ARDUİNO UYGULAMALARI

VELİ ÇAMAN

2023

© Copyright 2023, Veli ÇAMAN
Bu kitabın hakları Veli ÇAMAN' a aittir. Tüm hakları saklıdır. Kaynak gösterilmeden kitaptan alıntı yapılamaz; Veli ÇAMAN 'ın yazılı izni olmadan radyo ve televizyona uyarlanamaz; oyun, film, elektronik kitap, CD ya da manyetik bant haline getirilemez; fotokopi ya da herhangi bir yöntemle çoğaltılamaz, yayınlanamaz ve dağıtılamaz.

ÖNSÖZ

Bu kitap Mesleki ve Teknik Anadolu Lisesinde okutulan "Mikrodenetleyiciler ve Kodlama" ile "İleri Mikrodenetleyiciler" derslerinde, öğrencilerin yardımcı kaynak olarak kullanmaları amacıyla hazırlanmış **ücretsiz** bir kaynaktır.

İÇİNDEKİLER

PROGRAMLAMA NEDİR ?

Programlama, bilgisayarlar veya diğer bilgi işlem cihazları üzerinde belirli bir görevi gerçekleştirmek için talimatların yazıldığı bir süreçtir. Bu talimatlar bir programın temelini oluşturur ve genellikle bir programlama dilinde yazılır. Programlama, bilgisayarların insanların anlayabileceği şekilde yönlendirilmesini sağlar.

Programlama şunları içerir:

Algoritma Geliştirme: İlk adım, belirli bir problemi çözmek için bir algoritma geliştirmektir. Algoritma, problemi çözmek için yapılması gereken adımların mantıklı bir sırasını tanımlar.
Programlama Dili Seçimi: Algoritma geliştikten sonra, bu algoritmayı bir programlama dilinde ifade etmek gerekir. Programlama dili, bilgisayarın anlayabileceği bir formatta algoritmayı ifade etmenizi sağlar. Örneğin, popüler programlama dilleri arasında Python, C++, Java, JavaScript ve Ruby bulunmaktadır.
Kod Yazma: Programlama dili seçildikten sonra, algoritmayı bu dilde kodlamak gerekir. Bu adım, algoritmanın yazılı bir biçimde bilgisayar tarafından anlaşılabilir hale getirilmesini sağlar.
Kodun Derlenmesi veya Yorumlanması: Kod yazıldıktan sonra, genellikle bir derleyici veya yorumlayıcı kullanılarak kodun bilgisayar tarafından çalıştırılabilir hale getirilmesi gerekir. Derleme, kodun bir bilgisayar programına dönüştürülmesi işlemidir. Yorumlama ise kodun satır satır çalıştırılmasıdır.
Hata Ayıklama (Debugging): Yazılan kodun hatalarını tespit etmek ve düzeltmek için hata ayıklama işlemi yapılır. Bu, programın beklenen şekilde çalışmasını sağlamak için önemlidir.

Test Etme ve Geliştirme: Kod tamamlandıktan sonra, programın işlevselliğini test etmek ve gerektiğinde iyileştirmeler yapmak önemlidir. Bu, kullanıcıların sorunsuz bir deneyim yaşamasını sağlar.

1.1. Algoritma Nedir ?

Algoritma, belirli bir problemi çözmek veya belirli bir görevi gerçekleştirmek için tasarlanmış mantıklı ve düzenli bir adım sırasını içeren talimatlar kümesidir. Algoritmalar, bilgisayar programlarından fiziksel problemlerin çözülmesine kadar birçok farklı bağlamda kullanılabilir. Temel olarak, algoritma, başlangıç bir durumdan başlayarak, belirli bir hedefe ulaşmak için izlenmesi gereken yolun tanımlandığı bir yönergeler listesidir.

Algoritmaların temel özellikleri şunlar olabilir:

Başlangıç Durumu: Her algoritma, bir başlangıç noktasından başlar. Bu başlangıç durumu, problemi veya görevi başlatan durumdur.

Adım Adım Talimatlar: Algoritma, adım adım talimatlar içerir. Bu talimatlar, problemi çözmek veya görevi tamamlamak için izlenmesi gereken belirli adımları tanımlar. Her adım, açık ve anlaşılır olmalıdır.

Karar Noktaları: Algoritma, karar noktalarını içerebilir. Bu noktalarda, belirli koşulların karşılaştırılması ve belirli adımların seçimi söz konusu olabilir.

Döngüler: Algoritmalar, bazen belirli adımları tekrarlamak için döngüler içerebilir. Bu, belirli bir şart karşılandığında veya belirli bir süre boyunca devam etmesi gereken işlemleri tanımlamak için kullanılır.

Çıkış Durumu: Algoritma, hedefe ulaşıldığında veya belirli bir koşul karşılandığında sona erer. Bu, problemi çözme veya görevi tamamlama aşamasıdır.

Algoritmalar, bilgisayar programları oluşturmak için kullanıldığı gibi, matematiksel problemleri çözmek, veri sıralamak, grafikleri çizmek, veri tabanlarına erişmek ve daha pek çok farklı konuda kullanılabilir. Algoritmaların iyi tasarlanmış olması, verimlilik, doğruluk ve düşük kaynak kullanımı gibi önemli faktörleri etkileyebilir. Bu nedenle, algoritma tasarımı, bilgisayar bilimleri, yazılım geliştirme ve matematik gibi birçok alanda temel bir kavramdır.

Örnek : Bozuk bir elektrik prizinin değiştirilmesi algoritması.

Adım 1 : Başla.
Adım 2 : Sigortayı Kapat.
Adım 3 : Kontrol kalemi yardımıyla prizi sök.
Adım 4 : Çalışan prizi tak ve sigortayı aç.
Adım 5: Prizi kontrol et, çalışmıyorsa Adım 2' ye git.
Adım 6 : Bitir.

1.2. Akış Şeması

Akış şeması, bir işlemin veya sürecin adımlarını, karar noktalarını, giriş ve çıkışları ve bu adımlar arasındaki ilişkileri grafiksel olarak temsil eden bir diyagramdır. Akış şemaları, karmaşık iş süreçlerini veya algoritmaları anlamak, analiz etmek ve belgelemek için kullanılır. Bu şemalar, bir işlemin veya sürecin mantığini ve akışını görsel olarak açıklayarak anlaşılmasını kolaylaştırır.

Akış şeması oluşturmak için kullanılan temel semboller şunlardır:

Dikdörtgen (İşlem): İşlem sembolü, bir görevin veya adımın gerçekleştirildiği yeri temsil eder. Bu adımlar, işlem adımlarını veya algoritmik komutları içerebilir.

Daire (Başlangıç ve Bitiş): Daire sembolü, bir işlemin veya sürecin başlangıcını ve bitişini temsil eder. Başlangıç noktası bir ok ile işaretlenirken, bitiş noktası bir daire içine yazılan "Stop" veya "End" gibi bir kelime ile gösterilir.

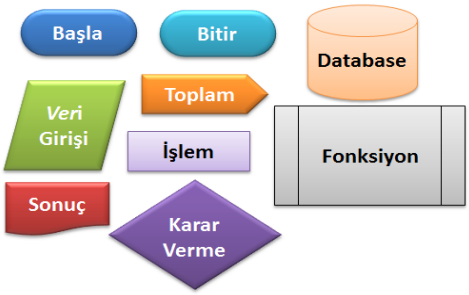
Paralelkenar (Karar veya Koşul): Paralelkenar sembolü, bir karar veya koşul noktasını temsil eder. Bu noktalarda genellikle "Evet" veya "Hayır" gibi iki seçenek bulunur ve bir yönlendirme oku, koşula göre nasıl devam edileceğini belirtir.

Veri Girişi/Çıkışı (Paralelkenarın Uzantısı): Paralelkenar sembolünün uzantıları, veri girişi veya çıkışını temsil eder. Bu semboller, kullanıcının veri girmesini veya sonuçları görmesini sağlar.

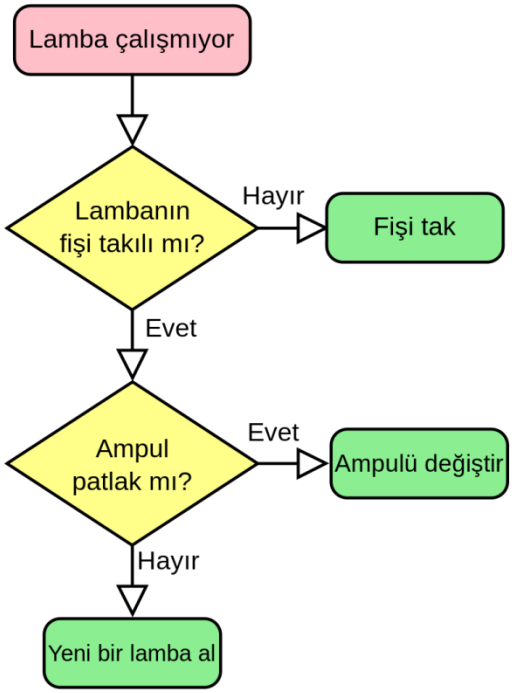
Yatay ve Dikey Yönlendirme Okları: Yatay oklar, işlem adımlarını ve akışın yatay yönde nasıl devam edeceğini gösterirken, dikey oklar işlemlerin altında veya üstünde sırayla nasıl ilerleyeceğini gösterir.

Akış şeması çizmek, bir sürecin veya algoritmanın mantığını daha iyi anlamak ve başkalarına açıklamak için kullanışlı bir araçtır. Bu nedenle, yazılım geliştirme, iş süreçleri tasarlama, veri analizi ve çeşitli mühendislik alanlarında sıkça kullanılır. Akış şemaları, karmaşık problemleri daha küçük ve daha yönetilebilir parçalara bölmek için de kullanılabilir, böylece her bir adım veya parça ayrı ayrı ele alınabilir.

Akış Diyagramı Sembolleri



Örnek :



1.3. Veri Türleri

Veri türleri, bilgisayar programlamasında ve veri işleme alanlarında kullanılan farklı veri değerlerini sınıflandırmak için kullanılan kategorilerdir. Her veri türü, belirli türdeki verileri saklamak veya işlemek için kullanılır ve programlama dilleri tarafından desteklenir. İşte yaygın olarak kullanılan bazı veri türleri:

Tam Sayı (Integer): Tam sayı veri türü, kesirli olmayan pozitif veya negatif tam sayı değerlerini temsil eder. Örnek olarak, -3, 0, 42 gibi değerler tam sayılara örnektir. Programlamada genellikle "int" olarak kısaltılır.

Kesirli Sayı (Float veya Double): Kesirli sayı veri türü, ondalık sayıları veya kesirli değerleri temsil eder. Örnek olarak, 3.14 veya -0.5 gibi değerler kesirli sayılara örnektir. "float" ve "double" veri türleri, farklı hassasiyet seviyeleri sunar.

Karakter (Char): Karakter veri türü, tek bir karakteri temsil eder. Bu karakterler harfler (A, b, c), rakamlar (0, 1, 2), özel semboller (!, $, %) veya boşluk gibi herhangi bir karakter olabilir. "char" veri türü olarak adlandırılır.

Dize (String): Dize veri türü, metin veya karakter dizilerini temsil eder. Örneğin, "Merhaba, Dünya!" gibi metinler bir dize olarak saklanır. Dizeler genellikle çift tırnak içinde ("") tanımlanır ve "string" olarak adlandırılır.

Boolean (Boolean): Boolean veri türü, yalnızca iki değeri temsil eder: "True" (doğru) ve "False" (yanlış). Boolean değerler, genellikle mantıksal ifadelerin sonuçlarını veya koşulları temsil etmek için kullanılır.

Liste (List): Liste veri türü, birden fazla veriyi bir araya getiren bir veri yapısıdır. Listeler, sıralı verileri saklamak için kullanılır ve elemanlara erişim sağlar. Programlama dillerine göre farklı isimlere sahip olabilirler, örneğin Python'da "list" olarak adlandırılırlar.

Dizin (Array): Dizin veri türü, sabit boyutta bir veri koleksiyonunu temsil eder. Diziler, benzer türdeki verileri depolamak ve indeks kullanarak erişmek için kullanılır. Özellikle programlama dillerinde veri koleksiyonları için etkilidirler.

Nesne (Object): Nesne veri türü, nesne yönelimli programlamada kullanılır. Bu tür, verileri ve bu verilere ait işlevleri bir araya getirir. Nesne tabanlı programlamada sıkça kullanılır.

NULL veya Nil (Null veya Nil): NULL veya Nil, bir değişkenin değerinin boş veya tanımsız olduğunu belirtmek için kullanılan bir değer veya işaretçidir.

Yapı (Struct veya Record): Yapılar, farklı veri türlerinden oluşan bir dizi veriyi bir araya getiren bir türdür. Her veriye ad verilir ve bu yapılar genellikle benzer türdeki verileri gruplamak için kullanılır.

Veri türü	ALT SINIR	ÜST SINIR
char	-128	127
int	-2,147,483,648	2,147,483,647
short int	-32,768	32,767
long int	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned char	0	255
unsigned short	0	65,535
unsigned int	0	4,294,967,295
unsigned long	0	18,446,744,073,709,551,615
float	-3.4e-38	3.4e+38
long double	-3.4e-4932	1.1e+4932
double	-1.7e-308	1.7e+308

1.4. Operatörler

Operatörler, programlama dillerinde belirli işlemleri gerçekleştirmek için kullanılan semboller veya anahtar kelimelerdir. Operatörler, değişkenler arasında matematiksel işlemler yapmak, karşılaştırmalar yapmak, mantıksal işlemler gerçekleştirmek, bit düzeyinde işlemler yapmak ve daha birçok işlemi yürütmek için kullanılır. İşte yaygın olarak kullanılan operatör türlerinin bir özeti:

Aritmetik Operatörler:

+: Toplama **-**: Çıkarma *****: Çarpma /: Bölme **%**: Mod (Bölümden kalan)

Karşılaştırma Operatörleri:
==: Eşittir !=: Eşit Değildir <: Küçüktür >: Büyüktür <=: Küçük veya Eşittir >=: Büyük veya Eşittir

Mantıksal Operatörler:

&& veya **and**: VE (Her iki koşul da doğru ise sonuç doğru)

|| veya **or**: VEYA (En az bir koşul doğru ise sonuç doğru)

! veya **not**: DEĞİL (Koşul yanlışsa sonuç doğru, koşul doğruysa sonuç yanlış)

Atama Operatörleri:

=: Değer atama +=: Toplama ve atama -=: Çıkarma ve atama

*=: Çarpma ve atama /=: Bölme ve atama %=: Mod ve atama

Artırma ve Azaltma Operatörleri:

++: Değeri 1 artırma (örneğin, **x**++) --: Değeri 1 azaltma (örneğin, **y**--)

Bit Düzeyinde Operatörler:

&: VE (bit düzeyinde) |: VEYA (bit düzeyinde)

^: XOR (bit düzeyinde) ~: Tersi (bit düzeyinde)

<<: Sol kaydırma (bit düzeyinde)

>>: Sağ kaydırma (bit düzeyinde)

1.5. Programlama Komut Yapıları

İf-Else Yapısı

"If-else" yapısı, programlarda belirli bir koşulu değerlendirmek ve bu koşula göre farklı işlemler yapmak için kullanılan bir kontrol yapısıdır. Bu yapının amacı, belirli bir şartın doğru veya yanlış olup olmadığını kontrol ederek, programın akışını farklı yollar boyunca yönlendirmektir.

"If-else" yapısının temel kullanımı şu şekildedir:

```
if (koşul):
    {# Koşul doğru ise buradaki işlemler yapılır }
else:
    { # Koşul yanlış ise buradaki işlemler yapılır }
```

Örnek:

```
if (x > 5)
  { digitalWrite (FAR,1); }
else
{   digitalWrite (FAR,0);   }
```

Eğer birden fazla koşul varsa <u>if-elseif</u> yapısı kullanılır.

```
if (x > 5)
{ digitalWrite (FAR,1); }

elseif (x>5)
{   digitalWrite (FAR,0);   }

else {   delay (500);   }
```

Switch- Case Yapısı

"Switch-case" yapısı, bir programın akışını belirli değerlere veya koşullara göre yönlendirmek için kullanılan bir kontrol yapısıdır. Genellikle bir değerin birden fazla farklı duruma (case) göre değerlendirilmesi gerektiğinde kullanılır. Bu yapı, if-else ifadeleri yerine daha okunaklı ve yönetilebilir kodlar oluşturmanıza olanak tanır.

Genel olarak "switch-case" yapısının yapısı şu şekildedir:

```
switch (değer)

{

 case durum1:
 // durum1 için işlemler
   break;
 case durum2:
 // durum2 için işlemler
 break;
 default:
  // hiçbir durum eşleşmezse buradaki işlemler
}
```

Örnek :

```
int ay = 3;
switch (ay)
 {
case 1:
   printl("Ocak");
   break;
case 2:
  printl("Şubat");
  break;
case 3:
  printl("Mart");
  break;
default:
 printl("Bilinmeyen ay");
}
```

1.6. Döngüler

Döngüler, programlama dillerinde belirli bir işlemi veya kod bloğunu belirli bir koşula veya belirli bir sayıda tekrarlamak için kullanılan yapısal öğelerdir. Döngüler, programlarda tekrarlayan görevleri gerçekleştirmek ve verileri işlemek için önemli bir rol oynarlar.

İşte yaygın olarak kullanılan döngü türleri:

For Döngüsü (For Loop)

For döngüsü, belirli bir başlangıç değeri, bitiş değeri ve artış miktarı ile belirli bir aralıktaki işlemleri tekrarlamak için kullanılır. Bu döngü, genellikle bir sayacın değeri üzerinde çalışır ve belirli bir koşul karşılandığında sona erer.

```
for (int i = 0; i < 5; i++)
    {
    // 0'dan 4'e kadar olan değerleri tekrarlar
     // Her tekrarda bu kod bloğu çalışır.
    }
```

While Döngüsü (While Loop):

While döngüsü, belirli bir koşul doğru olduğu sürece işlemleri tekrarlamak için kullanılır. Koşul her döngü başladığında değerlendirilir ve koşul yanlış hale geldiğinde döngü sona erer.

```
int sayac = 0;
while (sayac < 5)
{
        // Koşul doğru olduğu sürece bu kod bloğu çalışır
        sayac++;
}
```

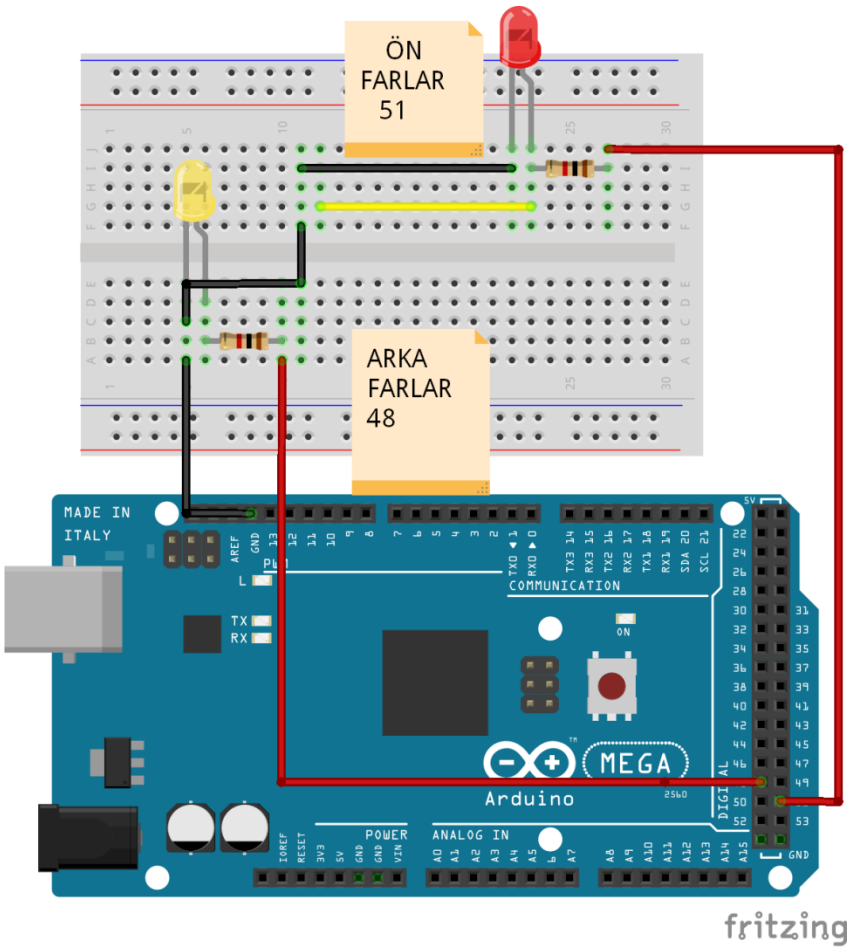
Do-While Döngüsü (Do-While Loop)

Do-while döngüsü, işlemi en az bir kez yapmak istediğinizde kullanılır. İşlemi yapar ve ardından belirli bir koşul doğru olduğu sürece işlemi tekrarlar.

```
 int sayac = 0;
do
{
        // Bu kod bloğu en az bir kez çalışır
        sayac++;
  } while (sayac < 5);
```

2. LED UYGULAMALARI

2.1. İKİ LED UYGULAMA



Malzeme Listesi

1- Arduino Mega
2- 2 Adet led diyot
3- 2 Adet 1K direnç

2.1.1. LED UYGULAMA (DİJİTALWRİTE)

//Ön ve Arka farları yakan programı yazınız.

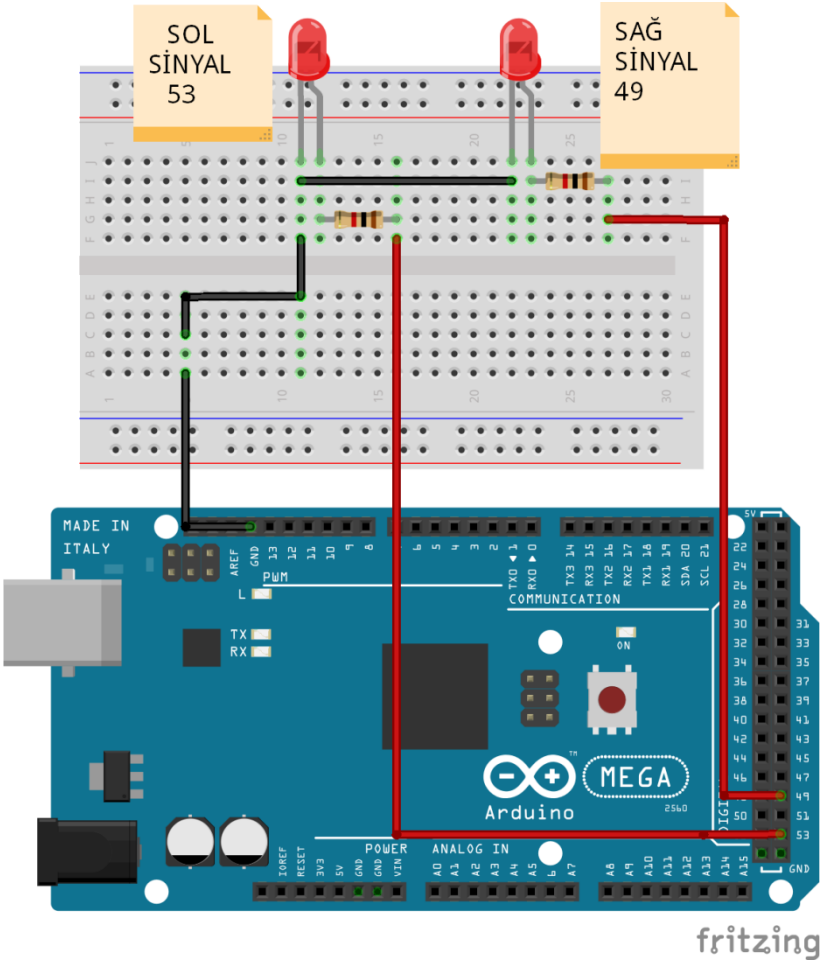
```
int ONFAR=51;                    //51 Nolu pine ONFAR ismi tanımlandı.
 int ARKAFAR=48;                  //48 Nolu pine ONFAR ismi tanımlandı.


  void setup()
  {
    pinMode(ONFAR,OUTPUT);    //48 VE 51 Nolu Pinler Çıkış olarak tanımlandı.
    pinMode(ARKAFAR,OUTPUT);
  }
  void loop()
    {
    digitalWrite (ONFAR,1);    //ONFAR ve ARKAFAR lojik "1" gönderilir..
   digitalWrite (ARKAFAR,1);
    }
```

2.1.2. LED UYGULAMA (DELAY)

//Ön ve Arka farları 1sn aralıklarla sürekli yakıp söndüren programı yazınız.

```
int ONFAR=51;              //51 Nolu pine ONFAR ismi tanımlandı.
int ARKAFAR=48;            //48 Nolu pine ONFAR ismi tanımlandı.
void setup()
{
  pinMode(ONFAR,OUTPUT);   //48 VE 51 Nolu Pinler Çıkış tanımlandı.
  pinMode(ARKAFAR,OUTPUT);
}

void loop()
 {
  digitalWrite (ONFAR,1);          //Ön Far yandı.
  digitalWrite (ARKAFAR,1);
  delay(1000);                     //1 sn gecikme olur.
  digitalWrite (ONFAR,0);          //Ön Far söndü.
  digitalWrite (ARKAFAR,0);
  delay(1000);
 }
```

2.1.3. LED UYGULAMA (İF-ELSE)



Malzeme Listesi

1- Arduino Mega
2- 2 Adet led diyot
3- 2 Adet 1K direnç

//Sağ ve Sol sinyalleri sırası ile 1sn aralıklarla 5 kere yakıp söndüren programı Sayaç değişkeni tanımlayarak ve IF-ELSE komutu kullanarak yazınız.

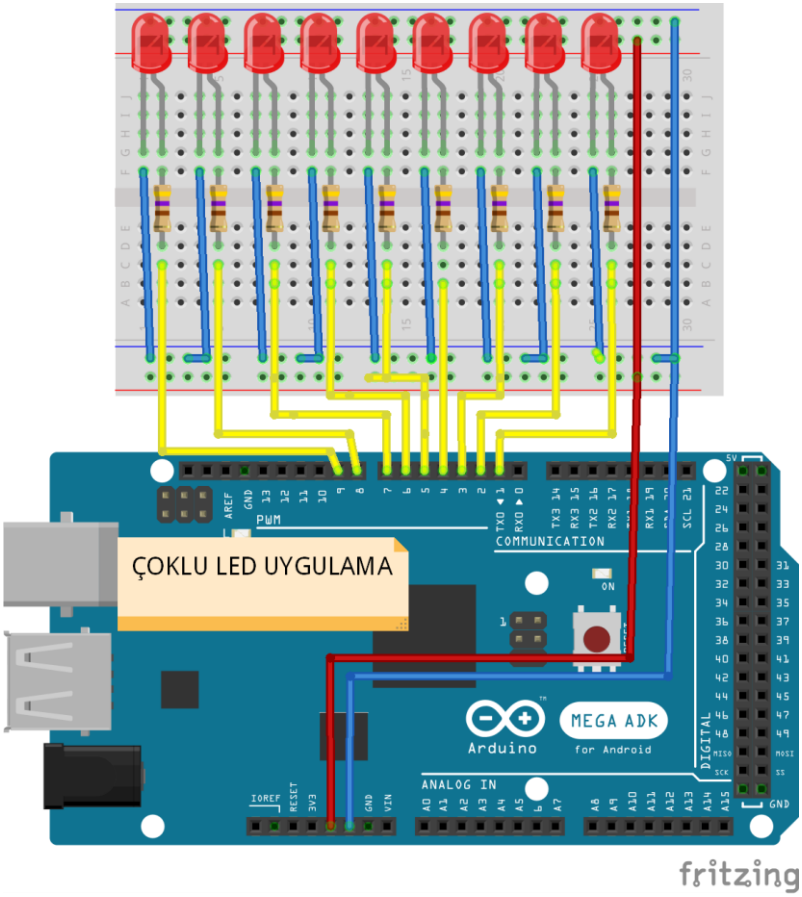
```
int SAGSINYAL=49;              //49 Nolu pine SAGSINYAL ismi tanımlandı.
int SOLSINYAL=53;              //53 Nolu pine SOLSINYAL ismi tanımlandı.
 int sayac=0;                  //sayaç değişkeni tanımlanıp 0 değeri yüklendi.
   void setup()
   { pinMode(SAGSINYAL,OUTPUT);   //49 VE 53 Nolu Pinler Çıkış olarak tanımlandı.
     pinMode(SOLSINYAL,OUTPUT)  }
void loop()
    {  X:
       digitalWrite (SAGSINYAL,1);     // Sağsinyali 1 sn aralıkla yak söndür.
         delay(1000);
         digitalWrite (SAGSINYAL,0);
         delay(1000);
         sayac++;                      //sayacı 1 artır.
         if (sayac<5)goto X;           //sayacın değeri 5 ten küçükse X adlı etikete git
         sayac=0;                      // değilse (5 ten büyükse) alt satırdan devam et
                                       //ve sayacı sıfırla
                                        Y:
         digitalWrite (SOLSINYAL,1);     // Sol sinyali 1 sn aralıkla yak söndür.
         delay(1000);
         digitalWrite (SOLSINYAL,0);
         delay(1000);
         sayac++;                       //sayacı 1 artır.
         if (sayac<5)goto Y;
         sayac=0;
       }
```

2.1.4. LED UYGULAMA (FOR)

//Sağ ve Sol sinyalleri sırası ile 1sn aralıklarla 5 kere yakıp söndüren programı FOR komutu kullanarak yazınız.

```
int SAGSINYAL=49;              //49 Nolu pine SAGSINYAL ismi tanımlandı..
int SOLSINYAL=53;              //53 Nolu pine SOLSINYAL ismi tanımlandı..
int i;                          //sayaç değişkeni tanımlanıp 0 değeri yüklendi.
  void setup()
  {
    pinMode(SAGSINYAL,OUTPUT);   //49 VE 53 Nolu Pinler çıkış tanımlandı.
    pinMode(SOLSINYAL,OUTPUT);
  }
  void loop()
   {
   for ( i=0 ; i<5 ; i++)
        {
        digitalWrite (SAGSINYAL,1);    // Sağsinyali 1 sn aralıkla yak söndür.
         delay(1000);
         digitalWrite (SAGSINYAL,0);
         delay(1000);
        }
  for ( i=0 ; i<5 ; i++)
        {digitalWrite (SOLSINYAL,1);   // Sol sinyali 1 sn aralıkla yak söndür.
         delay(1000);
         digitalWrite (SOLSINYAL,0);
         delay(1000);
        }
```

2.2. SEKİZ LED UYGULAMA



Malzeme Listesi

1- Arduino Mega
2- 8 Adet led diyot
3- 8 Adet 470 Ohm direnç

2.2.1. DİZİ TANIMLAMA UYGULAMASI

//LED1 den LED8 e kadar olan ledleri dİzi tanımlayarak çıkış seçiniz. Tüm ledleri 500 msn. Aralıklarla yakıp söndürünüz.

```
int LEDDIZI[] ={2,3,4,5,6,7,8,9} ;    //Ledler diziye tanımlandı
 int i;
void setup()
{
 for(i=0; i < 8 ; i ++)
     { pinMode(LEDDIZI [i] , OUTPUT); }//Ledler çıkış olarak tanımlandı


}
void loop()
  {
 for ( i=0 ; i < 8 ; i++)
      { digitalWrite (LEDDIZI[i],1);  }   // Tüm ledleri yak.
     delay(500);


 for ( i=0 ; i < 8 ; i++)
      { digitalWrite (LEDDIZI[i],0);  }   // Tüm ledleri söndür.
     delay(500);
  }
```

2.2.2. KARAŞİMŞEK UYGULAMASI

LED1 den LED8 e kadar olan ledleri 50 msn aralıklarla sağdan sola ve soldan sağa tek tek yakıp söndüren programı yazınız.

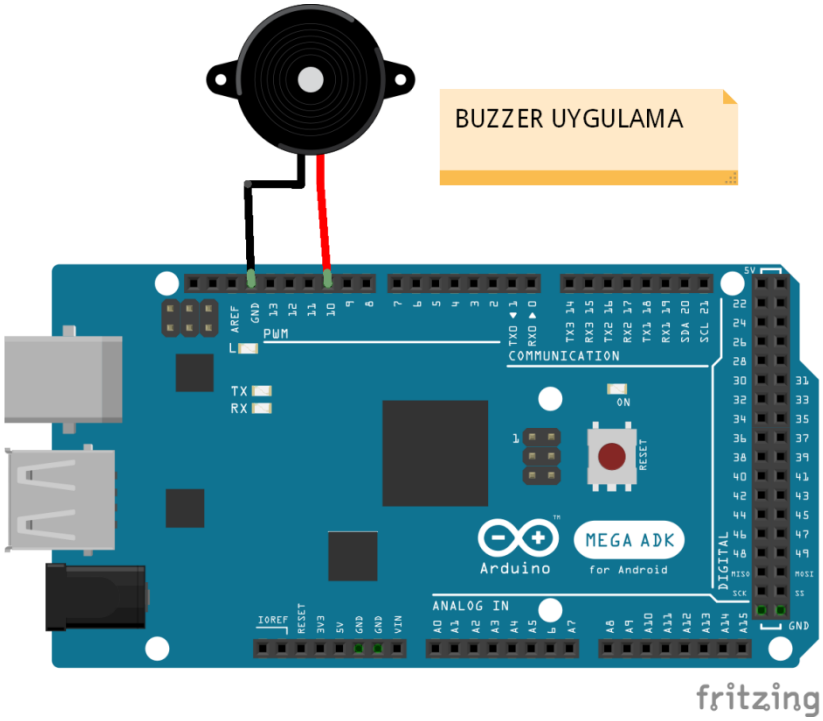
```
int LEDDIZI[] ={2,3,4,5,6,7,8,9} ;    //Ledler diziye tanımlandı
 int i;              //sayaç değişkeni tanımlanıp 0 değeri yüklendi.

 void setup()
 {
  for(i=0; i < 8 ; i ++)
      { pinMode(LEDDIZI [i] , OUTPUT); }
 }

 void loop()
  {
  for ( i=0 ; i < 8 ; i++)                    // İleri arttırma.
       {  digitalWrite (LEDDIZI[i],1);     // Ledleri soldan sağa yak söndür.
          delay(50);                        // 50 msn gecikme yapar.
          digitalWrite (LEDDIZI[i],0);
       }
   for ( i=7 ; i > -1 ; i --)                 // Geriye azaltma
       {  digitalWrite (LEDDIZI[i],1);     // Ledleri sağdan sola yak söndür.
          delay(50);
          digitalWrite (LEDDIZI[i],0);
       }

   }
```

2.2.3. POLİS ÇAKARI UYGULAMASI

//LED1 den LED4 e kadar olan ledleri 30 msn aralıklarla 15 defa yakıp söndüren, sonrasında LED5 den LED8 e kadar olan ledleri 30 msn aralıklarla 15 defa yakıp söndüren ve bunu sürekli tekrarlayan programı yazınız.

```
int LEDdizisi [] ={2,3,4,5,6,7,8,9};     //Ledler diziye tanımlandı
  int i;
void setup()
{for(i=0; i < 8 ; i ++)
      {pinMode( LEDdizisi [i] , OUTPUT); } //Ledler çıkış olarak tanımlandı
}
void loop()
 {
 for(int i=0; i<15; i++)  //15 defa tekrarlanır.
     {
       digitalWrite(LEDdizisi[0],1);    digitalWrite(LEDdizisi[1],1);
       digitalWrite(LEDdizisi[2],1);    digitalWrite(LEDdizisi[3],1);
       delay(30);    //Ledler 30 msn yanar.
       digitalWrite(LEDdizisi[0],0);    digitalWrite(LEDdizisi[1],0);
       digitalWrite(LEDdizisi[2],0);    digitalWrite(LEDdizisi[3],0);
       delay(30);   //Ledler 30 msn söner.
      }
     for(int i=0; i<15; i++)
     { digitalWrite(LEDdizisi[4],1);    digitalWrite(LEDdizisi[5],1);
       digitalWrite(LEDdizisi[6],1);    digitalWrite(LEDdizisi[7],1);
       delay(30);
       digitalWrite(LEDdizisi[4],0);    digitalWrite(LEDdizisi[5],0);
       digitalWrite(LEDdizisi[6],0);    digitalWrite(LEDdizisi[7],0);
       delay(30);       }
 }
```

3. BUZZER UYGULAMASI



Malzeme Listesi

1- Arduino Mega
2- 1 adet Buzzer

//10 nolu pinde bulunan buzerı fonksiyon kullanarak "analogWrite" komutu ile farklı tonlar elde eden programı yazınız.

```
 int buzer=10;  //10 nolu port büzer olarak tanımlandı.
void setup()
{
  pinMode(buzer,OUTPUT); //10 nolu port çıkış olarak tanımlandı.
}
void loop ()
{
  beep(30);              //beep adlı fanksiyonu çağır. Bekle değişkenine 30 yükle.
  analogWrite(buzer, 0); // Buzer 1 sn durur.
  delay(1000);
}
void beep( int bekle)  //beep adlı fanksiyon tanımlandı.
{
  for(int i=0;i<5;i++)
{
  analogWrite(buzer, 50);  // Analog 50 bilgisi buzer'a gönderilir.
  delay(bekle);
  analogWrite(buzer, 255);    // Analog 255 bilgisi buzer'a gönderilir.
   delay(bekle);
  }
 }
```

4. SERİ MONİTÖR UYGULAMASI-1

// PC ye veri gönderen programı yazınız.

```
void setup()
 {
   Serial.begin(9600);              //Seri haberleşme için baud rate oranı ayarla.
 }

 void loop()
 {
   Serial.println("MERHABA ");          //1. Satıra yaz ve alt satıra geç.
   Serial.print("SAAT=13.00    ");       //saati yaz yanına günü yaz.
   Serial.print("SALI");                 //Ekrana "SALI" yaz.
   Serial.println();                     //Alt satıra geç.
   delay(1000);                          //1000 msn gecikme yap.

 }
```

5. SERİ MONİTÖR UYGULAMASI-2

// PC ye saniye bilgisini gönderen programı yazınız.

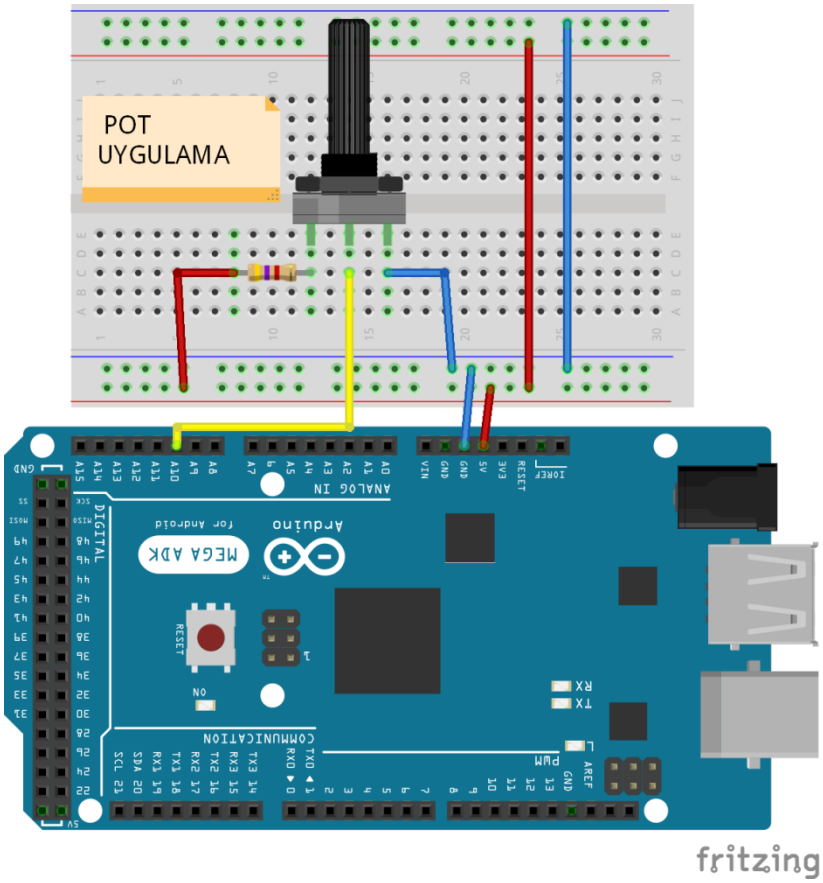
```
int x;                  //x 10 değişken olarak tanımlandı.
void setup()
{
Serial.begin(9600);     //Seri haberleşme için baud rate oranı ayarla.

}
void loop()
{
for (x=0;x<60;x++)        //İşlemi 60 defa tekrarlar.
{
  Serial.print("Saniye= ");  // Seri monitör ekranına 1sn aralıklarla yazılır.
  Serial.println(x);
  delay(1000);
  }
}
```

6. SERİ MONİTÖR UYGULAMASI-3

// PC den "f" bilgisi gönderilirse farları yakan, "c" bilgisi gönderilirse söndüren programı yazınız.

```
int x;
int far = 51 ,arkafar = 48;
char z;

void setup()
{
Serial.begin(9600);
pinMode(far,OUTPUT        //Farlar çıkış olarak ayarlandı.
pinMode(arkafar,OUTPUT);
}
void loop()
{
   if (Serial.available()>0)       //Seri monitörden bilgi geldi mi?
  {
   z = Serial.read();                 //Gelen bilgiyi z değişkenine yükle
   if (z=='f'){digitalWrite(far,1);} //Gelen bilgi "f" mi ? Evet ise farları yak.
   if (z=='f'){digitalWrite(arkafar,1);}
  if (z=='c'){digitalWrite(far,0);}
  if  (z=='f'){digitalWrite(arkafar,0);} //Gelen bilgi "c" mi?Evet ise farları söndür.
     Serial.println(z); //Gelen bilgiyi seri monitör ekranına yaz.
   }
 }
```

7. POTANSİYOMETRE UYGULAMASI (ANALOG OKUMA)



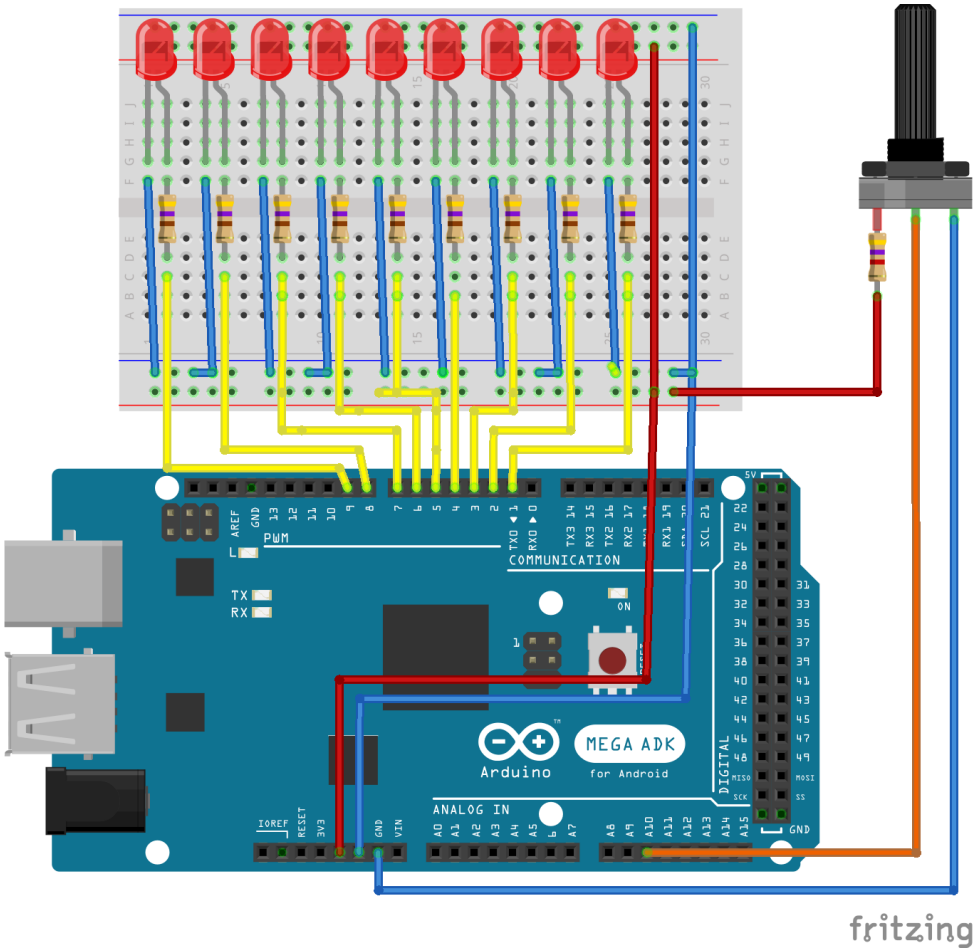
Malzeme Listesi

1- Arduino Mega
2- 1 Adet 4.7K direnç
3- 1 Adet Potansiyometre

// Potansiyometre değerini okuyup değeri seri monitöre gönderen programı yazınız.

```
 int x ;          // x isminde  değişken tanımlandı.
int pot=A10;  // A10 portu "pot" isminde  değişken tanımlandı.
int PotDegeri;
 void setup()
 {
   Serial.begin(9600);
 }
 void loop()
 {
   PotDegeri = analogRead(pot); // A10 pinine bağlı pot değerini oku.
   Serial.print("POT=");
   Serial.println(PotDegeri);     //POT değerini ekrana yaz.
   Serial.println();                 //1 satır atla

    int  x=map(PotDegeri, 0, 1023, 0, 255);    //POT değerini 0-255 arasına
                                                          dönüştür.
   Serial.print("POT YENI=");              // Seri monitöre "POT YENİ" yaz.
   Serial.println(x);                          // Yeni POT değerini ekrana yaz.
   Serial.println();
   delay(50);
 }
```

8. POTANSİYOMETRE UYGULAMASI (FONKSİYON)



Malzeme Listesi

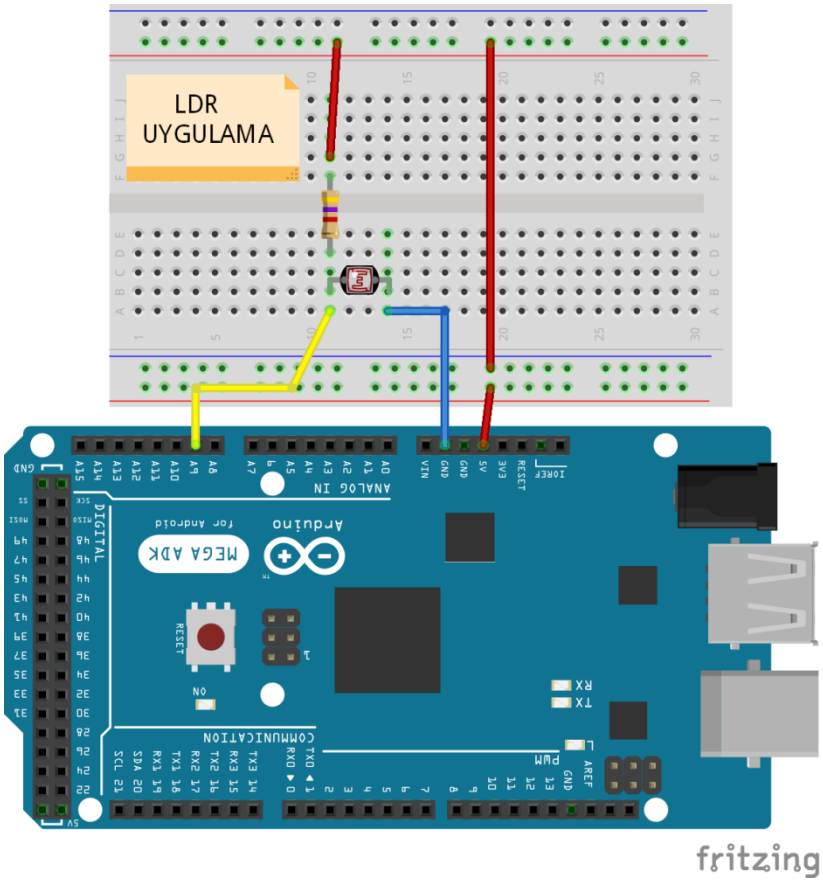
1- Arduino Mega
2- 8 Adet 470 Ohm direnç
3- 1 Adet Potansiyometre
4- 8 Adet led diyot

// Karaşimşek uygulamasında ledlerin yanıp sönme zaman değerini potansiyometre değerini okuyarak alıp çalıştıran programı yazınız.

```
int pot=A10, PotDegeri;
int LEDdizisi[] = {2,3,4,5,6,7,8,9}; // LED pinleri dizi olarak tanımlandı.
int bekle;
void setup()
{
  Serial.begin(9600);
   for(int i=0; i<8 ;i++)
    {    pinMode(LEDdizisi[i], OUTPUT); }   // LED pinleri cikis olarak ayarlandi
}
```

```
void loop()
{
  for(int i=0; i<8; i++)    //Tum LEDleri sirayla bekle değerikadar yakip sonduruyoruz.
  {
    potoku();                //potoku fonksiyonunu çağır. Bekle değerini al.
    digitalWrite(LEDdizisi[i],1);  // ledlere "1" bilgisi gönder
    delay(bekle);
    digitalWrite(LEDdizisi[i],0);   // ledlere "0" bilgisi gönder
      }
for(int i=7;i>-1; i--)
  {
    potoku();                    //potoku fonksiyonunu çağır. Bekle değerini al.
    digitalWrite(LEDdizisi[i],1);
    delay(bekle);
    digitalWrite(LEDdizisi[i], 0);
  }
}
  void potoku()          // potoku alt programı. Burada potansiyometre hareketi ile
                         süre ayarlanıyor.
{
  PotDegeri = analogRead(pot);  // Pot ucundaki analog bilgi okunarak "PotDegeri" 'ne
                                   yüklenir.
 bekle=map(PotDegeri, 0, 1023, 20, 255); //POT değerini 0-255 arasına dönüştür.
  Serial.print("POT=");
  Serial.println(bekle);
  Serial.println();
}
```

9. LDR UYGULAMASI (ANALOG OKUMA)



Malzeme Listesi

1- Arduino Mega
2- 1 Adet LDR
3- 1 Adet 4.7K direnç

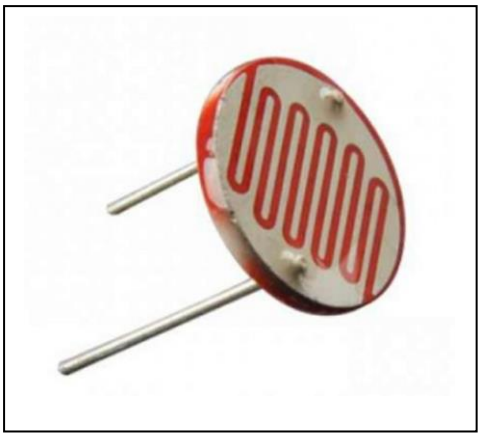


Foto dirençler, üzerlerine düşen ışık şiddetiyle ters orantılı olarak dirençleri değişen elemanlardır. Foto direnç, üzerine düşen ışık arttıkça direnç değeri lineer olmayan bir şekilde azalır. LDR'nin aydınlıkta direnci minimum, karanlıkta maksimumdur. Hem AC devrede, hem DC devrede aynı özellik gösterir. Bu iki iletken telden dışarıya uç çıkarılarak LDR'nin bağlantı terminalleri oluşturulmuştur. LDR'nin üst yüzeyi ışık etkisini algılayabilmesi için şeffaf bir malzemeyle kaplanmıştır.

//LDR sensörün aydınlıkta okunan değerini ve elimizle üstünü kapatıp karanlıktaki okunan değerini seri monitörde gösteren programı yazınız.

```
int LDR = A9 ;
int LDRDegeri;
void setup()
{
  Serial.begin(9600);
 }
void loop()
{
LDRDegeri = analogRead(LDR); // LDR ucundaki analog bilgi okunarak "LDRDegeri" 'ne yüklenir.

Serial.print("LDR = "); // Seri monitör ekranına okunan değer yazılır.
Serial.println(LDRDegeri);
delay(100);
}
```

NOT :

Burada ölçtüğünüz aydınlık ve karanlık LDR değerini bir kenara not ediniz. Bir sonraki uygulamada bu değerleri kontrol ederek uygulama yapacağız.

Benim ölçtüğüm değerler şöyle (Işık ve karanlık seviyesine göre sizinki farklılık gösterebilir.)

Aydınlık değeri = 193

Karanlık değeri = 43

10. LDR UYGULAMASI (ANALOG OKUMA VE FAR YAKMA)

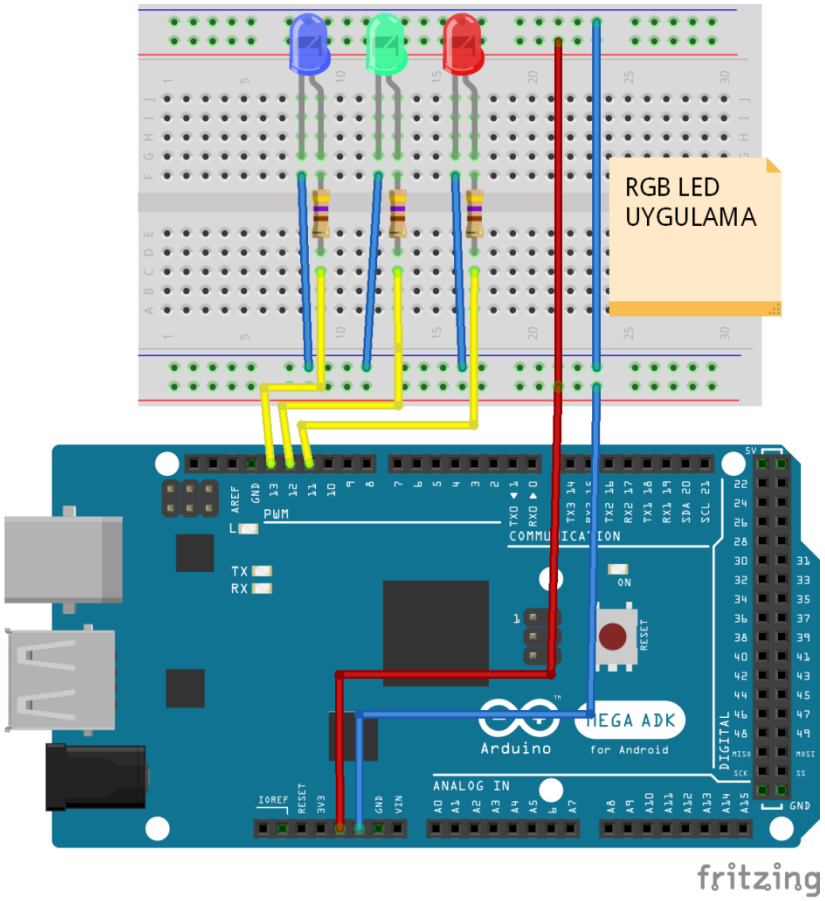
// LDR sensörünü kullanarak karanlıkta ön ve arka farları yakan, aydınlıkta söndüren programı yazınız

```
int LDR = A9 ;
int LDRDegeri;
 int arkafar=48  ,onfar= 51  ;
 void setup()
 {
 pinMode(onfar,OUTPUT);
 pinMode(arkafar,OUTPUT);
 Serial.begin(9600);
  }
 void loop()
 {
 LDRDegeri = analogRead(LDR); // LDR ucundaki analog değer okunur.
 Serial.print("LDR  =  ") , Serial.println(LDRDegeri);
 // Okunan değer seri monitör ekranına yazılır.
 if (LDRDegeri > 80) // Okunan değer 80' den büyük ise farlar söner.
                     //Değilse farlar yanar.
 {digitalWrite(onfar,0); digitalWrite(arkafar,0);}
else          .
 {digitalWrite(onfar,1); digitalWrite(arkafar,1);}
 }
```

NOT :

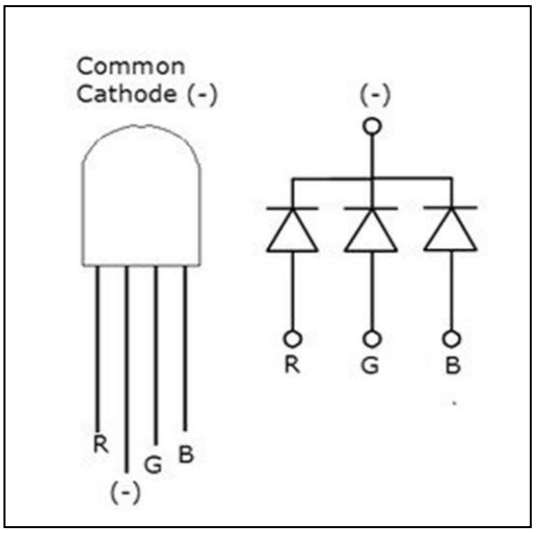
Seri monitörü açıp elinizi LDR üstüne yaklaştırıp uzaklaştırarak değerini gözlemleyiniz. Bu sırada farların durumundaki değişikliğe dikkat ediniz.

11. RGB LED UYGULAMASI



Malzeme Listesi

1- Arduino Mega
2- 3 Adet Led diyot
3- 3 Adet 470 ohm direnç



//RGB ledlerde 1500 msn aralıklarla önce kırmızı ledi, sonrasında sıra ile yeşil, mavi, kırmızı-mavi, kırmızı-yeşil, mavi-yeşil, kırmızı-mavi-yeşil ledleri birlikte yakan programı yazınız.

```
int Red= 11 , Green=12 , Blue = 13;  //Portlar tanımlandı.
int sure=1500;
void setup()
{ pinMode(Red,OUTPUT); // Tüm ledler çıkış olarak ayarlandı.
pinMode(Green,OUTPUT);
pinMode(Blue,OUTPUT);
}
```

```
void loop()
{
digitalWrite(Red , 1), digitalWrite(Green,0) , digitalWrite(Blue,0);
delay(sure);  //Kırmızı yandı.

digitalWrite(Red , 0), digitalWrite(Green,1) , digitalWrite(Blue,0);
delay(sure); //Yeşil yandı.

digitalWrite(Red , 0), digitalWrite(Green,0) , digitalWrite(Blue,1);
delay(sure); //Mavi yandı.

digitalWrite(Red , 1), digitalWrite(Green,0) , digitalWrite(Blue,1);
delay(sure); //Kırmızı + Mavi yandı.

digitalWrite(Red , 1), digitalWrite(Green,1) , digitalWrite(Blue,0);
delay(sure); //Kırmızı + Yeşil yandı.

digitalWrite(Red , 10), digitalWrite(Green,1) , digitalWrite(Blue,1);
delay(sure); //Yeşil + Mavi yandı.

digitalWrite(Red , 1), digitalWrite(Green,1) , digitalWrite(Blue,1);
delay(sure); //Kırmızı + Yeşil + Mavi yandı.
}
```

12. PWM- RGB LED UYGULAMASI

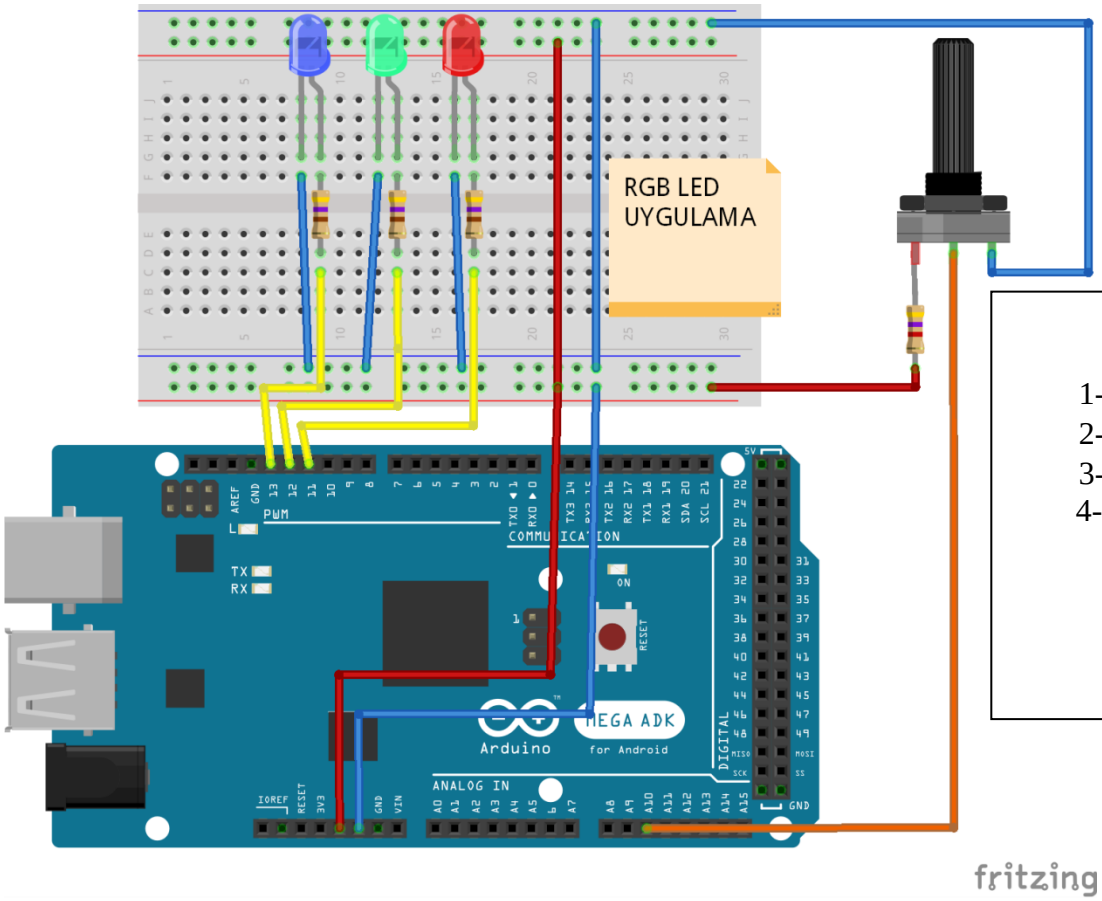
// Kırmızı ledleri analogWrite komutu kullanarak düşük parlaklıktan yükseğe gelip 2 sn. bekleyip, sonra tekrar düşük parlaklık seviyesine belirli hızda yakan programı yazınız.

```
int Red= 11 ; // 11 nolu pin tanımlandı.
int sure=20;
int i;
void setup()
{
pinMode(Red,OUTPUT); // Led çıkış olarak ayarlandı.
}
void loop()
{
for(i = 0 ; i < 256 ; i++)
    {
     analogWrite(Red,i); //Led parlaklığı 0-256 analog 20msn süre ile artar.
     delay(sure);
    }

  delay(2000);
for(i = 255 ; i > 0 ; i--)
    {
     analogWrite(Red,i); //Led parlaklığı 255-20 analog 20msn süre ile azalır..
     delay(sure);
    }
delay(2000);
}
```

13. PWM-RGB MİX UYGULAMASI

```
//RGB ledleri 2 şer 2 şer karıştırarak 256 renk elde eden programı yazınız.
int Red= 11 , Green = 12, Blue = 13 ;
int sure=20;
int i;
void setup()
{
pinMode(Red,OUTPUT) , pinMode(Green,OUTPUT) , pinMode(Blue,OUTPUT);
}
void loop()
{
for(i = 0 ; i < 256 ; i++)
    { analogWrite(Red,i); analogWrite(Blue,255-i);  delay(sure);  }
    //Kırmızıyı artırırken Maviyi azalt
 delay(2000);
for(i = 0 ; i < 256 ; i++)
    {  analogWrite(Red,255-i);  analogWrite(Green,i);  delay(sure);  }
    //Yeşili artırırken Kırmızıyı azalt
 delay(2000);

for(i = 0 ; i < 256 ; i++)
    { analogWrite(Green,255-i); analogWrite(Blue,i);  delay(sure);  }
    // Maviyi artırırken Yeşili azalt
 delay(2000);

}
```

14. PWM-RGB POT UYGULAMASI



Malzeme Listesi

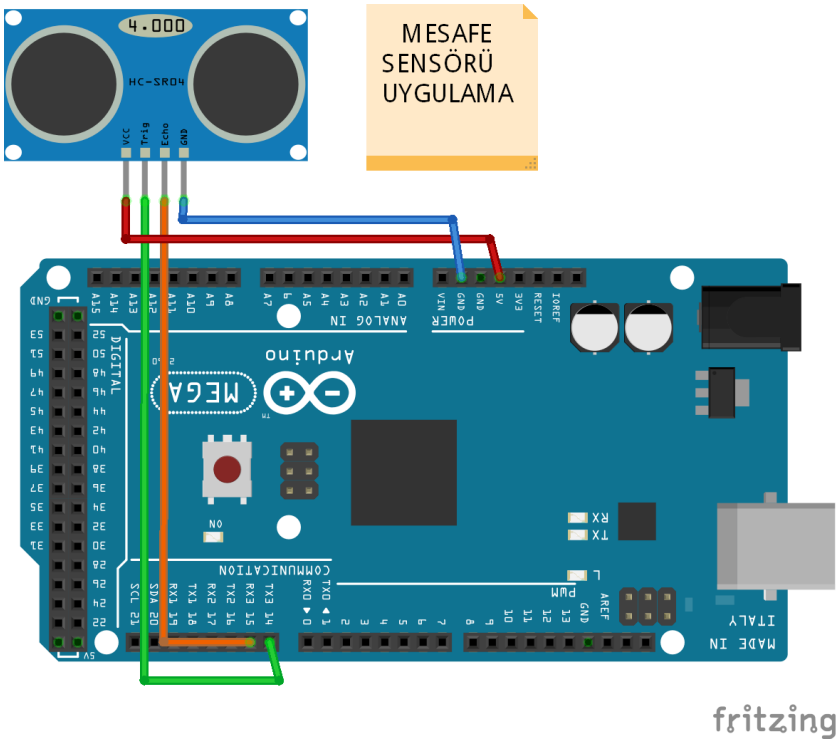
1- Arduino Mega
2- 3 Adet Led diyot
3- 1 Adet 470 Ohm direnç
4-1 Adet 10K Potansiyometre

//RGB ledleri 2 şer 2 şer karıştırarak 256 renk elde eden programın geçiş süresini potun değerine göre ayarlayan programı yazınız.

```
int Red= 11 , Green = 12, Blue = 13 , pot=A10;

int sure , potdegeri;

int i;

void setup()

{

pinMode(Red,OUTPUT) , pinMode(Green,OUTPUT) , pinMode(Blue,OUTPUT);

}
```

```
void loop()
{
for(i = 0 ; i < 256 ; i++)
    { potoku(); analogWrite(Red,i); analogWrite(Blue,255-i);  delay(sure);   }
    //Kırmızıyı artırırken Maviyi azalt
 delay(2000);

for(i = 0 ; i < 256 ; i++)
    {  potoku();  analogWrite(Red,255-i);  analogWrite(Green,i);  delay(sure);   }
    //Yeşili artırırken Kırmızıyı azalt
 delay(2000);

for(i = 0 ; i < 256 ; i++)
    { potoku();  analogWrite(Green,255-i); analogWrite(Blue,i);  delay(sure);   }
    // Maviyi artırırken Yeşili azalt
 delay(2000);
 }

void potoku()
{
  potdegeri = analogRead(pot); //Süre değerini potun konumu belirliyor
  sure=map(potdegeri, 0, 1023, 3,50); //0-1023 aralığı 3-50 arasına dönüşür.
 }
```

15. MESAFE SENSÖRÜ UYGULAMASI



Malzeme Listesi

1- Arduino Mega
2- 1 Adet HC-SR04 mesafe sensörü

HC-SR04 MESAFE SENSÖRÜ

Bu sensör, robotik projelerde Arduino ile kullanılan en popüler sensörlerden birisidir. Kullanımı oldukça kolaydır ve program kısmı doğru olduğu sürece **2cm – 400cm** arası uzaklıkları düzgün bir şekilde ölçebilmektedir.

Çalışma prensibi ise şu şekildedir:

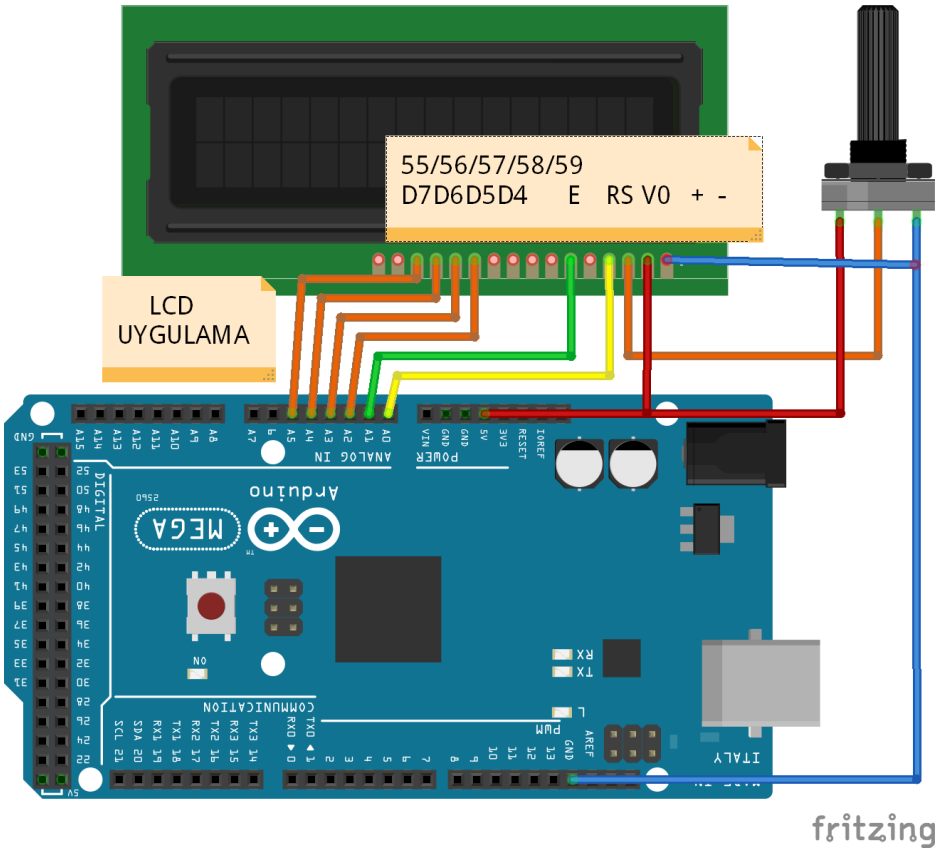
Sensörün Triger pininden uygulanan sinyal 40 kHz frekansında ultrasonik bir ses yayılmasını sağlar. Bu ses dalgası herhangi bir cisme çarpıp sensöre geri döndüğünde, Echo pini aktif hale gelir. Biz ise bu iki sinyal arasındaki süreyi ölçerek yani sesin yankısını algılayarak cismin sensörden uzaklığını tespit edebiliriz.

Arduino ile HC-SR04 sensörü düzgün bir şekilde kullanmamızı sağlayacak bir kütüphane mevcut. "**NewPing.h"** isimli kütüphanenin en güncel halini indirip Arduino yazılımına ekliyoruz.

// Mesafe sensörünün ölçtüğü değeri seri monitörde gösteren programı yazınız.

```
#include <NewPing.h> // Mesafe sensörün kütüphanesi tanıtıldı.
#define TRIGGER_PIN  14 // 14 nolu pin trigger olarak tanımlandı.
#define ECHO_PIN     15 // 15 nolu pin echo olarak tanımlandı.
#define MAX_DISTANCE 200  // Max mesafe 200.
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
  int x;

void setup()
  {
    Serial.begin(9600);
  }

void loop()
  {
  x=sonar.ping_cm(); // Ölçülen mesafe x değişkenine ata.
  Serial.print("mesafe  = "); // Ölçülen mesafeyi seri monitör ekranına yaz.
  Serial.print(x);
 Serial.println("cm");
delay(200);
  }
```

16. LCD DISPLAY UYGULAMASI



Malzeme Listesi

1- Arduino Mega
2- 1 Adet LCD Display
3- 1 Adet 10K potansiyometre

LCD ekran 5 volt ile çalışmaktadır. VCC ve GND bağlantıları buna göre yapılmalıdır. LCD'nin Vo bağlantısı, ekran üzerinde oluşacak karakterlerin görünürlüğünü ayarlamaktadır. Bu ayar ortama ve üretici firmaya göre değiştiği için Vo pini potansiyometreye bağlanır. Potansiyometrenin diğer iki ucu 5 volt ve GND'ye bağlanır. Böylece potansiyometre ile yazıların görünürlüğü ayarlanabilir. Eğer bu bağlantı düzgün bir şekilde yapılmaz ise ekran üzerinde görüntü oluşmayacaktır.

Tanımlanmış karakterleri kullanabilmeniz için öncelikle LCD kütüphanesini **'LiquidCrystal.h'** projenize eklemelisiniz. Kütüphane eklendikten sonra LCD'ye bağlanan Arduino pinleri programda belirtilmelidir. Setup fonksiyonu içerisinde LCD türünü de belirttikten sonra LCD ekran kullanıma hazırdır.

Önemli LCD Fonksiyonları:

- **lcd.begin(sutun_sayisi, satir_sayisi):** LCD ekranın tanınması için setup fonksiyonu içerisinde kullanılır. LCD kurulumu için fonksiyona sütun ve satır sayısı eklenmelidir.
- **lcd.print("MERHABA"):** LCD ekrana yazı yazdırmak için kullanılır.
- **lcd.setCursor(sütun_sayısı, satır_sayısı):** LCD ekran üzerinde imlecin yerini ayarlamak için kullanılır. Sütun ve satır sayıları 0'dan başlamaktadır. Örneğin alt satıra inmek için fonksiyon içerisine (0,1) yazılmalıdır. Böylece imleç, 0. sütun ve 1. satıra gidecektir. İmlecin yeri ayarlandıktan sonra yazma işlemi, imlecin bulunduğu yerden başlar.
- **lcd.clear():** LCD ekranda yazan her şeyi siler ve imleci en başa alır.

//LCD ekranın 1. Satırına "KODLAMAYA", 2. Satırına "HOSGELDINIZ" yazdıran programı yazınız.

```
#include <LiquidCrystal.h> // LCD ekran sensörün kütüphanesi tanıtıldı.
LiquidCrystal lcd(54,55,56,57,58,59); // LCD 'nin bağlı olduğu portlar tanımlandı.

void setup()
{
lcd.begin(16, 2);  // LCD 'nin 2 satır ve 16 sütun olduğu tanımlandı.
Serial.begin(9600);
}

void loop()
{
lcd.setCursor(0, 0);              //1.satırın 1. Karakterinden itibaren yaz.
lcd.print("KODLAMAYA");           //Metin LCD ekranına yazılır.
lcd.setCursor(0, 1);              //2.satırın 2. Karakterinden itibaren yaz.
lcd.print("HOSGELDINIZ");
}
```

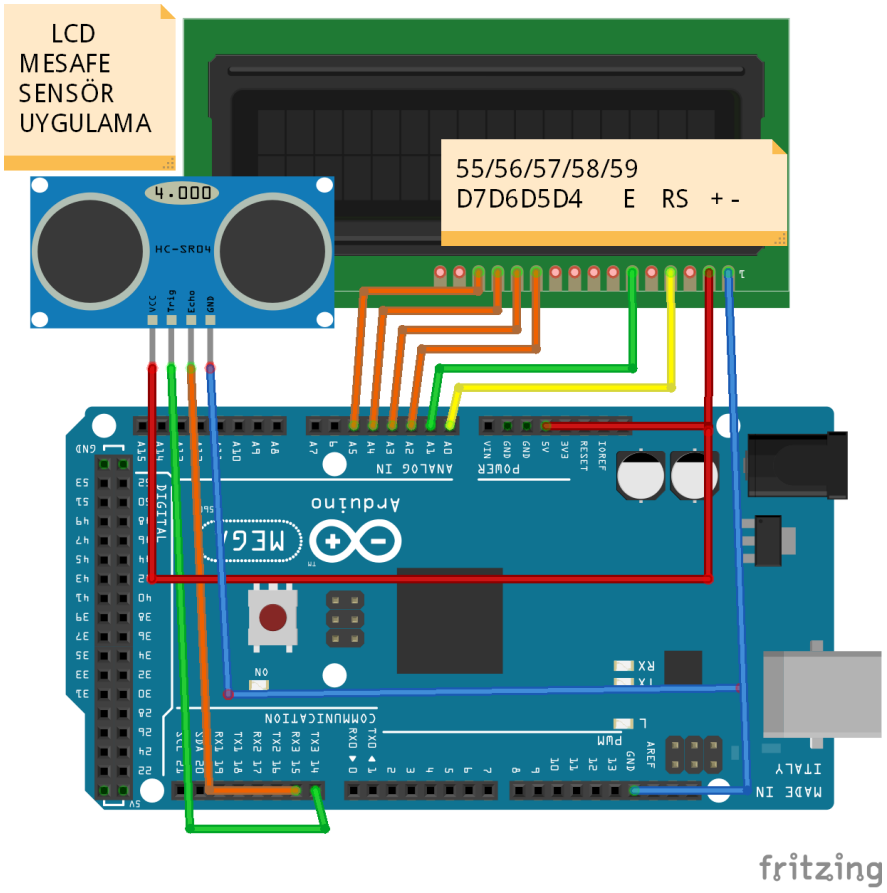
17. LCD – SAAT UYGULAMASI

// LCD ekranın 1. Satırına saat,dakika,saniye değerlerini yazıp saati çalıştıran programı yazınız.

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(54,55,56,57,58,59);
void setup()
{
lcd.begin(16, 2);
Serial.begin(9600);
saniye=0,dakika=0,saat=0; // Başlangıç süreleri sıfırlandı.
}
void loop()
{
      lcd.clear();    //LCD ekran temizlendi.
      for(saat=0;saat<24;saat++)
      {
       lcd.setCursor(0, 0);lcd.print("SAAT   "); // 0.sütun 1. Satıra SAAT yazar.
        lcd.setCursor(8, 0);lcd.print(saat); // 8.sütun 1. Satıra SAAT değerini yazar.

                for( dakika=0;dakika<60;dakika++)
                {
                 lcd.setCursor(0, 0);lcd.print("SAAT   ");
                lcd.setCursor(8, 0);lcd.print(saat);
                lcd.setCursor(10, 0); lcd.print(":"); // 10.sütun 1. Satıra ":" yazar
                    lcd.setCursor(11, 0);lcd.print(dakika);  // 10.sütun 1. Satıra dakika
                                                              değerini yazar.
```

```
for(saniye=0;saniye<60;saniye++)
                {
                 lcd.setCursor(0, 0);
                  lcd.print("SAAT    ");
                  lcd.setCursor(13, 0);
                  lcd.print(":");          // 13.sütun 1. Satıra ":" yazar
                    lcd.setCursor(14, 0);   // 14.sütun 1. Satıra saniye değerini
                                            yazar
                 lcd.print(saniye);
                 delay(1000);
                }
          }
     }
}
```

18. LCD - MESAFE SENSÖRÜ UYGULAMASI



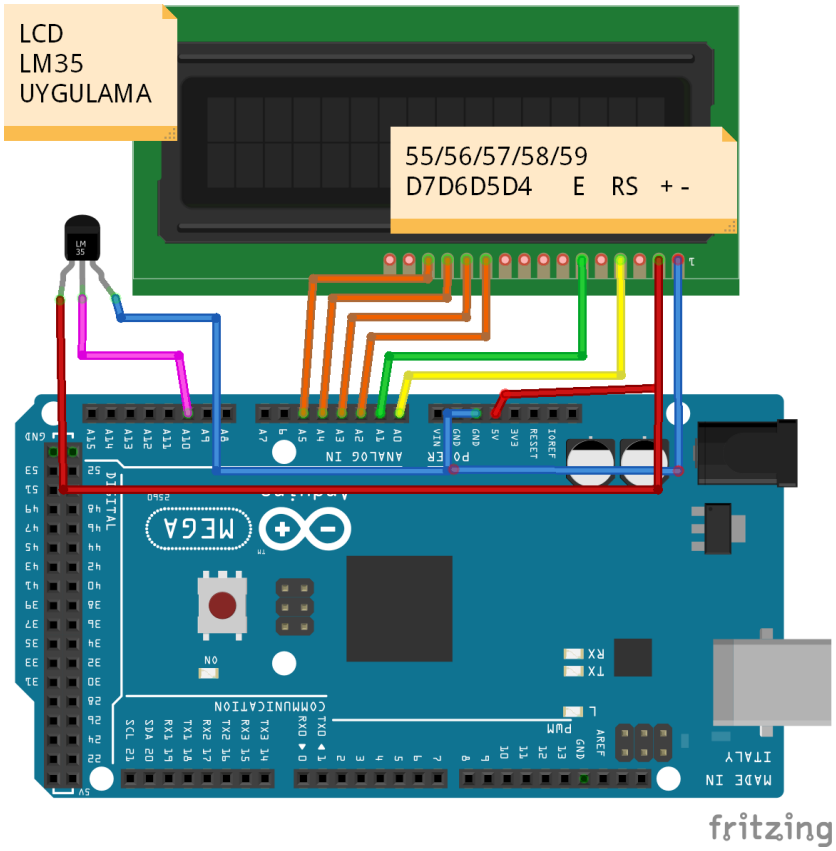
Malzeme Listesi

1-Arduino Mega
2- 1 Adet HC-SR04
3- LCD Display

//mesafe sensöründen gelen bilgiyi LCD ekranda gösteren programı yazınız.

```
#include <LiquidCrystal.h> // LCD ekran sensörün kütüphanesi tanıtıldı.
LiquidCrystal lcd(54,55,56,57,58,59); // LCD 'nin bağlı olduğu portlar tanımlandı.

 #include <NewPing.h> // Mesafe sensörün kütüphanesi tanıtıldı.
 #define TRIGGER_PIN  14 // 14 nolu pin trigger olarak tanımlandı.
 #define ECHO_PIN     15 // 15 nolu pin echo olarak tanımlandı.
  #define MAX_DISTANCE 200  // Max mesafe 200.
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing pin ayarları.
 int x;

void setup()
{
 Serial.begin(9600);
 lcd.begin(16, 2);
}
void loop()
{
 x=sonar.ping_cm();  // Ölçülen mesafe x değişkenine ata.
 lcd.setCursor(0, 0);
 lcd.print("MESAFE  = ");  // Ölçülen mesafeyi LCD ekranına yaz.
 lcd.print(x);
 lcd.print("cm   ");
 delay(500);
}
```

19. LCD - LM35 UYGULAMASI



Malzeme Listesi

1-Arduino Mega
2- 1 Adet LM35 ısı sensörü
3- 1 Adet LCD Display

Ortamın sıcaklığını ölçmeye yarayan LM35 sıcaklık sensörü analog çıkışlı bir sıcaklık sensörüdür. LM35 sıcaklık sensörü çıkış gerilimi sıcaklık ile doğru orantılı olarak değişir. Sıcaklık ölçüm aralığı -55 ile 150 derece arasında değişmektedir. 4-30 V arasında bir gerilim değeri ile beslendiğinde ve 60 mikro A'den az akım ile 0.5 derece hassasiyetle ölçüm yapabilmektedir.

Her bir derece için çıkış değeri 10mV değişim gösterir. Sıcaklık sensöründe ölçüm yapılabilmesi için sensörün üzerinde yazıların bulunduğu tarafın sol kısmına güç hattı sağ kısmına da toprak hattı bağlanır. Orta kısımda bulunan bacak analog çıkış verdiği için Arduino kartı üzerinde bulunan analog giriş pinleri ile bağlantısı kurulur.

//LCD ekranın 1. Satırına "ORTAM", 2. Satırına "SICAKLIK " ve yanına LM35 ' den okuduğu değeri yazdıran programı yazınız.

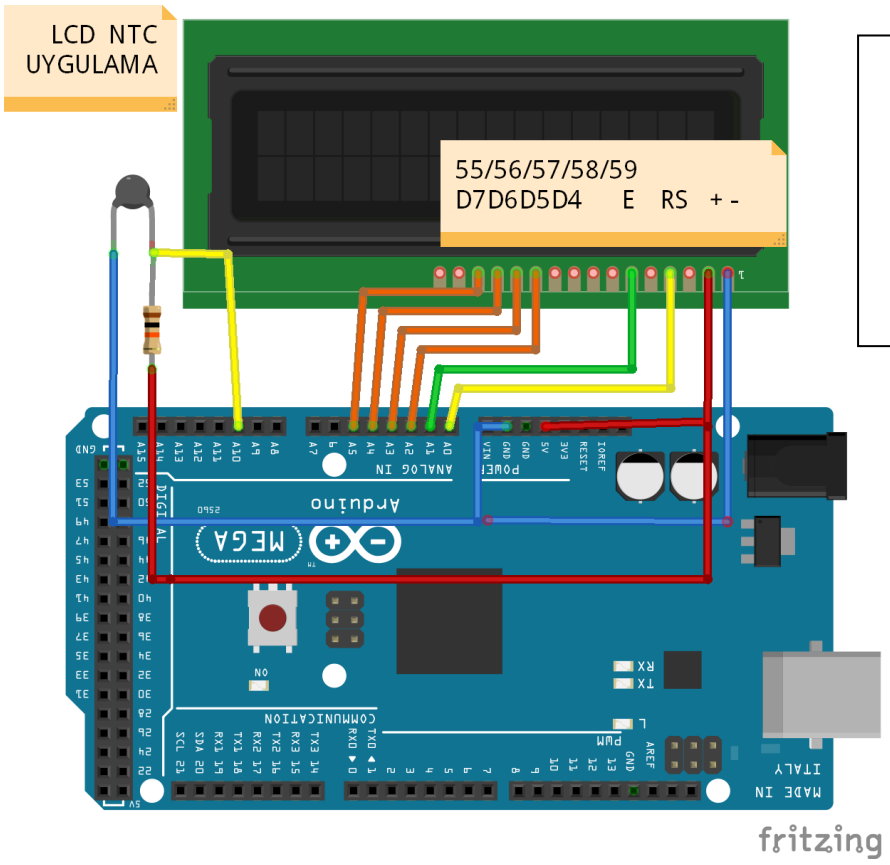
```
#include <LiquidCrystal.h>
LiquidCrystal lcd(54,55,56,57,58,59);
int deger;
float sicaklikC;

void setup()
{
  Serial.begin(9600);
  lcd.begin(16,2);
}

void loop()
{
 temphesap();  // Sıcaklık hesaplama alt programa git.
  lcd.setCursor(0, 0);     lcd.print("     ORTAM");
  lcd.setCursor(0, 1);     lcd.print("SICAKLIK  ");
  lcd.setCursor(11,1); lcd.print(sicaklikC,1 );lcd.print(" C ");
  delay(500);
}

void temphesap()
{
    sicaklikVolt = analogRead(lm35); //LM35 ısı sensörü çıkışındaki analog sinyali oku.
    sicaklikVolt = (sicaklikVolt / 1023) * 5000; //0 değeri = 0V 1023 değeri = 5V. 0(sıfır) ile 1023 değeri arasında okunan sensör değerinin 0 ile 5000mV değeri arasındaki karşılığı nedir?(1v=1000mV) gerilimDeger = (sensorDeger/1023)*5000;
    sicaklikC = sicaklikVolt / 10;  // Okunan değeri hesapla.
    lcd.setCursor(0, 0);     lcd.print("     ORTAM"); // Değeri LCD ekrana yaz.
    lcd.setCursor(0, 1);     lcd.print("SICAKLIK  ");
    lcd.setCursor(11,1); lcd.print(sicaklikC );lcd.print(" C ");

     }
```

20. LCD - LM35 UYGULAMASI (FAR YAKMA)

//LM35 sıcaklık sensöründen okunan değer 28 derecenin üzerinde ise ön ve arka farları yakan, altında ise söndüren programı yazınız.

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(54,55,56,57,58,59);
int far=51;
int arkafar=50;
float  sicaklikVolt;
int sicaklikC;
int lm35=A10;
void setup()
{pinMode(far,OUTPUT);  pinMode(arkafar,OUTPUT); lcd.begin(16, 2);  }
void loop()
{   temphesap();
    delay(1000);
    if(sicaklikC > 28) // Ölçülen sıcaklık 28' den büyük ise farları yak değilse söndür.
    {digitalWrite(far,1),digitalWrite(arkafar,1);}
    else
    {digitalWrite(far,0),digitalWrite(arkafar,0);}
}
void temphesap()
{   sicaklikVolt = analogRead(lm35);
    sicaklikVolt = (sicaklikVolt / 1023) * 5000;
    sicaklikC = sicaklikVolt / 10;
    lcd.setCursor(0, 0);     lcd.print("     ORTAM");
    lcd.setCursor(0, 1);     lcd.print("SICAKLIK  ");
    lcd.setCursor(11,1); lcd.print(sicaklikC );lcd.print(" C ");   }
```

21. LCD - NTC UYGULAMASI



Malzeme Listesi

1-Arduino Mega
2- 1 Adet NTC ısı sensörü
3- 1 Adet LCD Display

Negatif Katsayılı Direnç (NTC), sıcaklığın artmasıyla elektriksel direncin azaldığı bir tür termistor veya termal dirençtir. Bu, NTC'nin sıcaklık değişikliklerini algılamak ve ölçmek için kullanıldığı birçok uygulama için önemlidir. NTC'ler, elektronik cihazlarda sıcaklık izleme, sıcaklık telafisi ve sıcaklık kompanzasyonu gibi bir dizi uygulamada yaygın olarak kullanılırlar.

NTC'lerin temel özelliği, sıcaklığın artmasıyla direncin azalmasıdır. Bu, NTC'nin sıcaklık değişikliklerini hassas bir şekilde algılayabilmesini sağlar. Bu nedenle, birçok termal kontrol ve ölçüm uygulamasında kullanılırlar. Örneğin, bir termal koruma devresi bir cihazın aşırı ısınmasını önlemek için NTC'leri kullanabilir. Sıcaklık arttıkça NTC'nin direnci azalır ve bu da devreyi kapatır veya sıcaklığı düşürmek için başka bir işlemi başlatır.

NTC'lerin direnç-ısı karakteristiği, belirli bir NTC tipinin spesifikasyonlarına bağlı olarak değişebilir. Bu nedenle, bir uygulama için doğru NTC tipini seçerken, sıcaklık aralığı, direnç değerleri ve hassasiyet gibi faktörler göz önünde bulundurulmalıdır.

//LCD ekranın 1. Satırına "ORTAM", 2. Satırına "SICAKLIK " ve yanına NTC ' den okuduğu değeri yazdıran programı yazınız.

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(54,55,56,57,58,59);
int deger;
float sicaklik;

void setup()
{
  Serial.begin(9600);
  lcd.begin(16,2);
}

void loop()
{
  NTCOKU(); // Analog değeri okuyup sıcaklık karşılığını bulan alt program.
  lcd.setCursor(0, 0);     lcd.print("     ORTAM");
  lcd.setCursor(0, 1);     lcd.print("SICAKLIK  ");
  lcd.setCursor(11,1); lcd.print(sicaklik,1 );lcd.print(" C ");
  delay(500);
}

void NTCOKU()
 {
deger = analogRead(A10); //A10  portuna bağlı NTC ucunda analog değer okunur.
sicaklik = log(((10240000 / deger) - 10000));
sicaklik = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * sicaklik * sicaklik )) * sicaklik );
sicaklik = sicaklik - 273.15; // Okunan değer hesaplanarak sıcaklık değeri oluşur.
}
```

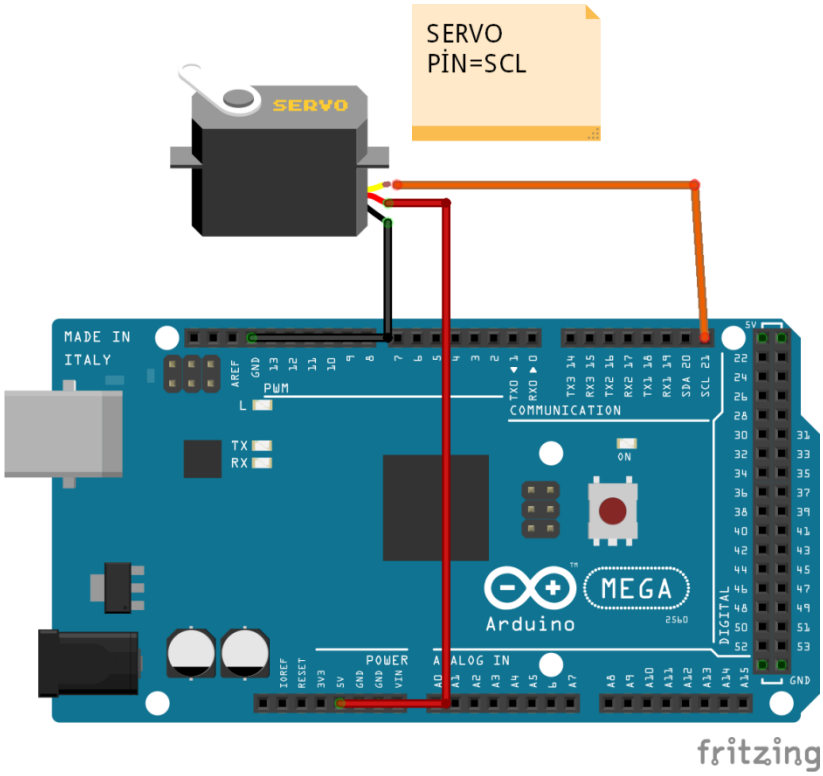
22. LCD - NTC UYGULAMASI (FAR YAKMA)

//NTC sıcaklık sensöründen okunan değer 25 derecenin üzerinde ise ön ve arka farları yakan, altında ise söndüren programı yazınız.

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(54,55,56,57,58,59);
int far=51,arkafar=48;
int deger;
float sicaklik;
void setup()
{ Serial.begin(9600);
  pinMode(far,OUTPUT);
  pinMode(arkafar,OUTPUT);
  lcd.begin(16,2);
}
void loop()
{
  NTCOKU();
  lcd.setCursor(0, 0);     lcd.print("     ORTAM");
  lcd.setCursor(0, 1);     lcd.print("SICAKLIK  ");
  lcd.setCursor(11,1); lcd.print(sicaklik,1 );lcd.print(" C ");
  if(sicaklik > 25)
    {digitalWrite(far,1),digitalWrite(arkafar,1);}
    else
    {digitalWrite(far,0),digitalWrite(arkafar,0);}
delay(500);
}
```

```
void NTCOKU()
  {
deger = analogRead(A10);
sicaklik = log(((10240000 / deger) - 10000));
sicaklik = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * sicaklik * sicaklik )) * sicaklik );
sicaklik = sicaklik - 273.15;
}
```

23. SERVO MOTOR UYGULAMASI



Malzeme Listesi

1-Arduino Mega
2- 1 Adet Servo motor

Servo motor 0 ila 180 derece arasında 1 derece hassasiyetle dönebilen motor çeşididir. Tam tur atamaz. Genellikle robot kol gibi tam tur dönmesine gerek olmayan, hassas açılı yerlerde kullanılır. Servo motor içerisinde bir adet DC motor bulunur. DC motorun ucuna bağlı dişli sisteminin yardımıyla servo mili daha fazla yük kaldırabilmektedir. Bu işlem sırasında servonun dönüş hızı da yavaşlamış olur. Kullanılan dişli sistemine göre servo motorların kaldırabileceği yük değişir. Kırmızı renk besleme (genellikle 5 volt) bağlantısını, siyah veya kahverengi renk de toprak bağlantısını göstermektedir. Geriye kalan turuncu kablo ise motorun açısını belirleyecek olan veri bağlantısıdır

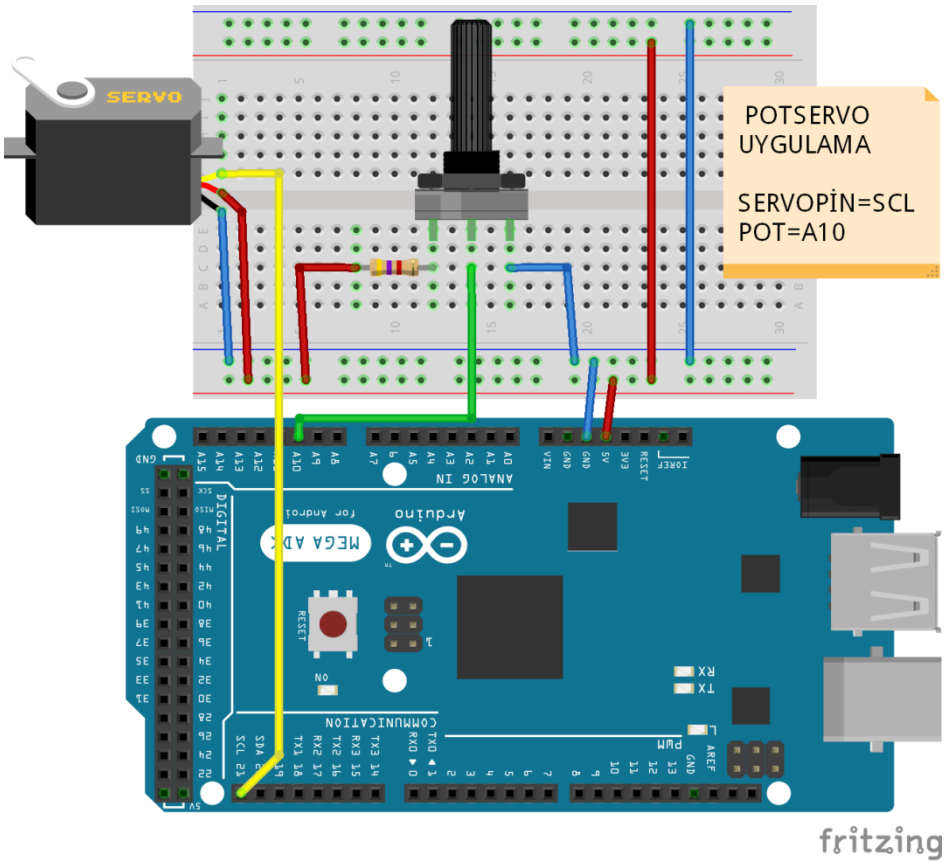
Arduino'da servo motor kontrolü için özelleştirilmiş PWM pinleri bulunmaktadır. PWM pin sayısı Arduino'nun türüne göre değişmektedir. Bu pinlerin yanında dalga (~) işareti bulunmaktadır.

Servo motor kontrolü için öncelikle **Servo.h** kütüphanesini projemize eklemeliyiz. Servo kütüphanesi eklendikten sonra Servo nesnesi kullanılarak yeni servo motorlar tanımlanır. Tanımlanan servo motorın bağlı olduğu pinler seçilir ve servo kullanıma hazır hale getirilir. Motor milinin konumunu değiştirmek için **Servo nesnesinin attach** metodu kullanılır. Bu metodun içerisine motor milinin gitmesi istenilen 0-180 derece arasında açı yazılır. Servonun yeni konumunu alması biraz zaman alabilir. Bu yüzden bekleme **(Delay)** komutu kullanılmalıdır.

//Servo motor 100 derece döndükten sonra 1sn beklesin ve 20 derece döndükten sonra 1sn bekleyen programı yazınız.

```
#include <Servo.h>  /* Servo kutuphanesi projeye dahil edildi */
Servo servoNesnesi;  /* servo motor nesnesi yaratildi */

void setup()
{
 servoNesnesi.attach(SCL);  /* Servo motor scl numarali pine baglandi */
}

void loop()
{
 servoNesnesi.write(100);  /* Motorun mili 100. dereceye donuyor */

 delay(1000);

servoNesnesi.write(20);   /* Motor mili 20. dereceye donuyor */

delay(1000);
}
```

24. SERVO MOTOR - POT UYGULAMASI



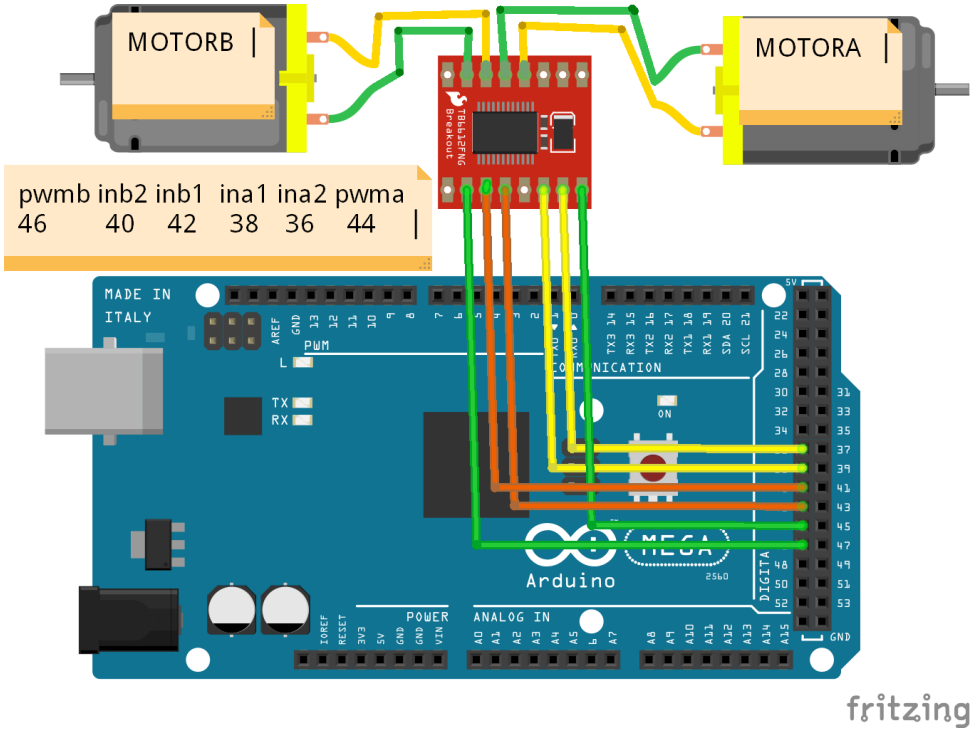
Malzeme Listesi

1-Arduino Mega
2- 1 Adet Servo motor
3- 1 Adet Potansiyometre

//A10 portuna bağlı potansiyometre ile servo motoru 0-180 derece aralığında döndüren programı yazınız.

```
  #include <Servo.h> /* Servo kutuphanesi projeye dahil edildi */
int potPin = A10; // A10 analog pini  potPin değişkenine tanımlandı.

int servoPin = SCL; // Servo motora açı bilgilerini göndereceğimiz SCL pini tanımlandı

Servo servo;  // servo ismiyle servo motorumuzu tanıttık.

void setup() {

  pinMode(potPin, INPUT); // potPin değişkenini giriş olarak ayarladık.

servo.attach(SCL);  // Servo'nun sinyali alacağı Arduino Pinini belirledik

}

 void loop() {

int okunan = analogRead(potPin); // 0'dan 1023'e, Pottan okunan değer

int aci = map(okunan, 0, 1023, 0, 180); // 0'dan 180 dereceye Açıya dönüştürülüyor.

servo.write(aci); }   // aci değişkeni ile açı bilgilerini servo motora gönderiyoruz.
```

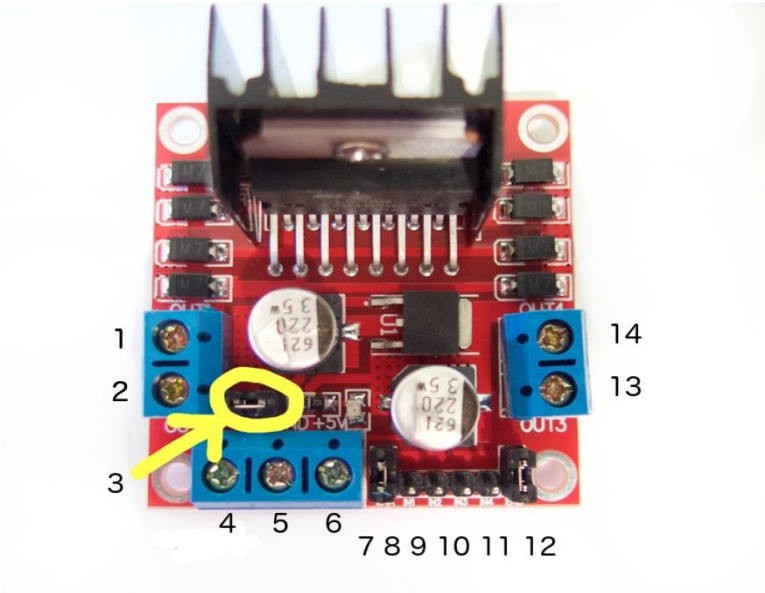
25. MOTOR SÜRÜCÜ UYGULAMASI



Malzeme Listesi

1-Arduino Mega
2- 2 adet DC motor
3- 1 Adet DC motor sürücü

L298 SÜRÜCÜ DEVRESİ



4- DC "+" 5- DC "- "

1- DC motor 1 "+" veya stepper motor A+

2- DC motor "-" veya stepper motor A-

13- DC motor 2 "+" veya stepper motor B+

14- DC motor 2 "-" veya stepper motor B-

7- ENA, DC motor 1 'in PWM çıkışı ile DC motorun hız kontrolü yapılır.

12- ENB, PWM çıkışı ile 2. DC motorun hız kontrolü yapabilirsiniz

8- IN1 1.motor yön belirleme pinleri / step motor komutasyon pinleri

9- IN2 1.motor yön belirleme pinleri / step motor komutasyon pinleri

10- IN3 2.motor yön belirleme pinleri / step motor komutasyon pinleri

11- IN4 2.motor yön belirleme pinleri / step motor komutasyon pinler

Motor yönü Arduino pinlerinde oluşturulan HIGH

veya LOW sinyalleri ile kontrol edilmektedir.

Örneğin motor 1 için

IN1 : HIGH IN2: LOW

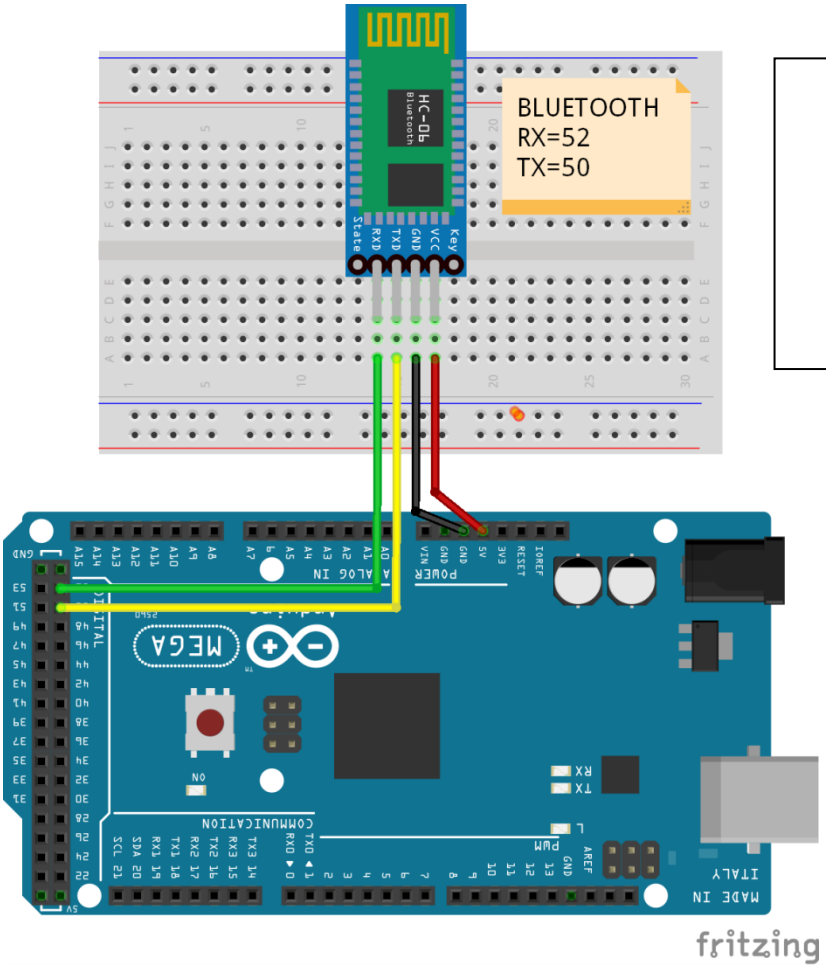
şeklinde olursa motor ileri yönde dönecektir.

IN1 : LOW IN2: HIGH

Şeklinde olursa da tam tersi yönde dönecektir.

```
// Motorlar 2 sn. ileri, 2 sn. geri, 2 sn. sağa ve 2 sn, sola hareket ettiren programı yazınız.
 int INA1=38; // Portlar tanımlandı.
 int INA2=36;
 int HIZA=44;
int INB1= 42;
int INB2=40;
int HIZB=46;
void setup()
{ // Bütün motor pinler çıkış olarak ayarlandı.
  pinMode(INA1, OUTPUT); pinMode(INA2, OUTPUT);pinMode(HIZA, OUTPUT);
pinMode(INB1, OUTPUT); pinMode(INB2, OUTPUT);pinMode(HIZB, OUTPUT);
}
void loop ()
{  // Motor 2 sn ileri hareket eder.
digitalWrite(INA1, HIGH);  digitalWrite(INA2, LOW);  analogWrite(HIZA,150);
 digitalWrite(INB1, HIGH);  digitalWrite(INB2, LOW);  analogWrite(HIZB,150);
delay(2000); // Motor 2 sn geri hareket eder.
digitalWrite(INA1, LOW);  digitalWrite(INA2, HIGH);  analogWrite(HIZA,150);
digitalWrite(INB1, LOW);  digitalWrite(INB2, HIGH);  analogWrite(HIZB,150);
delay(2000); //Motor 2 sn sağa hareket eder.
digitalWrite(INA1,LOW);  digitalWrite(INA2,HIGH);  analogWrite(HIZA,150);
digitalWrite(INB1,LOW);  digitalWrite(INB2,LOW);  analogWrite(HIZB,150);
delay(2000); //Motor 2 sn sola hareket eder.
 digitalWrite(INA1,LOW);  digitalWrite(INA2,LOW);  analogWrite(HIZA,150);
digitalWrite(INB1,LOW);  digitalWrite(INB2,HIGH);  analogWrite(HIZB,150);
 delay(2000);

}
```

26. BLUETOOTH UYGULAMASI



Malzeme Listesi

1-Arduino Mega
2- 1 Adet Bluetooth modül

Bluetooth, kablo bağlantısını ortadan kaldıran kısa mesafe radyo frekansı (RF) teknolojisinin adıdır. HC-06 modülü seri haberleşme esasına göre çalışırlar. Rx ucu veriyi alan uç iken, Tx ucu veriyi gönderen uçtur. HC-06 Bluetooth modülü 3.3V gerilim ile çalışmaktadır.Tabiki bu bluetooth modülüne bağlanmak için bir adı ve şifresinin olması gerekir. Bluetooth bağlantısı yaparken adı HC-06, şifresi 1234 veya 0000 olmaktadır ve baund hızı default olarak 9600 ayarlanmıştır.

```
//HC-06 BLUETOOTH modül uygulamasının programını yazınız.

#include <SoftwareSerial.h> //Servo kutuphanesi projeye dahil edildi
SoftwareSerial mySerial(50,52); // TX, RX BLUETOOTH MODÜL
int far = 51; // Portlar tanımlandı.
int arkafar = 48;
int sagsinyal =49;
int solsinyal = 53;
int buzzer=10;
int HIZ=200;
int INA2=36;int INA1=38;int hizA=44;
int INB2=40;int INB1=42;int hizB=46;

void setup()
{  // Portlar çıkış olarak ayarlandı.
 pinMode(far, OUTPUT);  pinMode(arkafar, OUTPUT);
 pinMode(sagsinyal, OUTPUT);pinMode(solsinyal, OUTPUT);

 pinMode(INA1, OUTPUT);pinMode(INA2, OUTPUT);pinMode(hizA, OUTPUT);
  pinMode(INB1, OUTPUT);pinMode(INB2, OUTPUT);pinMode(hizB, OUTPUT);

 pinMode(buzzer,OUTPUT);
 mySerial.begin(9600);
 mySerial.println("ARAC uygulamasi");

}
```

```
void loop()
{
    char gelenveri = mySerial.read();    // Okunan veri taranarak gelenveri karakterine atanır.
 switch (gelenveri)
{
  case'V': // gelenveri 'V' ise buzzer çalsın.
  digitalWrite(buzzer,1);
        break;

  case'v':
 digitalWrite(buzzer,0); // gelenveri 'v' ise buzzer dursun.

 break;

 case'W': // gelenveri 'W' ise far yansın.

 digitalWrite(far,1);
  break;

 case'w': // gelenveri 'w' ise far sönsün.

digitalWrite(far,0);
break;
```

```
case'U': // gelenveri 'U' ise arkafar yansın.
digitalWrite(arkafar,1);
break;

case'u': // gelenveri 'u' ise arkafar sönsün.
digitalWrite(arkafar,0);
break;

case'F': // gelenveri 'F' ise ileri hareket.
digitalWrite(INA1,1); digitalWrite(INA2,0); analogWrite(hizA,HIZ);
digitalWrite(INB1,1); digitalWrite(INB2,0); analogWrite(hizB,HIZ);
break;

case'B': // gelenveri 'B' ise geri hareket.
digitalWrite(INA1,0); digitalWrite(INA2,1); analogWrite(hizA,HIZ);
digitalWrite(INB1,0); digitalWrite(INB2,1); analogWrite(hizB,HIZ);
break;

case'R': // gelenveri 'R' ise sağ hareket.
digitalWrite(INA1,1); digitalWrite(INA2,0); analogWrite(hizA,HIZ);
digitalWrite(INB1,0); digitalWrite(INB2,0); analogWrite(hizB,HIZ);
break;

case'L': // gelenveri 'L' ise sol hareket.
digitalWrite(INA1,0); digitalWrite(INA2,0); analogWrite(hizA,HIZ);
digitalWrite(INB1,1); digitalWrite(INB2,0); analogWrite(hizB,HIZ);
break;
```

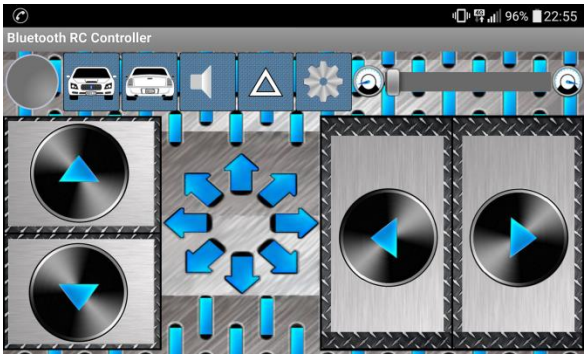
```
case'G':
digitalWrite(INA1,1); digitalWrite(INA2,0); analogWrite(hizA,HIZ);
digitalWrite(INB1,1); digitalWrite(INB2,0); analogWrite(hizB,HIZ-80);
break;

case'H':
digitalWrite(INA1,1); digitalWrite(INA2,0); analogWrite(hizA,HIZ-80);
digitalWrite(INB1,1); digitalWrite(INB2,0); analogWrite(hizB,HIZ);
break;

case'I':
digitalWrite(INA1,0); digitalWrite(INA2,1); analogWrite(hizA,HIZ-80);
digitalWrite(INB1,0); digitalWrite(INB2,1); analogWrite(hizB,HIZ);
break;

case'J':
digitalWrite(INA1,0); digitalWrite(INA2,1); analogWrite(hizA,HIZ);
digitalWrite(INB1,0); digitalWrite(INB2,1); analogWrite(hizB,HIZ-80);
break;

case'S':
digitalWrite(INA1,0); digitalWrite(INA2,0);
digitalWrite(INB1,0); digitalWrite(INB2,0);
delay(100);
break;
 }

}
```

ARDUINO BLUETOOTH RC CAR UYGULAMASI KULLANIMI

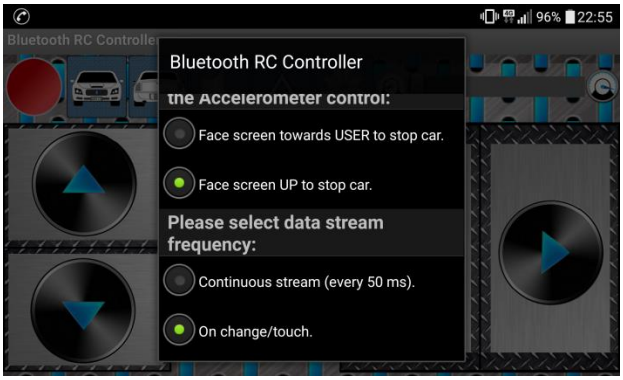
1-Play store dan bluetooth rc controller araması ile "ARDUINO BLUETOOTH RC CAR " uygulamasını indiriyoruz.

2-Cep telefonu veya tabletten Bluetooth özelliğini açıp arama yaptırıyoruz. Bulunan HC06 modülüne tıklayarak HC06 yı eşleştiriyoruz. Şifre = 1234

3-BLUETOOTH RC CONTROLLER programı çalıştırdığımızda aşağıdaki ekranı görüyoruz.



4-Settings 'den ayarları aşağıdaki gibi seçerek gerçekleştiriyoruz ve sayfanın en sonunda "ok " tuşuna basıp ayarları kaydediyoruz.



5-Son olarak CONNECT TO CAR komutu ile HC 06 yı seçip araçtaki bluetooth modülüne bağlanıyoruz.

KAYNAKÇA

1- Arduino İle Uzaktan Kontrol Sitemleri AHMET RAŞİTPETEKÇİ

2- HOŞGÖREN Mehmet, Mikroişlemciler, Ankara, 2003.

3-TOPBAŞOĞLU Oktay, Robotik ve Kodlama Ders Notları

4- http://www.megep.meb.gov.tr/

5- https://www.arduino.cc/

6-Vikipedia

7-https://fritzing.org/