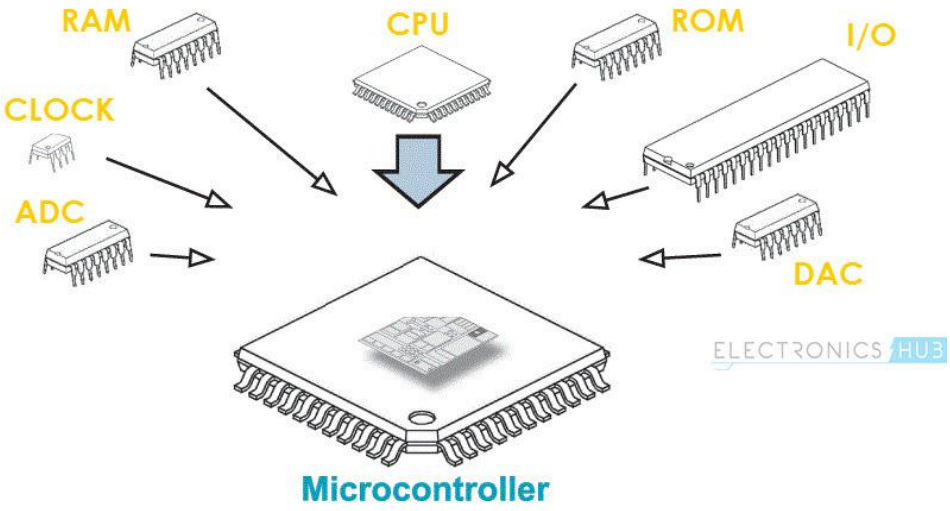
ROBOTİK VE KODLAMA DERS NOTLARI

Mikrodenetleyici Nedir?

Mikrodenetleyici, dışarıdan gelen bir veriyi (programı) hafızasına alan, derleyen ve sonucunda da çıktı elde eden bir bilgisayardır. Mikrodenetleyicinin yapısında:

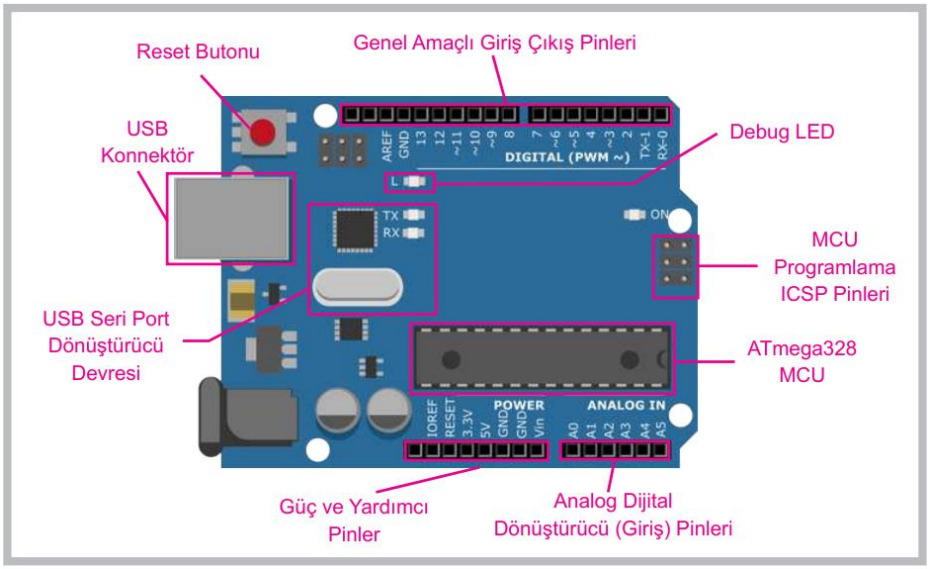
- CPU (İşlemci)
- RAM (Geçici bellek)
- ROM (Kalıcı, sadece okunabilir bellek)
- I/O Portları (Giriş/Çıkış Portları - INPUT/OUTPUT)
- Seri ve Paralel Portlar
- Sayıcılar
- Bazılarında da A/D (Analog to Digital) ve D/A (Digital to Analog) çeviriciler bulunur.



Mikrodenetleyici Uygulama Kartı

Bünyesinde **mikrodenetleyici** yapısına ek olarak robotik projelerin yapımını kolaylaştıran birimleri de üzerinde taşıyan elektronik geliştirme kartlarına **mikrodenetleyici uygulama kartları** denir.

En çok kullanılan ve dersimizin de konusu olan uygulama geliştirme kartı **ARDUINO** olarak bilinen geliştirme kartıdır.

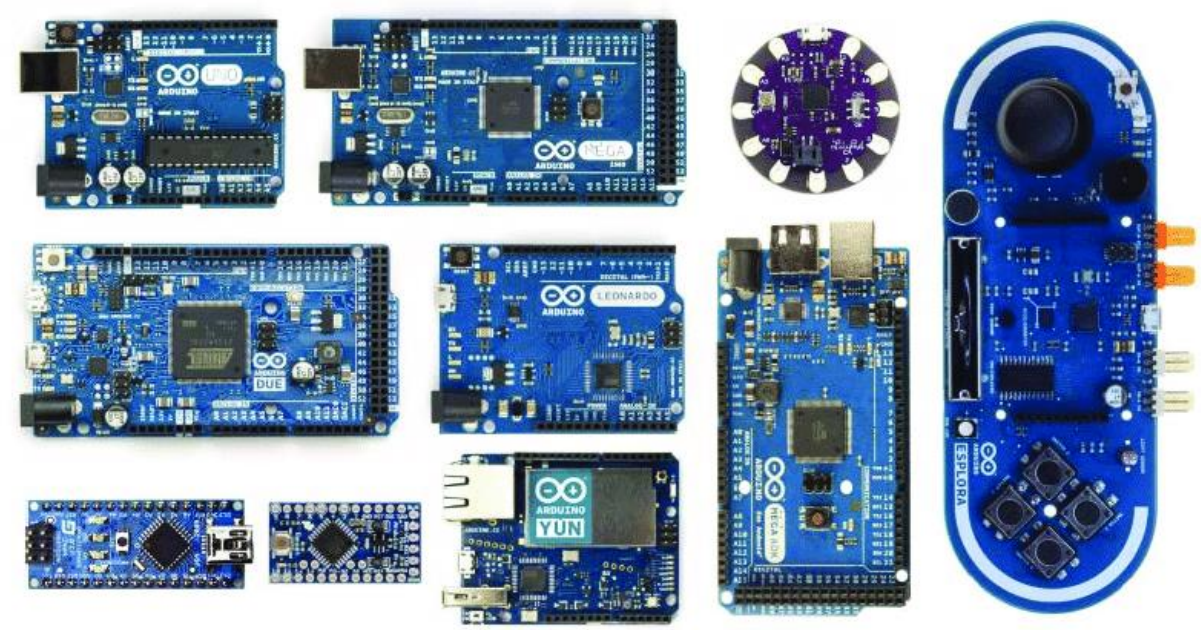


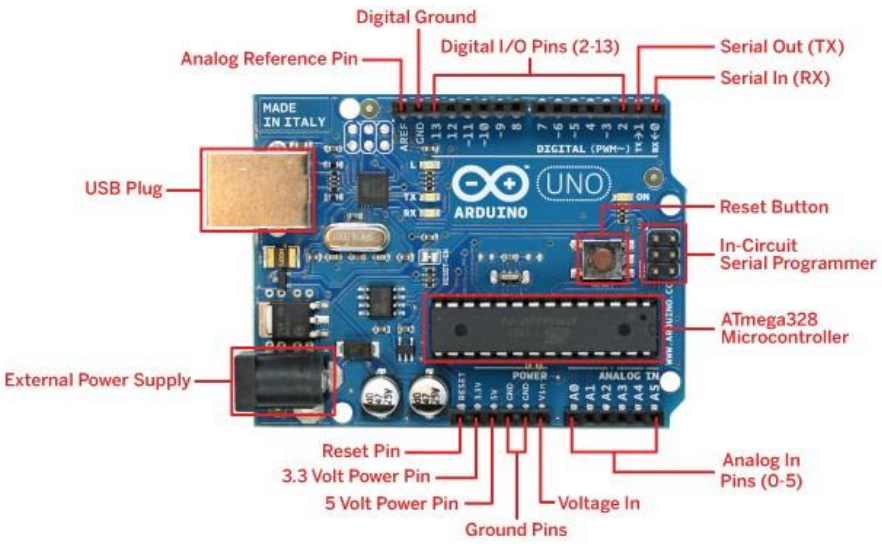
Arduino Nedir?

Kullanımı kolay, açık kaynaklı donanım ve yazılımdan oluşan, programlanabilir uygulama geliştirme kartıdır.

Arduino Çeşitleri:

- Arduino Uno
- Arduino Pro
- Arduino Pro Mini
- Arduino Nano
- Arduino Micro
- Arduino Leonardo
- Arduino Mega
- Arduino Mega ADK
- Arduino Zero veya Arduino M0 Pro
- Arduino Due
- Arduino Esplora
- Arduino 101
- LilyPad Arduino veya LilyPad Arduino USB
- Arduino Yun





ROBOT TÜRLERİ VE EĞİTSEL AMAÇLI ROBOTLAR

Robot Nedir?

Robot, bir programlama algoritması ile çalıştırılan elektromekanik parçalardan oluşan akıllı sistemlerdir. Robotlar dış dünyayı **sensörler (algılayıcı)** ile algılayıp veri toplayarak bir sonuç üretirler. Bunu **otonom (kendi kendine)** ya da bir kullanıcının yönlendirmesiyle yapabilirler.

Robotlar, farklı kontrol yöntemleriyle kullanılmaktadır.

Kontrol yöntemlerine göre robotlar şu şekilde ayrılmaktadır:

1. **Tepkisel (Reactive) Robotlar**: Etki-Tepki (Algılama-Cevap) prensibiyle çalışan robotlardır.
2. **Bilinçli (Deliberative) Robotlar**: Algılama-Planlama-Hareket prensibiyle çalışan robotlardır.
3. **Karma (Hibrit) Robotlar**: Tepkisel ve Bilinçli kontrol yöntemlerinin birleşmesinden oluşmaktadır.
4. **Davranışsal (Behavioral) Robotlar**: Karma kontrole alternatif olarak sunulmuştur. Tepkisel ve Bilinçli hareket özelliklerine sahiptir.

Kullanım alanlarına göre robotlar şu şekilde sınıflandırılmaktadır:

1. Ev robotları
2. Tıbbi robotlar
3. Askerî robotlar
4. Uzay robotları
5. Hobi robotları
6. Eğitsel robotlar
    a. Blok Tabanlı Robot Montaj Setleri
    b. Düşük Maliyetli Mobil Robot Tasarım Kitleri
    c. Açık Kaynak Mobil Robot Platformları
    d. Tam Monte Edilmiş Mobil Robotlar
    e. Minyatür Sürü Robotları

Robotta Mekanik / Elektromekanik Bileşenler

Robotlar, programlanabilen elektronik kartlar haricinde gövdesini ve hareketli parçalarını oluşturan plastik veya metal bileşenlerden meydana gelmektedir.

Robotun mekanik / elektromekanik bileşenleri 7 kısımda incelenebilir:

- Robot gövdeleri
- Motorlar
- Tekerlek, ayak ve paletler
- Eklenti ve bağlantı bileşenleri
- Vida, somun ve rondela bileşenleri
- Amortisör, yay ve esnek bileşenler
- Mekanik veya vakumlu nesne tutucu bileşenler

Gövde	DC Motor	Servo Motor	Tekerlek	Vida ve somunlar
				

Robotta Elektronik Bileşenler

Bir robottaki mekaniksel parçaları kolay bir şekilde kontrol etmek ve dışarıdan alınan fiziksel değişikliklere göre farklı yönlendirmelerde bulunmak için elektronik bileşenlere ihtiyaç vardır.

Robotlarda kullanılan elektronik bileşenler şunlardır:

- Mikrodenetleyici kartlar
- Motor sürücü kartları
- Sensörler
- Kablosuz erişim kartları

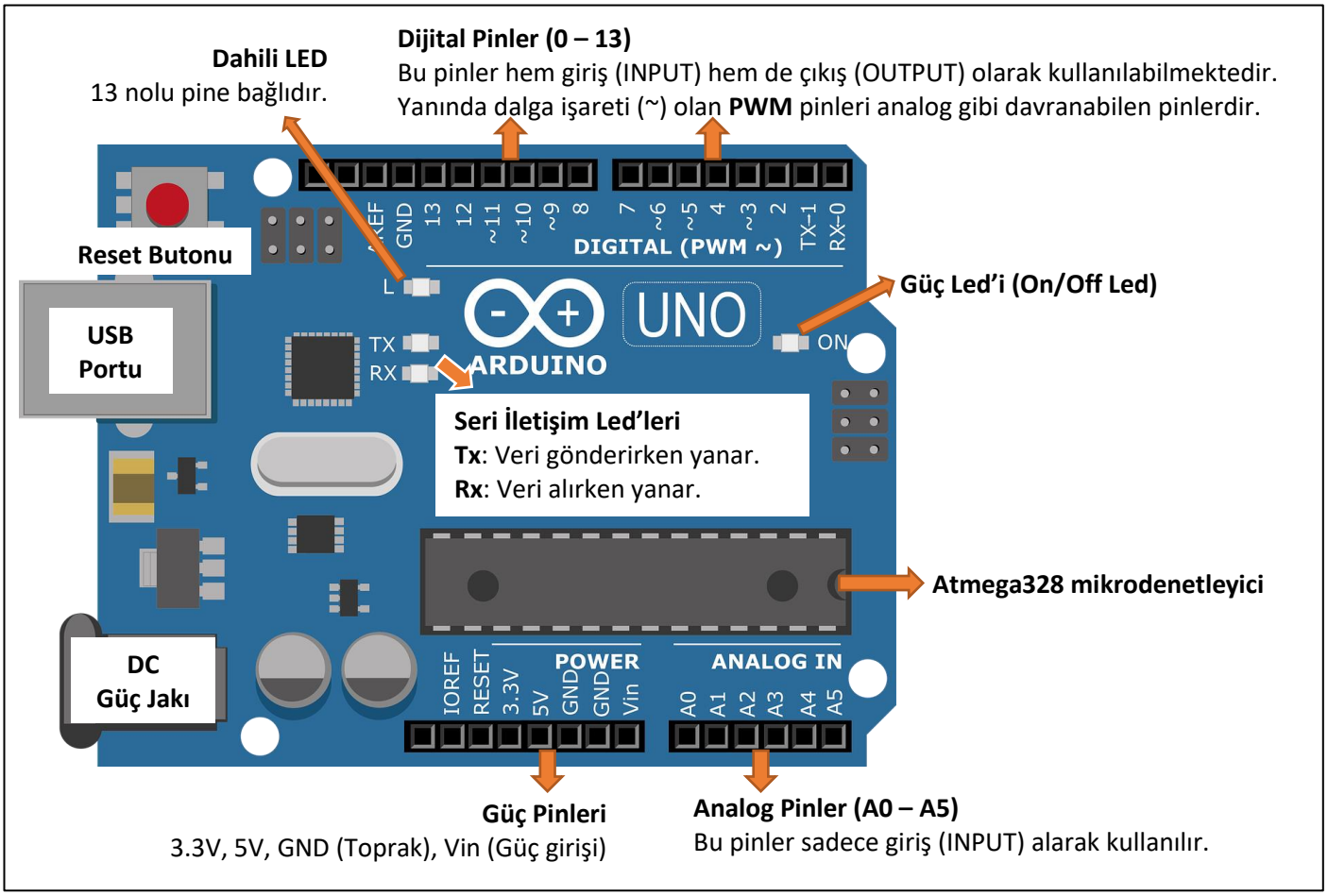
Mikrodenetleyici kart (Arduino)	Mikrodenetleyici kart (Microbit)	DC Motor Sürücü Kart	Servo Motor Sürücü Kart
			

Sensörler		
		
a. Çizgi izleme sensörü	b. Renk sensörü	c. Ultrasonik mesafe sensörü

Kablosuz Erişim Kartları		
		
a. Bluetooth	b. ESP8266 Wi-Fi	c. NRF24L01

Temel Bilgiler

Arduino'nun Yapısı Hakkında



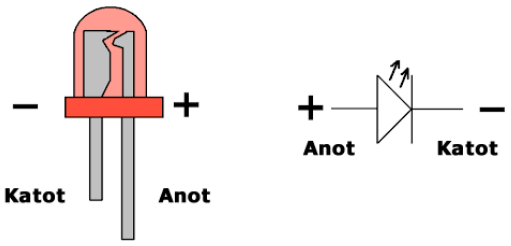
- Arduino'nun genel çalışma voltajı 0V – 5V'dur.
- 0V → LOW (0) olarak kabul edilir.
- 5V → HIGH (1) olarak kabul edilir.
- GND topraktır ve 0V kabul edilir.
- Dijital pinler hem **giriş** (**INPUT**) hem de **çıkış** (**OUTPUT**) olarak kullanılabilir.
    - **Çıkış** olarak kullanılan pini programlamak için o pine yazmalısınız. **digitalWrite(pin, değer)**

      Örnek: 2 nolu pine yazalım.
      digitalWrite(2, LOW) ya da **digitalWrite(2, HIGH)**
      Bir pine **LOW** yazarsanız o pini 0V yapmış olursunuz.
      Bir pine **HIGH** yazarsanız o pini 5V yapmış olursunuz.

    - Bir pini **giriş** olarak kullanıyorsanız o pini okumalısınız. **digitalRead(pin)**

      Örnek: 2 nolu pini okuyalım.
      digitalRead(2)
      Bir pini okuduğunuzda ya **LOW** ya da **HIGH** elde etmiş olursunuz. Elde ettiğiniz bu değeri **if** ile kontrol edebileceğiniz gibi bir değişkene de atayabilirsiniz. Örneğin;
      if(digitalRead(2) == HIGH) ya da **int deger = digitalRead(2);**
- Analog pinler ise sadece **giriş** (**INPUT**) olarak kullanılabilir. Dolayısıyla bu pinleri okumalısınız. Okuduğunuz değeri bir değişkene atamalısınız. Bu değer **0-1023** arasında bir değer olacaktır.
  int deger = analogRead(pin); → **Örnek**: `int deger = analogRead(2);`

LED Nedir?



LED, ışık yayan bir çeşit diyottur. LED'ler artı ve eksi uçlara sahiptir. Bu sebeple devrelere bağlanırken yönüne dikkat edilerek bağlanmalıdır.

Uzun ayağı anot (+), kısa ayağı ise katot (-) ayağıdır.

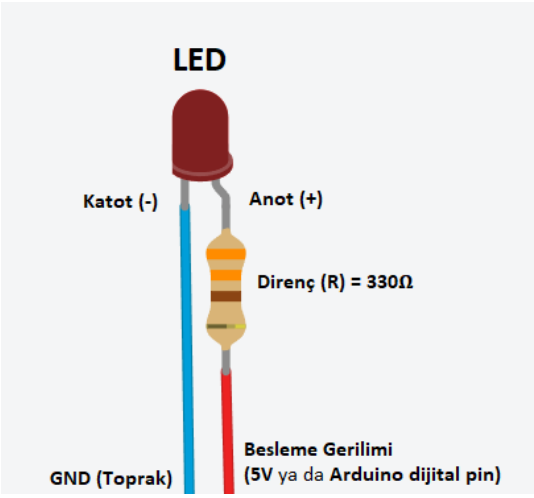
- LED'ler devrelerde **çıkış (OUTPUT)** elemanı olarak kullanılır.
- Led'ler çalıştıklarında **20 mA (0.02A)** akım çeker.
- Renklerine göre **Kırmızı 2V**, **Sarı 2.1V**, **Yeşil 2.1V**, **Mavi 3.5V**, **Beyaz 3.6V** gerilimle çalışmaktadır.
- Led'ler direk olarak Arduino pinlerine bağlanırsa bozulabilirler. LED için Arduino'nun besleme gerilimi olan 5V fazla olduğundan üzerine düşen gerilimi azaltmak için seri bir direnç ile beraber kullanılmalıdır.
- Seri direnç değerini şu formülle hesaplayabilirsiniz:

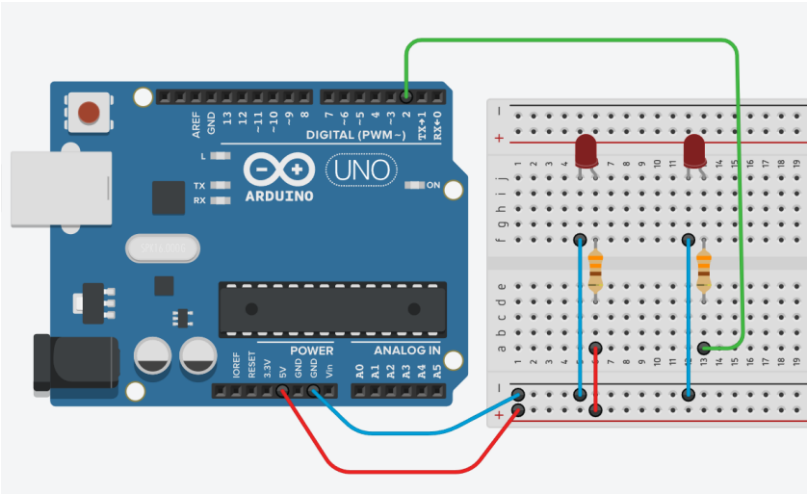
  Direnç = (Besleme Gerilimi – LED Çalışma Gerilimi) / LED Çalışma Akımı
  formülü ile bulabiliriz.

  Örnek: Kırmızı LED için direnci hesaplayalım.
  R = (5V – 2V) / 0.02A = **<u>150 Ohm</u>** (Minimum 150 Ohm)

  Not: Genellikle projelerimizde **220 Ohm** veya **330 Ohm** kullanacağız.

  - 220Ω →  **Kırmızı**(2) - **Kırmızı**(2) - **Kahverengi**(1) - **Altın** (%5 eksik ya da fazla olabilir)
  - 330Ω →  **Turuncu**(3) - **Turuncu**(3) - **Kahverengi**(1) - **Altın** (%5 eksik ya da fazla olabilir)





Kullanılan seri direncin değeri **220Ω** ya da **330Ω** olarak tercih edilebilir. Genellikle direnç anot (+) ayağına bağlanır. Ama böyle olması şart değildir. Direnci katot (-) ayağına da bağlayabilirsiniz.

Anot (+) ayağını doğrudan 5V'a bağlarsanız LED çalışmaya başlar ve herhangi bir kontrol söz konusu değildir.

Eğer LED'i kontrollü olarak yakıp söndürme gibi bir işlem yapacaksanız o zaman anot (+) ayağını Arduino'nun dijital pinlerinden birine bağlamalısınız. Sonrasında yazacağınız kod ile istediğiniz kontrolü yapabilirsiniz.

Direnç Nedir?

Elektrik akımına karşı zorluk gösteren devre elemanıdır.

- Elektronik devrelerde en sık kullanılan devre elemanıdır ve "**R**" harfiyle gösterilir.
- Direncin elektriksel büyüklüğü "**ohm**"dur ve "**Ω**" (omega) harfiyle gösterilir.
- Devreden geçen akımı sınırlamak ve istenilen gerilimi elde etmek için (Gerilim Bölücü) kullanılır.
- Dirençlerin değeri üzerindeki renklerinden hesaplanır. Her rengin sayısal bir karşılığı vardır.

- Direnç üzerindeki 1. ve 2. rengin sayısal değerleri yan yana yazılarak iki basamaklı bir sayı elde edildikten sonra 3. rengin değeri kadar yanına sıfır atılarak direncin değeri hesaplanır.

  Örnek: Aşağıdaki direncin değerini hesaplayalım.

  Sarı (4) – **Mor** (7) – **Kırmızı** (2) → **R = 4700Ω = 4.7KΩ**

  En sondaki **altın** rengi **%5 töleransı** yani **hata payını** temsil eder. Bunun anlamı direncin değeri %5 az ya da fazla çıkabilir demektir. O yüzden gerçek dirençler ölçüldüğünde olması gereken değere yakın bir değer verirler. Bu değer belirtilen tölerans aralığında ise o direnç sağlam kabul edilir.
  Eğer bu renk **gümüş** rengi olursa o zaman tölerans **%10** kabul edilir.

  Örnek: Aşağıdaki direncin değerini hata payını da dahil ederek hesaplayalım.

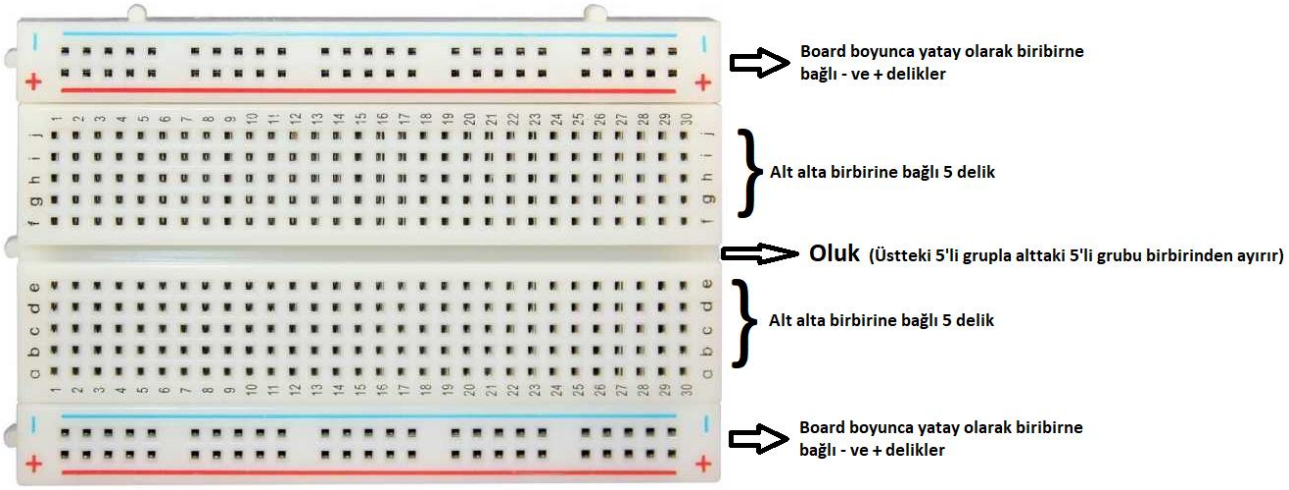
  

  Kahverengi (1) – Yeşil (5) – Sarı (4) – Altın (%5) → **R = 150000Ω = 150KΩ ± %5**
  150K'nin **%5**'i **7.5K** yapar. (150*0.05 = 7.5)
  Bu da şu anlama gelir: Direnç **7.5K** az ya da **7.5K** fazla çıkabilir.
  O zaman direncin aralığı şöyle olur:
  (**150K – 7.5K**) ile (**150K + 7.5K**) = **142.5K** ile **157.5K** aralığı elde edilmiş olur.

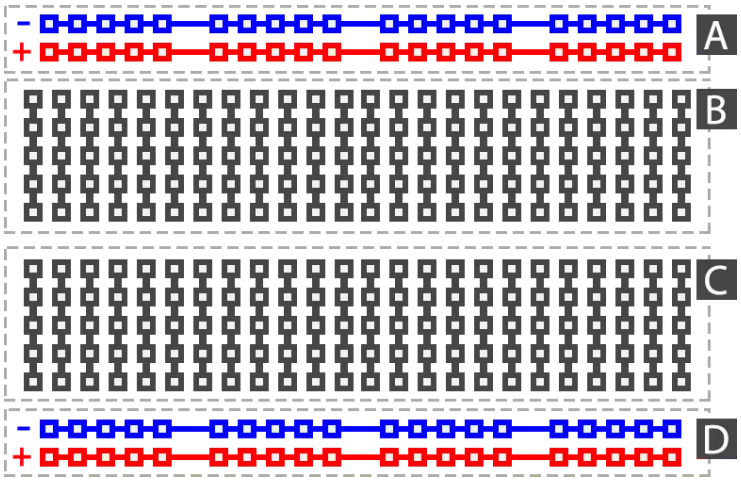
- Dirençler yönsüz devre elemanlarıdır. Yani artı (+) eksi (-) ayakları yoktur. Devreye nasıl bağlandıkları önemli değildir.

Breadboard Nedir?

Breadboard, üzerine elektronik devre elemanlarını takarak proje oluşturmamıza kolaylık sağlayan delikli bir karttır. Devreleri tak çıkar mantığı ile oluşturmamıza yarar. Belirli satır ve sütunları kendi aralarında birbirine bağlı olan delikli devre tahtasıdır.



Breadboard'ın iç yapısında delikler birbirleriyle şu şekilde bağlı yani iletken durumdadır.



- Kartın üst kısmındaki ve alt kısmındaki – ve + olarak gösterilen yatay satır kendi içinde soldan sağa bağlıdır.
- Orta kısımdaki alt alta olan 5 delik ise kendi arasında bağlı durumdadır.
- Bu 5'li delikleri kartın tam ortasındaki **oluk** denilen boşlukla birbirinden ayrılmakadır. Dolayısıyla üstteki 5 delik ile alttaki 5 delik birbirinden ayrıdır.
- Deliklerin birbirine bağlanmasına iletken ya da kısa devre denildiği de olur.
- Birbirine bağlı deliklerden herhangi birine takılan bir uç, bağlı olan diğer deliklerde çoğaltılmış olur. Örneğin kırmızı satıra takılan 5V gerilimi kendisine bağlı olan diğer tüm deliklerden elde edilebilir. Yani 5V, bağlı olan delik sayısı kadar çoğaltılmış olur.

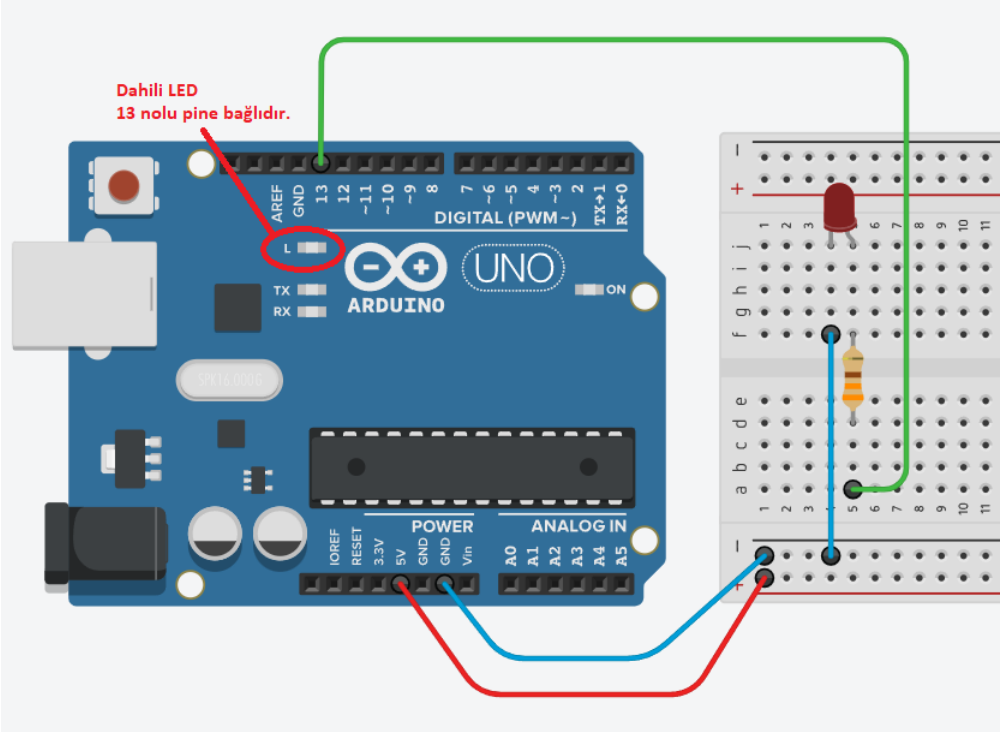
Arduino Örnek Devre Uygulamaları

1. Yanıp Sönen LED Uygulaması

13 nolu pine bağlı LED'i 1sn aralıklarla yakıp söndürelim.

- 13 nolu pine bağlı ayrıca dahili bir LED olduğu için o LED de yanıp sönecektir. Dikkat ediniz!
- LED'in üzerine düşen gerilimi azaltmak için **330Ω** direnç bağlanmalıdır.

Turuncu (3) – Turuncu (3) – Kahverengi (1) → **R = 330Ω**



- LED'in katot (-) ucu **GND** (toprak) ucuna bağlıdır.
- Anot (+) ucu **330Ω** dirençle arduino 13 nolu pine bağlıdır.
- Direnç dikey olarak oluğun üzerinden bağlanmıştır. Dikkat ediniz.
- LED çıkış (**OUTPUT**) devre elemanıdır. Kodumuzu ona göre yazalım.

Arduino Kodu	**Kullandığımız pini değişken tanımlayarak kodu tekrar yazalım.**

`void setup()`
`{`
`  pinMode(13, OUTPUT);`
`}`
 	
`void loop()`
`{`
`  digitalWrite(13, HIGH);`
`  delay(1000);`
 	
`  digitalWrite(13, LOW);`
`  delay(1000);`
`}`
	`int led = 13;`
	
	`void setup()`
	`{`
	`  pinMode(led, OUTPUT);`
	`}`
	
	`void loop()`
	`{`
	`  digitalWrite(led, HIGH);`
	`  delay(1000);`
	
	`  digitalWrite(led, LOW);`
	`  delay(1000);`
	`}`

Not: **LOW** yerine 0, **HIGH** yerine 1 de yazabilirsiniz.

```
digitalWrite(13, 0); // LOW
digitalWrite(13, 1); // HIGH
```

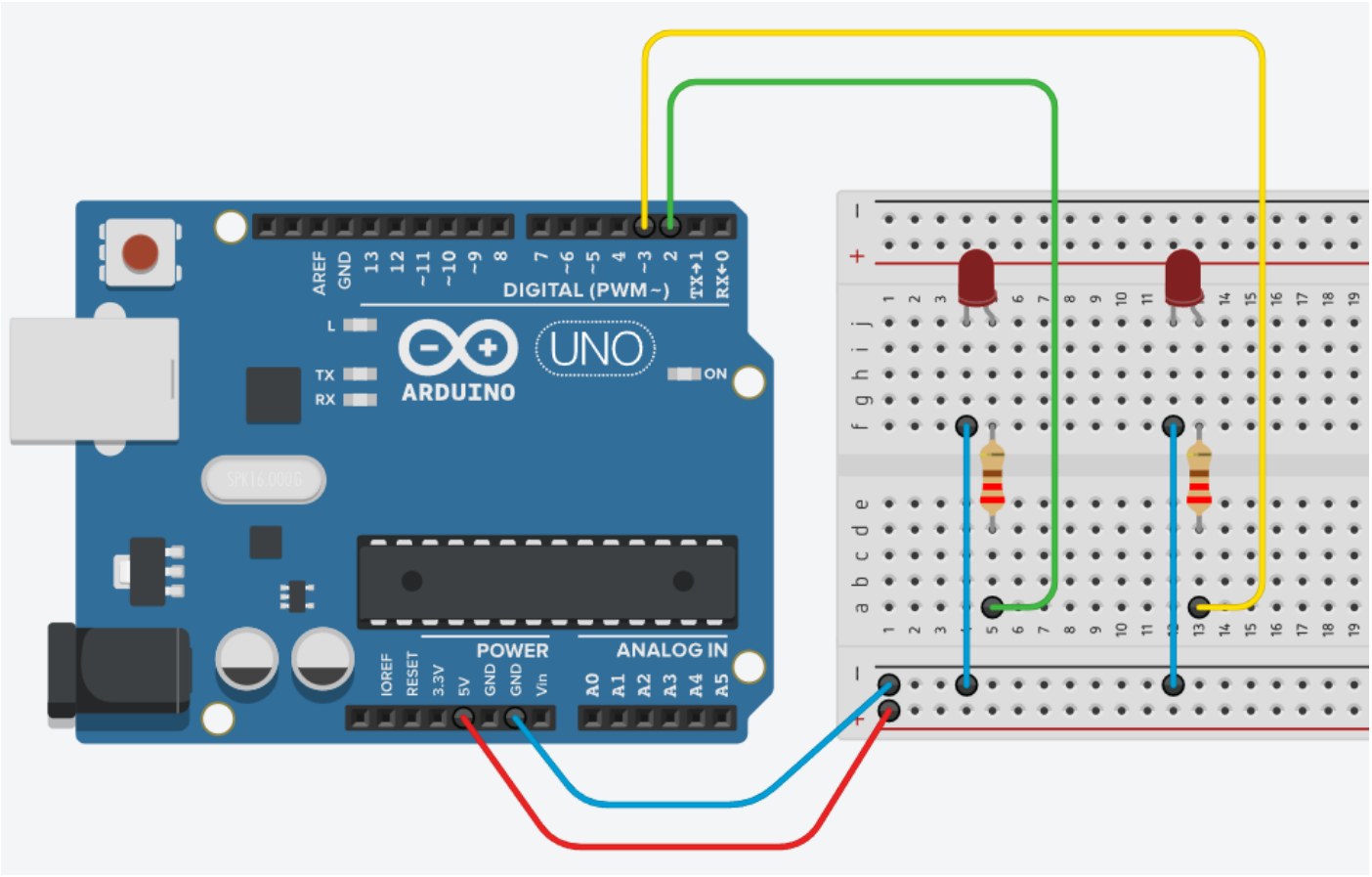
```
digitalWrite(led, 0);
digitalWrite(led, 1);
```

2. Karşılıklı Yanıp Sönen LED Uygulaması (Flip – Flop)

Arduino'da 2 ve 3 nolu pinlere bağlı iki tane led'i karşılıklı olarak yakıp söndürelim. Bir yandığında diğeri sönecektir.

- Bu devrede kullandığımız pinleri değişken olarak tanıtıp kullanalım.
- Bu devrede led'leri korumak için 220Ω değerinde bir direnç kullanalım. Oluğun üzerinden dikey olarak bağlayalım.

Kırmızı(2) - **Kırmızı**(2) - **Kahverengi**(1) → **R = 220Ω**



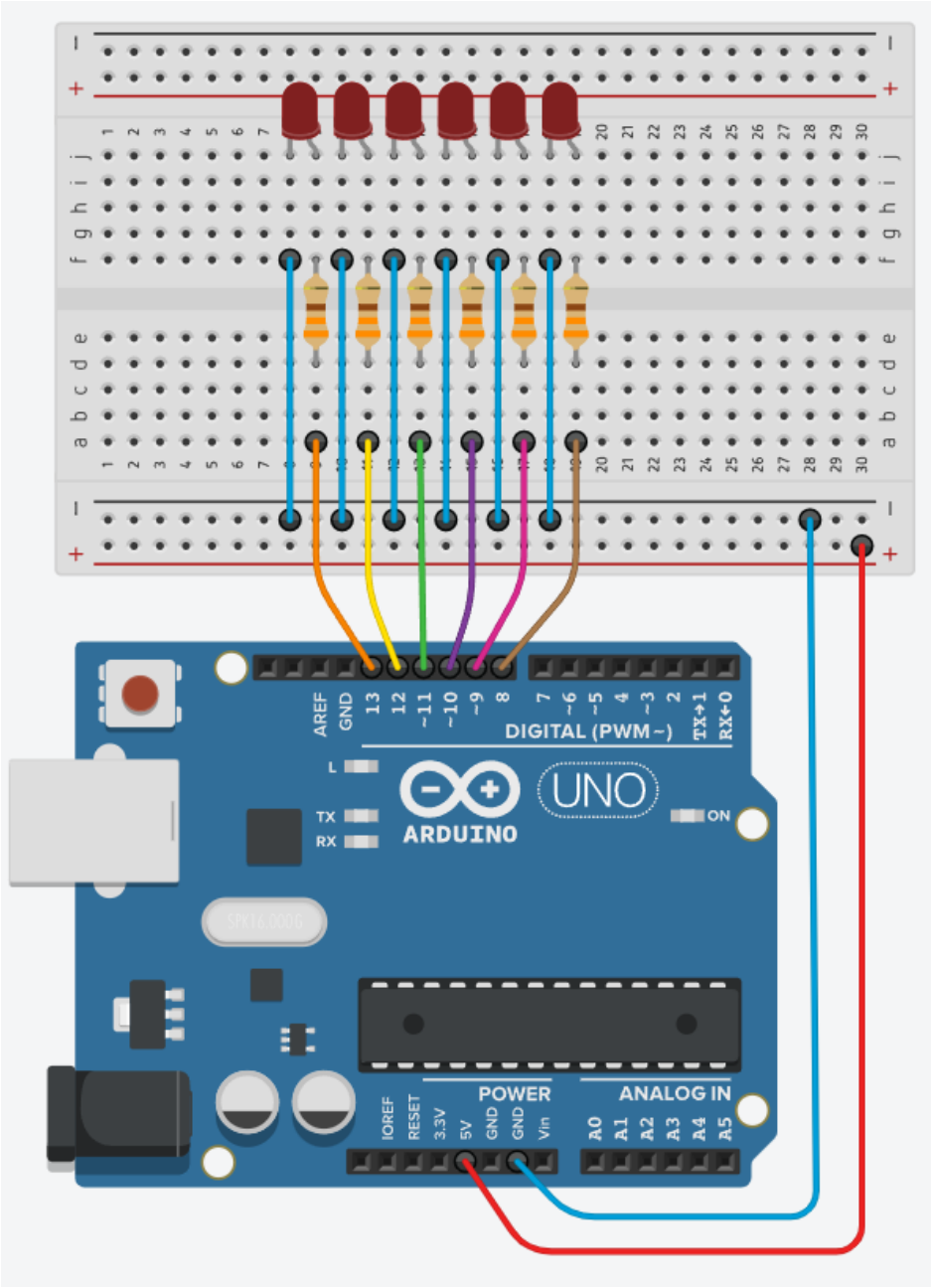
Arduino Kodu	**Kullandığımız pini değişken tanımlayarak kodu tekrar yazalım.**

`void setup()`
`{`
`  pinMode(2, OUTPUT);`
`  pinMode(3, OUTPUT);`
`}`
 	
`void loop()`
`{`
`  digitalWrite(2, HIGH);`
`  digitalWrite(3, LOW);`
`  delay(1000); // 1sn bekleme`
 	
`  digitalWrite(2, LOW);`
`  digitalWrite(3, HIGH);`
`  delay(1000);`
`}`
	`int led1 = 2;`
	`int led2 = 3;`
	
	`void setup()`
	`{`
	`  pinMode(led1, OUTPUT);`
	`  pinMode(led2, OUTPUT);`
	`}`
	
	`void loop()`
	`{`
	`  digitalWrite(led1, HIGH);`
	`  digitalWrite(led2, LOW);`
	`  delay(1000); // 1sn bekleme`
	
	`  digitalWrite(led1, LOW);`
	`  digitalWrite(led2, HIGH);`
	`  delay(1000);`
	`}`

3. Kara Şimşek LED Uygulaması

Arduino'nun 8.pininden başlayarak 6 adet led ile kara şimşek led uygulaması yapalım. Bu uygulamada led'ler sırasıyla soldan sağa ve sağdan sola yanıp sönecektir.

- Led'lerin bekleme süresi olarak **200ms** kullanalım.
- İlk ledi 13.pine, son ledi ise 8.pine bağlayalım
- Led'lerde **330Ω** direnç kullanalım. Dirençleri oluğun üzerinden dikey olarak bağlayalım.
- Kodumuzda değişkenleri kullanalım. Hem ledleri, hem de bekleme süresini değişken olarak tanımlayalım ki sonradan değişiklik yapması kolay olsun.



- Bağlantıları farklı şekillerde yapabilirsiniz.
- İlk ledi 13.pindeki led olarak kabul edip kodumuzu ona göre yazalım.

Arduino Kodu

```
int led1 = 13;
int led2 = 12;
int led3 = 11;
int led4 = 10;
int led5 = 9;
int led6 = 8;
int sure = 200;

void setup()
{
	pinMode(led1, OUTPUT);
	pinMode(led2, OUTPUT);
	pinMode(led3, OUTPUT);
	pinMode(led4, OUTPUT);
	pinMode(led5, OUTPUT);
	pinMode(led6, OUTPUT);
}

void loop()
{
	// SOLDAN SAĞA YAKALIM
	digitalWrite(led1, HIGH);
	delay(sure);
	digitalWrite(led1, LOW);

	digitalWrite(led2, HIGH);
	delay(sure);
	digitalWrite(led2, LOW);

	digitalWrite(led3, HIGH);
	delay(sure);
	digitalWrite(led3, LOW);

	digitalWrite(led4, HIGH);
	delay(sure);
	digitalWrite(led4, LOW);

	digitalWrite(led5, HIGH);
	delay(sure);
	digitalWrite(led5, LOW);

	digitalWrite(led6, HIGH);
	delay(sure);
	digitalWrite(led6, LOW);

	// SAĞDAN SOLA YAKALIM
	digitalWrite(led6, HIGH);
	delay(sure);
	digitalWrite(led6, LOW);

	digitalWrite(led5, HIGH);
	delay(sure);
	digitalWrite(led5, LOW);

	digitalWrite(led4, HIGH);
	delay(sure);
	digitalWrite(led4, LOW);

	digitalWrite(led3, HIGH);
	delay(sure);
	digitalWrite(led3, LOW);

	digitalWrite(led2, HIGH);
	delay(sure);
	digitalWrite(led2, LOW);

	digitalWrite(led1, HIGH);
	delay(sure);
	digitalWrite(led1, LOW);
}
```

Aynı kodları döngü ile tekrar yazalım

```
int sure = 200;

void setup()
{
	// 8'den 13'e kadar çıkış pinleri
	for (int i = 8; i <= 13; i++)
	{
		pinMode(i, OUTPUT);
	}
}

void loop()
{
	// SOLDAN SAĞA YAKALIM. Azalan döngü
	for (int i = 13; i >= 8; i--)
	{
		digitalWrite(i, HIGH);
		delay(sure);
		digitalWrite(i, LOW);
	}


	// SAĞDAN SOLA YAKALIM. Artan döngü
	for (int i = 8; i <= 13; i++)
	{
		digitalWrite(i, HIGH);
		delay(sure);
		digitalWrite(i, LOW);
	}
}
```

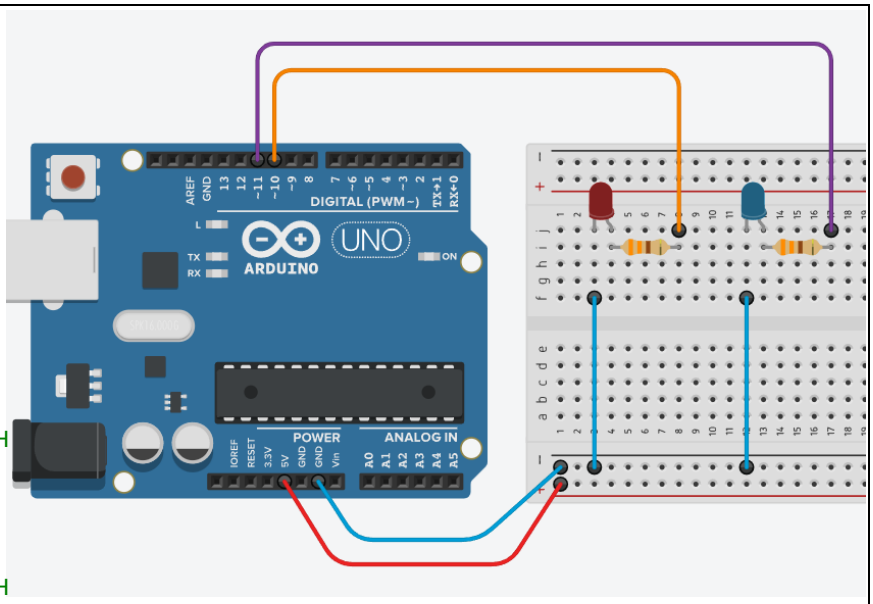
4. Polis Çakarı LED Uygulaması

Bir kırmızı bir de mavi led kullanarak sırasıyla ikişer kez hızlı bir şekilde yanıp sönen polis çakarı yapalım.

- Ledler **10** ve **11** nolu pinlere bağlı olsun.
- Ledlerde **330Ω** direnç kullanalım.
- Çakarın hızı **150ms** olsun.
- Çakarda ilk olarak **kırmızı** led 2 kere hızlı bir şekilde yanıp söndükten sonra aynı şekilde **mavi** led yanıp sönecektir.
- Bu devrede dirençlerimizi yatay bağlayalım.
- Kodumuzda **HIGH** yerine 1, **LOW** yerine 0 yazalım.

Arduino Kodu

```
int kirmizi = 10;
int mavi = 11;

int sure = 150;

void setup()
{
	pinMode(kirmizi, OUTPUT);
	pinMode(mavi, OUTPUT);
}

void loop()
{
	// MAVİ LED 2 kere yanıp sönsün
	digitalWrite(kirmizi, 1); // HIGH
	delay(sure);
	digitalWrite(kirmizi, 0); // LOW
	delay(sure);

	digitalWrite(kirmizi, 1); // HIGH
	delay(sure);
	digitalWrite(kirmizi, 0); // LOW
	delay(sure);


	delay(sure); // Ekstra bekleme ekleyelim


	// KIRMIZI LED 2 kere yanıp sönsün
	digitalWrite(mavi, 1); // HIGH
	delay(sure);
	digitalWrite(mavi, 0); // LOW
	delay(sure);

	digitalWrite(mavi, 1); // HIGH
	delay(sure);
	digitalWrite(mavi, 0); // LOW
	delay(sure);

	delay(sure); // Ekstra bekleme ekleyelim
}
```

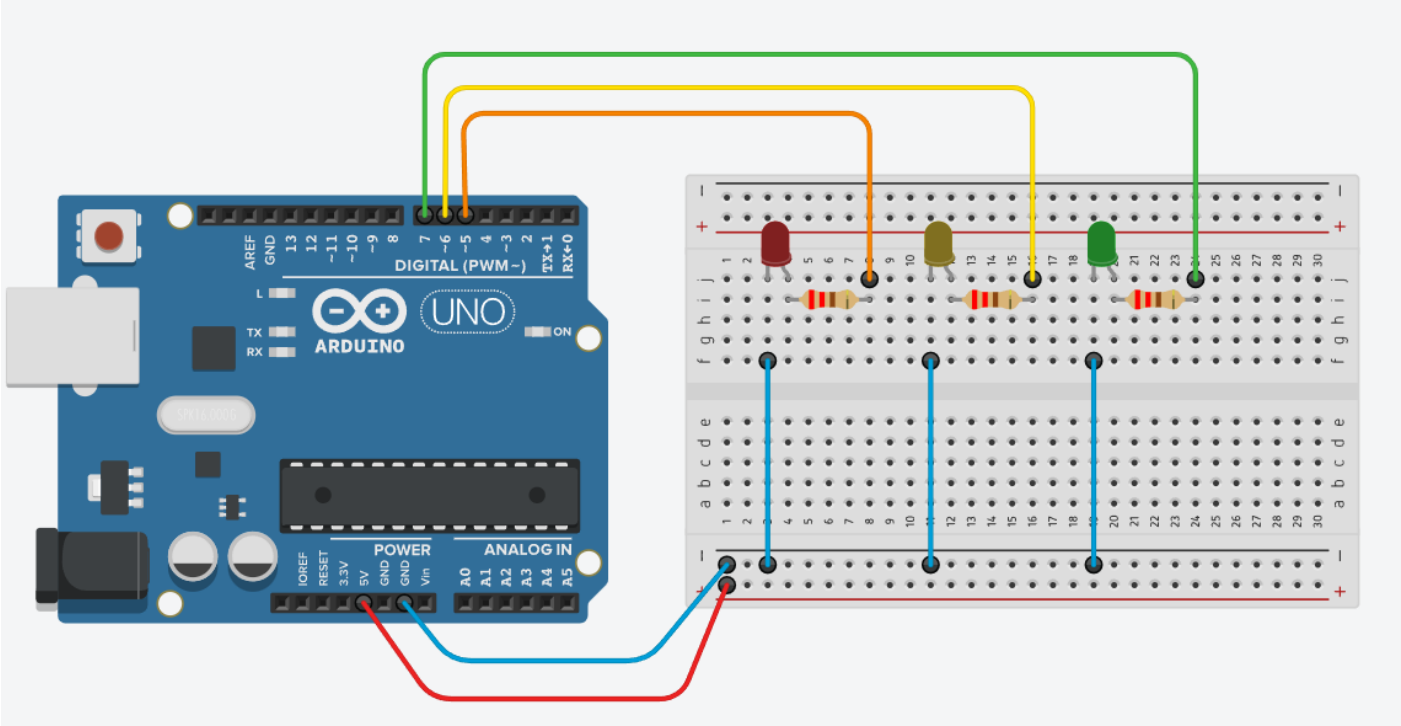


Not: Kodumuzdaki **loop** kısmı döngü kısmıdır. Buradaki kodlar bitince loop'un başına döner ve tekrar kodları çalıştırır. Ledler arasına ekstra bekleme süresi eklerken kodun sonundaki mavi ledden başındaki kırmızı lede geçerken de ekstra bekleme süresi ekledik. Bu bir simülasyon olduğu için gerçek devreye göre daha yavaş çalışıyor. O yüzden çakar etkisini görmek için süreyi biraz uzun tutmak istedik. Gerçek devrede bu kadar uzun süreye ihtiyacımız olmayacaktır.

5. Trafik Lambası LED Uygulaması

Kırmızı, sarı ve yeşil olmak üzere 3 tane ledimiz olsun. Bunları tarafik lambası gibi sırasıyla yakıp söndürelim.

- Ledler sırasıyla **5**, **6** ve **7** nolu pinlere bağlı olsun.
- Kırmızı **6** saniye, sarı **1.5** saniye ve yeşil **4** saniye yanacaktır.
- Sonra tekrar sarı ve kırmızı yanarak eski konumuna geri gelecektir.
- Led için bu kez **220Ω** kullanalım. Dirençleri yatay bağlayalım.
- Değişkenleri pinler için kullanalım.

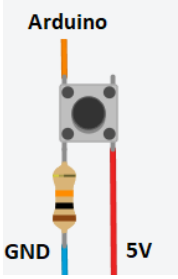


Arduino Kodu

```
int kirmizi = 5;
int sari = 6;
int yesil = 7;

void setup()
{
    pinMode(kirmizi, OUTPUT);
    pinMode(sari, OUTPUT);
    pinMode(yesil, OUTPUT);
}

void loop()
{
    digitalWrite(kirmizi, HIGH);
    delay(6000);
    digitalWrite(kirmizi, LOW);

    digitalWrite(sari, HIGH);
    delay(1500);
    digitalWrite(sari, LOW);

    digitalWrite(yesil, HIGH);
    delay(4000);
    digitalWrite(yesil, LOW);

    digitalWrite(sari, HIGH);
    delay(1500);
    digitalWrite(sari, LOW);
}
```

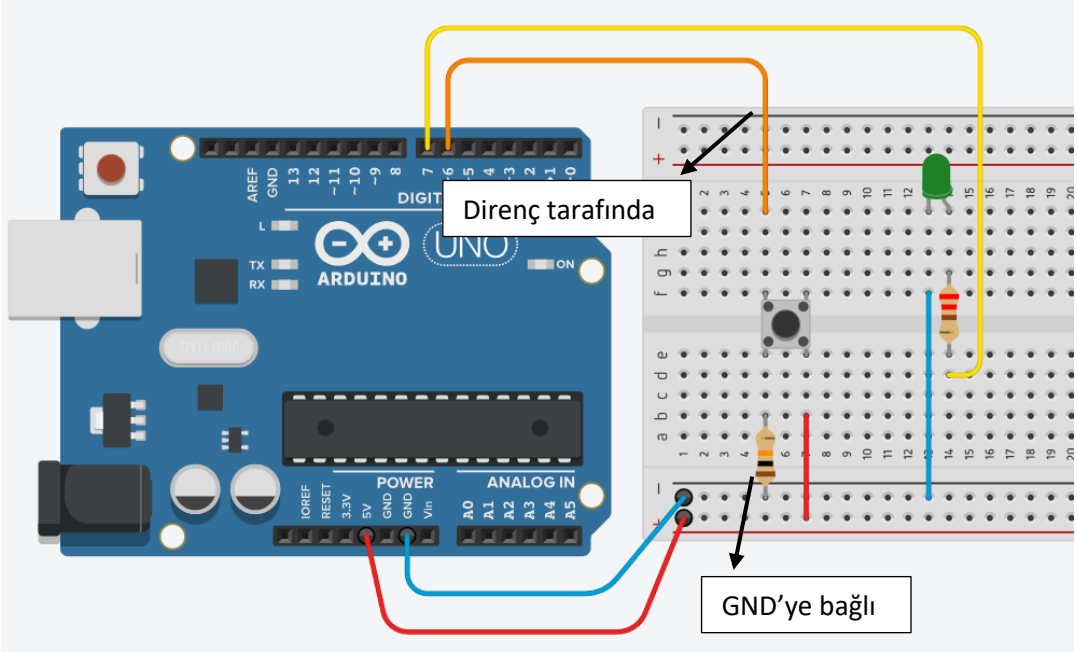
Not: Loop sonunda tekrar başa döneceği için sondaki sarıdan sonra baştaki kırmızıya dönecektir.

6. PullDown Buton ile LED Yakma Uygulaması (PullDown Buton)



Basma düğmesi olarak bilinen **push buton** ile bir ledi yakalım.

- **PullDown** dirençle kullanılan buton **6** nolu pine, led ise **7** nolu pine bağlı olsun.
- Buton basılı iken led yansın, bırakılınca led sönsün.
- Led için **220Ω** direnç kullanalım.
- Genelde **PullDown** direnç **10KΩ** olarak tercih edilir.
  Bu direnç GND tarafına bağlanır.
- Butonlar **giriş** (**INPUT**) elemanlarıdır. Dolayısıyla giriş pinini **digitalRead(pin)** ile okumalısınız. Okunan değer **HIGH** ya da **LOW** olabilir.
- **PullDown** direnç kullanan butona basılınca, bağlı olduğu pin **HIGH** olur. Bırakılınca **LOW** olur. Dolayısıyla butona basılıp basılmadığını bu şekilde anlayabilirsiniz.

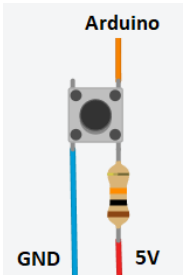


- Butonun bir ayağı doğrudan 5V'a bağlı
- Butonun ikinci ayağı 10K'lık bir dirençle GND'ye ve 6 nolu pine bağlı
- **PullDown** direnç **GND** hattında olur. Butonun normal durumu **LOW**, basılınca **HIGH** olur.

Arduino Kodu

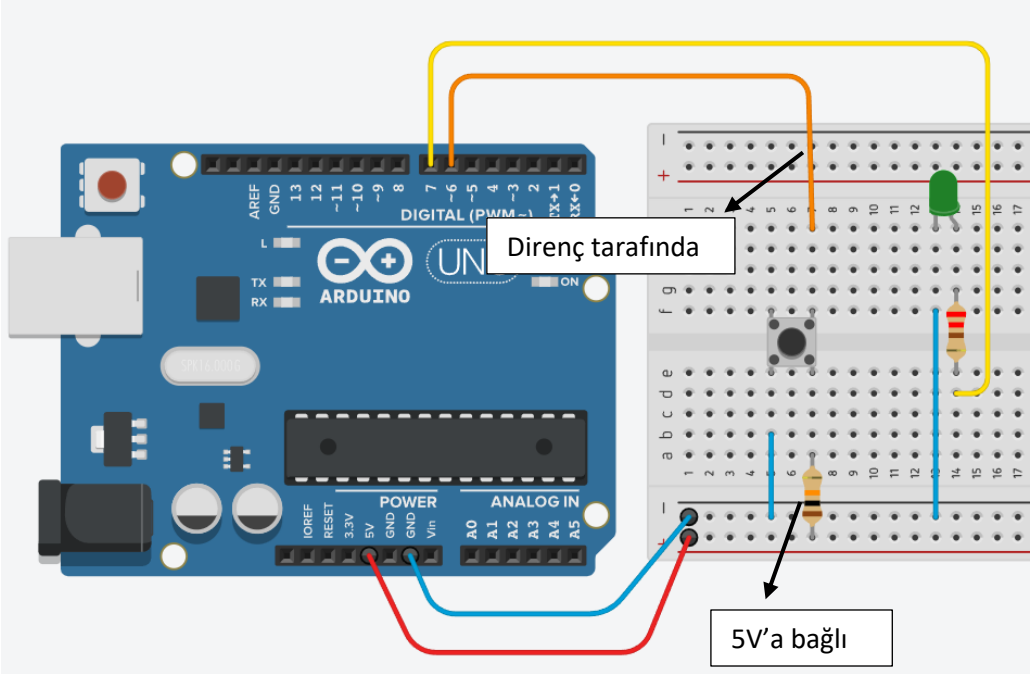
```
int buton = 6;
int led = 7;

void setup()
{
    pinMode(buton, INPUT);
    pinMode(led, OUTPUT);
}

void loop()
{
    if (digitalRead(buton) == HIGH)      // Buton basılı ise
    {
        digitalWrite(led, HIGH);
    }
    else                            // Buton basılı değilse
    {
        digitalWrite(led, LOW);
    }
}
```

7. PullUp Buton ile LED Yakma Uygulaması (PullUp Buton)



Basma düğmesi olarak bilinen push buton ile bir ledi yakalım.

- **PullUp** dirençle kullanılan buton **6** nolu pine, led ise **7** nolu pine bağlı olsun.
- Buton basılı iken led yansın, bırakılınca led sönsün.
- Led için **220Ω** direnç kullanalım. 
- Genelde **PullUp** direnç **10KΩ** olarak tercih edilir.
  Bu direnç 5V tarafına bağlanır. 
- Butonlar **giriş** (**INPUT**) elemanlarıdır. Dolayısıyla giriş pinini **digitalRead(pin)** ile okumalısınız. Okunan değer **HIGH** ya da **LOW** olabilir.
- **PullUp** direnç kullanan butona basılınca, bağlı olduğu pin **LOW** olur. Bırakılınca **HIGH** olur. Dolayısıyla butona basılıp basılmadığını bu şekilde anlayabilirsiniz.

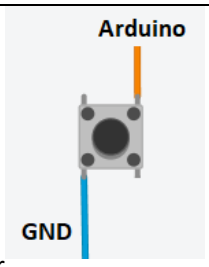


- Butonun bir ayağı **10KΩ**'lık direnç üzerinden **+5V**'a ve **6** nolu pin'e bağlı
- Butonun diğer ayağı ise **GND**'ye bağlı
- **PullUp** direnç **+5V** ayağında olur. Butonun normal durumu **HIGH**, basılınca **LOW** olur.

Arduino Kodu

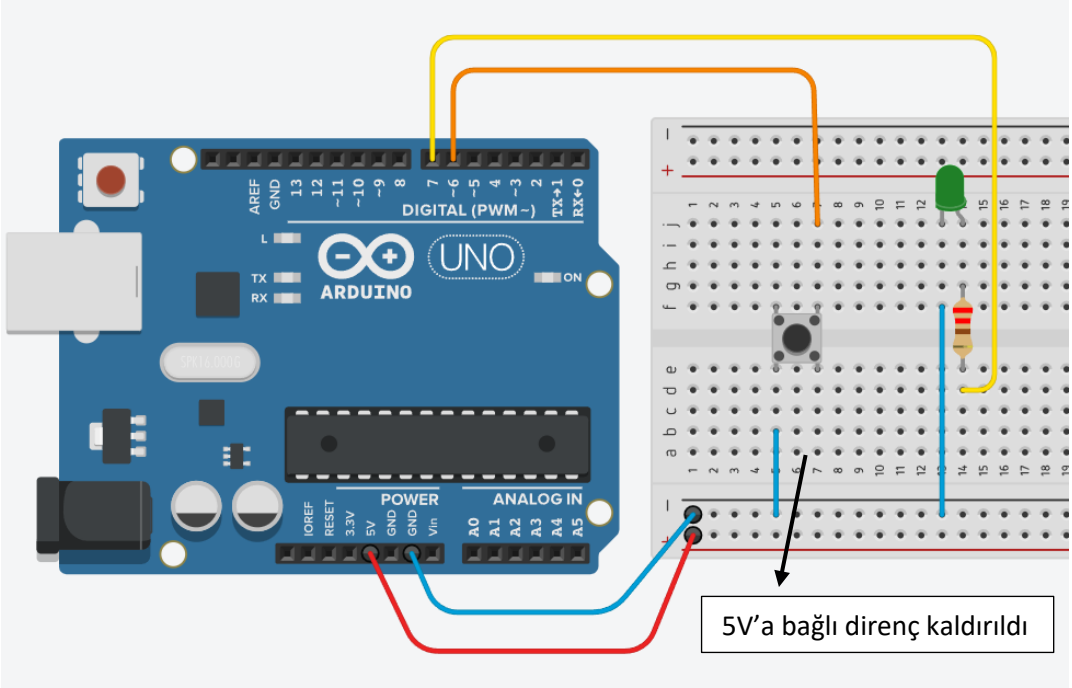
```
int buton = 6;
int led = 7;

void setup()
{
    pinMode(buton, INPUT);
    pinMode(led, OUTPUT);
}

void loop()
{
    if (digitalRead(buton) == LOW)       // Buton basılı ise
    {
        digitalWrite(led, HIGH);
    }
    else                                 // Buton basılı değilse
    {
        digitalWrite(led, LOW);
    }
}
```

8. Dahili PullUp Buton ile LED Yakma Uygulaması (Dahili PullUp Buton)



Arduino kartı üzerindeki dahili **20KΩ PullUp** direncini kullanarak led yakıp söndürelim.

- Buton **6**, led ise **7** nolu pine bağlı olsun.
- Buton basılı iken led yansın, bırakılınca led sönsün.
- Led için **220Ω** direnç kullanalım.
- Arduino üzerinde dahili olarak bulunan **20KΩ** değerindeki **PullUp** direnci aktif edilirse; normalde buton'nun **5V** ayağına bağlanan **PullUp** direncini artık kullanmaya gerek yoktur.
- Butonlar **giriş** (**INPUT**) elemanlarıdır. Dolayısıyla giriş pinini **digitalRead(pin)** ile okumalısınız. Okunan değer **HIGH** ya da **LOW** olabilir.
- **PullUp** direnç kullanan butona basılınca, bağlı olduğu pin **LOW** olur. Bırakılınca **HIGH** olur.
- Dahili **PullUp** direncini aktif etmek için buton **INPUT_PULLUP** olarak tanımlanmalıdır.



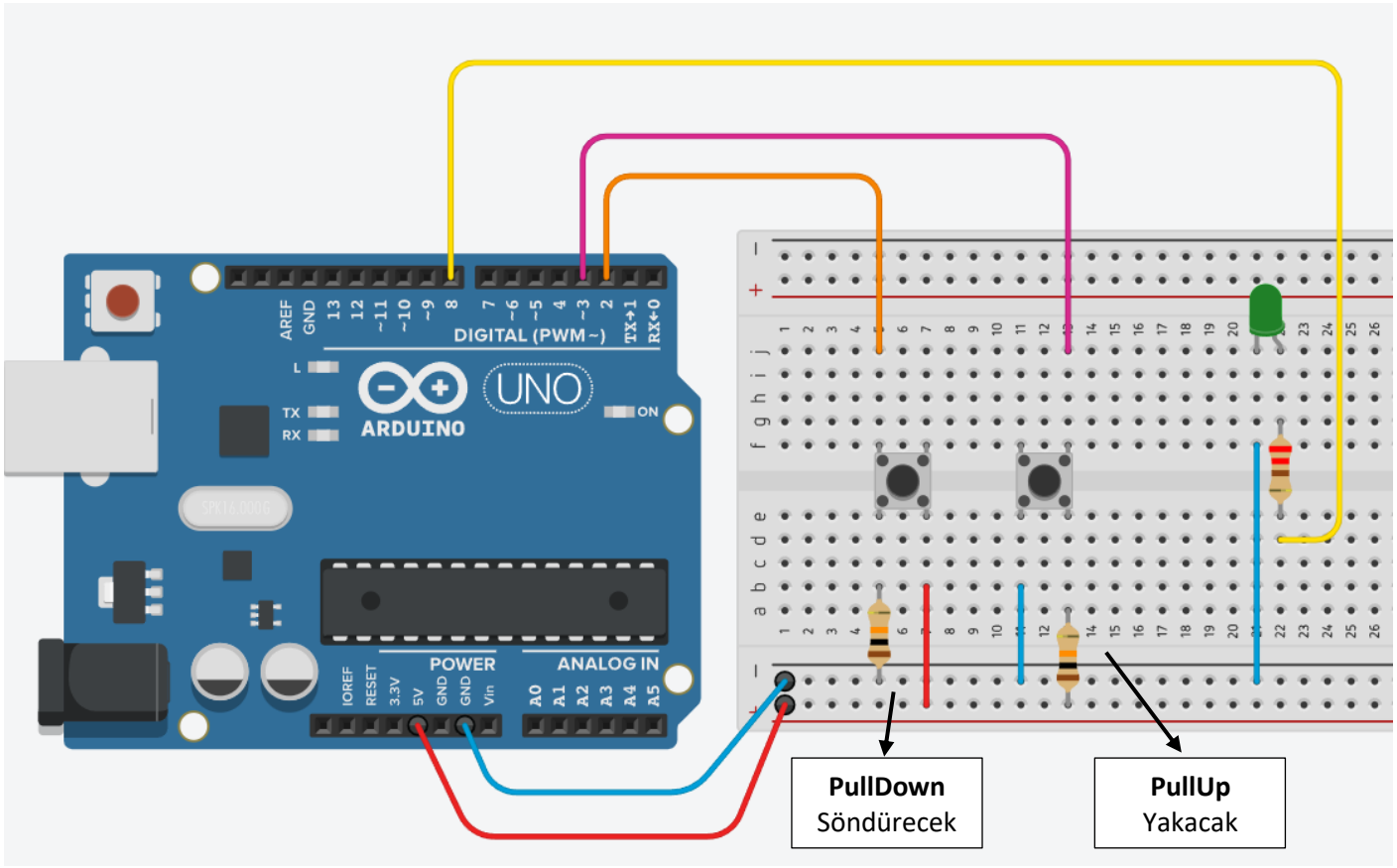
Arduino Kodu

```
int buton = 6;
int led = 7;

void setup()
{
    pinMode(buton, INPUT_PULLUP);  // dahili PullUp direnci aktif ediliyor
    pinMode(led, OUTPUT);
}

void loop()
{
    if (digitalRead(buton) == LOW)       // Buton basılı ise
    {
        digitalWrite(led, HIGH);
    }
    else                                 // Buton basılı değilse
    {
        digitalWrite(led, LOW);
    }
}
```

9. PullDown ve PullUp Buton ile LED Yakma Uygulaması

İki çeşit buton kullanarak led yakıp söndürelim.

- **PullDown** buton **2**, **PullUp** buton **3**, LED ise **8** nolu pine bağlı olsun.
- **PullDown** buton ledi söndürsün. **PullUp** buton ise ledi yaksın.



Arduino Kodu

```
int buton1 = 2;
int buton2 = 3;
int led = 8;

void setup()
{
    pinMode(buton1, INPUT);
    pinMode(buton2, INPUT);
    pinMode(led, OUTPUT);
}

void loop()
{
    if (digitalRead(buton1) == HIGH)     // Buton1 pulldown basılınca HIGH olur
    {
        digitalWrite(led, LOW);          // LED'i söndür
    }

    if (digitalRead(buton2) == LOW)      // Buton2 pullup basılınca LOW olur
    {
        digitalWrite(led, HIGH);         // LED'i yak
    }
}
```

10. Dahili PullUp Buton ile Anahtar Uygulaması

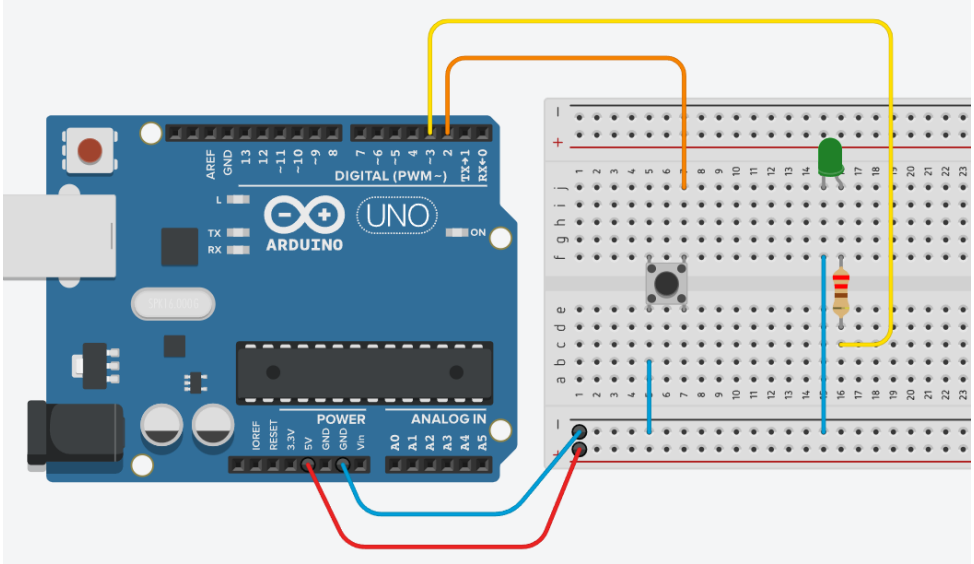
Dahili **PullUp** direncini kullanarak butonu anahtar gibi kullanarak LED yakıp söndürelim.

- **Butonlar** basılıp bırakıldığında eski durumlarına geri dönerler. O yüzden basma düğmesi olarak adlandırılırlar.
- **Anahtarlar** ise basıldığındaki konumunda kalırlar. Tekrar basılırlarsa önceki konumuna geri dönerler.
- Kapılardaki zil düğmeleri basma düğmelerdir. Lambaları yakıp söndürdüğümüz elemanlar ise birer anahtardır.

Push Buton (Basma Düğmesi) **Switch (Anahtar)** **Slide Switch (Sürgülü Anahtar)**



- Bu örneğimizde basma düğmesini **switch** yani **anahtar** gibi kullanalım.
- Yani butona basınca led yansın, butona tekrar basınca led sönsün. Bunu için led'in durumunu takip edeceğimiz mantıksal bir değişken kullanacağız.
- Buton dahili **PullUp** direncini kullansın ve 2 nolu pine, led ise 3 nolu pine bağlansın.



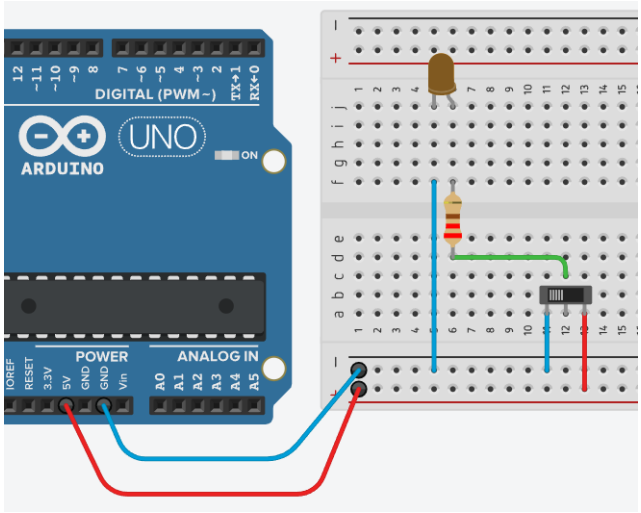
Arduino Kodu

```
int buton = 2;
int led = 3;
bool durum = false;  // led'in durumunu takip edeceğimiz mantıksal değişken

void setup()
{
    pinMode(buton, INPUT_PULLUP);
    pinMode(led, OUTPUT);
}

void loop()
{
    if (digitalRead(buton) == LOW)          // PullUp Düğmeye basıldı ise
    {
        if (durum == false)                 // led sönükse
        {
            digitalWrite(led, HIGH);        // ledi yak
            durum = true;                   // durumu true olarak ayarla
        }
        else                                // DEĞİLSE
        {
            digitalWrite(led, LOW);         // ledi söndür
            durum = false;                  // durumu false olarak ayarla
        }

        while (digitalRead(buton) == LOW); // düğme basılı olduğu sürece döngü ile beklet
    }
}
```

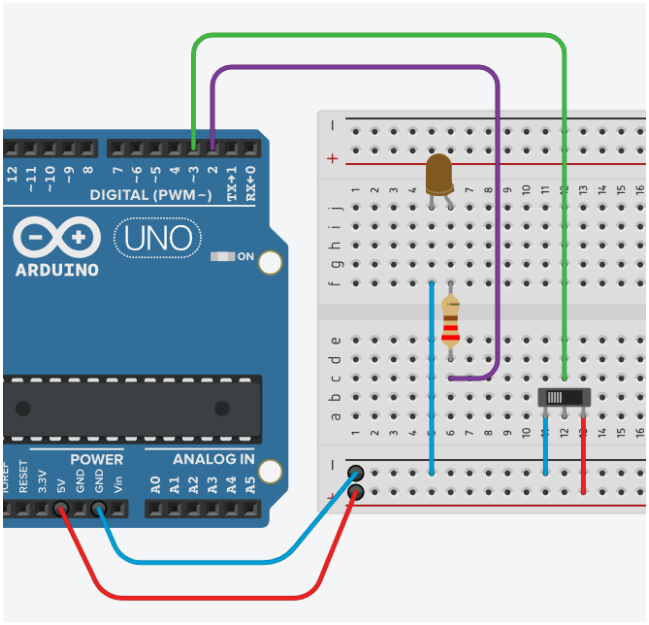
11. Sürgülü Anahtar ile LED Uygulaması (Slide Switch)

Sürgülü anahtarın sürgüsü sola ya da sağa çekildiğinde LED'i yakıp söndürelim.

- Uygulamayı gerçekleştirmenin 2 yolu vardır.
  - Birincisi ledi doğrudan sürgülü anahtara bağlayarak herhangi bir kod yazmadan çalıştırmak.
  - İkincisi ise hem anahtarı hem de ledi dijital pinlere bağlayarak program ile kontrolünü gerçekleştirmek.
- Her iki yöntemi de yapalım. Sürgülü anahtarın bir ayağı **GND**, bir ayağı **5V**, orta ayağı ise çıkış olarak kullanılır.
  - Birinci yöntemde orta ayaktaki çıkışı doğrudan led'e bağlayacağız.
  - İkinci yöntem de ise orta ayaktaki çıkışı dijital pine bağlayıp programlayacağız. Anahtar 3, led 2 nolu pine bağlansın.



Birinci Yöntem



İkinci Yöntem (Programlanabilir)

Arduino Kodu

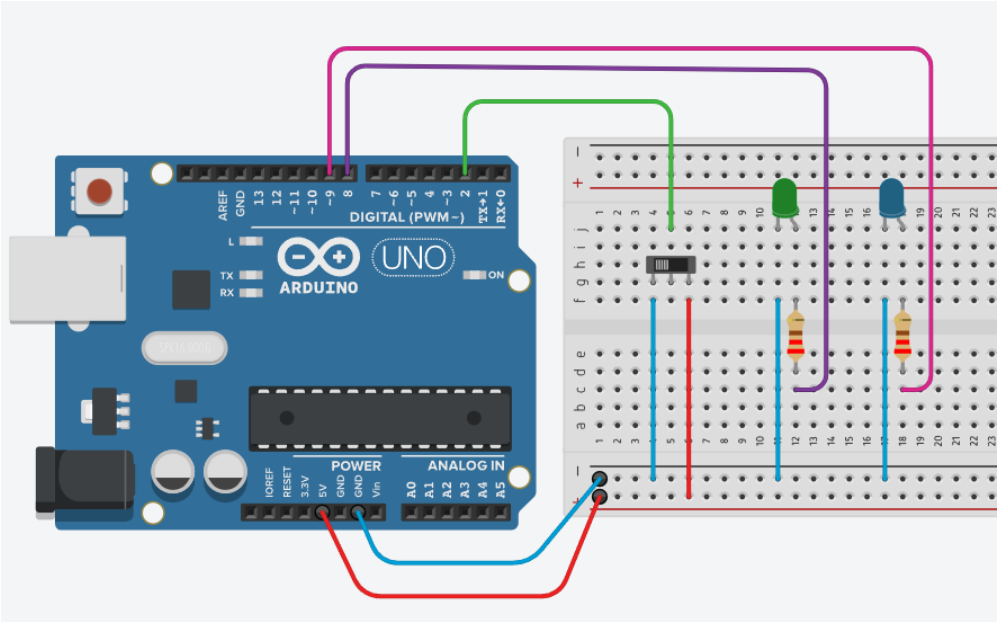
```
int led = 2;
int anahtar = 3;

void setup()
{
    pinMode(led, OUTPUT);
    pinMode(anahtar, INPUT);
}

void loop()
{
    if(digitalRead(anahtar) == LOW) // sürgü solda ise
    {
        digitalWrite(led, LOW);      // ledi söndür
    }
    else                             // değilse
    {
        digitalWrite(led, HIGH);     // ledi yak
    }
}
```

Sürgülü anahtarın **GND** bağlantısı solda olduğundan sürgü solda ise 3 nolu pin **LOW**, sürgü sağda ise 3 nolu pin **HIGH** olur.

12. Sürgülü Anahtar ile İki LED Uygulaması (Slide Switch)

Sürgülü anahtar ile iki tane ledi yakıp söndürelim.

- Sürgülü anahtarın sürgüs solda iken soldaki led, sağda iken sağdaki ledi yakalım. Diğer ledi ise bu durumda söndürelim.
- Led'ler 8 ve 9 nolu pine, anahtar ise 2 nolu pine bağlı olsun.



Arduino Kodu

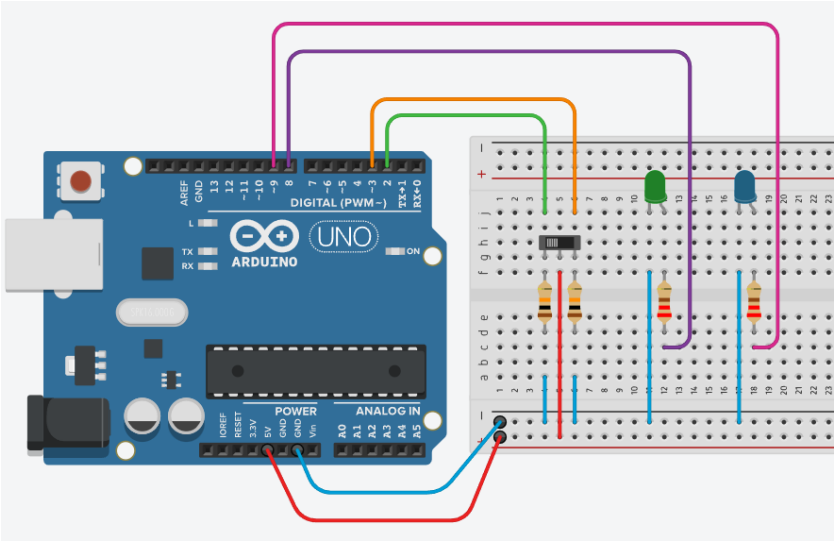
```
int led1 = 8;
int led2 = 9;
int anahtar = 2;

void setup()
{
    pinMode(led1, OUTPUT);
    pinMode(led2, OUTPUT);
    pinMode(anahtar, INPUT);
}

void loop()
{
    if (digitalRead(anahtar) == LOW)     // sürgü solda ise
    {
        digitalWrite(led1, HIGH);        // soldaki ledi yak
        digitalWrite(led2, LOW);         // sağdaki ledi söndür
    }
    else                                 // değilse
    {
        digitalWrite(led1, LOW);         // soldaki ledi söndür
        digitalWrite(led2, HIGH);        // sağdaki ledi yak
    }
}
```

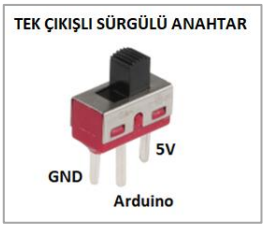
- Sürgülü anahtarın solunda **GND** olduğundan sürgü sola çekildiğinde **LOW** olur,
- Sağında ise **5V** olduğunda sürgü sağa çekildiğinde **HIGH** olur.

13. Sürgülü Anahtar ile İki Çıkış Almak

Sürgülü anahtardan iki çıkış alarak iki ledi yakalım. Yani sürgülü anahtarı iki anahtar gibi kullanalım.

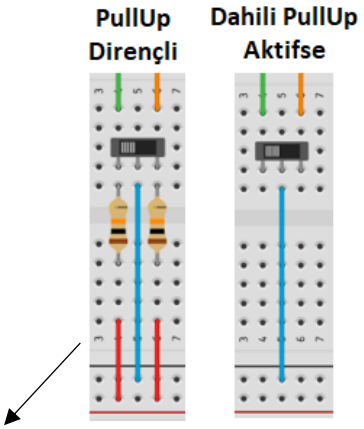
- Normalde sürgülü anahtarda bir ayak **GND**, bir ayak **5V** ve ortadaki ayak da **çıkış** olarak kullanılır.
- Bu uygulamamızda ortadaki ayağı **giriş**, diğer iki ayağı **çıkış** olarak kullanalım. Çıkışlar Arduino'ya gider.
- Girişi **5V**'a, çıkışları ise **Arduino**'ya bağlayalım.
- Anahtar **2** ve **3** nolu pinlere, ledler ise **8** ve **9** nolu pinlere bağlı olsun.
- Eğer sürgülü anahtardan iki çıkış alacaksanız anahtarın karasızlığa düşmemesi için **Arduino**'ya giden çıkışlar **10KΩ**'luk dirençlerle topraklanmalıdır. Bu dirençler bir nevi **PullDown** dirençtir. Dolayısıyla sürgünün çekildiği taraf **HIGH** olarak kabul edilir.





Arduino Kodu

```
int led1 = 8;
int led2 = 9;
int anahtar1 = 2;
int anahtar2 = 3;

void setup()
{
    pinMode(led1, OUTPUT);
    pinMode(led2, OUTPUT);
    pinMode(anahtar1, INPUT);
    pinMode(anahtar2, INPUT);
}
void loop()
{
    if (digitalRead(anahtar1) == HIGH) // sürgü solda ise
    {
        digitalWrite(led1, HIGH); // soldaki ledi yak
        digitalWrite(led2, LOW); // sağdaki ledi söndür
    }
    else                                    // değilse
    {
        digitalWrite(led1, LOW); // soldaki ledi söndür
        digitalWrite(led2, HIGH); // sağdaki ledi yak
    }
}
```

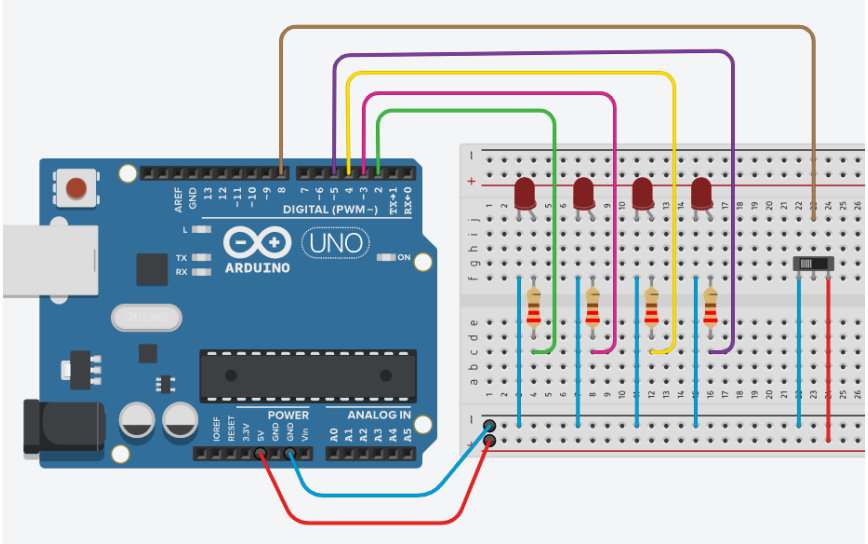


- Aynı devreyi **PullUp** dirençlerle de kurabilirsiniz.
- O durumda orta ayağı **GND**, çıkış ayaklarını ise **10KΩ**'luk dirençlerle **5V**'a bağlayarak sürgünün çekildiği tarafı **LOW** kabul edip devrenizi programlayabilirsiniz.
- Hatta dahili **20KΩ**'luk **PullUp** direncini aktif ederseniz. Anahtardaki **10KΩ**'luk **PullUp** dirençlerini kaldırıp devrenizi aynı şekilde programlayabilirsiniz. Bu durumda anahtarları **INPUT_PULLUP** olarak tanımlamayı unutmayınız.

14. Sürgülü Anahtar ile Kayar LED Uygulaması

Sürgülü anahtarın konumuna göre 4 adet ledi soldan sağa ya da sağdan sola sırasıyla yakalım.

- Led'ler sırasıyla **2**, **3**, **4** ve **5** nolu pinlere bağlı olsun. Anahtar ise **8** nolu pine bağlı olsun.
- Led'lerde **220Ω** direnç kullanalım.
- Sürgü solda iken ledler sola doğru **0.5sn** aralıklarla sırasıyla yansın. Sürgü sağda ise aynı şekilde sağa doğru sırasıyla yansın.



Arduino Kodu

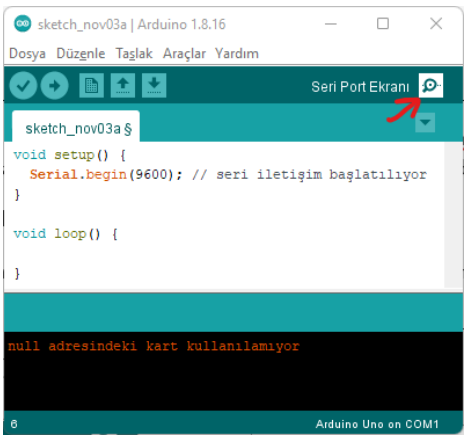
```
int anahtar = 8;

void setup()
{
    // 2'den 5'e kadar olan ledleri çıkış olarak ayarlayalım
    for (int i = 2; i <= 5; i++)
    {
        pinMode(i, OUTPUT);
    }

    pinMode(anahtar, INPUT);
}

void loop()
{
    if (digitalRead(anahtar) == HIGH) // sürgü sağda ise
    {
        // LED'ler sağa doğru sırasıyla yanacak

        for (int i = 2; i <= 5; i++)
        {
            digitalWrite(i, HIGH);
            delay(500);
            digitalWrite(i, LOW);
        }
    }
    else            // anahtar HIGH değilse yani sürgü solda ise
    {
        // LED'ler sola doğru sırasıyla yanacak
        for (int i = 5; i >= 2; i--)
        {
            digitalWrite(i, HIGH);
            delay(500);
            digitalWrite(i, LOW);
        }
    }
}
```

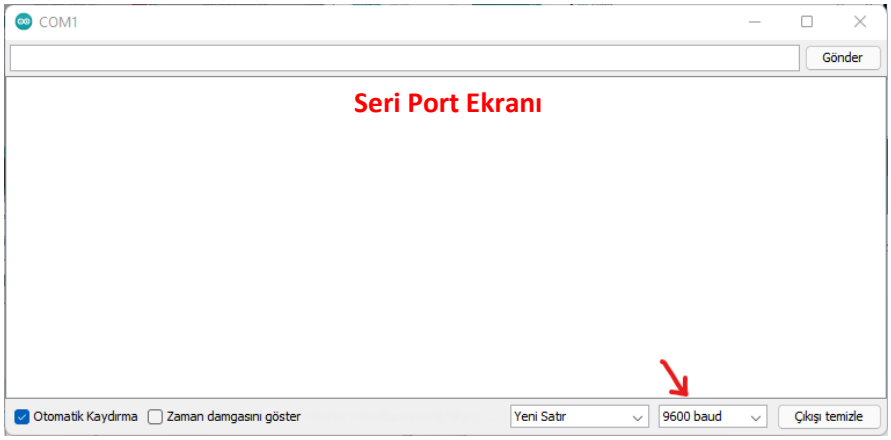
Seri Port Ekranı (Serial Monitor) Kullanımı

Arduino başka sistemler seri port üzerinde iletişim kurabilmektedir. Bunu anlamı; **Arduino** başka sistemlere veri gönderebilir, o sistemlerden veri alabilir demektir. Gönderilen ya da alınan verileri takip edebilmek için **seri port ekranı** denilen ekran kullanılır.

Seri monitör ekranı ile seri iletişim kanalına veri yazabilir (gönderebilir) ya da seri iletişim kanalındaki veri okunabilir.

- Arduino'da seri iletişim kanalını başlatmak için **setup()** içinde **Serial.begin(*İletişim Hızı*)** komutu kullanılır.
- İletişim hızı genelde **9600** bound olarak ayarlanır.
- Bu hız istenilirse değiştirilebilir. Bu durumda seri port ekranında değiştirdiğiniz hızı seçmelisiniz.





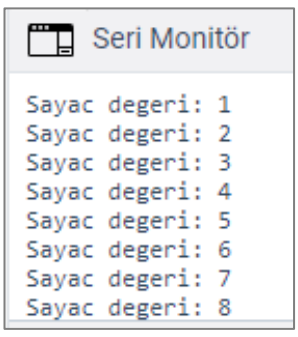
Seri monitöre veri yazdırmak için aşağıdaki komutlar kullanılır.

- **Serial.print(veri);** → Veriyi ekrana yazar. Alt satıra geçmez.
- **Serial.println(veri);** → Veriyi ekrana yazar. Alt satıra geçer. (Sondaki **ln** ifadesi **line** kelimesinden gelir. **Line** ise satır demektir)

Örnek: Bir sayaç tanımlayıp döngüde arttıralım. Nasıl artığını seri port ekranına yazdırarak takip edelim.

Arduino Kodu

```
int sayac = 0;

void setup() {
	Serial.begin(9600);			// seri iletişim başlatılıyor
}

void loop() {
	sayac++;
	Serial.print("Sayac degeri: ");	// alt satıra geçmez
	Serial.println(sayac);			// sayacı yazıp alt satıra
geçer
}
```

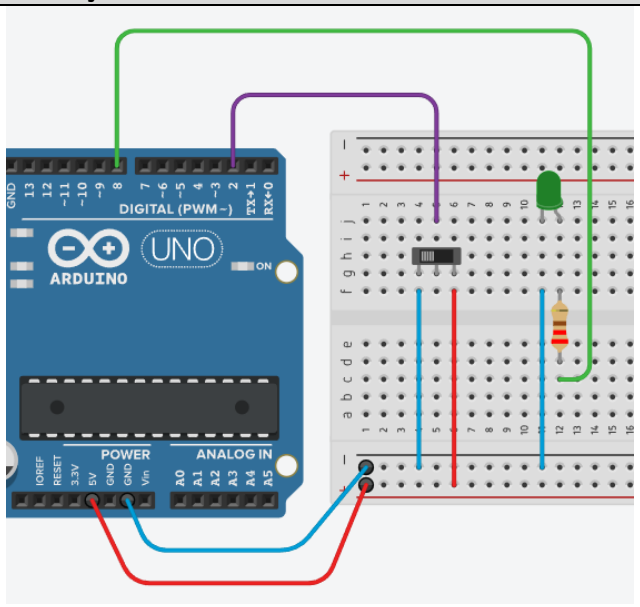


Örnek: Sürgülü anahtar ile yakılan bir led devresinde anahtarın ve ledin durumlarını seri port ekranına yazdıralım.

Arduino Kodu

```
int led = 8;
int anahtar = 2;

void setup()
{
    pinMode(led, OUTPUT);
    pinMode(anahtar, INPUT);

    Serial.begin(9600);
    Serial.println("DEVRE BASLADI");
    Serial.println("--------------------");
}

void loop()
{
    if (digitalRead(anahtar) == LOW)
    {
        digitalWrite(led, LOW);

        Serial.println("Surgu solda");
        Serial.println("LED kapali");
        Serial.println();
    }
    else
    {
        digitalWrite(led, HIGH);

        Serial.println("Surgu sagda");
        Serial.println("LED acik");
        Serial.println();
    }
}
```

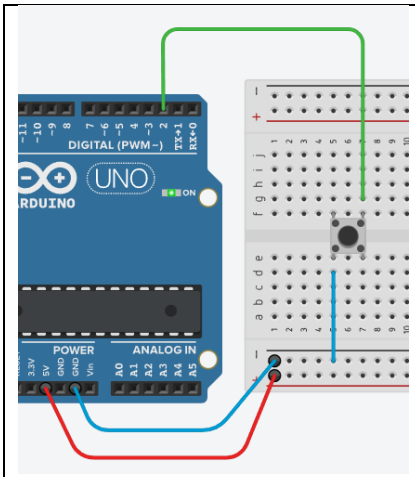
Devre Şeması



Seri Monitör

```
DEVRE BASLADI
--------------------
Surgu solda
LED kapali

Surgu sagda
LED acik
```

Örnek: Bir tane **sayaç** tanımlayalım. Dahili **pullup** direncini kullanan **push butona** basıldığında tanımlı olan sayacı arttıralım ve seri port ekranına yazdıralım. Burada butona 1 kere bastığımızda sayacın kaç kere arttığına dikkat ediniz.




```
int buton = 2;
int sayac = 0;

void setup()
{
    pinMode(buton, INPUT_PULLUP);

    Serial.begin(9600);
    Serial.println("Sayac: 0");
}

void loop()
{
    if (digitalRead(buton) == LOW)
    {
        sayac++;
        Serial.print("Sayac: ");
        Serial.println(sayac);
        // while (digitalRead(buton) == LOW);
    }
}
```

Devre ilk çalıştığında

Seri Monitör

```
Sayac: 0
```

Butona 1 kere basıldığında sayacı 11 kere arttırdı

Seri Monitör

```
Sayac: 0
Sayac: 1
Sayac: 2
Sayac: 3
Sayac: 4
Sayac: 5
Sayac: 6
Sayac: 7
Sayac: 8
Sayac: 9
Sayac: 10
Sayac: 11
```

***Arduino Uno**'nun çalışma frekansı **16MHz**'dir. Yani saniyede 16 milyon işlem yapabiliyor demektir. Bizim butona basma süremiz yaklaşık **50ms** olun. Yani butonu **50ms** basılı tutuyoruz demektir. Bu **50ms** boyunca Arduino **if** kodunu defalarca çalıştırmakta ve butonun basılı olduğunu gördüğü için defalarca sayacı arttırıp ekrana yazmaktadır. Sayacı, bir kere basınca bir kere arttırsın istiyorsanız butondan elimizi çekene kadar Arduino'yu oyalamamız lazım. Bunun için buton basılı olduğu sürece boş dönen bir döngü yazılır.* `while (digitalRead(buton) == LOW);`