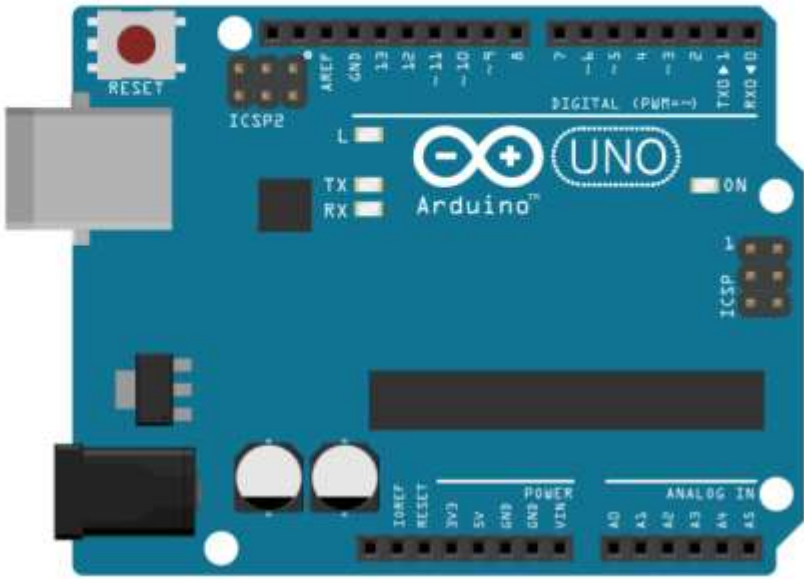
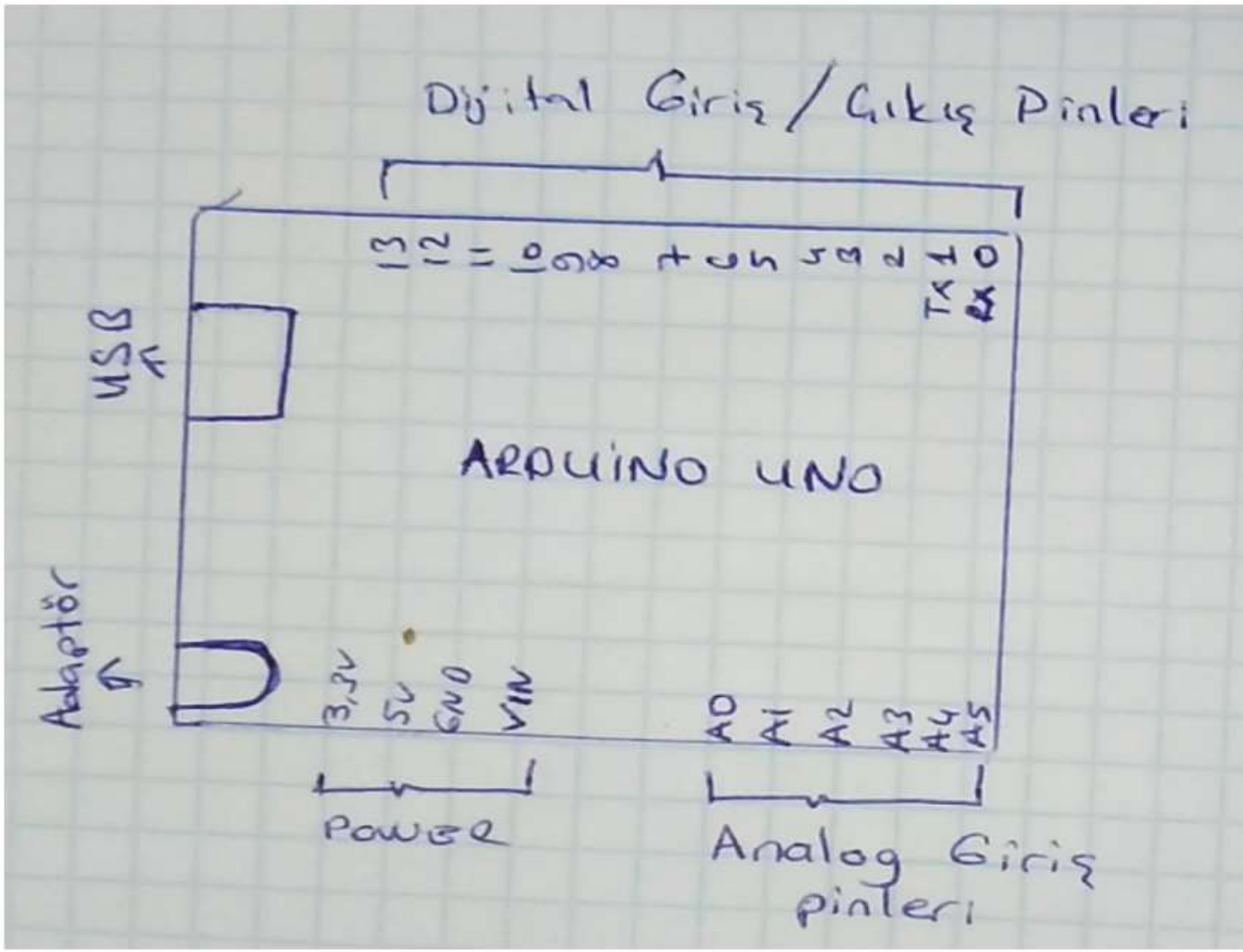
Ardunio

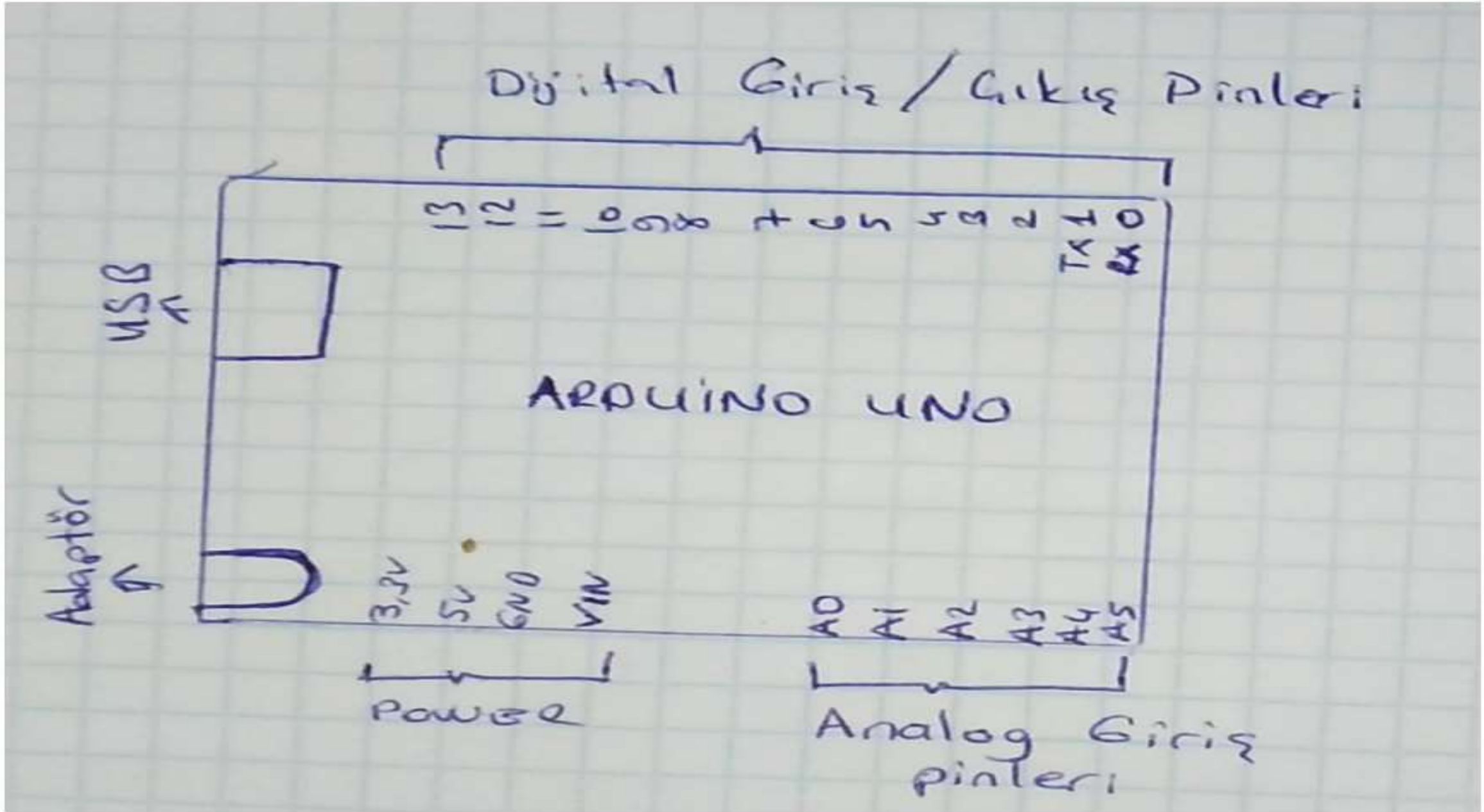
Arduino UNO Tanıyalım



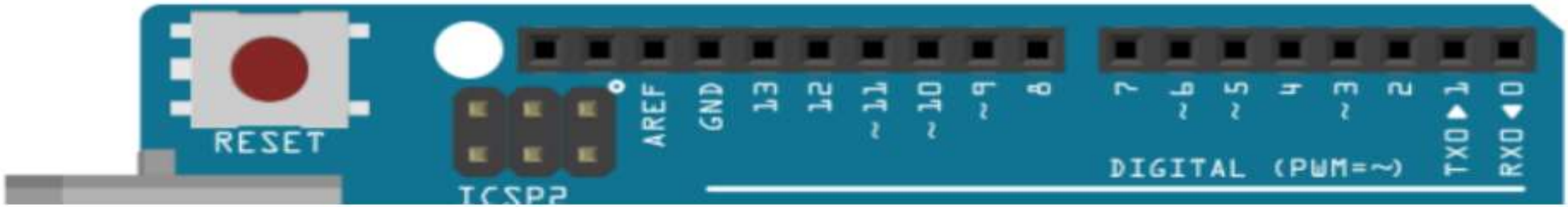
Arduino Taslak Çizimi



Arduino Taslak Çizimi



Dijital Giriş/Çıkış Pinleri
(0-13 arası pinler)



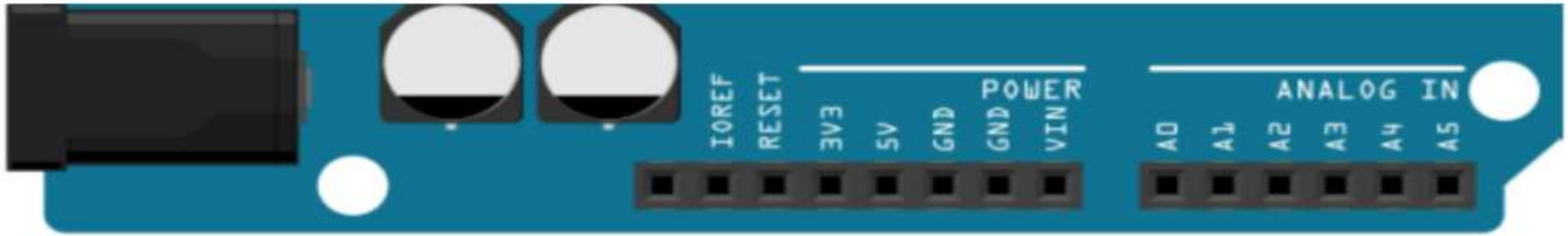
14 adet dijital pin bulunur. Bu pinler genellikle bağlı oldukları led, step motor, LCD ekran, buzzer gibi bileşenlere kontrollü bir şekilde elektrik akımı vermek(çıkış-OUTPUT) için kullanılır. Bu pinlerden dijital veri olarak sadece 0 ve 1 değeri alınır.

Ayrıca Button gibi devre elemanları ile kullanılarak 0 ve 1 verisi girilebilir(INPUT).

0 dijital verisi 0 Voltu temsil ederken

1 dijital verisi ise 5 Voltu temsil eder.

Analog Pinler(A0-A5)



Arduino UNO kartında 6 adet analog giriş pini bulunur. Analog giriş pinlerine çeşitli sensörler(LDR, LM35 TCRT5000 gibi) bağlanarak dış ortamdaki veriler 0-5 volt arasında değişen elektrik sinyali olarak alınır. Bu voltaj değerleri ADC denen ünite yardımı ile

0-1023 arasındaki sayıya dönüştürülür.

Örnek:

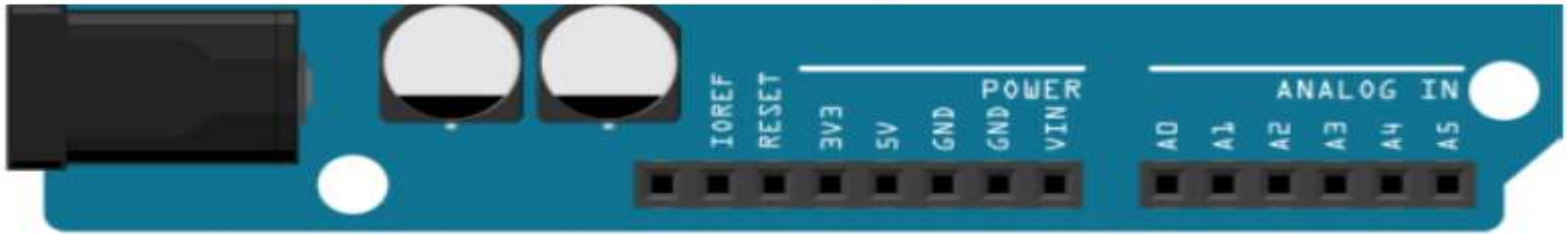
Aygıt sisteme

0 Volt gönderirse ADC birimi 0 değerini

2.5 Volt gönderirse ADC birimi 512 değerini

5 Volt gönderirse ADC birimi 1023 değerini verir

POWER-GÜÇ PİNLERİ

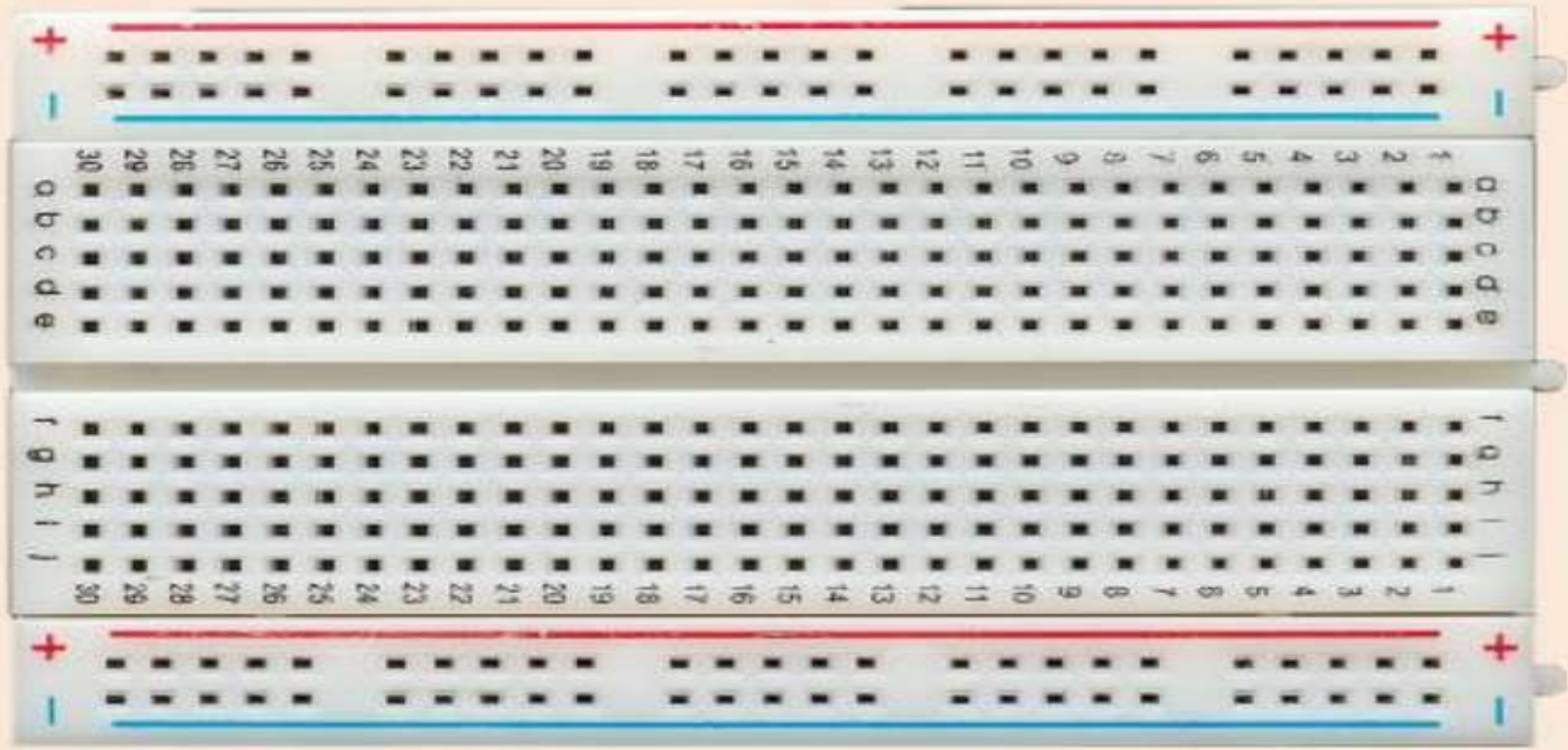


5V ve 3.3V:Arduino kartımızdan 5V ve 3.3V pinlerinden (+) Voltaj alabiliriz. Devremizi kurarken gerekli elektrik enerjisi bu pinlerden alınacaktır.

VIN: Batarya veya pil takılacaksa Voltaj girişi buradan yapılır

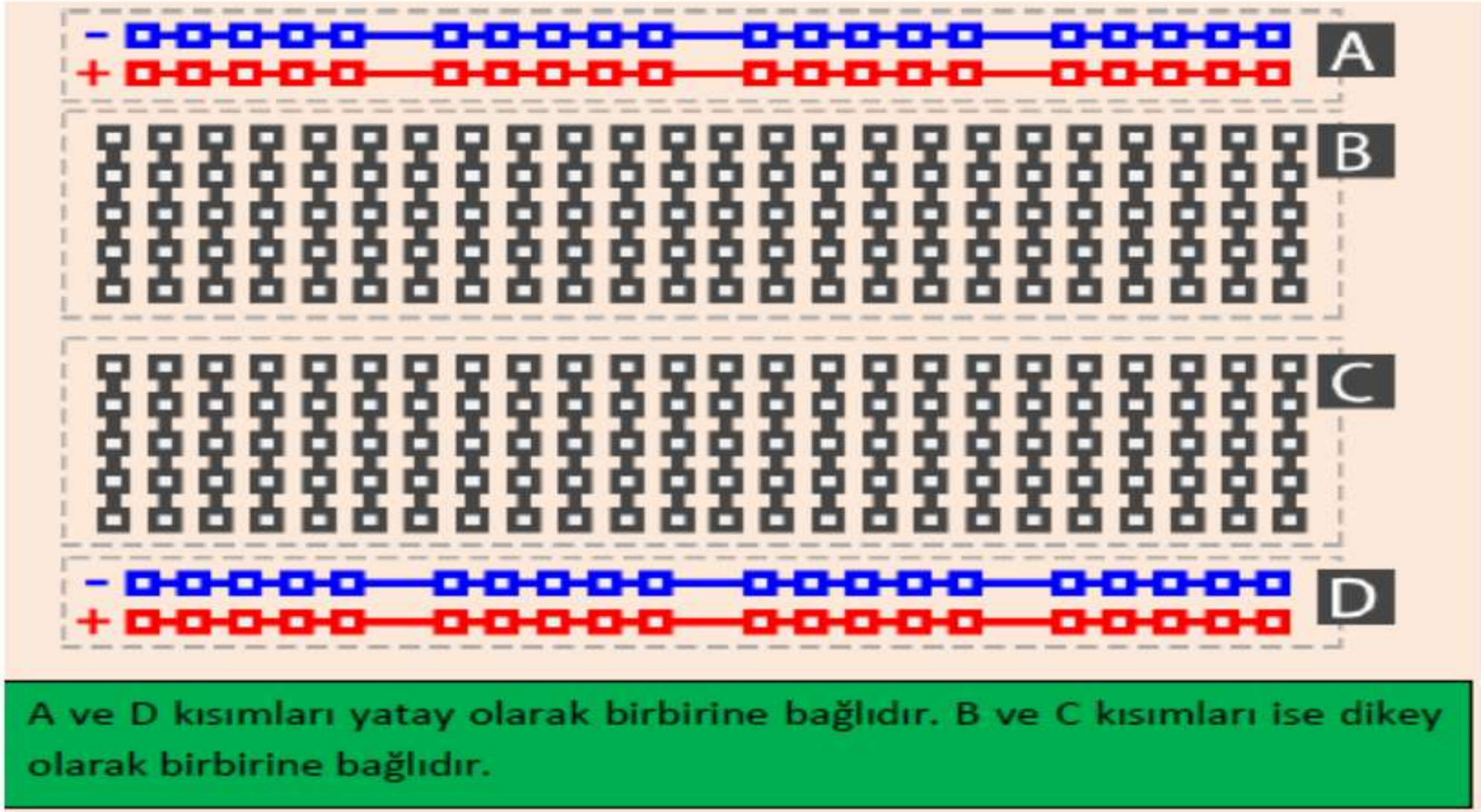
GND: Bu bin ise toprak yani(-) ucu temsil eder

BREADBOARD



Breadboard devre elemanlarını birbirine bağlamak için kullanılır. Tekrar tekrar kullabilirsiniz.

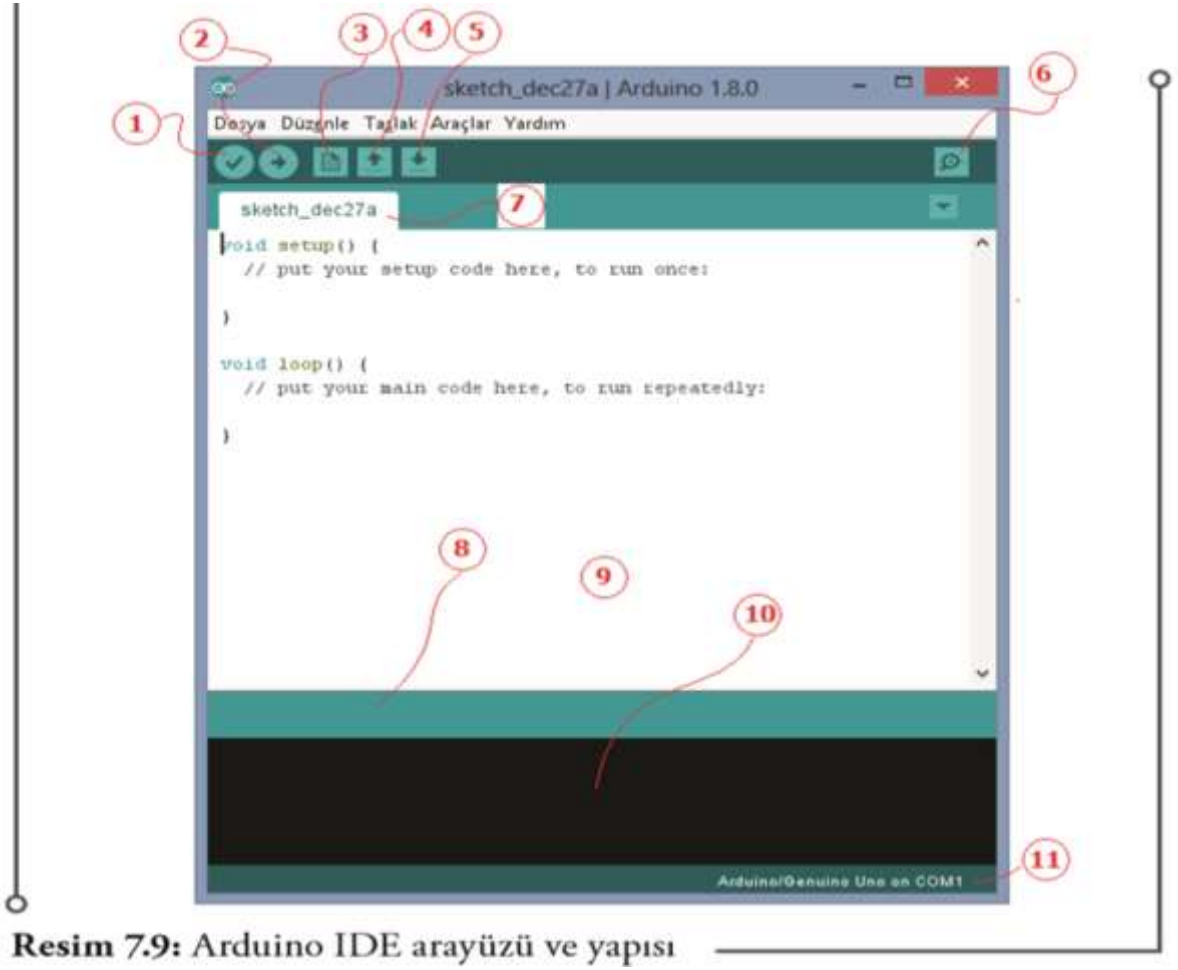
BREADBOARD YAPISI



A ve D kısımları yatay olarak birbirine bağlıdır. B ve C kısımları ise dikey olarak birbirine bağlıdır.

Arduino IDE

1. **Kontrol Et:** Yazılan kodları derler ve hataları bulur.
2. **Yükle:** Yazılan programı Arduino kartına yükler.
3. **Yeni:** Yeni çalışma sayfası açar.
4. **Aç:** Kayıtlı bir programı açar.
5. **Kaydet:** Yazılan programı kaydeder.
6. **Seri Port Ekranı:** Arduino ile seri iletişim yaparak ekran aç
7. **Sketch:** Yazılan programın dosya ismini gösterir.
8. **Gösterge:** Yaptığı işlemin ilerleme durumunu gösterir.
9. **Boş alan:** Yazılacak program alanıdır.



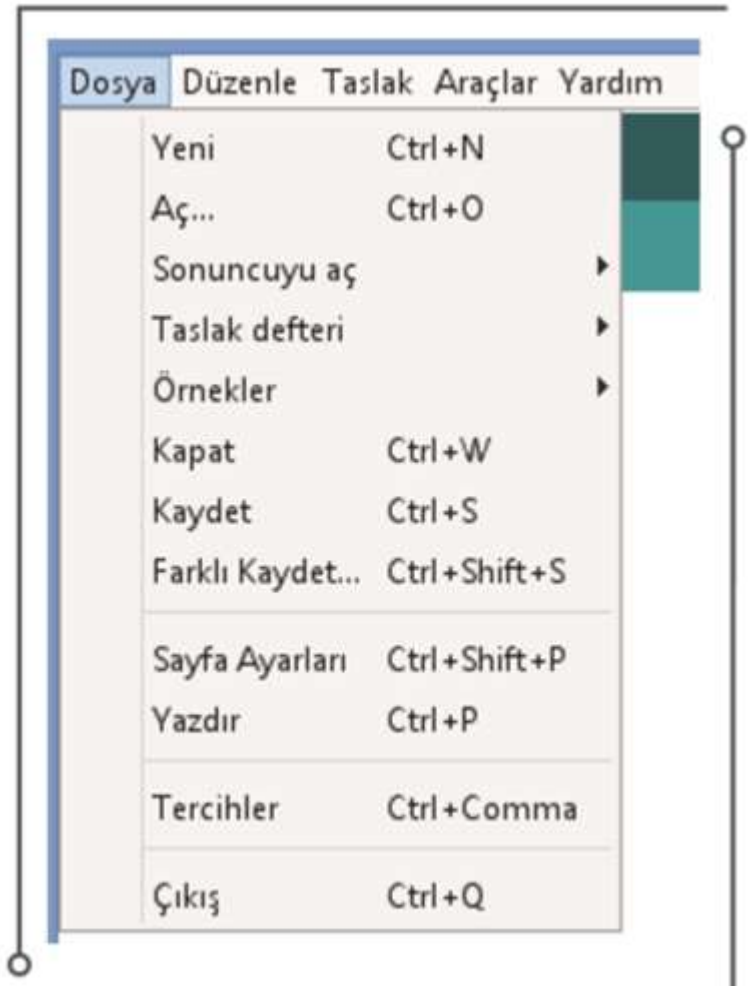
Resim 7.9: Arduino IDE arayüzü ve yapısı

10. **Rapor:** Derleme sonucu varsa yapılan hataları, yoksa programın yükleme sonrası mikrodenetleyicide kapladığı alanı gösterir.
11. **Gösterge:** Bilgisayara usb ile bağlanan Arduino'nun bağlandığı portu ve hangi Arduino modeli ile çalışılıyorsa onu gösterir.

Arduino IDE’de Menüler

Dosya Menüsü

Dosya menüsünde temel komutlar yer almaktadır. Ayrıca “Örnekler” komutu ile program kütüphanesinde yer alan kodlar incelenebilmektedir. Bu bölümde çok sayıda örnek bulunmakta, gerekli bağlantılar yapılarak görsel olarak incelenebilmekte, küçük değişiklikler yaparak etkisi görülebilmektedir. Bu bölüm yeni başlayan kullanıcılar için büyük kolaylık sağlamaktadır



Resim 7.10: Arduino IDE dosya menüsü

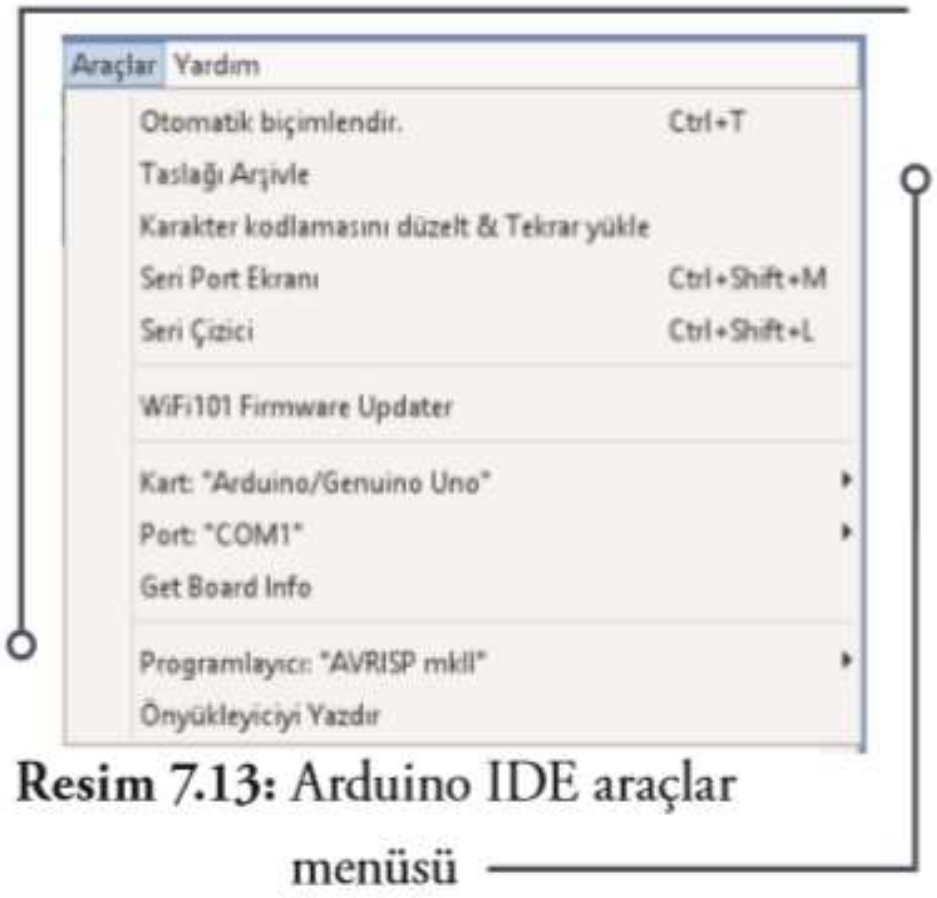
Düzenle Menüsü

Düzenle menüsünde standart işlemlerin dışında “Forum için Kopyala” ve “HTML olarak Kopyala” seçenekleri kullanılarak internet ortamında, Arduino arayüzünde yazılan biçimde (renk- yazı tipi) paylaşabilmektedir.

Resim 7.11: Arduino IDE düzenle menüsü

Taslak Menüsü

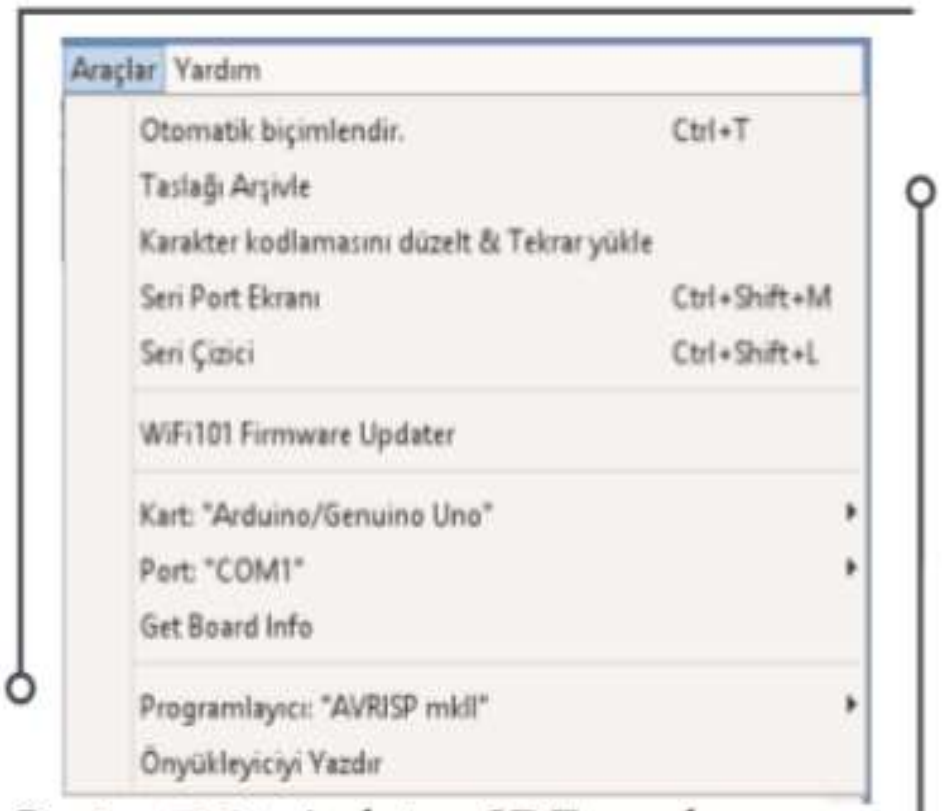
Taslak menüsünde standart işlemlerin dışında “library ekle” sekmesini kullanarak istenilen kütüphane otomatik olarak çalışılan koda eklenebilmektedir. Ayrıca başka bir kütüphane eklemek için “Dosya Ekle” kullanılabilmektedir.



Resim 7.13: Arduino IDE araçlar menüsü

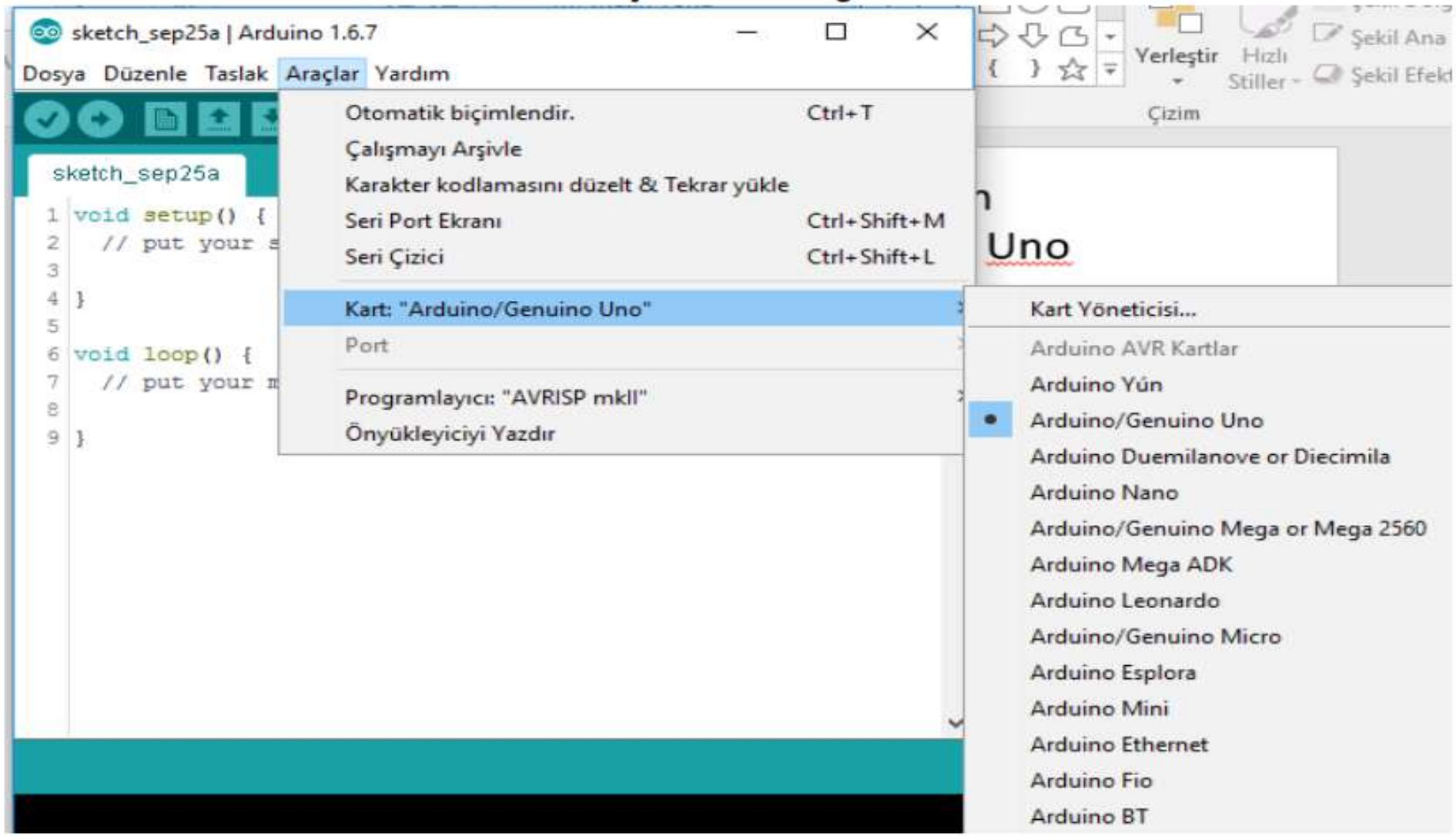
Araçlar Menüsü

- Araçlar menüsünde standart işlemlerin dışında “Kart” sekmesini kullanarak programlanacak Arduino modelinin seçimi, “Port” kısmından kullanılan Arduino’nun bilgisayarın hangi portuna bağladığı seçilmelidir. Arduino’nun bağlı olduğu port ile arayüzde seçili olan portun aynı olması gereklidir. Arduino destekli robotla bağlantı bu şekilde yapılmalıdır.

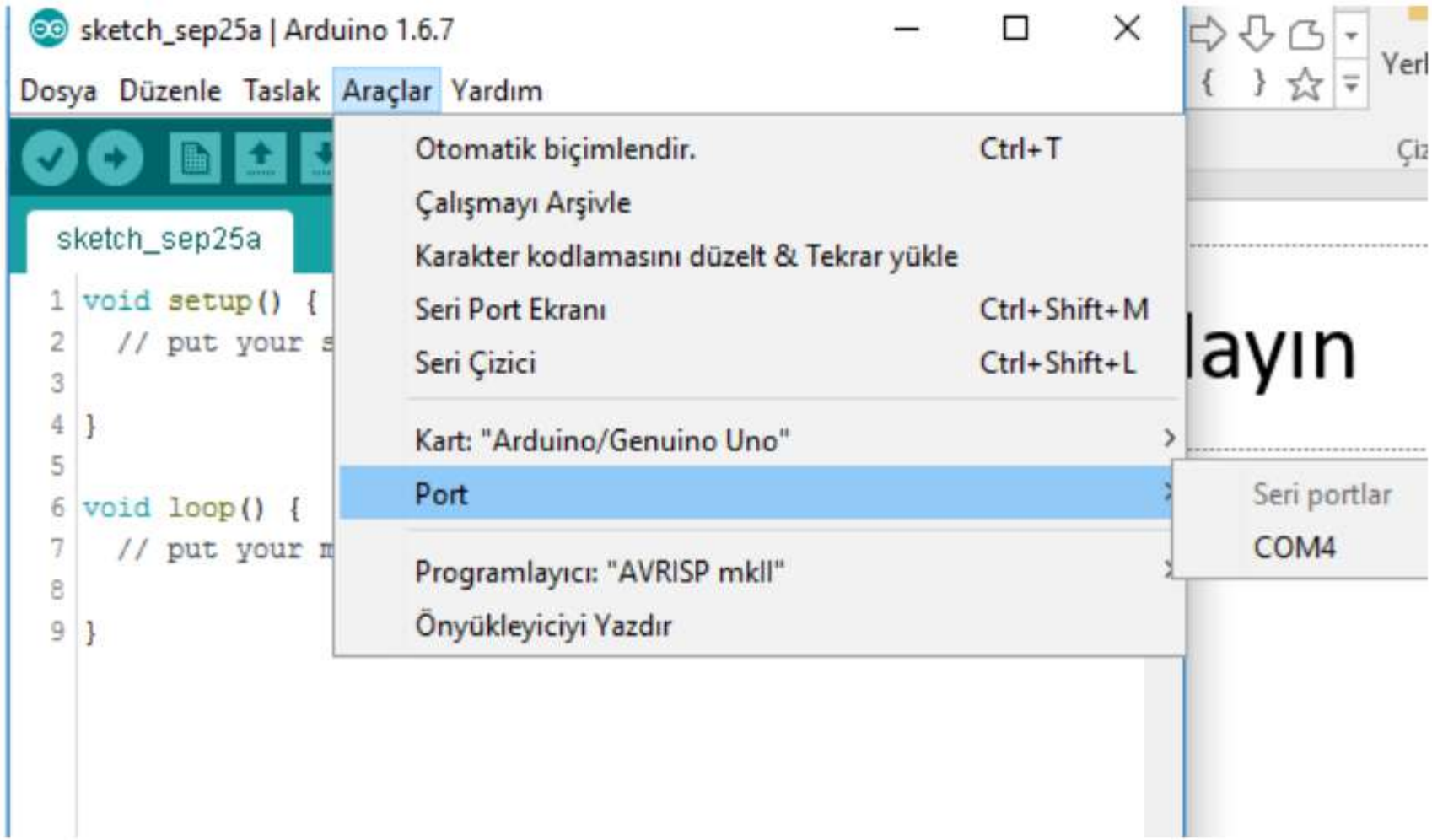


Resim 7.13: Arduino IDE araçlar menüsü

Kullanacağımız Kartı Seçelim
Araçlar>Kart>Arduino/Geniuno Uno
menü yolu seçilmeli



Arduino kartı bilgisayara taktıktan sonra port seçilmelidir. Port seçimi yapılmazsa yazılım yüklemesi yapılamaz



Arduino IDE'nin Temel Özellikleri

- Arduino IDE tümleşik geliştirme ortamında **basitleştirilmiş C++** kullanılır.
- Arduino programları genellikle **tanımlamalar, kurulum ve ana program** bloğu olmak üzere üç bölümden oluşur.
- Program yazımı **belirli kalıpta, bloklar** halinde gerçekleştirilir.
- Program kodları renkli olarak gösterilir. Kodların bulunduğu yerlerde **gri renkte olan yazılar kodun ne işe yaradığı hakkında** bilgi vermek için kullanılır.
- **Arduino'ya yüklenen programlar kaldırılana kadar Arduino içinde kalır**. Yüklemeden sonra bağımsız olarak çalıştırılabilir.
- Bloklar, { } parantezleri ile oluşturulur. Komutlar aynı veya alt alta satırlara yazılabilir. Fakat programın anlaşılabilirliği açısından alt alta yazmak daha uygundur.

Arduino IDE’nin Temel Özellikleri

.**Tüm komutlar noktalı virgül (;) ile biter. Fakat blok başlatan ifadelerden sonra noktalı virgül kullanılmaz**.

- Programda kullanılan tüm değişkenler ve bilgi tipleri bildirilir.
- Programın başında kullanılacak kütüphaneler aktifleştirilir /çağrılır.
- **Açıklamalar “//” ve “/* */ ”** (birden fazla satır için) ile yazılır. Türkçe karakter kullanılmamalıdır. Fakat açıklama satırları içerisinde (derleme işlemine dâhil edilmediğinden) kullanılabilir.
- Eşdeğer ifadeler #define ile atanır.
- Kütüphaneler #include ile çağrılır.

Arduino IDE'nin "Program" Yapısı

```
int buttonPin = 4; // butonu 4. pine tanımlandı
void setup()
{
  Serial.begin(9600); // Seri haberleşme başlatıldı
  pinMode(buttonPin, INPUT); // Pin modu tanımlandı
}
void loop()
{
  // ...
}
```

Resim 7.14: void setup() fonksiyonu örneği

void setup() fonksiyonu program yüklenip enerji verildikten veya tekrar başlatıldıktan **sonra 1 defa çalışan fonksiyondur**.

"void setup()" ile başlayan satır, takip eden **tırnak içindeki bölümde temel ayarla**rın yapılacağını belirtmektedir.

Bu fonksiyon içine **pin modları, kütüphaneyi başlatma ve değişkenler yazılmaktadır. Burada yapılan ayarlamalarda hangi mikrodenetleyici pininin (veri bacağının) giriş** (input-veri çekilen port-) ya da çıkış (output-veri gönderilen port-) olduğu belirtilmektedir.

Arduino IDE'nin "Program" Yapısı

```
int buttonPin = 4; // butonu 4. pine tanımlandı
void setup()
{
  Serial.begin(9600); // Seri haberleşme başlatıldı
  pinMode(buttonPin, INPUT); // Pin modu tanımlandı
}
void loop()
{
  // ...
}
```

Resim 7.14: void setup() fonksiyonu örneği

```
int buttonPin = 4; // butonu 4. pine tanımlandı

void setup()
{
  Serial.begin(9600); // Seri haberleşme başlatıldı
  pinMode(buttonPin, INPUT); // Pin modu tanımlandı
}

void loop()
{
  if (digitalRead(buttonPin) == HIGH) //okunan buton 1 ise
  Serial.write('Basıldı'); // Seri ekrana yaz
  else
  Serial.write('Basılmadı');
  delay(500);
}
```

Resim 7.15: void loop() fonksiyonu örneği

void loop() fonksiyonuna setup işleminden sonra eklenen ve mikrodenetleyici ya da **Arduino'nun beslemesi devam ettiği sürece tekrarlanan komutlar yazılmaktadır**.

Buraya yazılan komutlar ile **Arduino pinleri arasından karşılaştırma, ilişkilendirme, matematiksel işlemler vs. yapılmaktadır. Yazılan program burada sonsuz döngü içinde çalışmaktadır**.

Yukarıdaki örnekte tanımlanmış olan butuna basıldıysa seri ekrana "Basıldı", basılmadıysa "Basılmadı" yazan küçük bir program "void loop" içine yazılmıştır.

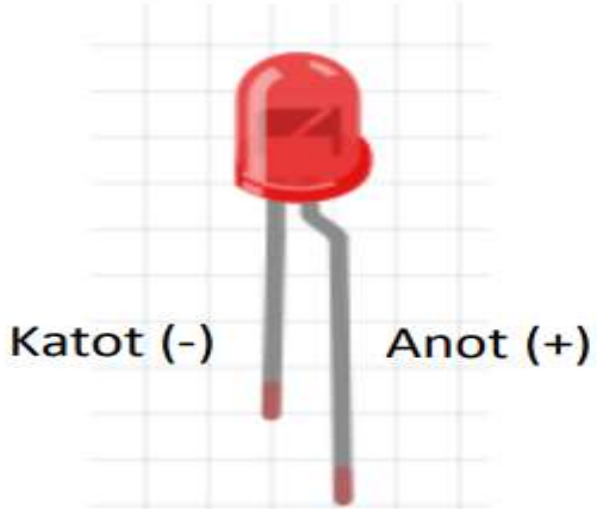
Arduino IDE'nin "Program" Yapısı

```
int buttonPin = 4; // butonu 4. pine tanımlandı

void setup()
{
  Serial.begin(9600); // Seri haberleşme başlatıldı
  pinMode(buttonPin, INPUT); // Pin modu tanımlandı
}

void loop()
{
  if (digitalRead(buttonPin) == HIGH) //okunan buton 1 ise
  Serial.write('Basıldı'); // Seri ekrana yaz
  else
  Serial.write('Basılmadı');
  delay(500);
}
```

Resim 7.15: void loop() fonksiyonu örneği

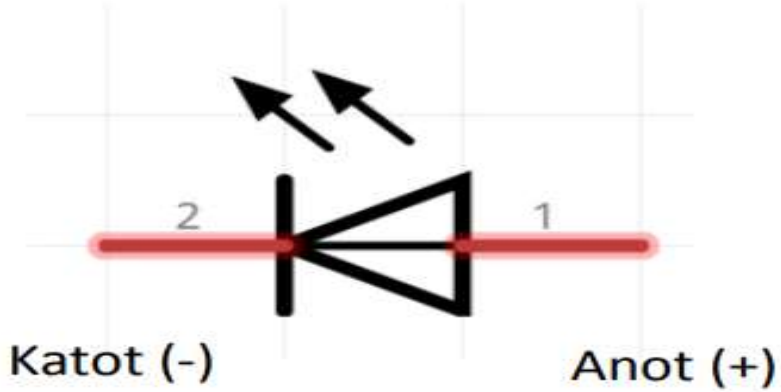
örnekte tanımlanmış olan butuna basıldıysa seri ekrana "Basıldı", basılmadıysa "Basılmadı" yazan küçük bir program "void loop" içine yazılmıştır.

LED NEDİR?



Gerçek Görünüm

İsmi Light Emited Diyot kelimelerinin ilk harfinden gelir. Üzerinden akım geçince ışık veren devre elemanıdır. Anot(+) bacağı elektrik akımı gönderebileceğimiz pinlerden birine bağlanır. Katot(-) ucu ise GND hattına bağlanır. Sarı, mavi, yeşil ,kırmızı vb. renklerde olabilir. LED ler genel olarak 20mA (mili Amper) yani 0,020 Amper akıma ihtiyaç duyar.



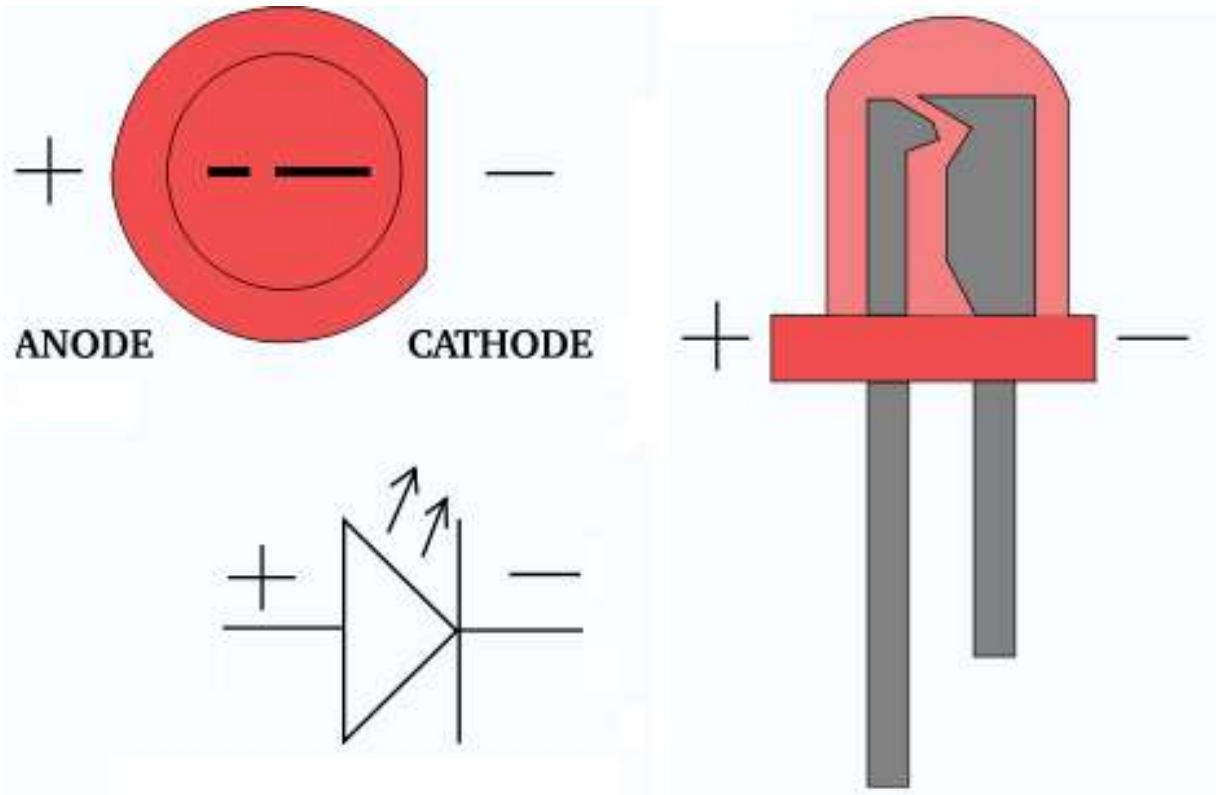
Sembolü

Projelerimizde LED'lerin yüksek akımdan dolayı zarar görmemesi için LED'in herhangi bir bacağının önüne direnç bağlanacaktır

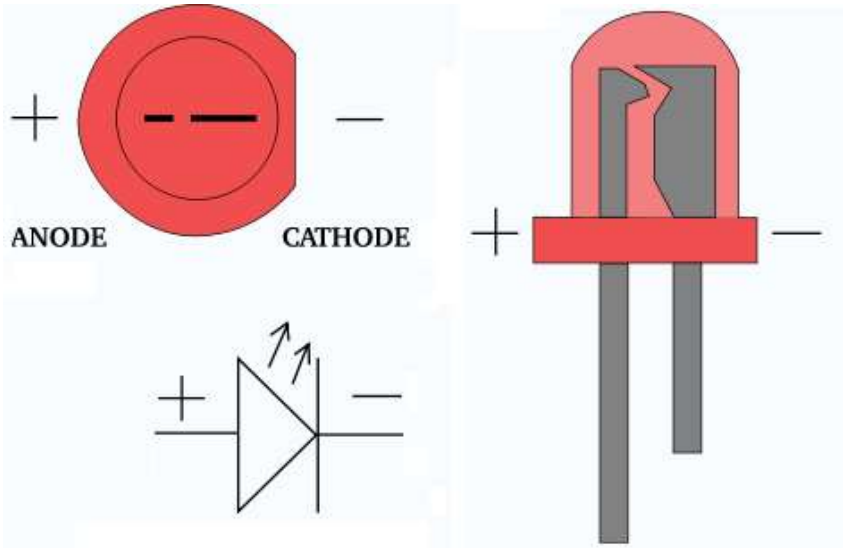
LED (**L**ight **E**mitting **D**iode)

- “Işık Yayan Diyot” anlamına gelir.
- Elektrik enerjisini ışığa dönüştüren yarı iletken devre elemanlarıdır.

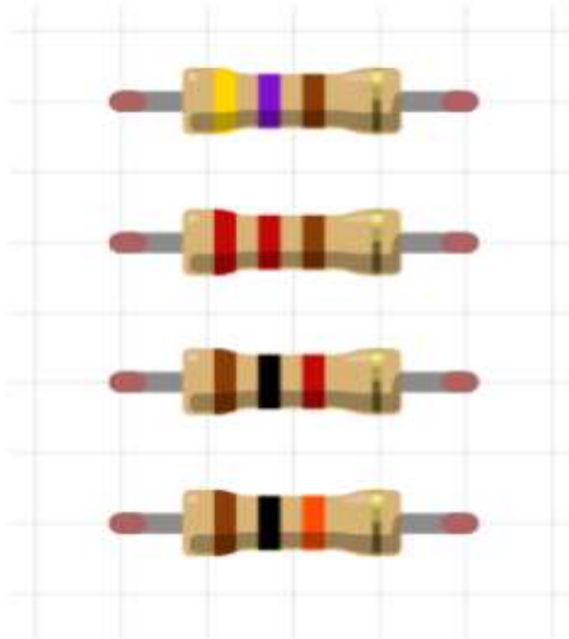
Devre Gösterimleri



LED (**L**ight **E**mitting **D**iode)

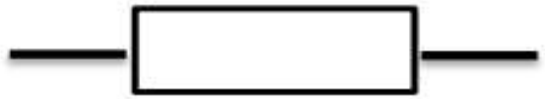


- **LED**‘in uzun bacağı, Arduino kart üzerindeki **dijital sinyal pinine**,
- kısa bacak ise yine kart üzerindeki **GND** pinine bağlanmalıdır.
- Arduino pinlere 5V enerji gönderir, **LED** ler ise ortalama 3V ile çalışmaktadır. İşte aradaki 2V kadar enerjiyi 220 ohm direnç ile dengeleriz



SEMBOLÜ

veya

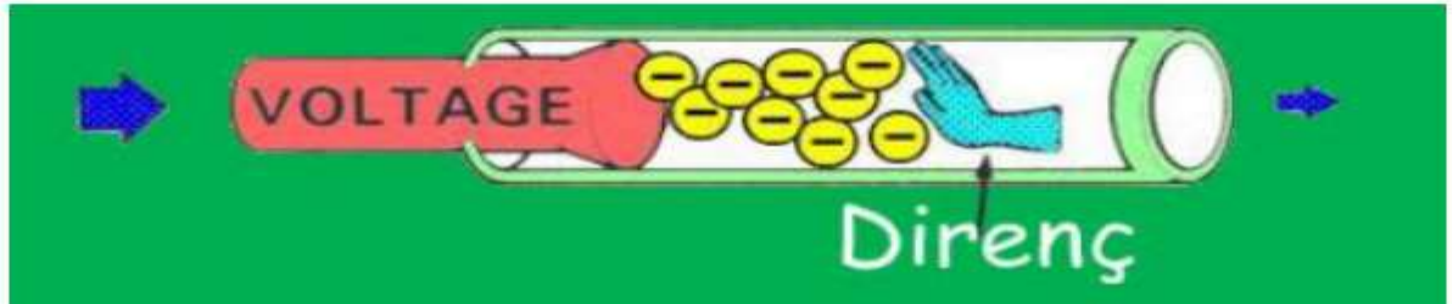


Direnç Nedir?

Direnç Devrede Akıma Karşı zorluk gösteren ve bu sayede akımı sınırlandırma işlemlerinde kullanılan devre elemanıdır.

Birimi Ohm sembolü Ω dır

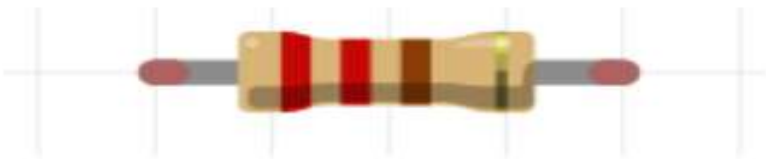
Direnç Üzerindeki Renkler direncin değerini bilmemizi sağlar.



DİRENÇ OKUMA VE RENK KODLARI

Direnç değeri hesaplanırken 1. ve 2. renklere karşılık gelen sayılar yan yana yazılır sonrasında 3. renk değeri kadar 0 (sıfır) eklenir

RENK	SAYI
Siyah	0
Kahverengi	1
Kırmızı	2
Turuncu	3
Sarı	4
Yeşil	5
Mavi	6
Mor	7
Gri	8
Beyaz	9
Altın	-
Gümüş	-



2 2 1(sıfır)
Kırmızı-kırmızı-kahverengi
2 2 0

220Ω

4 7 1(sıfır)
sarı- mor -kahverengi
4 7 0

470Ω

Renk Kodlarını Kolay Hatırlamak için aşağıdaki yazıyı kullanabiliriz

S o K a K T a S a Y a M a M G i B i

0	1	2	3	4	5	6	7	8	9
Siyah	Kahverengi	Kırmızı	Turuncu	Sarı	Yeşil	Mavi	Mor	Gri	Beyaz

1000 Ω (ohm) =1 K Ω(Kilo Ohm)

1	0	3(sıfır)
kahve-	siyah -	Turuncu
1	0	000

10000Ω=10K Ω

6	8	2(sıfır)
mavi-	gri -	kırmızı
6	8	00

6800Ω=6,8KΩ

6	8	1(sıfır)
mavi-	gri -	kahve
6	8	0

680Ω

1	5	2(sıfır)
kahve-	yeşil -	kırmızı
1	5	00

1500Ω=1,5KΩ

3	3	1(sıfır)
turuncu-turuncu-		kahve
3	3	0

330Ω

Led önüne Bağlanılacak direncin Hesaplanması

OHM kanununa göre Bu hesaplamada R=V/ I formülü kullanılacaktır

R>> direnç >>birimi ohm>>Ω
V>> voltaj>>birimi volt>>V
I >>akımdır>>birimi Amper>>A

Led 0,020 amper akıma ihtiyaç duyar. Arduino dijital pinleri ise 5 V çıkış verir o halde gerekli direnci hesaplayalım.

R=5/0,020
R=250 Ω direnç (yaklaşık) bağlanılabilir.

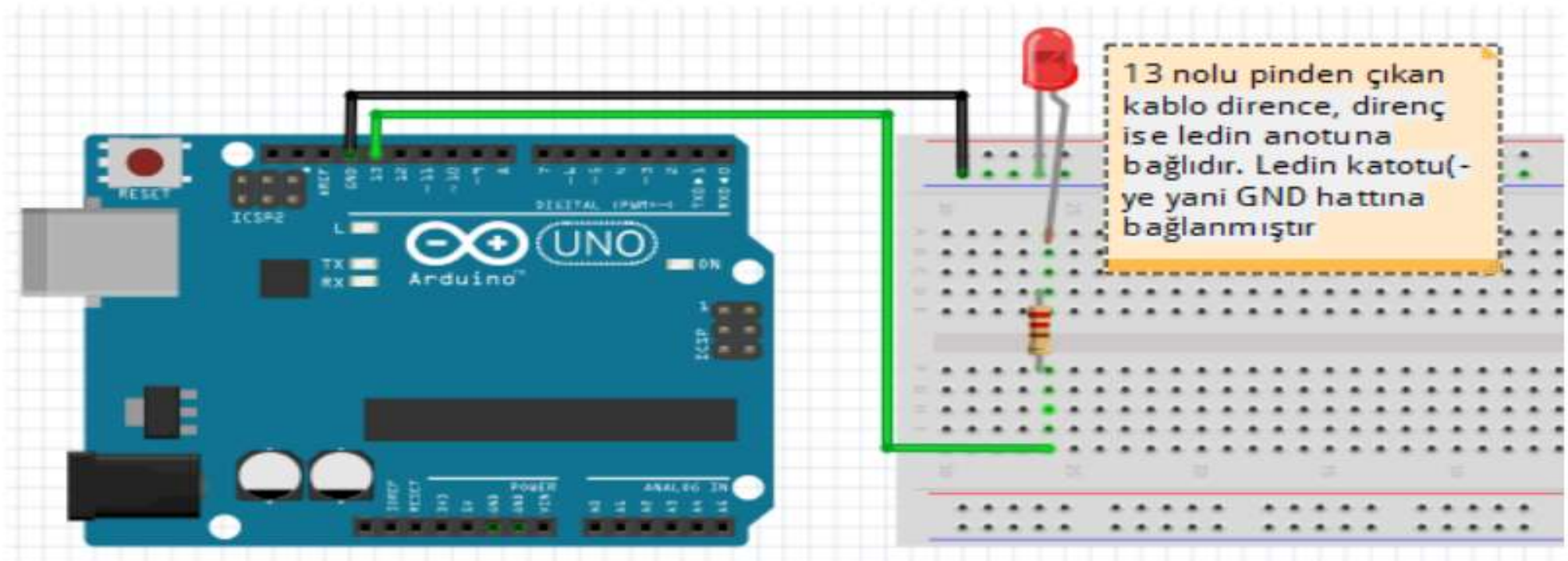
Biraz daha yüksek değerde bağlanırsa led'in parlaklığı direnç değerine göre değişecektir.

LED Yakma

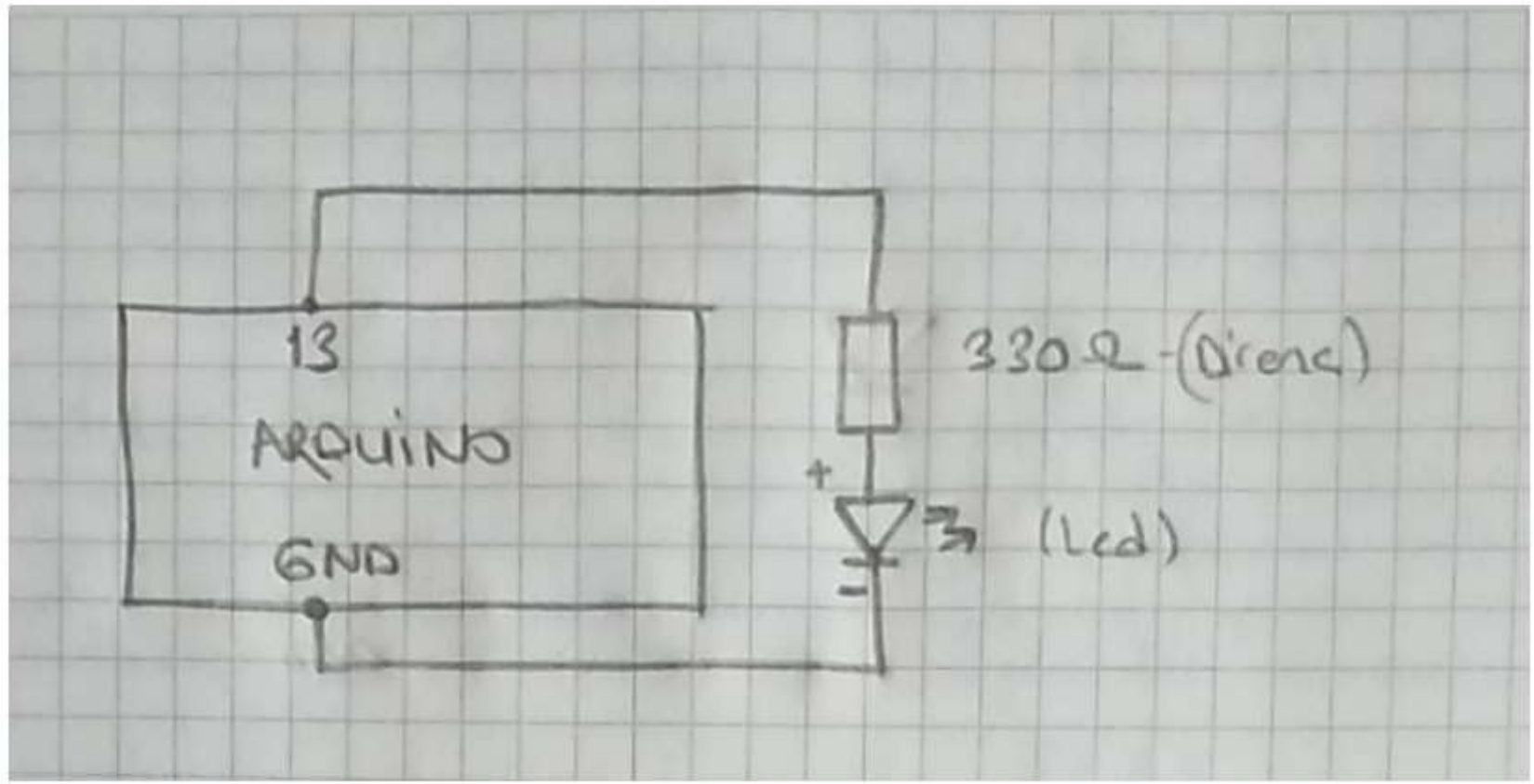
LED YAKMA - Malzemeler

1. Arduino Uno
2. Breadboard
3. LED
4. 220 Ohm Direnç
5. İki ucu erkek kablo

Proje 1- LED Yakıp Söndürme

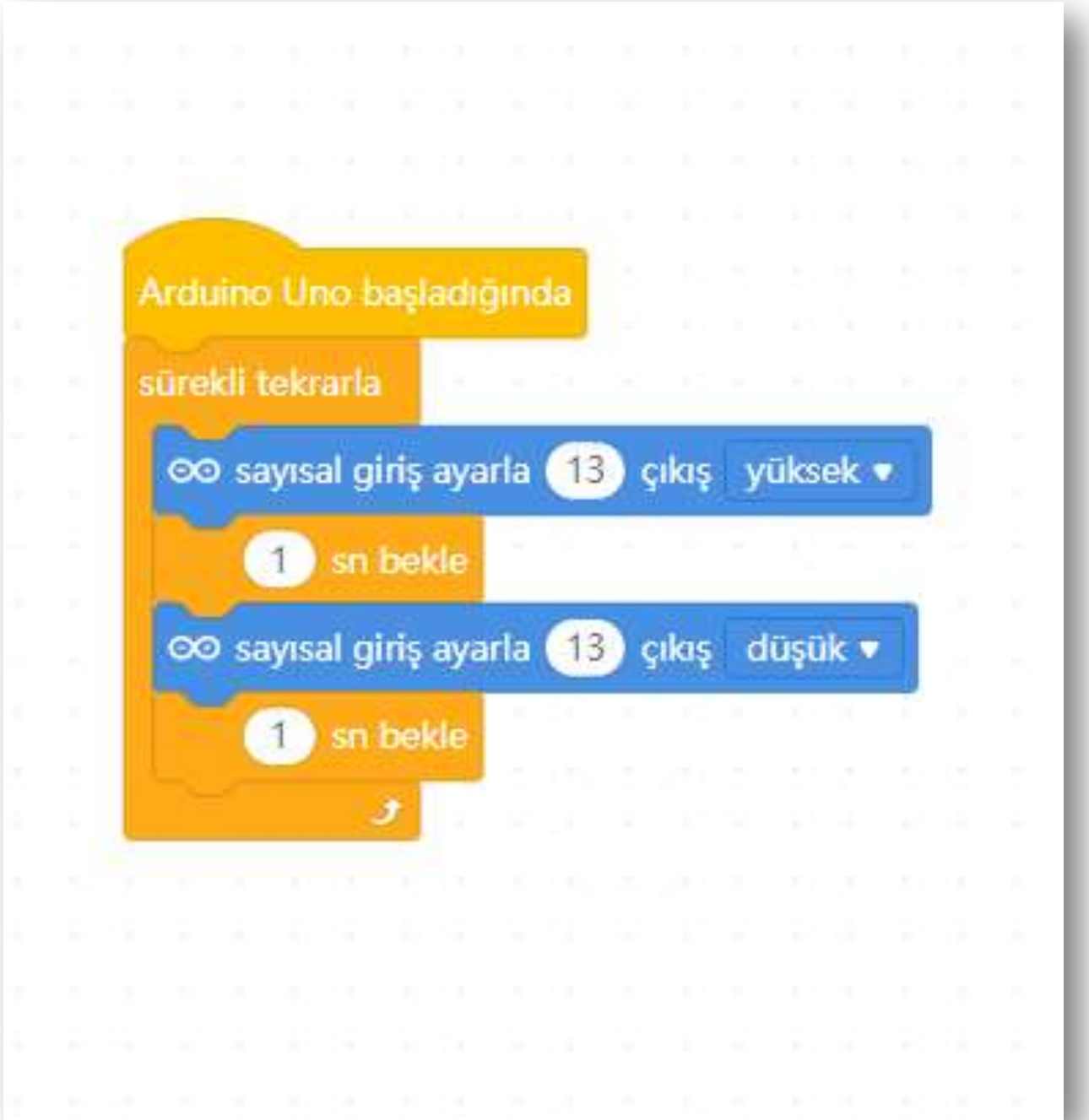


Proje 1- Led yakıp söndürme devre çizimi



LED YAKMA

Kodlar



Arduinoya Kodlarımızı Yazalım

ledyakson §

```
void setup() {
  pinMode(13,OUTPUT);//13 nolu dijital pin çıkış verecek şekilde ayarlandı

}

void loop() {
  digitalWrite(13,HIGH); //13 nolu digital pini 1 yap yani +5V yap Bu esnada LED yanacaktır
  delay(1000);           //1000 ms=1sn bekle LED 1 sn Yanacak
  digitalWrite(13,LOW); //13 Nolu Pini 0 yap yani 0V yap Bu esnada LED sönecek
  delay(1000);           //1 sn LED sönük bekleyecek

}
```

PROGRAM YAPISI

```
4  void setup() {
5
6    pinMode(13, OUTPUT);//13 nolu dijital pini çıkış olarak ayarla
7  }
8
```

1

```
10 void loop() {
11   digitalWrite(13, HIGH);   // 13 nolu pini YÜKSEK yap +5V verir dijital değeri 1(bir) dir
12   delay(1000);              // 1000 ms=1 sn bekle
13   digitalWrite(13, LOW);    // 13 nolu pini DÜŞÜK yap 0V verir dijital değeri 0(sıfır) dir
14   delay(1000);              // 1000ms=1 sn bekle
15 }
```

2

1-void setup()

Program yüklenip enerji verildikten veya reset atıldıktan sonra bir defa çalışır. Bu fonksiyon içine genellikle kurulum ve ayar için gerekli kodlar yazılır. Aynı zamanda bir defa çalışmasını istediğimiz kodları da yazabiliriz.

2. void loop()

Setup() fonksiyonumuz tamamlandıktan sonra loop fonksiyonumuza geçer ve burada sonsuz döngü içinde yazdığımız programı sürekli çalıştırır.

pinMode(); Komutu

- Kullanılacak olan pinlerin, hangi görevde çalışacağı bu komut ile belirlenir

pinMode(***pinNo*** , ***Görev***)

```
4  void setup() {
5
6    pinMode(13, OUTPUT);//13 nolu dijital pini çıkış olarak ayarla
7  }
8
```

pinMode(13,OUTPUT); => 13 Nolu pin Çıkış olarak tanımlandı.

pinMode(12,OUTPUT); => 12 Nolu pin Çıkış olarak tanımlandı.

digitalWrite(); Komutu

```
10 void loop() {
11   digitalWrite(13, HIGH);
12   delay(1000);
13   digitalWrite(13, LOW);
14   delay(1000);
15 }
```

- Çıkış olarak tanımlanan pinlerden, dijital veri olarak 1 veya 0 çıkışı verilmesini sağlar.

- ***digitalWrite(13,HIGH); => 13 nolu pini YÜKSEK yap dijital Veri olarak 1, Voltaj olarak +5V çıkış verilir. (dijital değeri -1)***
- ***digitalWrite(13, LOW); => 13 nolu pini DÜŞÜK yap Dijital veri olarak 0 (sıfır), Voltaj değeri olarak 0V çıkış verilir. (dijital değeri-0)***

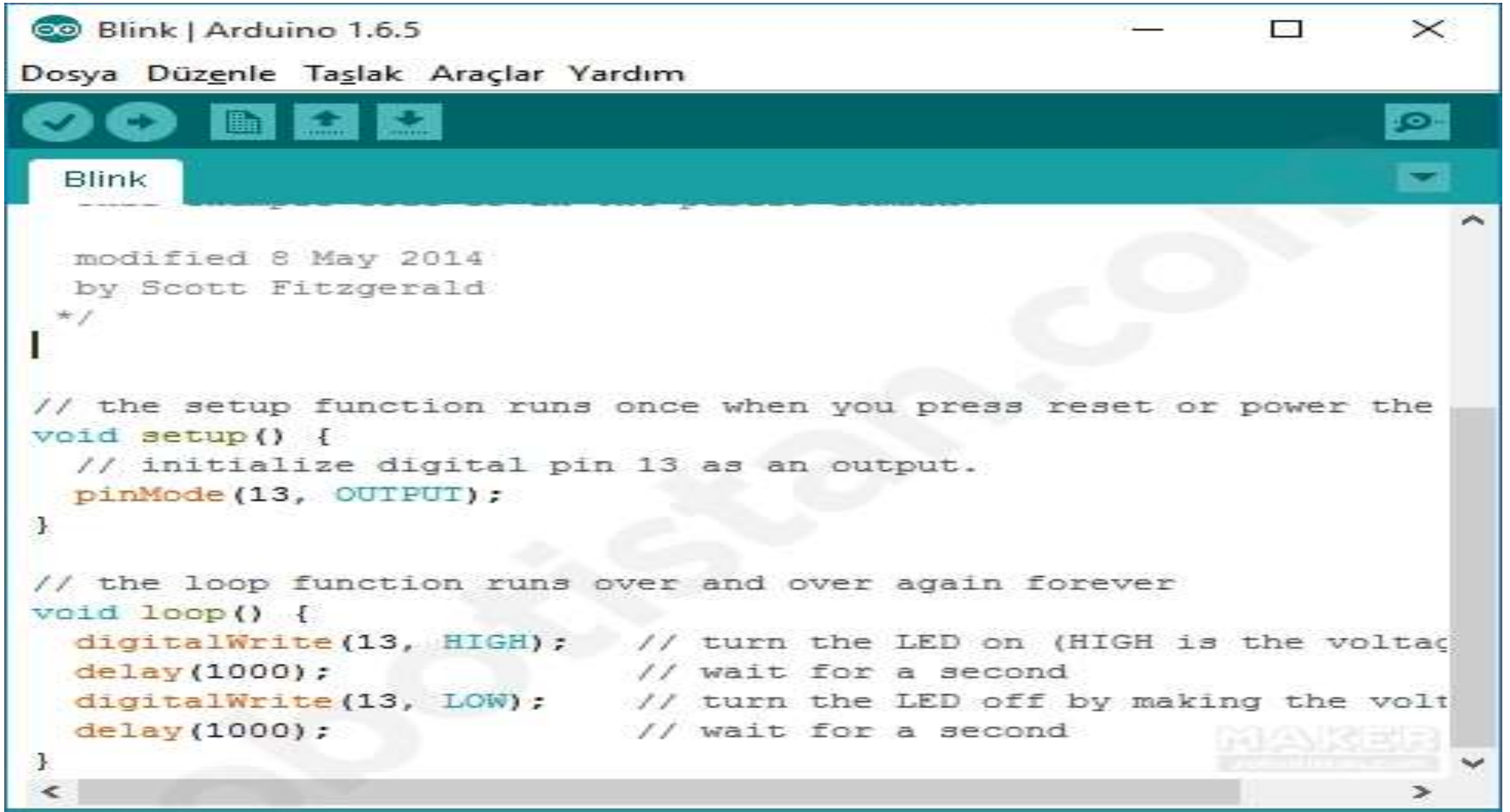
delay(); Komutu

```
10 void loop() {
11   digitalWrite(13, HIGH);
12   delay(1000);
13   digitalWrite(13, LOW);
14   delay(1000);
15 }
```

- Parantez içerisinde belirtilen mili Saniye (ms) süre kadar bekleme yapılmasını sağlar. Komutlar arasında bekleme yapmak için kullanılır.

	delay(1000)	***1000 ms=1sn bekle***
Örnek:	***delay(5000)***	***5000 ms=5 sn bekle***
	delay(7500)	***7500 ms=7.5 sn bekle***

```
void setup() {
pinMode(13, OUTPUT); // 13 numaralı digital pini elektrik verecek şekilde çıkış olarak tanımlıyoruz.
}

void loop() {
digitalWrite(13, HIGH); // 13 numaralı pin den HIGH komutu ile elektrik çıkışı sağlıyoruz.
delay(1000); // kodumuzu 1000 milisaniye (1 saniye) bekletiyoruz.
digitalWrite(13, LOW); // 13 numaralı pin den LOW komutu ile elektrik çıkışını d
delay (1000); // kodumuzu 1000 milisaniye (1 saniye) bekletiyoruz.
}
```



- **Dosya > Örnekler 01.Basics > Blink**

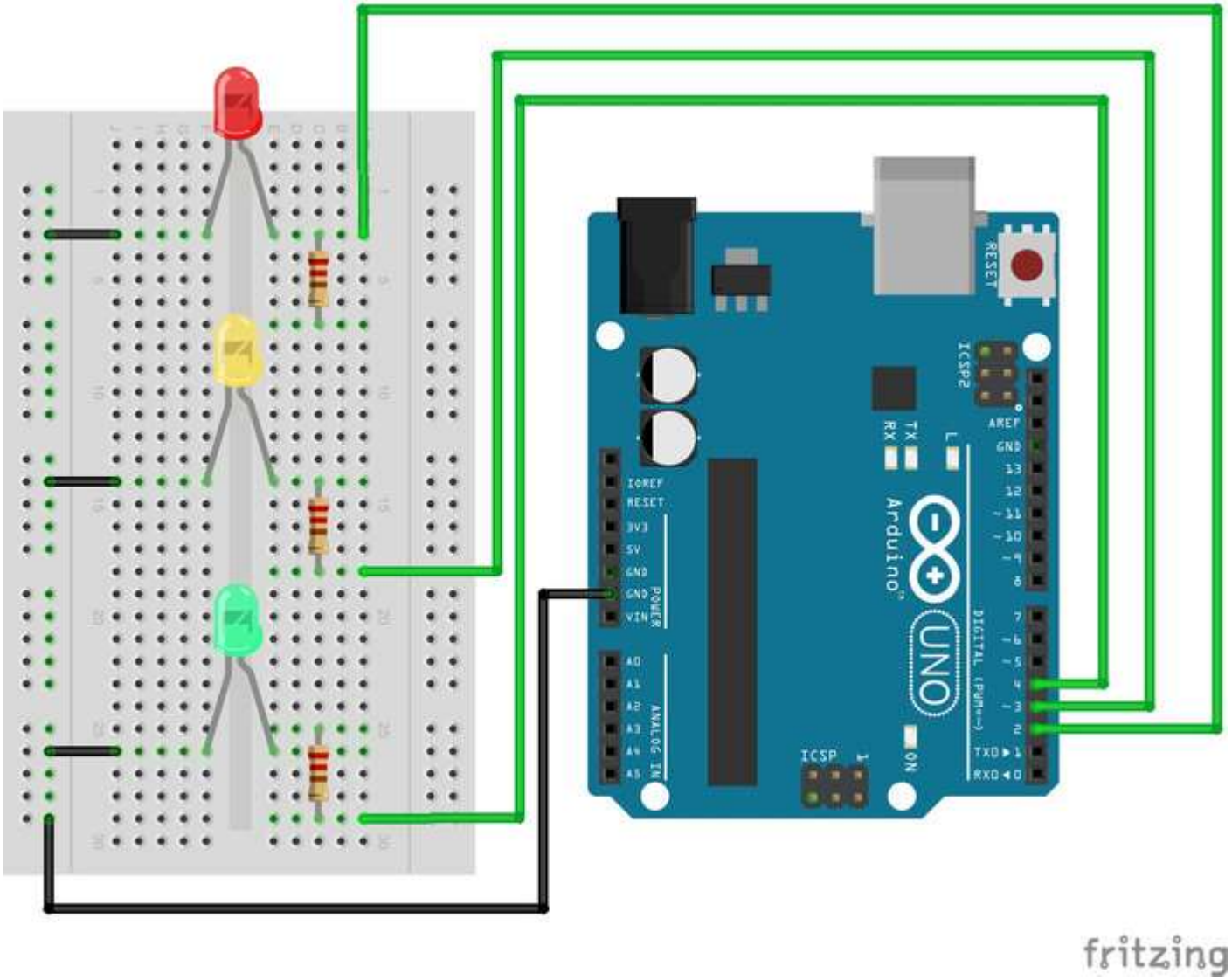
Trafik Lambası

Trafik Lambası - Malzemeler

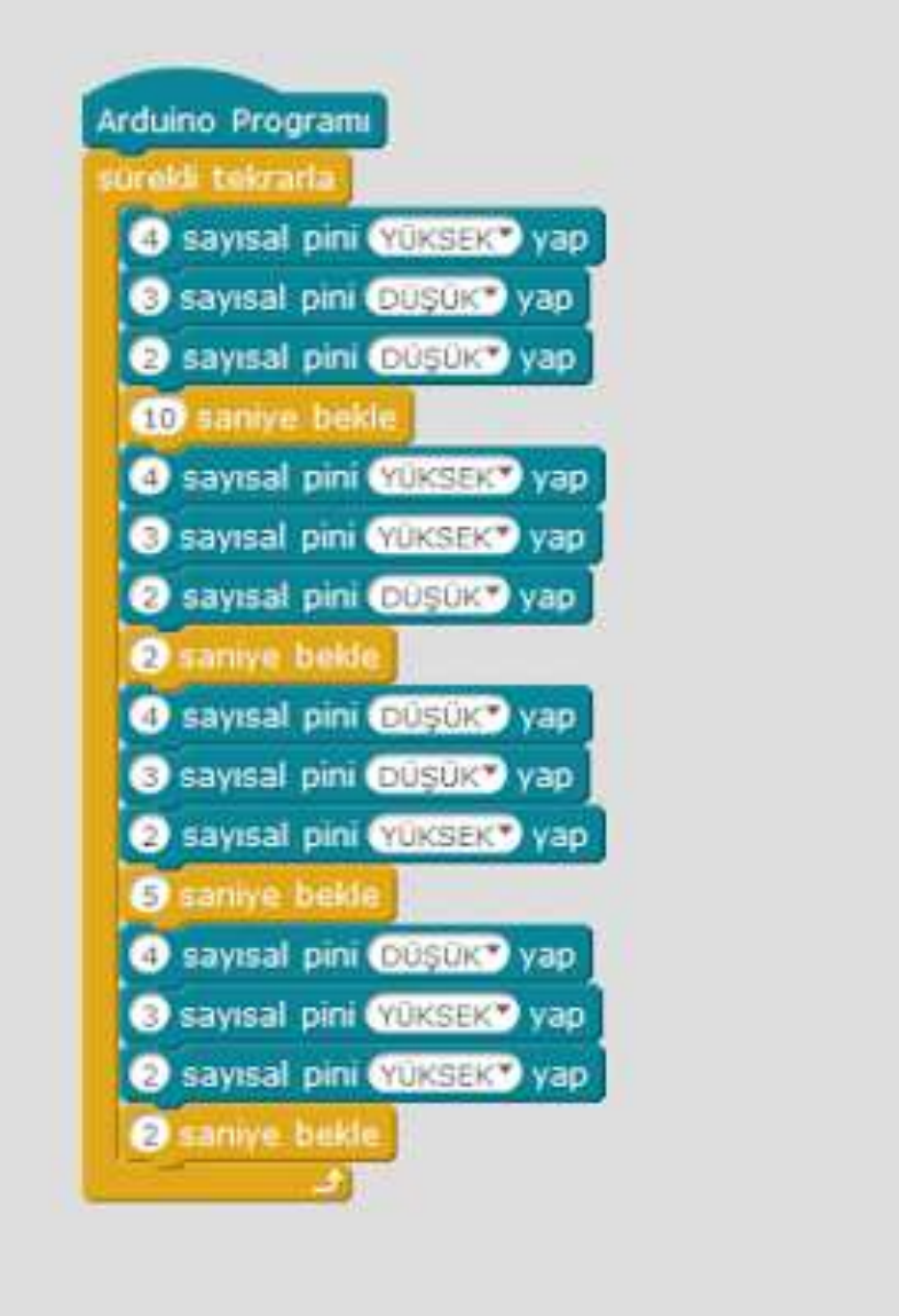
1. Arduino Uno
2. Breadboard
3. LED *(sarı – kırmızı - yeşil)*
4. 220 Ohm Direnç *x3*
5. İki ucu erkek kablo

Çalışma Mantığı: program ilk başta kırmızı ışığı yakacak. Kırmızı ışık 5 saniye yandıktan sonra sönecek ve 1 saniye boyunca sarı ışık yanacak. Sarı ışık söndükten sonra da 3 saniye boyunca yeşil ışık yanacak.

Trafik Lambası - Devre



Trafik Lambası Kodlar



Blink | Arduino 1.8.12

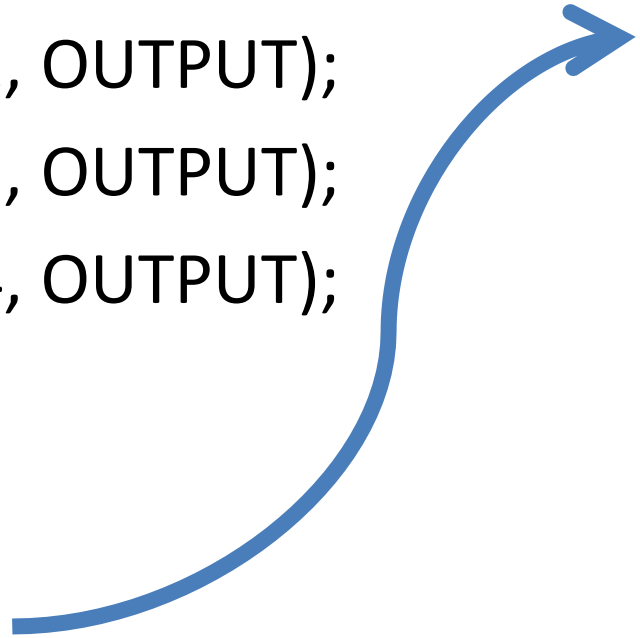
Dosya Düzenle Taslak Araçlar Yardım

Blink §

```
void setup()
{/* LED pinleri çıkış olarak ayarlandı */
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
}
void kirmiziIsik(){ /* Sadece kırmızı ışığı yakan fonksiyon */
  digitalWrite(2,HIGH);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
}
void sariIsik(){/* Sadece sarı ışığı yakan fonksiyon */
  digitalWrite(2,LOW);
  digitalWrite(3,HIGH);
  digitalWrite(4,LOW);
}
```

```
void yesilIsik(){/* Sadece yeşil ışığı yakan
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,HIGH);
}
void loop()
{ kirmiziIsik();
  delay(5000);
  sariIsik();
  delay(1000);
  yesilIsik();
  delay(3000);
}
```

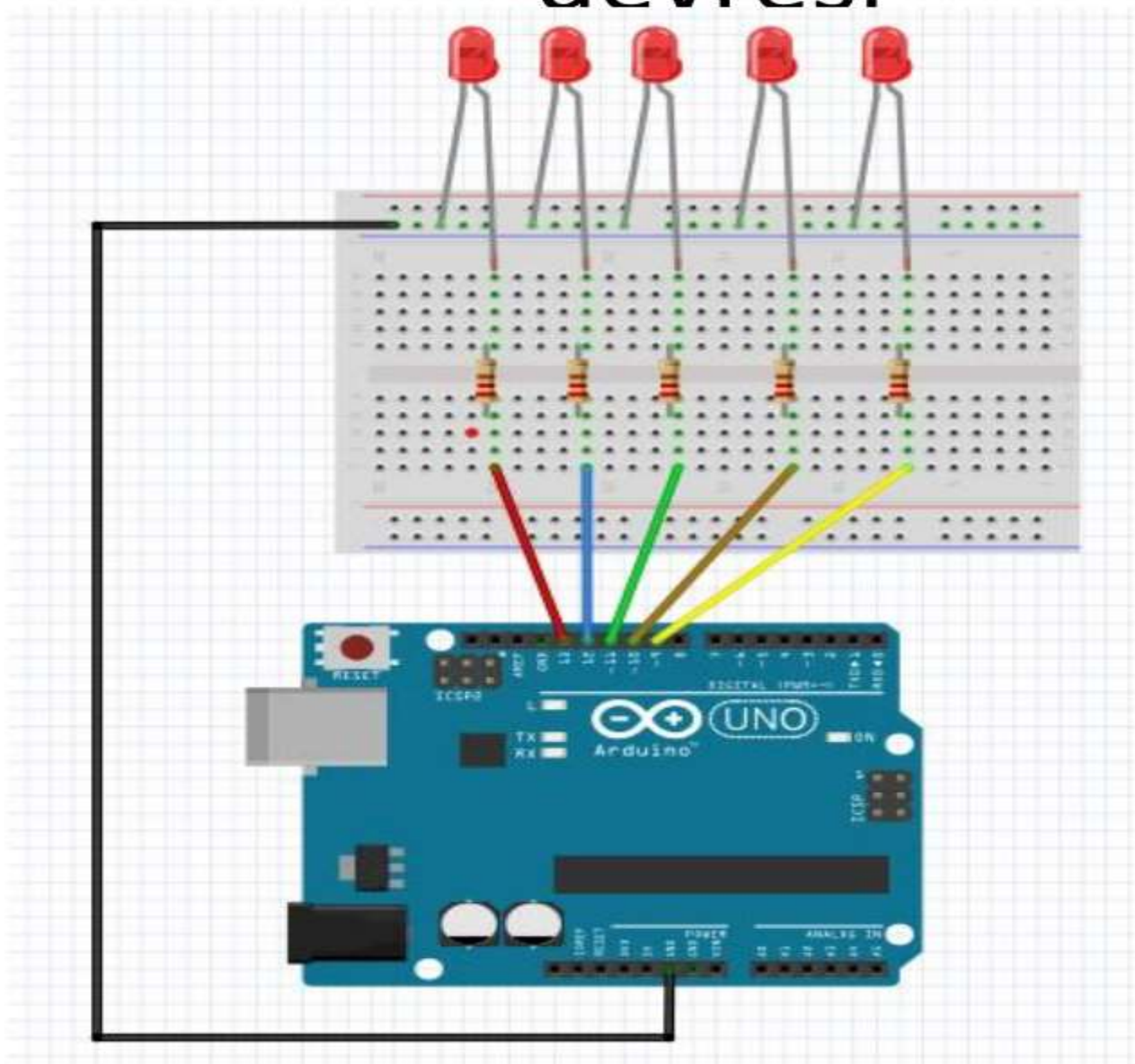
```
void setup() {
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
}
```



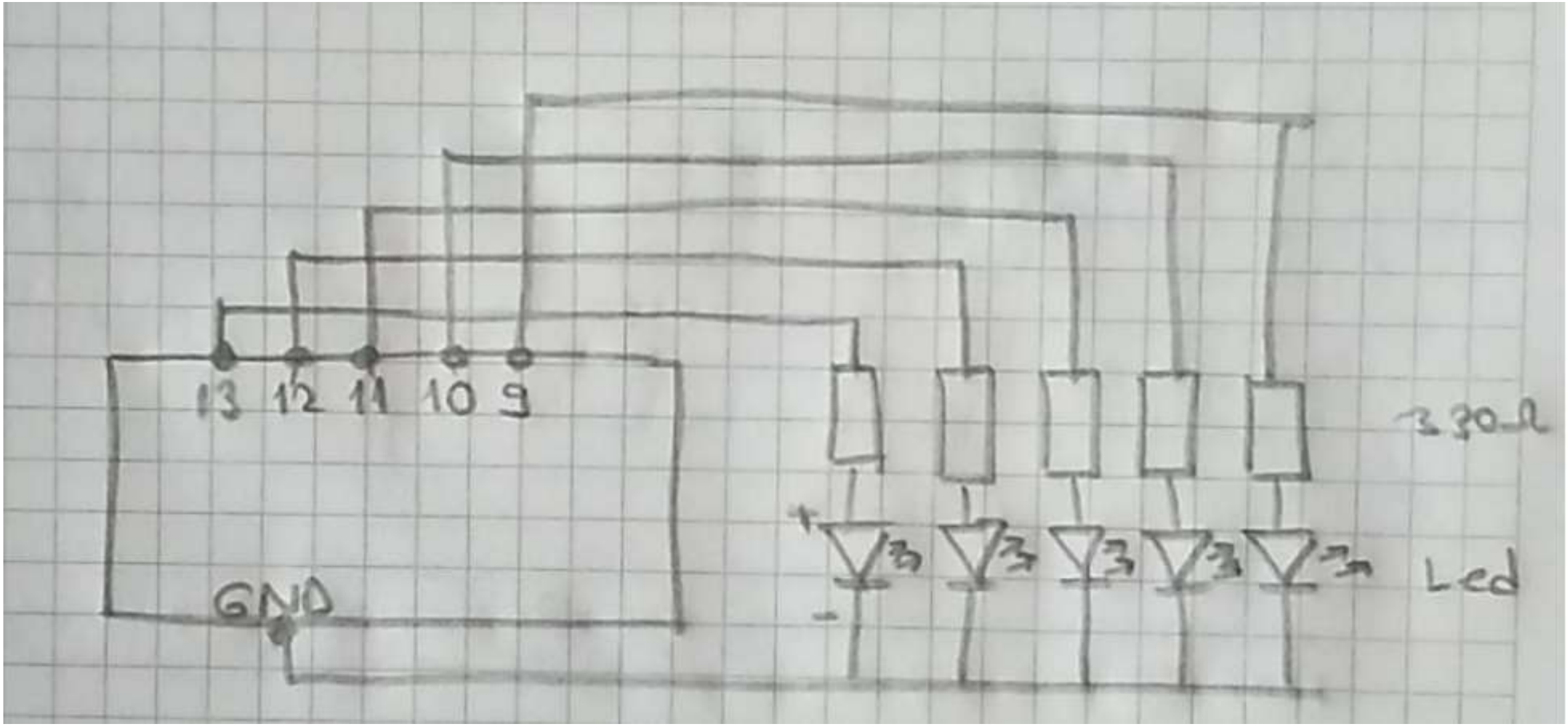
```
void loop() {
digitalWrite(2, HIGH)
digitalWrite(3, LOW);
digitalWrite(4, LOW);
delay(5000);
digitalWrite(2, LOW);
digitalWrite(3, HIGH);
digitalWrite(4, LOW);
delay(1000);
digitalWrite(2, LOW);
digitalWrite(3, LOW);
digitalWrite(4, HIGH);
delay(3000);
}
```

```
void setup() {
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
}
void loop() {
digitalWrite(2, HIGH)
digitalWrite(3, LOW);
digitalWrite(4, LOW);
delay(5000);
digitalWrite(2, LOW);
digitalWrite(3, HIGH);
digitalWrite(4, LOW);
delay(1000);
digitalWrite(2, LOW);
digitalWrite(3, LOW);
digitalWrite(4, HIGH);
delay(3000);
}
```

Proje 2- Yürüyen Işık devresi bağlantı devresi



9,10,11,12,13 nolu pinlere bağlı



Aşağıdaki kodlara göre yürüyen ışık devresi kodlarını yazalım.

ledyakson §

```
void setup() {
  pinMode(13,OUTPUT);//13 nolu dijital pin çıkış verecek şekilde ayarlandı


}

void loop() {
  digitalWrite(13,HIGH); //13 nolu digital pini 1 yap yani +5V yap Bu esnada LED yanacaktır
  delay(1000);           //1000 ms=1sn bekle LED 1 sn Yanacak
  digitalWrite(13,LOW); //13 Nolu Pini 0 yap yani 0V yap Bu esnada LED sönecek
  delay(1000);           //1 sn LED sönük bekleyecek


}
```

```
void setup() {                  //ilk ayar ve kurulum yapılır (1 kere çalışacak fonksiyon)
      pinMode(13,OUTPUT);       //13 nolu dijital pini çıkış olarak ayarla
      pinMode(12,OUTPUT);       //12 nolu dijital pini çıkış olarak ayarla
      pinMode(11,OUTPUT);       //11 nolu dijital pini çıkış olarak ayarla
      pinMode(10,OUTPUT);       //10 nolu dijital pini çıkış olarak ayarla
      pinMode(9,OUTPUT);        //9 nolu dijital pini çıkış olarak ayarla
}
```

```
void loop() {                   //Sürekli Tekrar edecek olan fonksiyon
      digitalWrite(13,HIGH);    //13 nolu pine enerji ver +5 V yani diğital olarak 1 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(13,LOW);     //13 nolu pindeki enerjiyi kes 0 V yani dijital olarak 0 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(12,HIGH);    //12 nolu pine enerji ver +5 V yani diğital olarak 1 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(12,LOW);     //12 nolu pindeki enerjiyi kes 0 V yani dijital olarak 0 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(11,HIGH);    //11 nolu pine enerji ver +5 V yani diğital olarak 1 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(11,LOW);     //11 nolu pindeki enerjiyi kes 0 V yani dijital olarak 0 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(10,HIGH);    //10 nolu pine enerji ver +5 V yani diğital olarak 1 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(10,LOW);     //10 nolu pindeki enerjiyi kes 0 V yani dijital olarak 0 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(9,HIGH);    //9 nolu pine enerji ver +5 V yani diğital olarak 1 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(9,LOW);     //9 nolu pindeki enerjiyi kes 0 V yani dijital olarak 0 yap
      delay(1000);              //1000 ms=1sn  bekle
```

```
void setup() {                  //ilk ayar ve kurulum yapılır (1 kere çalışacak fonksiyon)
      pinMode(13,OUTPUT);       //13 nolu dijital pini çıkış olarak ayarla
      pinMode(12,OUTPUT);       //12 nolu dijital pini çıkış olarak ayarla
      pinMode(11,OUTPUT);       //11 nolu dijital pini çıkış olarak ayarla
      pinMode(10,OUTPUT);       //10 nolu dijital pini çıkış olarak ayarla
      pinMode(9,OUTPUT);        //9 nolu dijital pini çıkış olarak ayarla
}
void loop() {                     //Sürekli Tekrar edecek olan fonksiyon
      digitalWrite(13,HIGH);    //13 nolu pine enerji ver +5 V yani diğital olarak 1 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(13,LOW);     //13 nolu pindeki enerjiyi kes 0 V yani dijital olarak 0 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(12,HIGH);    //12 nolu pine enerji ver +5 V yani diğital olarak 1 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(12,LOW);     //12 nolu pindeki enerjiyi kes 0 V yani dijital olarak 0 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(11,HIGH);    //11 nolu pine enerji ver +5 V yani diğital olarak 1 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(11,LOW);     //11 nolu pindeki enerjiyi kes 0 V yani dijital olarak 0 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(10,HIGH);    //10 nolu pine enerji ver +5 V yani diğital olarak 1 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(10,LOW);     //10 nolu pindeki enerjiyi kes 0 V yani dijital olarak 0 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(9,HIGH);    //9 nolu pine enerji ver +5 V yani diğital olarak 1 yap
      delay(1000);              //1000 ms=1sn  bekle
      digitalWrite(9,LOW);     //9 nolu pindeki enerjiyi kes 0 V yani dijital olarak 0 yap
      delay(1000);              //1000 ms=1sn  bekle
```

Arduino’da Yapılar

- Söz Dizimi için kullanılanlar;
  - ;
  - {}
  - //
  - /*….*/
  - #define – *tanımlama işlemleri için kullanılır*
  - #include

Arduino’da Yapılar

```
void loop() {
int x = analogRead(1);
```

- ; Noktalı Virgül

C programlama dilinde her satır programından sonra noktalı virgül konulması, sonlandırılması gerekmektedir. Noktalı virgül konulmadığı durumlarda derleme hatası oluşmaktadır.

```
void setup() {
pinMode(2, OUTPUT);
}
```

Arduino’da Yapılar

- {} Süslü Parantez

Fonksiyonlarda, **döngülerde ve koşullu ifadelerin bildirilmesinde** süslü parantez kullanılmaktadır.

İç içe olan fonksiyonlarda en dıştaki süslü parantezin en baştaki fonksiyona ait olduğuna dikkat edilmeli, aksi takdirde derleme hatası ortaya çıkmaktadır

```
void fonksiyonlarim()
{
//yapılacaklar }
while () {
//yapılacaklar }
do {
//yapılacaklar
}
for (x=0;x<=100;x++)
{
//yapılacaklar }
```

Arduino’da Yapılar

– // Çift Slash

Program satırından, program başında veya herhangi bir yerde programın çalışma şekli hakkında **açıklama** yapmak isteniyorsa çift slash kullanılmalıdır. **Çift slashtan sonra yazılanlar program kodu olarak dikkate alınmaz. Derleyici tarafından yok sayılır ve mikroişlemciye aktarılmaz**.

```
X = 5; // Bu, tek satırlı bir açıklamadır. Çift slash sonrasındaki her şey bir açıklamadır
       // satırın sonuna çift slash eklenmelidir.
/ * bu satırlı bir açıklamadır - tüm kod bloklarını açıklamak için kullanabilirsiniz
(gwb == 0) {  // Tek satırlık açıklama tek satır olarak kullanılabilir
X = 3;        / * Ancak başka satırlar kullanılacaksa - yukarıdaki şekilde geçersizdir.
Bu şekilde kullanılmalıdır. * /
}
// "kapanış" slahsını unutmayın
* /
```

Arduino’da Yapılar

– /*....*/ Yıldızlı Slash

/* Buraya yazılan her türlü satır, sütun dahil program olarak alınmamakta, açıklama olarak dikkate alınmaktadır. */

```
X = 5; // Bu, tek satırlı bir açıklamadır. Çift slash sonrasındaki her şey bir açıklamadır
       // satırın sonuna çift slash eklenmelidir.
/ * bu satırlı bir açıklamadır - tüm kod bloklarını açıklamak için kullanabilirsiniz
(gwb == 0) {   // Tek satırlık açıklama tek satır olarak kullanılabilir
X = 3;         / * Ancak başka satırlar kullanılacaksa - yukarıdaki şekilde geçersizdir.
Bu şekilde kullanılmalıdır. * /
}
// "kapanış" slahsını unutmayın
* /
```

Arduino'da Yapılar

```
#include "SPI.h" //Tanımlama 1. tanımlama şekli
#include <SPI.h> //Tanımlama 2. tanımlama şekli
```

- Söz Dizimi için kullanılanlar;
  - ***#include;*** Arduino için yazılmış **kütüphanelere erişimi** sağlamak için kullanılmaktadır.
  - SPI kütüphanesini kullanmak ve onun içerisindeki komutlara ulaşmak istendiğinde program başı **tanımlanması ("...") ya da <> şekilde** olmak üzere iki şekilde yapılmaktadır.

#include <SPI.h> **#include "SPI.h«**

  - #include deyiminden sonra **hiçbir noktalı virgül veya eşittir bulunmamalıdır.**

Arduino’da Yapılar

- Söz Dizimi için kullanılanlar;
  - ***#define*** – **ön işlemci komutu olup kullanılan bir isim yerine başka bir ismin kullanımını ve değişimini sağlamaktadır.** Programın derlenmesinden önce programcının sabit bir değere isim vermesine izin vermektedir. “#” işaretinin kullanılması gereklidir. #define deyiminden sonra hiçbir noktalı virgül veya eşittir bulunmamalıdır.

 # define PI 3.14

.

.

alan= PI * yaricap * yaricap

```
/* Çember alanını hesaplar */

#include<stdio.h>
#define PI 3.14
int main( void )
{
        int yaricap;
        float alan;
        printf( "Çemberin yarı çapını giriniz> " );
        scanf( "%d", &yaricap );
        alan = PI * yaricap * yaricap;
        printf( "Çember alanı: %.2f\n", alan );
        return 0;
}
```

Değişken nedir? Nasıl Tanımlanır?

Değişken içerisinde veri tutabileceğimiz hafıza alanlarıdır.

Değişken tanımlanırken türü ve ismi belirtilir. Tanımlama yapıldıktan sonra istenirse değişkene değer ataması yapılabilir.

ÖRNEK;

int

Uzun ismi integer olan bu veri tipi ile değişkenler içerisinde tam sayı verisi atanabilir. -32768 ile 32768 arasında değer alabilir.En çok Kullanılan veri tipidir.

Arduinoda Kullanacağımız Veri Türleri ve sabit tanımlama

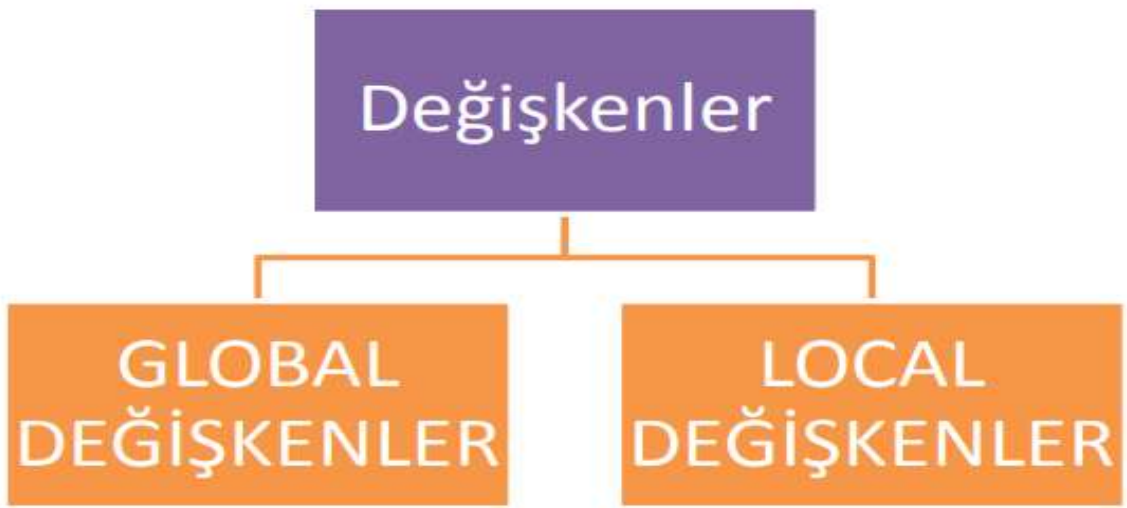
VERİ TÜRÜ	ALABİLECEĞİ DEĞER ARALIĞI	ÖRNEK
int	-32768 ile 32768 arası	int sayi=100;
char	-127 ile127 arası	char karakter=67 char karakter='A'
long	-2 147 483 648 ile 2 147 483 647 arası	long uzunTamsayi=1000000
float	Ondalıklı sayılar için kullanılır	float pi=3.1415

const ile değişken değerini sabitlemek

Program boyunca değiştirilmesini istemediğimiz değişkenlerin önüne const yazarak sabit Haline getirebiliriz

```
const float pi=3.14;
```

GLOBAL VE LOCAL DEĞİŞKENLER



Global değişkenler program içerisinde her yerde kullanılabilir. Fonksiyonların dışında tanımlama yapılmalıdır

Local değişkenler sadece tanımlandığı fonksiyon içerisinde kullanılabilir. Değişken Kullanılacağı fonksiyon içerisinde tanımlanmalıdır.

GLOBAL VE LOCAL DEĞİŞKEN KULLANIMI

```
int sayi1=10;    //Global değişken,her yerde kullanılabilir
int sayi2=20;   //Global değişken,her yerde kullanılabilir
int toplam;     //Global değişken,her yerde kullanılabilir

void setup() {
  int sayi3=500; // setup fonksiyonu içerisinde geçerlidir LOCAL değişken

}

void loop() {
  int sayi4=600; // loop fonksiyonu içerisinde geçerlidir.. LOCAL değişken


}
```


OPERATÖRLER

Artimetiksel Operatörler

- **= Atama İşleci**

C programlama dilindeki tek eşit işareti atama operatörü olarak adlandırılır. Bir denklem veya eşitlik gösterdiği cebir sınıfından farklı bir anlam taşır. Atama işleci, mikrodenetleyiciye eşittir işaretinin sağ tarafında bulunan herhangi bir değer veya ifadeyi değerlendirmesini ve eşit işaretin solundaki değişkende saklamasını söyler.

int algiDeg; // algiDeg adını taşıyan bir tamsayı değişkeni tanımlanmış

algiDeg=analogRead (1); //analog pin 1'in değerini saklar

```
int algiDeg;                  // algiDeg adını taşıyan bir tamsayı değişkeni
algiDeg = analogRead(1); // analog pin 1 in giriş gerilimi algiVal de saklanmaktadır
```

Artimetiksel Operatörler

- **+ Toplama, - Çıkarma, * Çarpma ve / Bölme** C'de programlama yaparken matematiksel işlemlerde aritmetik operatörler kullanılır. int ya da float (virgüllü) değerinden sonuçlar bulunabilir.

```
Y = y + 4;
X = x - 8;
I = j * 7;
R = r / 6;
```

Artimetiksel Operatörler

- **% Mod**

Bir tam sayı belirlenen bir bölüme ayrıldığında kalanını hesaplar. Belirli bir aralıktaki bir değişkeni tutmak için de kullanılmaktadır.

```
x = 7% 5; // X şimdi 2 içerir
x = 9% 5; // X şimdi 4 içerir
x = 5% 5; // X şimdi 0 içerir
x = 4% 5; // x şimdi 4 içerir
```

ÖRNEK UYGULAMA:

```
1  float a=5;    //float, double veri tipi gibi  noktalı sayılar için kullanılan veri tipidir
2  float b=3;
3  float toplam,fark,carpim,bolme;
4  void setup() {
5  Serial.begin(9600);
6  toplam=a+b;
7  fark=a-b;
8  carpim=a*b;
9  bolme=a/b;
10 Serial.print("Toplama işleminin Sonucu=");Serial.println(toplam);
11 Serial.print("Çıkarma işleminin Sonucu=");Serial.println(fark);
12 Serial.print("Çarpma  işleminin Sonucu=");Serial.println(carpim);
13 Serial.print("Bölme işleminin Sonucu=");Serial.println(bolme);
14 }
15
16 void loop() {
17   // put your main code here, to run repeatedly:
18
19 }
```

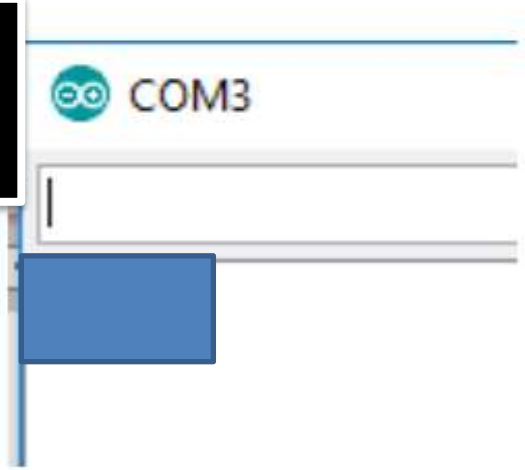
```
float a=5;
float b=3;
float carpim,bolme;
void setup() {
Serial.begin(9600);
carpim=a*b;
bolme=a/b;
Serial.print("çarpım işlemi sonucu=");Serial.println(carpim);
Serial.print(«bölme işlemi sonucu=");Serial.println(bolme);
}
void loop(){

}
```

MATEMATİKSEL İŞLEMLERDE İŞLEM ÖNCELİĞİ

Sıra	isim	Sembol	Örnek İşlem	İşlem Sonucu
1.sıra	Parantez	()	(5+10+15)/3	10
3.sıra	Çarpma bölme	*,/	12/3 ile 6*3	4 ile 18
4.sıra	Toplama çıkarma	+,-	5+6 ile 6-3	11 ile 3

```
1 float denklem;
2 void setup() {
3   Serial.begin(9600);
4   denklem=(3+3)/2+10*3;
5   Serial.print(denklem);
6 }
7 void loop() {
8   // put your main code here, to run repeatedly:
9 }
```

Denklem=(3+3)/2+10*3

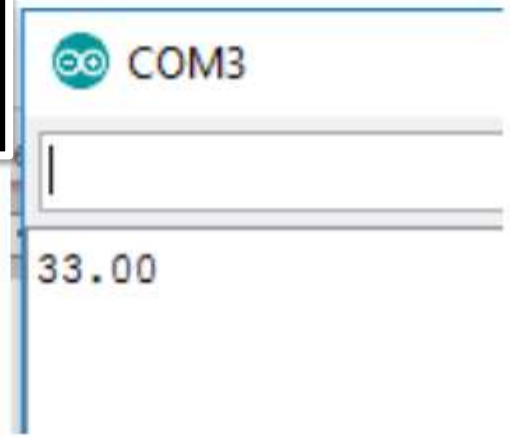


MATEMATİKSEL İŞLEMLERDE İŞLEM ÖNCELİĞİ

Sıra	isim	Sembol	Örnek İşlem	İşlem Sonucu
1.sıra	Parantez	()	(5+10+15)/3	10
3.sıra	Çarpma bölme	*,/	12/3 ile 6*3	4 ile 18
4.sıra	Toplama çıkarma	+,-	5+6 ile 6-3	11 ile 3

Denklem=(3+3)/2+10*3

```
1 float denklem;
2 void setup() {
3   Serial.begin(9600);
4   denklem=(3+3)/2+10*3;
5   Serial.print(denklem);
6 }
7 void loop() {
8   // put your main code here, to run repeatedly:
9 }
```



Karşılaştırma Operatörleri

- **== (eşit eşit),** **!= (eşit değil),**
- **< (küçük),** **> (büyük),**
- **<= (küçük eşit),** **>= (büyük eşit)**

- **Genellikle if içerisinde** karşılaştırma yaparken kullanılan operatörlerdir. “if” karşılaştırma operatörüyle birlikte kullanıldığında, belli bir değerin üzerinde olup olmadığı test edilir.

- Parantez içindeki ifadeler doğruysa parantez içindeki ifadeler çalıştırılır. Doğru değilse program kod üzerinde atlanır.

```
if (degisken> 100)
{
//değişken 100 den büyükse buraya girilir
}
```

x==y

x!=y

x<y

x>y

x<=y

x>=y

```
x == y (x, y ye eşit)
x != y (x, y ye eşit değil )
x < y (x, y den küçük)
x > y (x, y den büyük)
x <= y (x, y den küçük eşit)
x >= y (x, y den büyük eşit)
```

ARDUİNO DA KARŞILAŞTIRMA OPARATÖRLERİ

Arduinoda'ki veriler üzerinde karşılaştırma yapmak için Karşılaştırma operatörleri kullanılır.

Doğru-True-1

Yanlış-False-0

Operatör	Anlamı	Örnek	Açıklama
==	Eşittir	x==y	X değeri y değerine eşitse Sonuç doğru(true)=1, değilse Yanlıştır(False)=0
<	Küçüktür	x<y	X değeri y değerinden küçükse sonuç doğru(true)=1 ,değilse Yanlıştır(False)=0
>	Büyüktür	x>y	X değeri y değerinden büyükse sonuç doğru(true)=1, değilse Yanlıştır(False)=0
<=	Küçük veya eşittir	x<=y	X değeri y değerinden küçük veya eşitse sonuç doğru(true) = 1 ,değilse Yanlıştır(False)=0
>=	Büyük veya eşittir	x>=y	X değeri y değerinden büyük veya eşitse sonuç doğru(true)=1 değilse Yanlıştır(False)=0
!=	Eşit değildir?	x!=y	X değeri y değeri ile eşit değilse Sonuç doğru(true)=1 değilse Yanlıştır(False)=0

Örnek

Operatör	Anlamı	Örnek	sonuç
==	Eşittir	6==6	1-Doğru-True
<	Küçüktür	3<2	0-Yanlış-false
>	Büyüktür	8>5	1-Doğru-true
<=	Küçük veya eşittir	3<=3	1-Doğru-true
>=	Büyük veya eşittir	6>=7	0-Yanlış-False
!=	Eşit değil	4!=3	1-Doğru-True

Boolean Operatörleri

- **&& (mantıksal ve)**

Boolean ifadeleri genellikle “if” yapısının içerisinde ve belirli şart gerektiren durumlarda kullanılmaktadır. **Yalnızca her iki işlem de doğruysa “if” şartı doğrudur.** Bu durumda “if” içerisindeki komut çalıştırılır. Herhangi biri yanlış ya da ikisi de false, yanlış sonuç ise if yapısı çalıştırılmaz. Örnekte oku1 ve oku2 HIGH ise LED yanmaktadır.

```
void loop(){ //sonsuz döngü
int okul= digitalRead(1);
int oku2= digitalRead(2);
if (okul == HIGH && oku2 == HIGH) {
digitalWrite(led, HIGH); // ledi yak
}
}
```

Boolean Operatörleri

- **&& (mantıksal ve)**

Örnekte aynı anda oku1 ve oku2 HIGH ise LED yanmaktadır.

```
void setup(){
digitalMod(1,OUTPUT);
digitalMod(2, OUTPUT)
}
void loop(){
int oku1=digitalRead(1);
int oku2=digitalRead(2);
if (oku1==HIGH && oku2==HIGH){
digitalWrite(led, HIGH); //ledi yak }
}
```

```
void loop(){ //sonsuz döngü
int okul= digitalRead(1);
int oku2= digitalRead(2);
if (okul == HIGH && oku2 == HIGH) {
digitalWrite(led, HIGH); // ledi yak
}
}
```

&& (VE) OPERATÖRÜ:

&& operatörü bütün karşılaştırmaların doğru yani (1) olması durumunda 1, diğer durumlarda yanlış (0) çıkışı verir. && sembolleri yerine “and” komutuda kullanılabilir

Karşılaştırma 1	Karşılaştırma 2	Sonuç
1	1	1
1	0	0
0	1	0
0	0	0

```
1  int a=5;
2  int b=10;
3  void setup() {
4    Serial.begin(9600);
5    Serial.println(a>2 && b==10);
6    Serial.println(a==2 && b==3);
7    Serial.println(a<6 && b>15);
8    Serial.println(a>=10 && b<=12);
9  }
10 void loop() {
11   // put your main code here, to run repeatedly:
12
13 }
```

a=5 b=10

a>2 && b==10;
a==2 && b==3;
a<6 && b>15;
a>=10 && b<=12;

```
1  int a=5;
2  int b=10;
3  void setup() {
4    Serial.begin(9600);
5    Serial.println(a>2 && b==10);  //1 ve 1 sonuç 1
6    Serial.println(a==2 && b==3);  //0 ve 0 sonuç 0
7    Serial.println(a<6 && b>15);   //1 ve 0 sonuç 0
8    Serial.println(a>=10 && b<=12);//0 ve 1 sonuç 0
9  }
10 void loop() {
11   // put your main code here, to run repeatedly:
12
13 }
```

a>2 && b==10; →1
a==2 && b==3;→ 0
a<6 && b>15; →0
a>=10 && b<=12;→0

Boolean Operatörleri

- **|| (mantıksal veya)**

Her iki işlemden herhangi birisi doğruluk şartını taşıyorsa if içerisindeki komut çalıştırılır. **Örnekte oku1 veya oku2 HIGH ise led yanmaktadır**

```
void loop(){ //sonsuz döngü
int okul= digitalRead(1);
int oku2= digitalRead(2);
if (okul == HIGH || oku2 == HIGH) {
digitalWrite(led, HIGH); // ledi yak
}
}
```

işaretleri yerine " or" komutuda kullanılabilir

Karş. 1	Karş. 2	Sonuç
1	1	1
1	0	1
0	1	1
0	0	0

a=5 b=10

a>2 || b==10;
a==2 || b==3;
a<6 || b>15;
a>=10 || b<=12;

```
1  int a=5;
2  int b=10;
3  void setup() {
4    Serial.begin(9600);
5    Serial.println(a>2 || b==10);
6    Serial.println(a==2 || b==3);
7    Serial.println(a<6 || b>15);  /
8    Serial.println(a>=10 || b<=12);
9  }
10 void loop() {
11   // put your main code here, to run repeatedly:
12
13 }
```

a>2 || b==10; → 1
a==2 || b==3; → 0
a<6 || b>15; → 1
a>=10 || b<=12;→ 1

```
1  int a=5;
2  int b=10;
3  void setup() {
4    Serial.begin(9600);
5    Serial.println(a>2 || b==10);  //1 veya 1 sonuç 1
6    Serial.println(a==2 || b==3);  //0 veya 0 sonuç 0
7    Serial.println(a<6 || b>15);  //1 veya 0 sonuç 1
8    Serial.println(a>=10 || b<=12);//0 veya 1 sonuç 1
9  }
10 void loop() {
11   // put your main code here, to run repeatedly:
12
13 }
```

Boolean Operatörleri

- **! (mantıksal değil)**

Değer olarak verilen ifadenin sıfır olma durumudur. İfadenin sıfır olma şartı sağlanıyor ise if yapısı çalıştırılmaktadır.

Örnekte buton1'e basılmıyor ise LED'i söndürülmektedir

```
if (!buton1) {   //buton 1 baslı değil ise
digitalWrite (led, LOW ) ; // ledi söndür
```

```
if (!butonl) {
digitalWrite(led, LOW); // ledi söndür
}
```

```
1
2  void setup() {
3    Serial.begin(9600);
4    Serial.println(5>3); // bu karşılaştırma sonucu Doğru (1) dir
5    Serial.println(!(5>3));  //! işareti
6    }
7
8  void loop() {
9    // put your main code here, to run repeatedly:
10
11 }
```

```
1
2  void setup() {
3    Serial.begin(9600);
4    Serial.println(5>3); // bu karşılaştırma sonucu Doğru (1) dir
5    Serial.println(!(5>3));  //! işareti doğru(1) olan karşılaştırmanın tersini alarak sonucu Yanlış(0) yap
6    }
7
8  void loop() {
9    // put your main code here, to run repeatedly:
10
11 }
```

Birleşik Operatörler

- **a++ (arttırma) ve -- (azaltma)**

Bir değişkende arttırma veya azaltma yapılmasını sağlamaktadır.

```
X = 2;
Y = ++ x; // şimdi x 3 içerir   // y 3 içerir
y = x--; // x tekrar 2 içerir, y 2  içerir
```

```
X ++; // x değerini birer arttırır ve eski x değerini döndürür
++ x; // x değerini birer artırır ve x'in yeni değerini döndürür

X--; // x değerini birer azaltır ve eski x değerini döndürür
--x; // x değerini birer azaltır ve yeni x değerini döndürür
```

Birleşik Operatörler

- **+= (birleşik artırma),**
- **-= (birleşik çıkarma),**
- ***= (birleşik çarpma),**
- **/= (birleşik bölme)**
- **%=(birleşik mod)**

Bu operatörler değişken üzerinde başka bir sabit veya değişkenle matematiksel işlem yapmak için kullanılmaktadır.

Atama işlemi	Açıklama	Örnek	Sonuç
++	1 arttır	a=1 a++ Serial.print(a)	2
--	1 azalt	b=5 b - - Serial.print(b)	4
+=	Topla Ata	x=10 x+=5 Serial.print(x)	15
-=	Çıkar Ata	y=20 y-=10 Serial.print(y)	10
*=	Çarp Ata	z=5 z*=6 Serial.print(z)	30
/=	Böl Ata	t=50 t/=5 Serial.print(t)	10

```
x = 2;
x += 4;        // x şimdi 6 içerir
x -= 3;        // x şimdi 3 içerir
x *= 10;       // x şimdi 30 içerir
x /= 2;        // x şimdi 15 içerir
x %= 5;        // x şimdi 0 içerir**
```

**Mod alınıyor. Mod alma x in y ile bölümünden kalan sayıdır.
x=15%5; olduğunda x=0 olacaktır.

```
X + = y; // x = x + y ifadesine eşdeğerdir
X - = y; // x = x - y ifadesine eşdeğerdir
X * = y; // x = x * y ifadesine eşdeğerdir
X / = y; // x = x / y ifadesine eşdeğerdir
X% = y; //  x = x% y; ifadesine eşdeğerdir
```

X: herhangi bir değişken türü
Y: herhangi bir değişken türü veya sabit

```
float a=10;
float b=20;
float c=30;
float d=40;
float e=50;
float f=60;
void setup() {
  Serial.begin(9600);
  a++;  //++ opatatörü a değişkenin değerini
  Serial.println(a);
  b--; //-- oparatörü b
  Serial.println(b);
  c+=10; // topla ata:c değişkenin
  Serial.println(c);
  d-=3;  // Çıkar ata:d değişkenin
  Serial.println(d);
  e*=2;  // Çarp ata:  e değişkeninde
  Serial.println(e);
  f/=3;  //Böl ata: x değişkeni
  Serial.println(f);
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

```
float a=10;
float b=20;
float c=30;
float d=40;
float e=50;
float f=60;
void setup() {
  Serial.begin(9600);
  a++;  //++ opatatörü a değişkenin değerini
  Serial.println(a);
  b--; //-- oparatörü b
  Serial.println(b);
  c+=10; // topla ata:c değişkenin
  Serial.println(c);
  d-=3;  // Çıkar ata:d değişkenin
  Serial.println(d);
  e*=2;  // Çarp ata:  e değişkeninde
  Serial.println(e);
  f/=3;  //Böl ata: x değişkeni
  Serial.println(f);
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

a=10
b=20
c=30
d=40
e=50
f=60

a++
b--
c+=10
d-=3
e*=2
f/=3

a=10	a++
b=20	b--
c=30	c+=10
d=40	d-=3
e=50	e*=2
f=60	f/=3

```
float a=10;
float b=20;
float c=30;
float d=40;
float e=50;
float f=60;
void setup() {
  Serial.begin(9600);
  a++;  //++ opatatörü a değişkenin değerini 1 arttırır. Çıktı 11 olacak
  Serial.println(a);
  b--; //-- oparatörü b değişkeni değerini 1 azaltır. Çıktı 19 olacak
  Serial.println(b);
  c+=10; // topla ata:c değişkenin var olan değeri üzerine 10 ekler. Çıktı 40 olaca
  Serial.println(c);
  d-=3;  // Çıkar ata:d değişkenin var olan değerinde 3 değerini çıkarır. Çıktı 37
  Serial.println(d);
  e*=2;  // Çarp ata:  e değişkeninde var olan değeri 2 ile çarpar. Çıktı 100 olaca
  Serial.println(e);
  f/=3;  //Böl ata: x değişkeni var olan değeri 3 e böler :Çıktı 20 olacak
  Serial.println(f);
}

void loop() {
  // put your main code here, to run repeatedly:

}
```