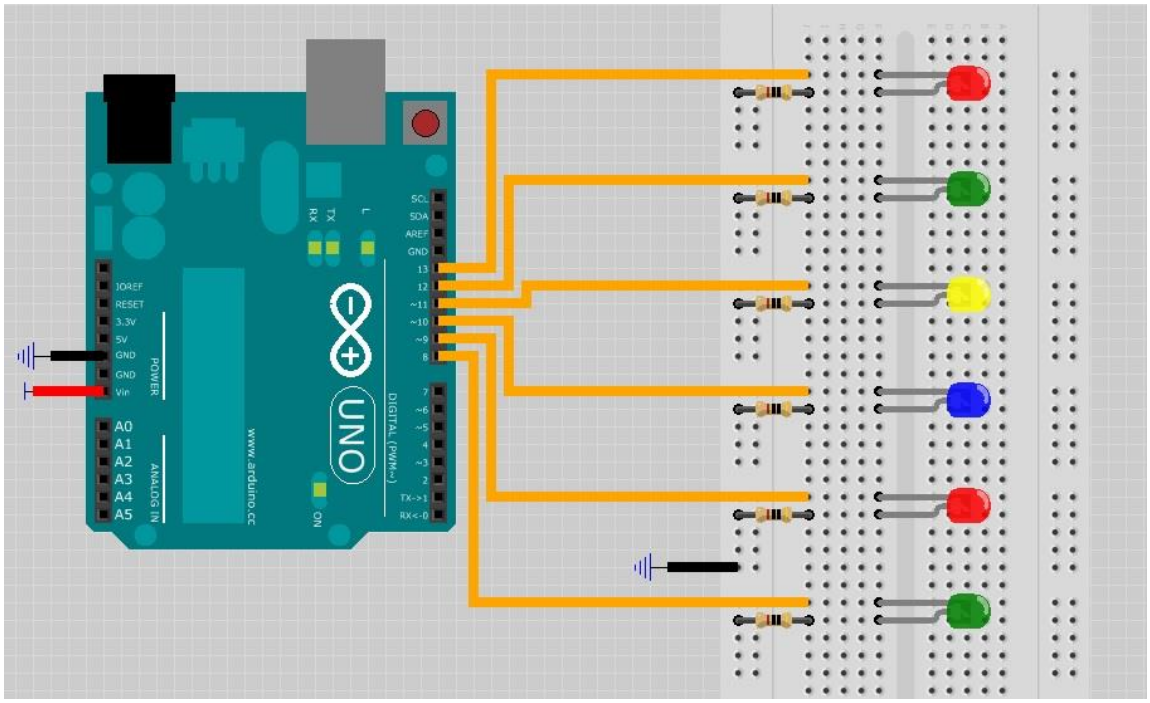
// ledPin pini çıkış olarak tanımlandı .
// buttonPin pini giriş olarak tanımlandı.

```
void loop(){

  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }
}
```

//buttonPin' den okunan değeri buttonState değişkenine yaz .
//buttonState değişkeni HIGH değeri aldığında, yani buton aktif olduğunda, aşağıdaki komutu işle.
// ledPin pinini HIGH yap, yani LED' i yak.

//aksi halde, yani buttonState değişkeni HIGH değil ise ledPin pinini LOW yap, yani LED' i söndür.

4.2. Arduino ile Kara Şimşek Uygulaması



Şekil 4.2. Kara şimşek uygulama devresi.

Deneyin Yapılısı:
1- Şekil 4.2' te görülen bağlantıları Arduino ve board üzerinde gerçekleştiriniz.
2- Arduino Vin pininden 12v besleme ile Arduino' ya güç veriyoruz.
3- Giriş/çıkış pinlerinin de bağlantılarını yapıp program aşamasına geçiniz.

Program:

```
const int ledPinleri[]={8, 9, 10, 11, 12, 13};
const int beklemeSuresi=50;
```

// 6 adet LED' in pin numaraları dizi oluşturularak atandı.
// beklemeSuresi sabitine 50 değeri atandı. Bu değer, LED'lerin yanıp sönmesi esnasındaki gecikmeler için kullanıldı.

```
void setup(){
  for (int led=0; led < 6; led++)
  {
    pinMode(ledPinleri[led], OUTPUT);
  }

}
```

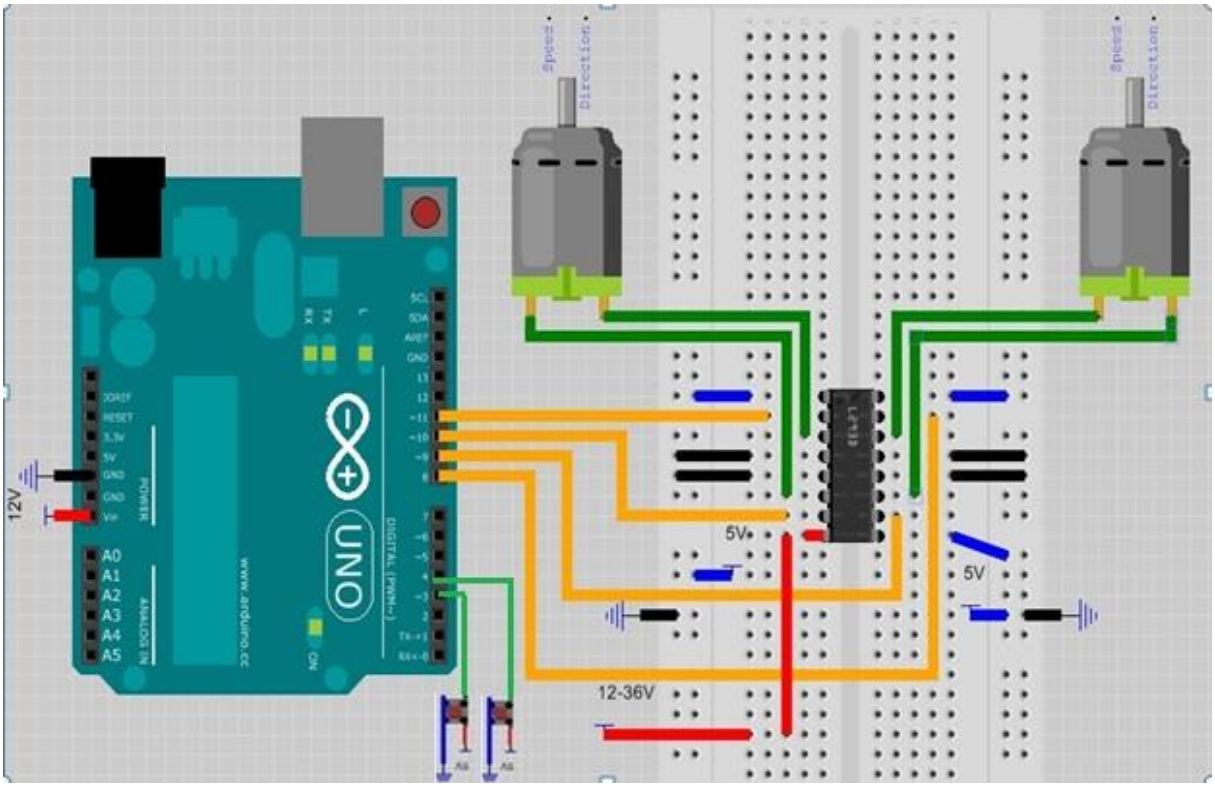
// Bir led değişkeni tanımlandı ve for döngüsü içinde 0' dan 6' ya kadar birer artırılarak döngü içerinde kullanıldı.
// led değişkeni for döngüsünün her bir adımında dizinin bir elemanını çıkış yaparak, LED için kullanılan tüm pinlerin çıkış olarak tanımlanmasında kullanıldı.

```
void loop() {
  for (int led=0; led < 5; led++)

  {
  digitalWrite(ledPinleri[led],HIGH);
  delay(beklemeSuresi);
  digitalWrite(ledPinleri[led+1],HIGH);
  delay(beklemeSuresi);
  digitalWrite(ledPinleri[led],LOW);
  delay(beklemeSuresi*2);
  }


  for (int led=5; led >0; led--)

  {

  digitalWrite(ledPinleri[led],HIGH);
  delay(beklemeSuresi);
  digitalWrite(ledPinleri[led-1],HIGH);
  delay(beklemeSuresi);
  digitalWrite(ledPinleri[led],LOW);
  delay(beklemeSuresi*2);
  }
}
```

//led değişkeni for döngüsünün her bir adımında ilk LED' den son LED' e kadar ileri yönde LED 'lerin yanmasını sağlar.
//Sırası gelen LED yanar ve 50ms beklenir, daha sonra bir sonraki yanar ve bir 50ms daha beklendikten sonra önceki LED söner. Bu döngü son LED yanıp bir önceki sönene kadar devam eder.

//Bu döngüde aynı işlem geri yönlü olarak tekrarlanır. Bu döngü ilk LED yanıp bir önceki sönene kadar devam eder.

4.3. Arduino ile DC Motor Kontrol



Şekil 4.3. DC motor uygulama devresi.

Deneyin Yapılısı:

1- Şekil 4.3' te görülen bağlantıları Arduino ve board üzerinde gerçekleştiriniz.

2- Arduino Vin pininden 12v besleme ile Arduino' ya güç veriyoruz.

3- Giriş/çıkış pinlerinin de bağlantılarını yapıp program aşamasına geçiniz.

Program:

```
int  sag_motor1 = 8;
int  sag_motor2 = 9;
int  sol_motor1 = 10;
int  sol_motor2 = 11;
int  sag_kontrol = 3;
int  sol_kontrol = 4;

void setup() {

  pinMode(sag_motor1, OUTPUT);
  pinMode(sag_motor2, OUTPUT);
  pinMode(sol_motor1, OUTPUT);
  pinMode(sol_motor2, OUTPUT);
  pinMode(sag_kontrol, INPUT);
  pinMode(sol_kontrol, INPUT);

}
```

// İlk bölümde değişkenler tanımlandı ve pin numaraları bağlantı şemasına göre belirlendi. Burada sag_motor1 ve sag_motor2 sağdaki motorun + ve – uçlarına bağlı pinleri temsil ederken diğer 2 değişken sol motorun + ve – uçlarına bağlı pinleri temsil ediyor. sag_kontrol, sol_kontrol değişkenlerinin atandığı pinlere ise butonlar bağlanmıştır.

//setup() fonksiyonunun içerisinde ilgili pinler giriş/çıkış olarak tanımlandı.

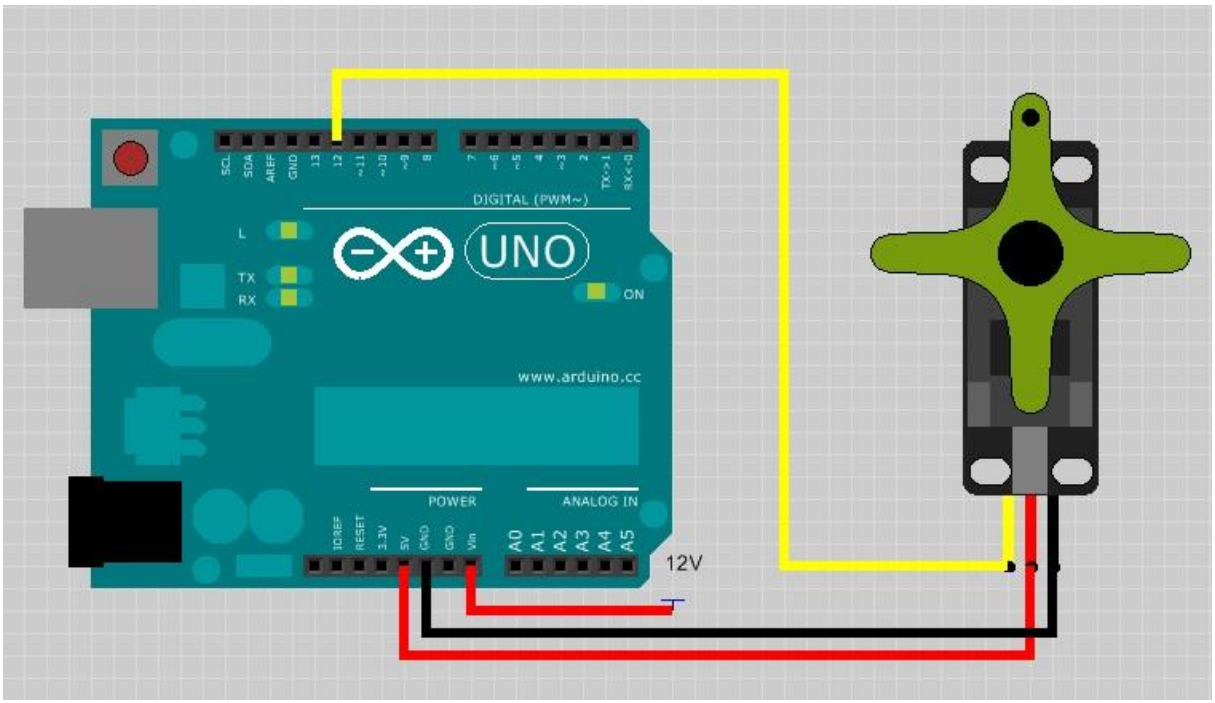
```
void loop() {

   if(sag_kontrol==HIGH)
   {
      digitalWrite(sag_motor1, HIGH);
      digitalWrite(sag_motor2, LOW);
      digitalWrite(sol_motor1, LOW);
      digitalWrite(sol_motor2, LOW);
   }
   if(sol_kontrol==HIGH)
   {
      digitalWrite(sag_motor1, LOW);
      digitalWrite(sag_motor2, LOW);
      digitalWrite(sol_motor1, HIGH);
      digitalWrite(sol_motor2, LOW);
   }
  else
   {
      digitalWrite(sag_motor1, LOW);
      digitalWrite(sag_motor2, LOW);
      digitalWrite(sol_motor1, LOW);
      digitalWrite(sol_motor2, LOW);
   }
  }
```

//Sağ motoru kontrol eden buton aktif ise sağ motoru çalıştır.

//Sol motoru kontrol eden buton aktif ise sol motoru çalıştır.

//aksi durumlarda 2 motora da gerilim uygulama yani motorları durdur.

4.4. Arduino ile Servo Motor Kontrol



Şekil 4.4. Servo motor uygulama devresi.

Deneyin Yapılısı:

1- Şekil 4.4’ te görülen bağlantıları Arduino ve board üzerinde gerçekleştiriniz.

2- Arduino Vin pininden 12v besleme ile Arduino’ ya güç veriyoruz.

3- Giriş/çıkış pinlerinin de bağlantılarını yapıp program aşamasına geçiniz.

Program:

```
#include <Servo.h>
Servo myservo;

int pos = 0;

void setup()
{
myservo.attach(9);
}
```

```
void loop()
{
   for(pos = 0; pos < 180; pos += 1)
   {
    myservo.write(pos);
    delay(5);
   }
   for(pos = 180; pos>=1; pos-=1)
   {
    myservo.write(pos);
    delay(5);
   }
}
```

1. DENEYİN AMACI

Algılayıcılar hiç şüphesiz mekatronik bir sistem için olmazsa olmaz elemanlardır. Temel olarak analog ve dijital olmak üzere 2 gruba ayrılırlar. Mikroişlemci ya da mikrodenetleyici gibi komut işleme yeteneğine sahip entegrelerin işleyebilecekleri komutlardan oluşan makine dili dijital '1' ve '0' lardan oluşur. Bu sebeple dijital sensörlerden elde edilen çıkışlar direkt olarak işlenebilmekte fakat analog sensörlerden elde edilen verilerin mi

2. ÖN BİLGİ

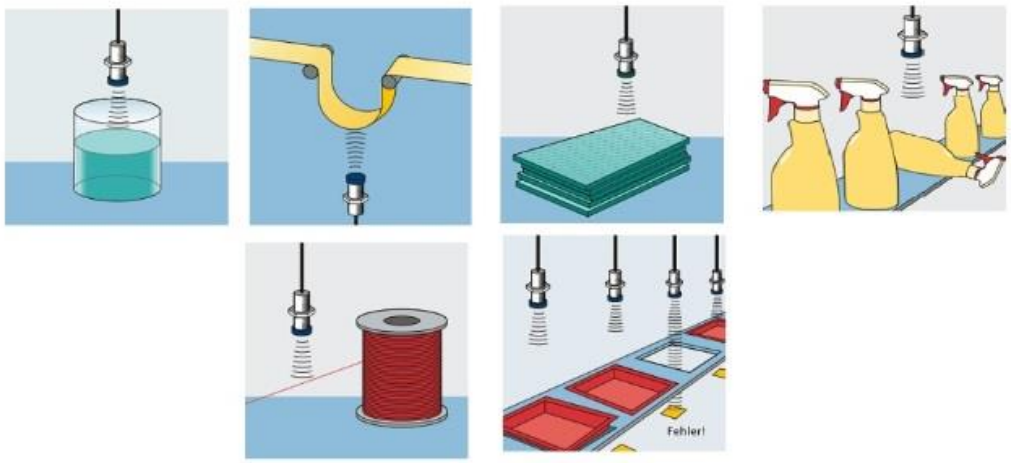
2.1. HC-SR04 Ultrasonik Sensör

Ses dalgaları sınıflandırılmasında 20Khz-1Ghz aralığındaki ses sinyalleri ultrasonik ses olarak tanımlanmıştır. Bizim sensörümüz ve bir çok ultrasonik sensör 40Khz frekansında ultrasonik ses üretmektedir. Burada önemli olan sesin yüksekliğinde belirleyici olan etken frekanstır. Ses yüksekse frekansta yüksektir. Ultrasonik ses sinyallerini insan kulağı algılayamaz.

Şekil 2.1. Ultrasonik sensörün çalışma mantığı

Ultrasonik darbe t=0 zamanında transdüser tarafından iletiliyor. X pozisyonundaki hedef tarafından yansıtıldıktan sonra t= tx zamanında darbe alınıyor. tx , X mesafesi ile orantılıdır. t=0 zamanında darbe iletilir (ultrasonik ses sinyali), cisimden yansır, transdüser tarafından algılanır ve tekrar gönderilir. Sonraki darbe ilk darbenin ultrasonik enerjisinin hepsi absorbe edildiğinde iletilmelidir. Bu yüzden sensöre bir pals gönderilir sensör okunur ve sensörün datasheetinde yazan süre kadar sensöre tekrar pals gönderilmez. Eğer bekleme yapmaksak sensör saçma değerler döndürür. Çünkü ilk yolladığımız sinyal bir yerden yansıyarak sensöre geri dönmeye devam eder.

Tüm katı ve sıvı cisimler ultrasonik dalgayı cok iyi oranda yansıtırlar. Hem katı hemde sıvı cisimlerden ultrasonik enerjinin %99u yansıtılır. Çok ufak oranlardaki enerji miktarı cisim tarafından emilir. Bundan dolayı sensörü çok çeşitli uygulamalarda sorunsuz kullanabilmemiz mümkündür. Ayrıca endüstriyel tipteki ultrasonik sensörler bir çok otomasyon sistemlerinde kullanılmaktadır.



Şekil 2.2. Ultrasonik sensörün endüstride kullanımına bazı örnekler

Şekil 2.1. HC-SR04 ultrasonik sensör.

HC-SR04 Sensörü dijital bir ultrasonik sensördür ve üzerinde 4 adet pin mevcuttur. Bunlar; VCC, GND, Trig, Echo pinleridir.
Çalışma mantığını da kısaca şöyle özetleyebiliriz;

Ses sinyallerinin boşlukta yayılma hızı 340 m/sn' dir. Sensörden ilk olarak bir ses dalgası gönderilir, yansıyan ses dalgası transducer tarafından algılanır ve tekrar gönderilir ancak gönderilmeden önce biraz beklenir. Çünkü bir önceki ses dalgasının cisim yada ortam tarafından tam anlamıyla emilmiş olması gerekir. Yansıyan sinyal Echo bacağından okunur ve bu sayede mesafe ölçümü yapılabilir. Önemli olan nokta Echo pininin ne kadar süre lojik 1 de kaldığıdır. Ölçülmesi gereken değer budur. Çünkü mesafenin bulunması için öncelikle

zamanın hesaplanması gerekir. Ses dalgası 340 m/sn de hareket ediyorsa ve biz ölçüm aralığını 1 us de yapıyorsak sesin aldığı yol = zaman x (34000 cm /1000000) = 1/29, yani zaman/29 olması gerekir. Ancak sinyalin gidiş ve gelişi göz ardı edilmemelidir. O halde bulduğumuz değeri 2' ye bölüp aradaki mesafeyi bulabiliriz; mesafe = zaman / (29*2) =zaman/58 olur.

2.2. NTC Sıcaklık Sensörü

Önemli Yapılar

void setup(): setup() fonksiyonu, program(sketch) başladığında çağırılır. Başlangıç değerleri, pin durum tanımlamaları, kullanılan kütüphanelerin başlatılması gibi işlemler bu fonksiyon içinde gerçekleştirilir. setup() fonksiyonu dışarıdan herhangi bir müdahale olmadığı sürece programda bir kez çalışır.
void loop(): Arduino' yu kontrol eden ana program bu döngü içerisine yazılır ve dışarıdan müdahale olmadığı sürece sürekli olarak çalışır.
if: If karşılaştırma operatörünün Arduino' da kullanımına aşağıda örnek verilmiştir.

```
if (x > 120) digitalWrite(LEDpin, HIGH);

if (x == 120 && y <=100)
digitalWrite(LEDpin, HIGH);

if (x != 120){ digitalWrite(LEDpin, HIGH); }

if (x < 120 ||  ){
  digitalWrite(LEDpin1, HIGH);
  digitalWrite(LEDpin2, HIGH);
}
```

for: Temel olarak bir kod bloğunu belirli bir sayıda ve üst üste çalıştırmak için kullanılan döngüdür.

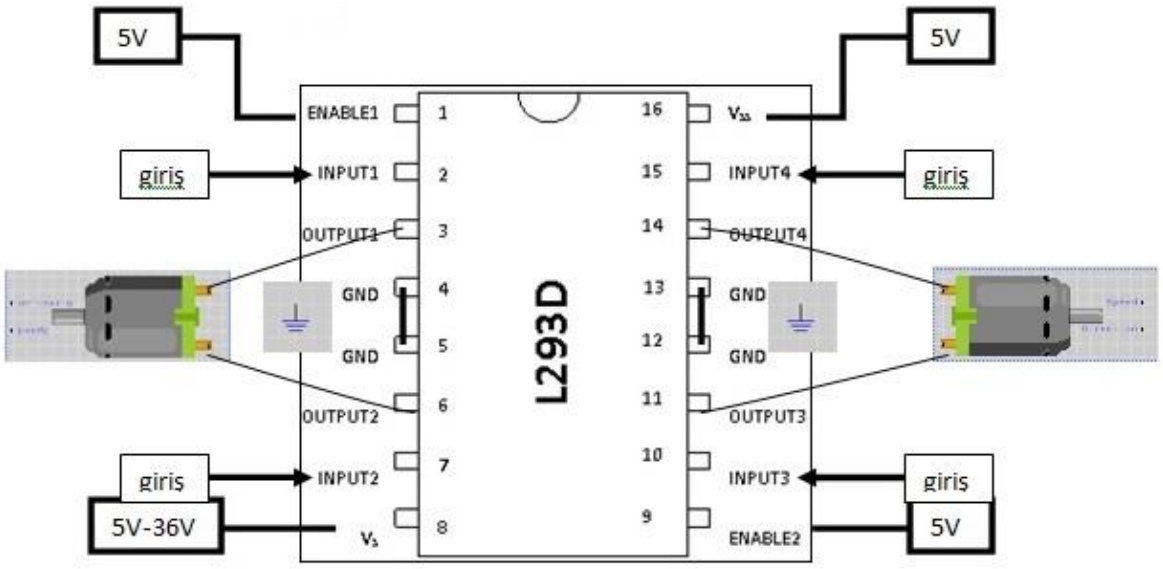
```
for ( başlangıç ; koşul ; artış yada azalış)
```

#include: Önişlemci direktifleri # işareti ile başlar ve program derlenmeden önce C önişlemcisi tarafından işletilir. #include direktifi program içerisinde kullanılan fonksiyonlar için gerekli kodları programa dahil etmek için kullanılır.

#define: #define komutu bir sembol tanımlamak için kullanılır.

2.2. Motor Sürücü Devresi

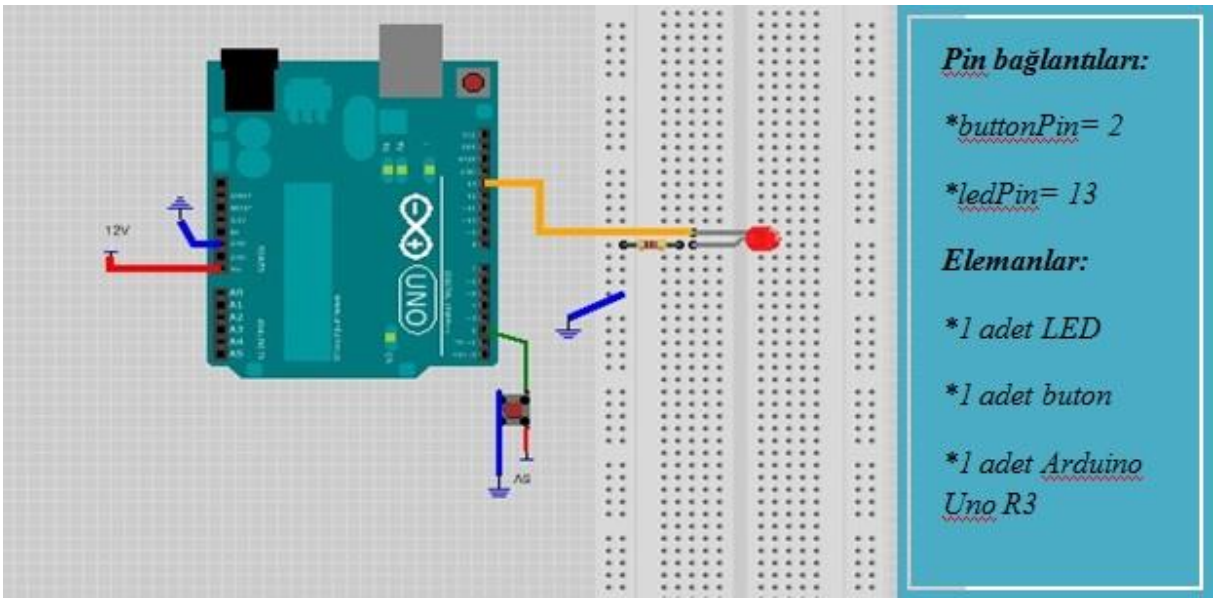
L293D: L293D motor sürücü entegreleri içerisinde en basit yapılı entegredir diyebiliriz. 16 bacaklı kılıf yapısındadır ve 2 motoru birbirlerinden ayrı şekilde ileri- geri kontrol edebilir. Enable bacağı da kullanılarak motor PWM ile de sürülebilir. 4.5V – 36V aralığında motorlar L293D ile kontrol edebilmektedir. L293D' nin maksimum akım sınırı 600mA' dir. L293B' nin akım sınırı ise 1A' dir.



Şekil 3.3. L293D bağlantı şeması.

3. DENEYİN YAPILIŞI

3.1. Arduino İle Led Yakma



Şekil 4.1. Arduino led yakma uygulama devresi.

Deneyin Yapılısı:

1- Şekil 4.1' te görülen bağlantıları Arduino ve board üzerinde gerçekleştiriniz.
2- Arduino Vin pininden 12v besleme ile Arduino' ya güç veriyoruz.
3- Giriş/çıkış pinlerinin de bağlantılarını yapıp program aşamasına geçiniz.

Program:

```
const int buttonPin = 2;
const int ledPin =  13;

int buttonState = 0;
```

// Butonun Arduino ile bağlantısı 2 nolu pin ile gerçekleştirildi ve bu pine buttonPin ismi verildi.
// LED' in Arduino ile bağlantısı 13 nolu pin ile gerçekleştirildi ve bu pine ledPin ismi verildi.
// Butonun değerini kontol eden buttonState adında bir değişken atandı.

```
void setup()
{

  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}
```

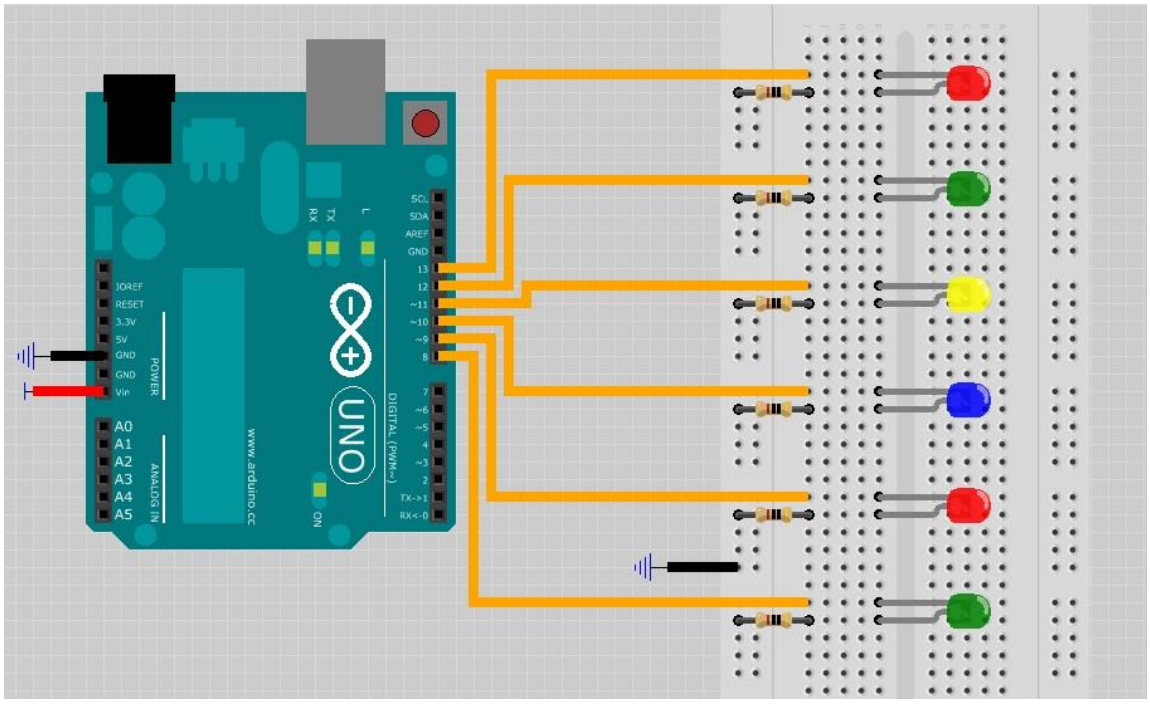
// ledPin pini çıkış olarak tanımlandı .
// buttonPin pini giriş olarak tanımlandı.

```
void loop(){

  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }
}
```

//buttonPin' den okunan değeri buttonState değişkenine yaz .
//buttonState değişkeni HIGH değeri aldığında, yani buton aktif olduğunda, aşağıdaki komutu işle.
// ledPin pinini HIGH yap, yani LED' i yak.

//aksi halde, yani buttonState değişkeni HIGH değil ise ledPin pinini LOW yap, yani LED' i söndür.

3.2. Arduino ile Kara Şimşek Uygulaması



Şekil 4.2. Kara şimşek uygulama devresi.

Deneyin Yapılısı:

1- Şekil 4.2’ te görülen bağlantıları Arduino ve board üzerinde gerçekleştiriniz.

2- Arduino Vin pininden 12v besleme ile Arduino’ ya güç veriyoruz.

3- Giriş/çıkış pinlerinin de bağlantılarını yapıp program aşamasına geçiniz.

Program:

```
const int ledPinleri[]={8, 9, 10, 11, 12, 13};
const int beklemeSuresi=50;
```

// 6 adet LED’ in pin numaraları dizi oluşturularak atandı.
// beklemeSuresi sabitine 50 değeri atandı. Bu değer, LED’lerin yanıp sönmesi esnasındaki gecikmeler için kullanıldı.

```
void setup(){
  for (int led=0; led < 6; led++)
  {
    pinMode(ledPinleri[led], OUTPUT);
  }

}
```

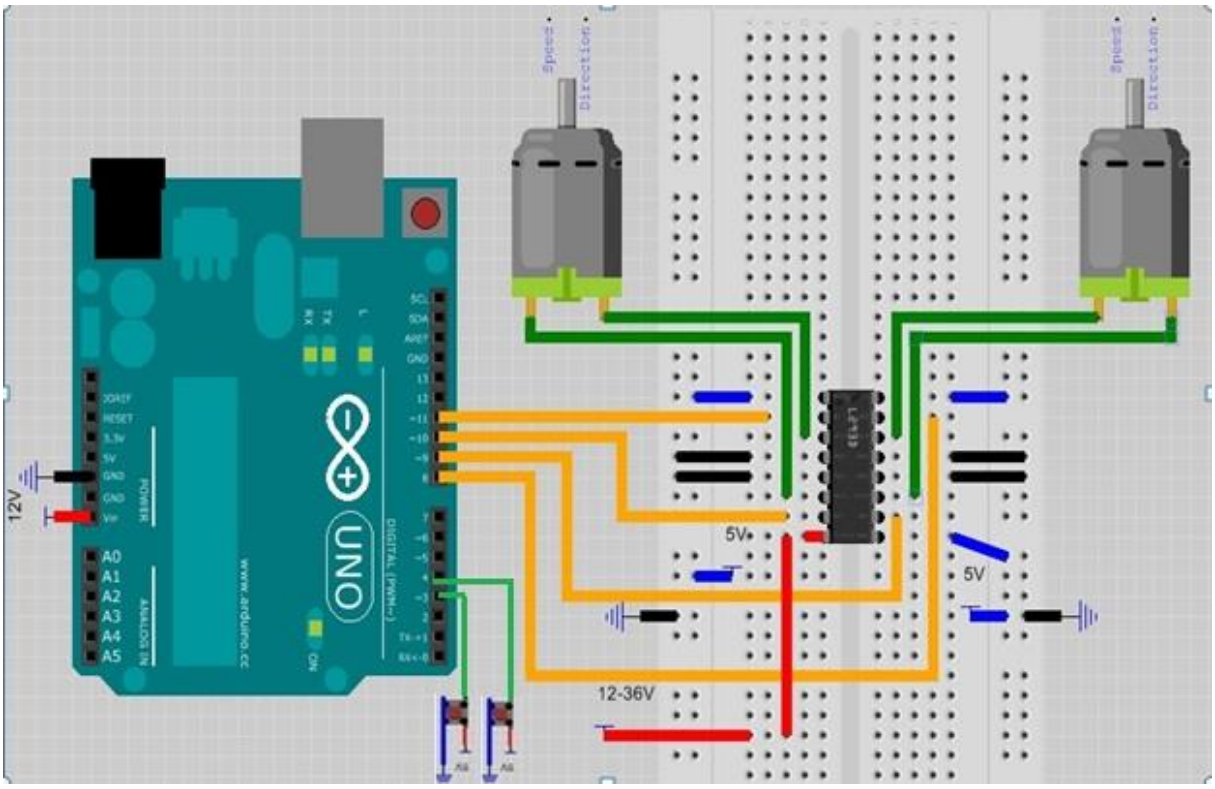
// Bir led değişkeni tanımlandı ve for döngüsü içinde 0’ dan 6’ ya kadar birer artırılarak döngü içerinde kullanıldı.
// led değişkeni for döngüsünün her bir adımında dizinin bir elemanını çıkış yaparak, LED için kullanılan tüm pinlerin çıkış olarak tanımlanmasında kullanıldı.

```
void loop() {
  for (int led=0; led < 5; led++)

  {
  digitalWrite(ledPinleri[led],HIGH);
  delay(beklemeSuresi);
  digitalWrite(ledPinleri[led+1],HIGH);
  delay(beklemeSuresi);
  digitalWrite(ledPinleri[led],LOW);
  delay(beklemeSuresi*2);
  }


  for (int led=5; led >0; led--)

  {

  digitalWrite(ledPinleri[led],HIGH);
  delay(beklemeSuresi);
  digitalWrite(ledPinleri[led-1],HIGH);
  delay(beklemeSuresi);
  digitalWrite(ledPinleri[led],LOW);
  delay(beklemeSuresi*2);
  }
}
```

//led değişkeni for döngüsünün her bir adımında ilk LED' den son LED' e kadar ileri yönde LED 'lerin yanmasını sağlar.
//Sırası gelen LED yanar ve 50ms beklenir, daha sonra bir sonraki yanar ve bir 50ms daha beklendikten sonra önceki LED söner. Bu döngü son LED yanıp bir önceki sönene kadar devam eder.

//Bu döngüde aynı işlem geri yönlü olarak tekrarlanır. Bu döngü ilk LED yanıp bir önceki sönene kadar devam eder.

3.3. Arduino ile DC Motor Kontrol



Şekil 4.3. DC motor uygulama devresi.

Deneyin Yapılısı:

1- Şekil 4.3' te görülen bağlantıları Arduino ve board üzerinde gerçekleştiriniz.

2- Arduino Vin pininden 12v besleme ile Arduino' ya güç veriyoruz.

3- Giriş/çıkış pinlerinin de bağlantılarını yapıp program aşamasına geçiniz.

Program:

```
int  sag_motor1 = 8;
int  sag_motor2 = 9;
int  sol_motor1 = 10;
int  sol_motor2 = 11;
int  sag_kontrol = 3;
int  sol_kontrol = 4;

void setup() {

  pinMode(sag_motor1, OUTPUT);
  pinMode(sag_motor2, OUTPUT);
  pinMode(sol_motor1, OUTPUT);
  pinMode(sol_motor2, OUTPUT);
  pinMode(sag_kontrol, INPUT);
  pinMode(sol_kontrol, INPUT);

}
```

// İlk bölümde değişkenler tanımlandı ve pin numaraları bağlantı şemasına göre belirlendi. Burada sag_motor1 ve sag_motor2 sağdaki motorun + ve – uçlarına bağlı pinleri temsil ederken diğer 2 değişken sol motorun + ve – uçlarına bağlı pinleri temsil ediyor. sag_kontrol, sol_kontrol değişkenlerinin atandığı pinlere ise butonlar bağlanmıştır.

//setup() fonksiyonunun içerisinde ilgili pinler giriş/çıkış olarak tanımlandı.

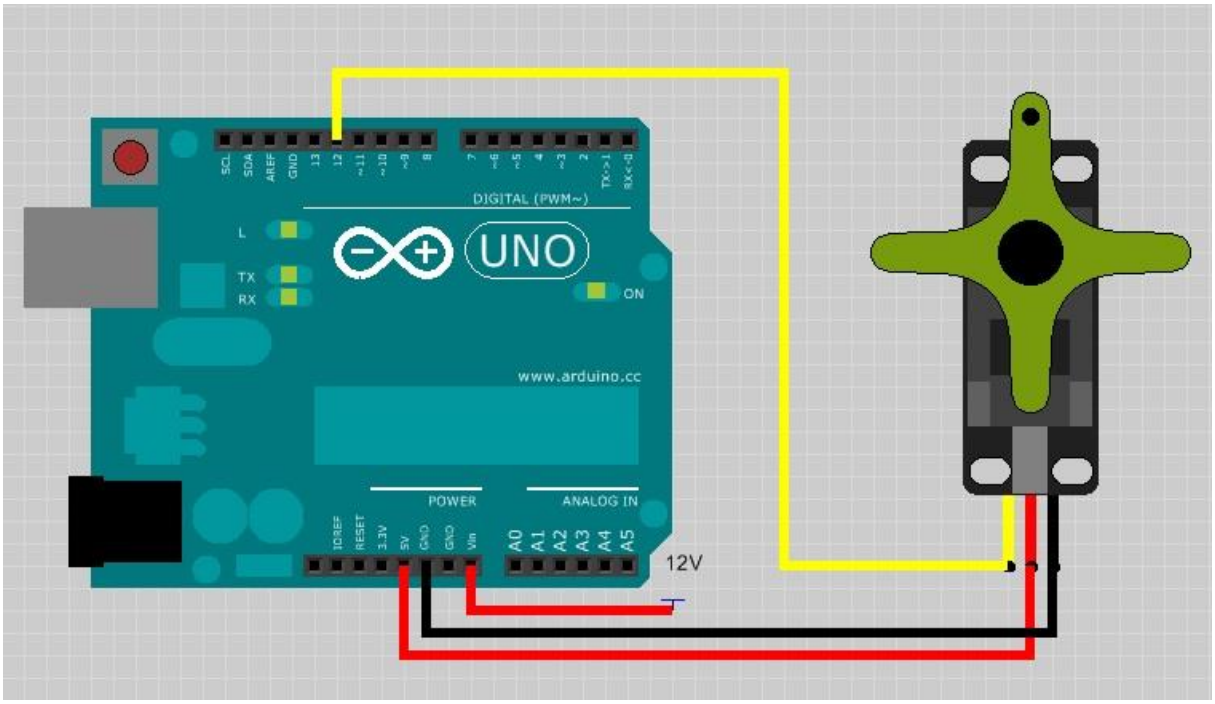
```
void loop() {

   if(sag_kontrol==HIGH)
   {
      digitalWrite(sag_motor1, HIGH);
      digitalWrite(sag_motor2, LOW);
      digitalWrite(sol_motor1, LOW);
      digitalWrite(sol_motor2, LOW);
   }
   if(sol_kontrol==HIGH)
   {
      digitalWrite(sag_motor1, LOW);
      digitalWrite(sag_motor2, LOW);
      digitalWrite(sol_motor1, HIGH);
      digitalWrite(sol_motor2, LOW);
   }
  else
   {
      digitalWrite(sag_motor1, LOW);
      digitalWrite(sag_motor2, LOW);
      digitalWrite(sol_motor1, LOW);
      digitalWrite(sol_motor2, LOW);
   }
  }
```

//Sağ motoru kontrol eden buton aktif ise sağ motoru çalıştır.

//Sol motoru kontrol eden buton aktif ise sol motoru çalıştır.

//aksi durumlarda 2 motora da gerilim uygulama yani motorları durdur.

3.4. Arduino ile Servo Motor Kontrol



Şekil 4.4. Servo motor uygulama devresi.

Deneyin Yapılısı:

1- Şekil 4.4' te görülen bağlantıları Arduino ve board üzerinde gerçekleştiriniz.

2- Arduino Vin pininden 12v besleme ile Arduino' ya güç veriyoruz.

3- Giriş/çıkış pinlerinin de bağlantılarını yapıp program aşamasına geçiniz.

Program:

```
#include <Servo.h>
Servo myservo;

int pos = 0;

void setup()
{
myservo.attach(9);
}
```

```
void loop()
{
   for(pos = 0; pos < 180; pos += 1)
   {
    myservo.write(pos);
    delay(5);
   }
   for(pos = 180; pos>=1; pos-=1)
   {
    myservo.write(pos);
    delay(5);
   }
}
```