

PROGRAMLAMAYA GİRİŞ

Özenç GÜR

Serdar GEDİK

Mahmuthan TUNAKAYA

EET ve EOT Alanı Öğretmenleri



ÖZEL İZMİR ATATÜRK ORGANİZE SANAYİ BÖLGESİ
NEDİM UYSAL MESLEKİ VE TEKNİK ANADOLU LİSESİ
Elektrik-Elektronik ve Endüstriyel Otomasyon Teknolojileri

Program Yapısı

1. void setup()

Setup() fonksiyonu program yüklenilip enerji verildikten veya reset atıldıktan sonra 1 defa çalışır.

Bu fonksiyon içine yazdıklarımız **pin modları** ve **seri iletişimini başlatma** gibi işlemler yapılır.

```
int buton = 3;  
//butonu 3. pine tanımladık  
  
void setup () {  
  
    Serial.begin (9600);  
    //Seri haberleşme  
    pinMode(buton, INPUT);  
    //Pin modu tanımladık  
}  
void loop () {  
  
}
```

Program Yapısı

2. void loop()

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

Dosya Düzenle Taslak Araçlar Yardım



```
void setup()
{
    pinMode (8, OUTPUT);
}

void loop()
{
    digitalWrite (8, HIGH);
    delay (1000);
    digitalWrite (8, LOW);
    delay (1000);
}
```

Program Yapısı

3. #define

#define ön işlemci komutu olup, bir isim yerine başka bir ismi değişimini sağlar.

```
#define ledpin 13
/* programda ledpin gördüğü yere 13
rakamını yerleştirecektir. */

void setup () {
    Serial.begin (9600);
    //Seri haberleşme
}

void loop () {
}
```

Program Yapısı

4. #include

#include programımız dışındaki kütüphanelere erişmek için kullanılır. Programımızda SPI kütüphanemizi kullanmak ve onun içerisindeki komutlara ulaşmak istediğimizde program başına tanımlamamız iki şekilde olabilir:

("..") yada (<..>)

#include "SPI.h" //Tanımlama 1

#include <SPI.h> //Tanımlama 2

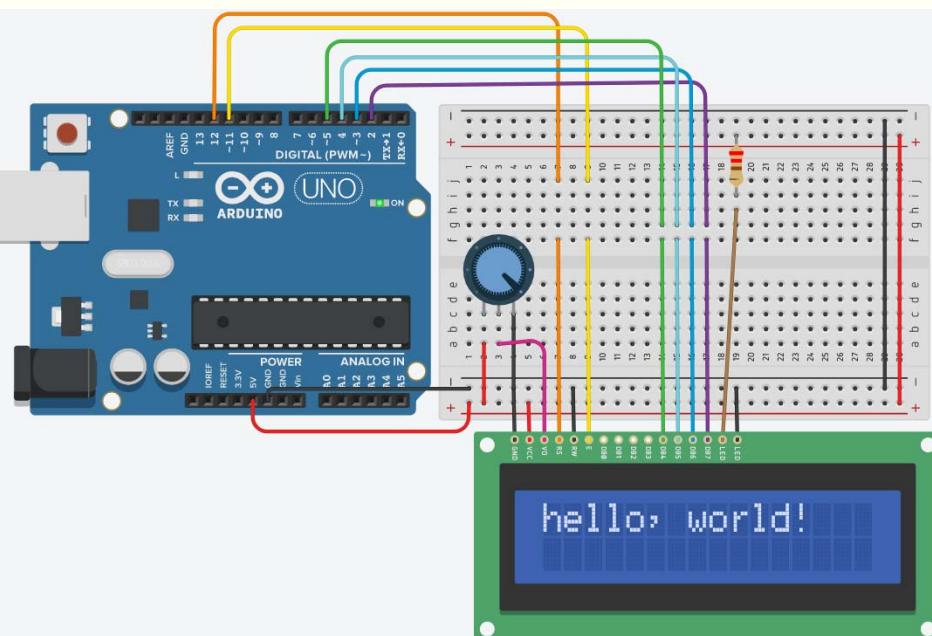
```
#include <LiquidCrystal.h>

void setup () {
    Serial.begin (9600);
    //Seri haberleşme
}

void loop () {
}
```

#include< >

Şekildeki devrede LCD ekranı çalıştırabilmemiz için ekranın kütüphanesini çağrıyoruz



```
#include <LiquidCrystal.h>
// Arduino nun LCD ekrana bağlanan pinleri yazılır
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
    // LCD ekranın sütun ve satır sayısı belirtilir
    lcd.begin(16, 2);
    // LCD ekrana yazı yazdırılır
    lcd.print("hello, world!");
}

void loop() {
```

Program Yapısı

Fonksiyonlar

Fonksiyonlar işlevsel bir program için olmazsa olmazlardandır, fonksiyonların diğer bir adı alt programlardır, her fonksiyon bir takım işlemleri yerine getirmek için tasarlanır, gerektiğinde çağrıılır.

```
void setup () {  
    pinMode (6, OUTPUT);  
    //Pin modu tanımladık  
}  
void loop () {  
  
}  
int deger()  
{  
    digitalWrite(6,1);  
    delay(1000);  
}
```

{ } Süslü Parantez

Bir fonksiyon tanımlandığında bu fonksiyon işleyeceği zaman iki süslü parantez arasındaki kodlar işlenir.

Hangi komutu kullanırsak kullanalım, o komutun işleyeceği kodlar iki süslü parantezi arasındaki kodlardır.

```
void setup ()  
{  
Serial.begin (9600);  
}  
void loop () {  
  
kodlar;  
}
```

; Noktalı Virgül

Program yazarken eğer sadece bir komut yazılmış ise mutlaka bu komut noktalı virgül ile kapanmalıdır.

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
    lcd.begin(16, 2);
    lcd.print("hello, world!");
}

void loop()
{}
```

/* */ Toplu Açıklama Satırı

Açılışı ve kapanışı olan bu açıklama satırı birden fazla satırda oluşturulabilir. Açılmış ve kapanış arasındaki tüm bilgiler açıklama satırıdır. Açıklama satırları derlenmez ve programın çalışmasına bir etkisi yoktur.

```
/*Aşağıdaki program 5 nolu girişin durumuna  
göre 13 nolu çıkışını lojik-1- veya lojik-0-  
yapmaktadır.*/
int giris=5; cikis=13;
void setup()
{
pinMode(giris, INPUT);
pinMode(cikis, OUTPUT);
}
void loop()
{
if(digitalRead(giris)==1)
{
digitalWrite(cikis, HIGH);
}
}
```

// Tek Açıklama Satırı

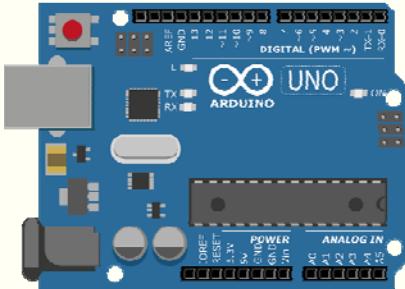
Yazıldığı yerden itibaren satır sonuna
kadar olan kısım açıklama satırı olarak
kabul edilir.

```
#define ledpin 13
// programda ledpin gördüğü yere 13
// rakamını yerlestirecektir.

void setup () {
    Serial.begin (9600);
//Seri haberleşme

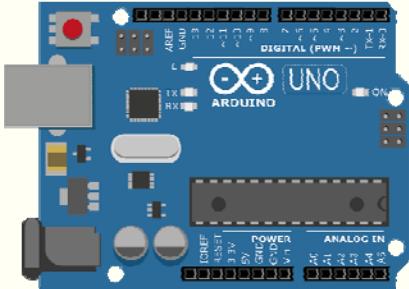
}

void loop () {
}
```



Veri Tipleri

- ✓ Bütün işlemleri **veri tipleri** üzerinde gerçekleştiriyoruz.
- ✓ Verileri tutan belleklere '**değişken**' isimleri vererek kullanıyoruz.
- ✓ **Tam sayı tipler** (`int`, `byte`, `long...`)
- ✓ **Ondalıklı sayı tipler** (`float`, `double`)
- ✓ **Karakter** (`char`, `string`)



Değişkenler

Değişken isimleri,

- ✓ Türkçe karakter içermemelidir.
- ✓ Değişken adı sayı ile başlamamalıdır ancak sayı içerebilir.
- ✓ Büyük – küçük harfe duyarlıdır.
- ✓ Program dilinin anahtar kelimeleri isim olarak kullanılamaz.
- ✓ ‘_’ Alt tire ile başlayabilir alt tire içerebilir.

Değişkenler



Değişkenler arduino hafızasında saklamak istediğimiz verileri tutmak için kullanılırlar.

Örneğin kum saklamak istersek bu kumu dolabın rafına öylece boşaltamayız. Öncelikle onu tutabilecek bir kaba ihtiyaç duyuyoruz bu kabımız

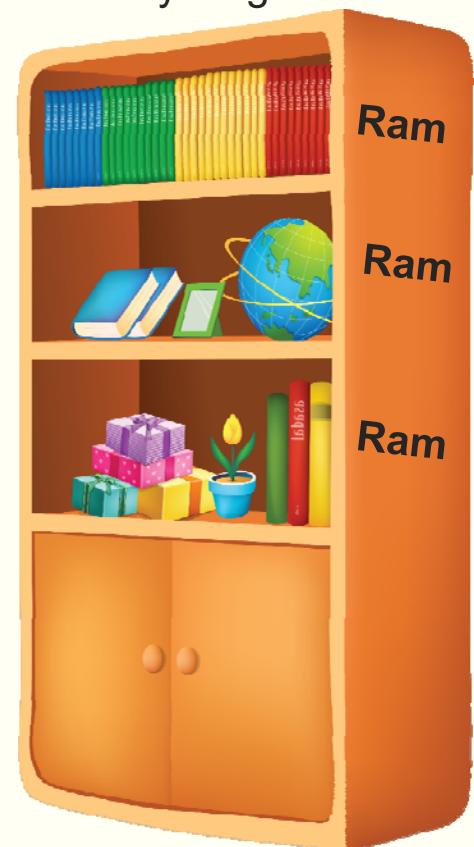


Veri Tipi



Değişken

Örnek olarak arduinoyu bir dolap olarak düşünelim. Dolap içerisinde farklı şeyler saklayacağız. Bu dolabın rafları bizim ram belleğimiz olsun.



Değişkenler

Veri tipleri ise değişkenin içerisinde tutulabilecek malzeme tipimizdir. Örneğin kumu bez torba ile saklayabiliriz ancak aynı bez torbadı su saklayamayız. Çünkü bez torba suyu tutmayacaktır. Bunun için plastik ya da cam şişeye ihtiyacımız var. Tabi her değişken tanımlarken o değişkene de bir isim vermeliyiz, yani dolaba koyduğumuz her kabın üzerine bir etiket yapıştırarak içinde ne olduğunu belirtmek gibi.

Değişkenler

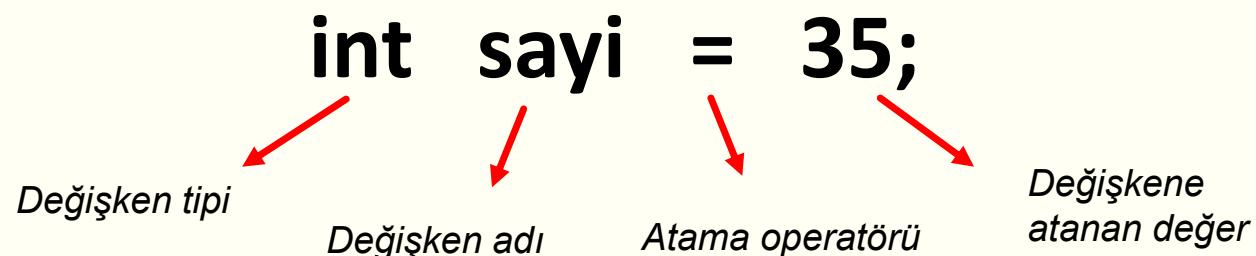
Her maddenin bir boyutu olduğu gibi her değişkenin de veri tutma kapasitesi var.

Değişken tanımlamak 3 adımdan oluşur. Bunlar;

1. *Değişken türünü belirtmek*
2. *Değişken adını belirtmek*
3. *Değişkene değer atamak.*

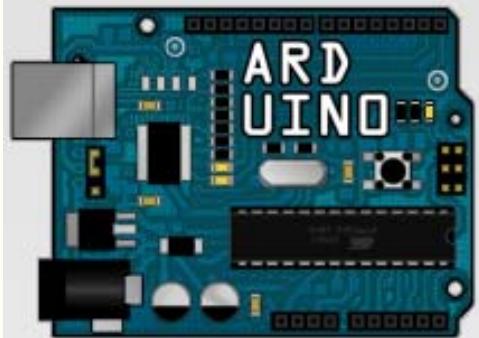
Değişken tanımlama işlemi aşağıdaki gibi yapılır.

int sayı = 35;



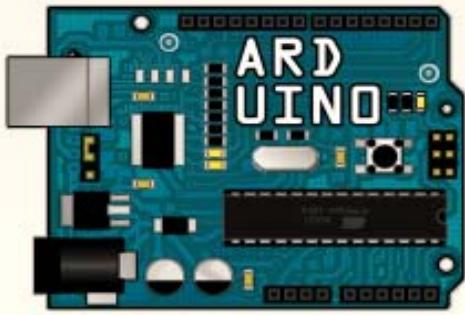
Değişken tipi *Değişken adı* *Atama operatörü* *Değişkene atanana değer*

TİP	BOYUT	MİN	MAX	AÇIKLAMA
void	-	-	-	-
boolean	1 byte	0 (FALSE)	1 (TRUE)	
char	1 byte	-128	+ 128	ASCII – bir karakterin saklanması için kullanılır
unsigned char	1 byte	0	255	ASCII – bir karakterin saklanması için kullanılır
byte	1 byte	0	255	0 – 255 işaretsiz sayılar
int	2 byte	-32.768	+ 32.767	İşaretli tam sayılar
unsigned int	2 byte	0	65535	Pozitif tam sayılar
word	2 byte	0	65535	Pozitif tam sayılar
short	4 byte	-32.768	+32.767	
long	4 byte	-2.147.483.648	+2.147.483.647	İşaretli uzun tam sayılar
unsigned long	4 byte	0	4.294.967.295	Pozitif uzun tam sayılar
float	4 byte	- 3,40E+45	+ 3,40E+45	ondalık sayılar
double		- 3,40E+45	+ 3,40E+45	ondalık sayılar
string	1 byte + x			Karakter veya dizi
array (dizi)	1 byte + x			Veri topluluğu



Veri Tipleri

- ✓ Arduino **8 bitlik** bir mikrodenetleyiciye sahip
- ✓ **Kısıtlı bir belleğe** sahip olduğundan veri tiplerini doğru kullanmak önemli
- ✓ **Noktalı-sayılar** fazla işlem gücü gerektiriyor!



Değişkenlerin Ömürleri

- ✓ Değişkenler program içerisinde **belirli bölgelerde tanımlı olup daha sonra yok edilebilirler.**
- ✓ **Global değişkenler** bütün program boyunca tanımlıdır, yok edilemezler
- ✓ **Lokal değişkenler** tanımlı oldukları blok / fonksiyon boyunca çalışıp fonksiyon veya bloktan çıkışınca yok edilir

Sabit Yapılar

1. true / false

TRUE komutu “1” , FALSE komutu “0” değeri ile aynı işlevdedir.

True ifadesi yerine 1 de yazılabilir.

```
void setup ()  
{  
Serial.begin (9600);  
}  
void loop () {  
  
if (x == TRUE)  
{  
    // x değişkeni 1 (TRUE)  
    //olduğunda kodlar çalışır  
    kodlar;  
}  
}
```

Sabit Yapılar

2. HIGH / LOW

digitalWrite() komutu ile “1” ve “0” bilgisi HIGH ve LOW kelimeleri ile dijital olarak gönderilebilir. HIGH kelimesi “1” değerine, LOW kelimesi “0” değerine eşittir

```
void setup ()  
{  
Serial.begin (9600);  
}  
void loop ()  
{  
digitalWrite(5, HIGH);  
 // 5. pine "1" bilgisini yaz  
 // (5. Pin 5V gerilim verir)  
digitalWrite(7, LOW);  
 //7. pine "0" bilgisini yaz  
//(7. Pin 0V gerilim verir)  
}
```

Sabit Yapılar

3. input / output

pinMode () komutu ile dijital giriş ve çıkış tanımlamak için kullanılır.

```
void setup ()
{
pinMode(5, INPUT) ;
//5. pin giriş olarak tanımlandı

pinMode(6, OUTPUT) ;
//6. pin çıkış olarak tanımlandı
}

void loop ()
{}
```

Mantıksal Operatörler

1. And (VE) Operatörü (&&)

Ancak iki durumda doğru ise TRUE (doğru) sonuç döndürür. Diğer durumlarda FALSE (yanlış) değer döndürür. Çift ve “&&” karakterleri ile belirtilir.

Örnek olarak Arduino 3 ve 4 dijital pinine bağlı butonların her ikisine birden basınca 5 numaralı pindeki ledi yanın uygulamamızın kodlarını yazalım:

```
void setup ()  
{  
    pinMode (3, INPUT) ;  
    pinMode (4, INPUT) ;  
    pinMode (5, OUTPUT) ; //Led pin çıkış  
}  
void loop ()  
{  
    bool buton1 = digitalRead (3) ;  
    bool buton2 = digitalRead (4) ;  
  
    if (buton1 == 1 && buton2 == 1) {  
        digitalWrite (5, HIGH) ;  
    }  
    else {  
        digitalWrite (5, LOW) ;  
    }  
}
```

Mantıksal Operatörler

2. OR (VEYA) Operatörü (||)

İki durumdan herhangi biri ya da her ikiside doğru ise TRUE (doğru) sonuç döndürür. Diğer durumlarda FALSE (yanlış) değer döndürür. Çift dikey çizgi “||” karakterleri ile belirtilir.

Örnek olarak Arduino 3 ve 4 dijital pinine bağlı butonların herhangi birine ya da ikisine birden basınca 5 numaralı pindeki ledi yakan uygulamamızın kodlarını yazalım:

```
void setup ()  
{  
    pinMode (3, INPUT) ;  
    pinMode (4, INPUT) ;  
    pinMode (5, OUTPUT) ; //Led pin çıkış  
}  
void loop ()  
{  
    bool buton1 = digitalRead (3) ;  
    bool buton2 = digitalRead (4) ;  
  
    if (buton1 == 1 || buton2 == 1) {  
        digitalWrite (5, HIGH) ;  
    }  
    else {  
        digitalWrite (5, LOW) ;  
    }  
}
```

Mantıksal Operatörler

3. Not (DEĞİL) Operatörü (!)

Değişkenin değerini terslemek için kullanılır. Eğer değer TRUE (doğru) ise FALSE (yanlış) , eğer değer FALSE (yanlış) ise TRUE (doğru) değer döndürür. Ünlem “!” karakteri ile belirtilir.

Örnek olarak Arduino 4 nolu dijital pinine bağlı butona basınca 5 numaralı pindeki ledi söndüren, bırakınca ise 5 numaralı pindeki ledi yakan uygulamamızın kodlarını yazalım:

```
void setup ()  
{  
pinMode (4, INPUT) ;  
pinMode (5, OUTPUT); //Led pin çıkış  
}  
void loop ()  
{  
bool buton = digitalRead(4);  
  
if (!buton) {  
//Eğer butona basılı DEĞİLSE!!  
digitalWrite(5, HIGH);  
}  
else {  
digitalWrite(5, LOW);  
}  
}
```

4. ? Operatörü

Durum kontrollerinde tek bir veri veya işlem sonucunu döndürmek istediğimizde ? Operatörü kullanabiliriz.

Veri kalabalığını ve satır sayısını azaltan oldukça kullanışlı bir operatördür.

Soru işaretinin öncesi koşulumuz sağlanıyorsa, soru işaretinden sonraki ilk değer, değilse ikinci değer döndürülüyor.

Aynı durumu if-else kontrolü ile de yapılabilir. Ancak ? operatörü ile daha kısa yapabiliz. İki yapının karşılaştırmasına bakalım:

```
void setup ()  
{  
Serial.begin(9600);  
}  
void loop ()  
{  
int sayı=4;  
string durum;  
  
if (sayı%2==0){  
durum= "çift";  
} else  
durum= "tek";  
// ? operatörü  
durum= (sayı%2 == 0) ? "çift" : "tek";  
}
```

Kontrol Yapısı

1. if

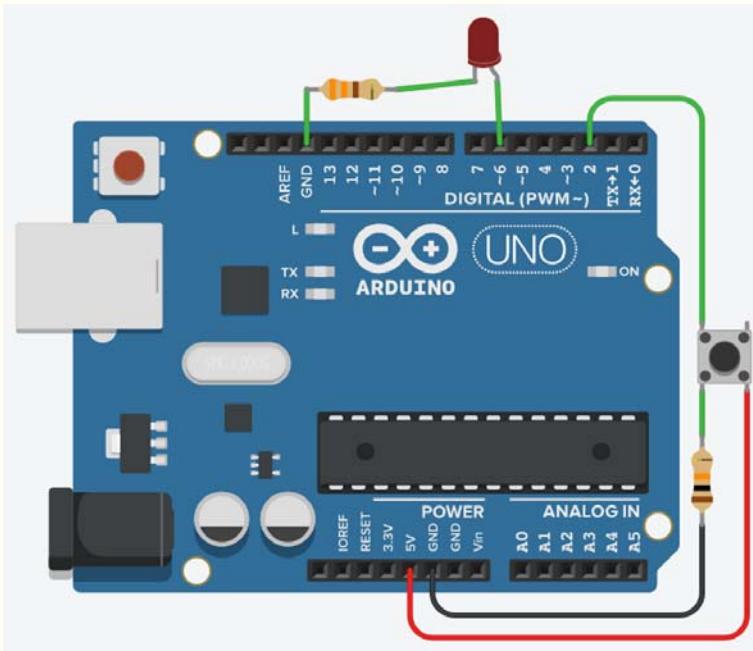
Eğer anlamına gelen bu yapıda if içerisinde yazılan ifade doğru ise yani “1”ise if komutunun iki süslü parantezi arasındaki kodlar çalışmaya başlar, ifadenin içindeki şart yanlış ise şart cümlesinden sonra gelen komut veya komut bloğunu çalışmaz, bir sonraki komuta geçilir.

Bu yapı ‘*Eğer butona basılmışsa Led i yak*’ gibi durumlarda veya karşılaştırmalarda kullanılır.

```
void setup () {  
Serial.begin (9600);  
  
}  
void loop () {  
  
if( şart ifadesi yazılır) {  
Doğru ise süslü parantez  
içindeki satırlar çalışır  
}  
Yanlış ise süslü parantez içini  
çalıştırmadan alt satıra geçer  
}
```

if

Şekildeki devreyi kurup program mantığını anlamaya çalışalım



```
void setup()
{
    pinMode(2, INPUT); // 2 numaralı pin giriş olarak
                        // ayarlandı
    pinMode(6, OUTPUT); // 10 numaralı pin giriş olarak
                        // ayarlandı
}
void loop()
{
    if (digitalRead(2) == HIGH) { //eğer butona basıldıysa
        digitalWrite(6, HIGH); //6 numaralı pine 5 V gönder
        delay(2000);           // 2 saniye bekle
    }
    digitalWrite(6, LOW); //6 numaralı pini pasif yap
}
```

Kontrol Yapısı

2. if / else

İf else değimi koşullu ifade yürütmek için kullanılır. İf eğer demekti, else değil ise demektir. if ve else birlikte kullanılır.

Programlama dilinde else tek başına kullanılamaz.

İf parantez içerisindeki değer şartı sağlıyorsa ("1") ise if yapısının kod veya kod bloğu çalışacaktır.

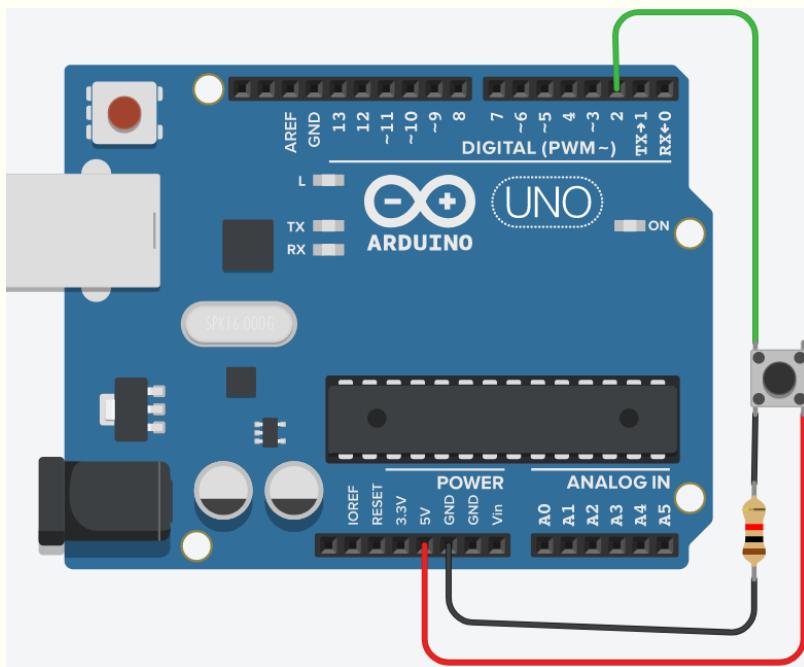
Yanlış ise ("0" ise) else kod veya kod bloğu çalışacaktır.

```
void setup()
{
    pinMode(2, INPUT);
    pinMode(6, OUTPUT);
}

void loop()
{
    if (digitalRead(2) == HIGH) {
        //eğer butona basıldıysa
        digitalWrite(6, HIGH);
            //6 numaralı pine 5 V gönder
        delay(2000);
            // 2 saniye bekle
    }
    digitalWrite(6, LOW);
            //6 numaralı pini pasif yap
} else {
    digitalWrite(6, LOW);
}
}
```

if / else

Şekildeki devreyi kurup program mantığını anlamaya çalışalım



```
void setup()
{
    Serial.begin(9600);          // seri haberleşme
    pinMode(2, INPUT);          // 2. pin giriş oldu
}

void loop()
{

    if (digitalRead(2) == HIGH) {
        Serial.println("Basildi"); // Seri ekrana yaz
        delay(500);
    }
    else{
        Serial.println("Basilmadi");
        delay(1000);
    }
}
```

Kontrol Yapısı

3. if / else if

if else yapısında sadece iki durumu kontrol edebiliyorduk, eğer ikiden fazla durum olacak ise if / else-if yapısını kullanmaktayız.

```
void setup()
{
}
void loop()
{
    // x 50'den küçük 40'dan büyük ise
    if( (x<50) && (x>40) )
    {
        y = 50; //y değişkeni 50 olsun
    } // x 40'dan küçük 30'dan büyük ise
    else if ( (x<40) && (x>30) )
    {
        y = 40; // y değişkeni 40 olsun
    } // x 30'dan küçük 20'den büyük ise
    else if ( (x<30) && (x>20) )
    {
        y = 30; // y değişkeni 30 olsun
    }
    else //hiçbiri değil ise
    {
        y = 0; // y değişkeni 0 olsun
    }
}
```

Kontrol Yapısı

4. switch / case

switch/case bir ifade veya değişkenin aldığı değere göre programın akışını istenen seçeneklere yönlendirmek için kullanılır.

switch yapısının parantezi içerisindeki değer case içerisinde hangisine eşit ise o case çalışır.

Bir switch/case yapısından çıkışı sağlamak ya da sonlandırmak için break yada return kullanılır.

```
void setup()
{
}

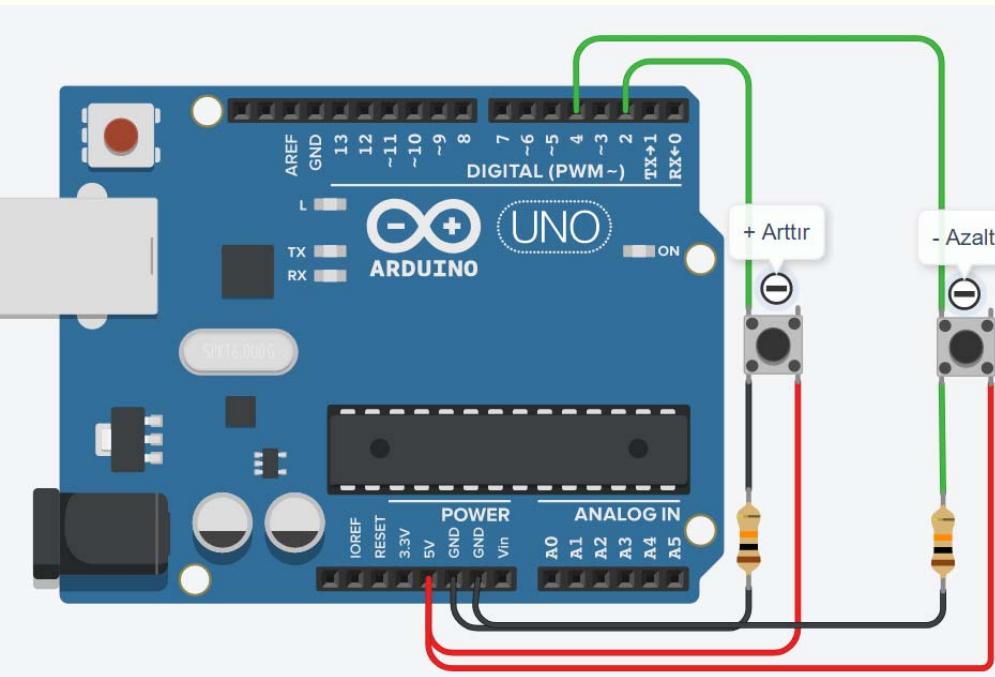
void loop()
{
    switch (sayi)
    {
        case 1:
            //Eğer sayı değişkeni 1 ise bu bölümdeki
            //kodlar işleyecek
            break; //daha sonra program break komutıyla
            //switch yapısından çıkacaktır

        case 2:
            //Eğer sayı değişkeni 2 ise bu bölümdeki
            //kodlar işleyecek
            break; //komutuyla program switch
            //yapısından çıkacaktır

        default:
            // Eğer hiçbir case çalışmassa bu bölüm
            // çalışır
            break; //switch yapısından çıkmak için
            // break komutu kullanılır.
    }
}
```

Kontrol Yapısı

switch / case



```
#define arttir 2
#define azalt 4
int durum_arttir;
int durum_azalt;
int sayi=0;

void setup() {
    pinMode(arttir, INPUT); pinMode(azalt, INPUT); Serial.begin(9600);}

void loop() {
    durum_arttir=digitalRead(arttir);    durum_azalt=digitalRead(azalt);

    if(durum_arttir==HIGH) {
        sayi++;    Serial.print("Sayi = ");    Serial.print(sayi);    delay(300);}

    else if(durum_azalt==HIGH){
        sayi--;    Serial.print("Sayi = ");    Serial.println(sayi);    delay(300);}

    switch (sayi) {
        case 1:
            Serial.println("motor 1 calisiyor");    break;
        case 2:
            Serial.println("motor 2 calisiyor");    break;
        case 3:
            Serial.println("motor 3 calisiyor");    break;
        default:
            Serial.println("motor durdu");
    }
}
```

switch / case

Uygulama:

Bir buton kullanarak programdaki sayı değişkeninin değerini Sıfırdan başlayarak butona her basısta birer birer arttırınız.

Sayı değeri 1 olduğunda seri monitöre :

'1. motor çalıştı' ifadesini

Sayı değeri 2 olduğunda:

'2. motor çalıştı' ifadesini

Bunların dışındaki durumlarda :

' Motorlar durdu' ifadesini yazdırınız.

Sayı değeri 7 olduğunda sayı değişkenini tekrar sıfıra eşitleyiniz.

Döngü Yapıları

1. while

while (ifade) döngüsü ifadenin şartı sağlanıyorsa ya da doğru olduğu durumlarda while döngüsünün süslü parantez arasındaki komutlarını sürekli olarak baştan sona çalıştırır.

```
void setup()
{
}
void loop()
{

while (değer == 100)
{
    x++; //değer değişkeni 100'e eşit olduğu sürece
          // x değişkeni 1 artar
}
}
```

Döngü Yapıları

1. while

while ifadesinde içerisindeki şart ifadesi doğru olduğu sürece 1 değeri üretilir, yanlış olduğu durumda “0” değeri üretilir.

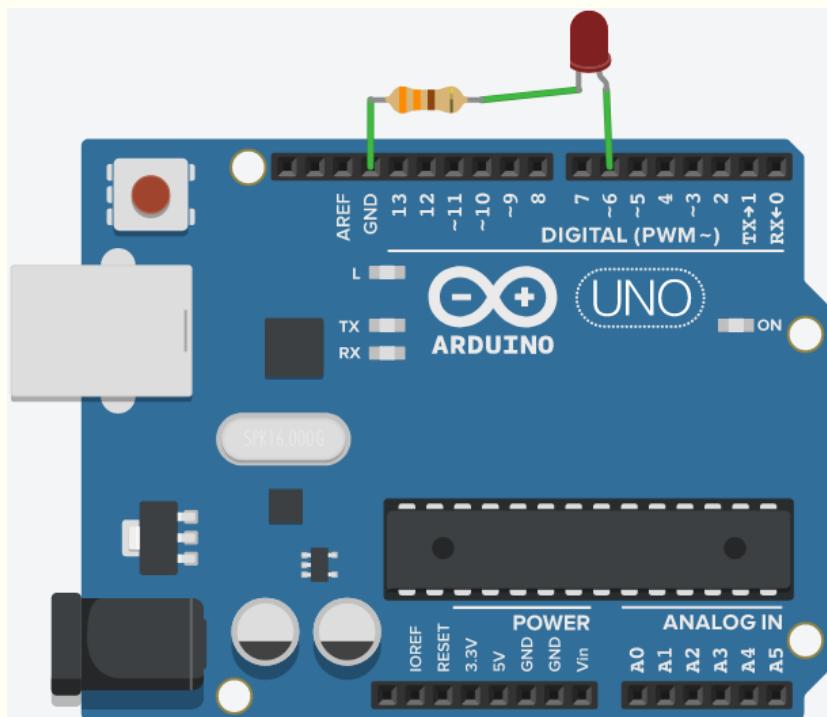
while içerişi “1” (true) olduğu sürece döngü devam eder, “0” olduğunda döngüden çıkarılır.

yandaki örnekte ifade içerişi her zaman “1” olduğundan, program while içerisinde sonsuz döngüye girer.

```
void setup()
{
}
void loop()
{
    while (1)
    {
        komutlar;
    }
}
```

while

Aşağıdaki devrede while döngüsüne girerek ledi yakacak, 1 saniye bekleyip ledi söndürecek ve ifadeyi 1 arttıracaktır. While döngüsünde 49 kere aynı işlemi yaptıktan sonra tekrar while döngüsüne girmeyecektir



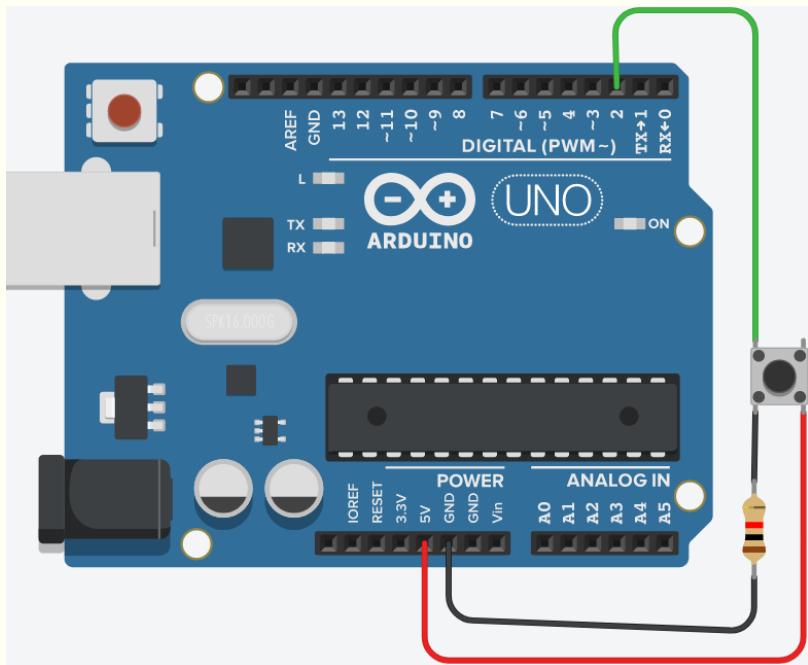
```
int ifade=0;

void setup() {
}

void loop()
{
    while (ifade<50)
    {
        digitalWrite(6, HIGH); // ledi yak
        delay(1000);          // 1 saniye bekle
        digitalWrite(6, LOW); // ledi söndür
        ifade;ifade+1;
    }
}
```

while

Uygulama: Buton kilitleme



Bir buton kullanarak programdaki sayı değişkeninin değerini Sıfırdan başlayarak butona her basışta birer birer arttırınız.

Butona her bastığımızda sayı değeri butondan elimizi çekene kadar bir defa artacaktır.

Buton basılı olduğu sürece sayı değeri bir kere değişecektir

Döngü Yapıları

2. do / while

do-while döngüsü öncelikle **do** parantez içindeki işlemi yapar ve **while** ile yapılan işlemi kontrol eder. while komutu ile aynı şekilde çalışır, tek farkı şart döngünün sonunda kontrol edilir. Yani **ifade yanlış olsa da hı döngü en az bir kere çalışır.**

Sağ taraftaki uygulamada **do** döngüsüne girecek sayıyı beş arttıracak, seri ekranımıza yazdıracak **while** ile durumu kontrol edecktir. Sayımız 25 ten büyük olduğu zaman döngüden çıkışacaktır.

```
int sayı = 0;  
void setup() {  
Serial.begin(9600); }  
void loop() {  
do {  
    sayı = sayı+ 5;  
    Serial.print("sayı = ");  
    Serial.println(sayı);  
    delay(500);  
}  
while (sayı < 25);  
}
```

do / while

Uygulama:

1. Programda ilk değeri sıfır olan bir değişken tanımlanacak.
2. Seri ekranın ilk olarak ‘Loop dongusu baslıyor’ ifadesi yazdırılıp altına belirtilen değişkenin sayı değeri yazdırılacak.
3. Program ilk durumda değişkenin değerini 1 artıracak. Seri ekrana sayının yeni değeri ile birlikte ‘Dongu Calisiyor’ ifadesini yazdırıracak
4. Sayı değeri 10 olduğunda program do-while döndüsünden çıkacak ve ekrana ‘Donguden Cikildi’ yazacak

Döngü Yapıları

3. for

Verilen şart doğru olduğu sürece belirtilen işlemlerin tekrarlanması sağlar.

```
for ( başlangıç değeri; döngünün devam etme koşulu; değerdeki değişim )
```

```
{
```

```
komutlar;
```

```
}
```

Değişken tanımlama burada da yapılabilir

```
for(int x=0; x<100; x++) {  
    Serial.println(x);  
}
```

başlangıç değeri
bitiş değeri
artış / azalış değeri

Döngü Yapıları

for

For döngüsünün başlangıç deyimi $x=10$, $y=100$ dür. Şart ifadesi ($x < 1000 \&\& y > 10$) dur. Şart ifadesi birden fazla olduğu için araya $\&\&$ işaretini konmuştur. Adım ifadesinde her değişken için ayrı bir artış/azalış miktarı verilmiştir.
 $x+=2$ ifadesi ile x her döngü çalıştığında 2 artacağını, $y-=5$ ifadesi ile her döngüde y değişkeninin 5 azalacağını belirtmektedir.

```
int x=0;
byte y=0;

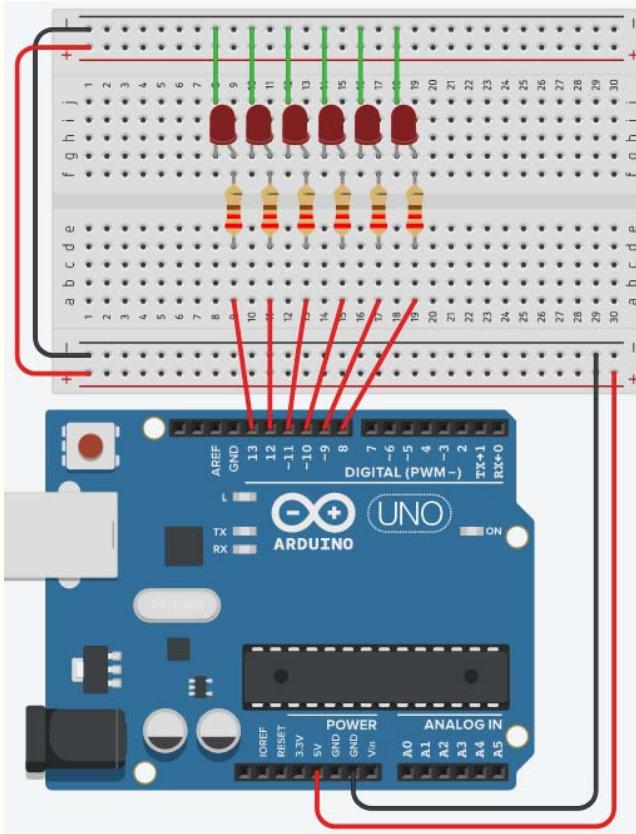
void setup() {
    Serial.begin(9600);
}

void loop() {

    for( x=10, y=100; ( x<1000 && y>10 ) ; x+=2, y-=5)
    {
        Serial.print(x);
        Serial.print("      ");
        delay(600);
        Serial.println(y);
        delay(600);
    }
}
```

for

Kara şimşek uygulaması



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

void setup () {
    for(int i=0; i<=5 ;i++)
    {
        pinMode(LEDdizisi[i], OUTPUT);
    }
}

void loop() {
    for(int i=0; i<=5; i++)
    {
        digitalWrite(LEDdizisi[i],HIGH);
        delay(200);
        digitalWrite(LEDdizisi[i],LOW);
    }

    for(int j=5;j>0; j--)
    {
        digitalWrite(LEDdizisi[j],HIGH);
        delay(200);
        digitalWrite(LEDdizisi[j], LOW);
    }
}
```

Döngü Yapıları

4. break

Break do, for, ve while döngülerinden döngü çalışması bittiğinde döngü dışına çıkmak için kullanılır. Switch case yapısında da kullanılır.

Sağdaki programda sensör değeri 300 den büyükse döngü dışına çıkarılır.

```
byte x = 0;  
int sens;  
void setup() {  
    Serial.begin(9600); }  
void loop() {  
    for (x = 0; x < 255; x++) {  
        sens = analogRead(sensorPin);  
        if (sens > 300) {  
            x = 0;  
            break;  
        }  
    }  
}
```

Döngü Yapıları

5. continue

Continue *do, for, while* döngülerinde bir satırın, işlem yapılmadan geçilmesini istediğimiz durumlarda kullanılır. continue komutu çalıştığında program döngünün başına döner.

if ifadesi doğru olmadığından program delay(3000); komutuna gelerek 3 sn bekleyecektir.

if ifadesi doğru olduğunda **continue;** komutuna gelinecektir, böylece tekrar for döngüsünün başına gidecektir, bu nedenle 3 sn beklemeyecektir.

```
byte x = 0;  
  
void setup() {  
    Serial.begin(9600); }  
  
void loop() {  
    for (x = 0; x < 255; x++) {  
        digitalWrite(5,1);  
        delay(500);  
        digitalWrite(5,0);  
        if (x>40 && x<120) {  
            continue;  
        }  
        delay(3000);  
    }  
}
```

Döngü Yapıları

6. return

Bu komut fonksiyonu sonlandırır. Bir değer veya değişkenle kullanılması durumunda fonksiyonun sonucu olarak o değeri verir.

Parametre döndüren fonksiyon yapılarında kullanılır.

Return komutunun kullanıldığı satırdan sonraki satırlar asla çalıştırılmaz.

Bazı uzun ve karmaşık programlarda hataları bulmamıza yardımcı olur.

```
byte degisken;  
byte deger;  
void setup() {  
    Serial.begin(9600); }  
void loop() {  
    int fonksiyon() {  
        if (degisken > deger) {  
            return degisken;  
        }  
        else{  
            return deger;  
        }  
    }  
}
```

return

Sağ taraftaki uygulamada değer alan ve parametre döndüren fonksiyon ile return uygulaması yapılmıştır. Bu uygulama değişkenlere atanmış sayı değerlerinin toplamını ve bölümünü yapmaktadır.

```
int fonksiyonarti(int r, int y) {  
    int top = r + y ;  
    return top; }  
  
float fonksiyonbolme (int x , float y) {  
    float bol = x / y;  
    return bol; }  
  
void setup(){  
    Serial.begin(9600);}  
  
void loop(){  
    int a = 5;  
    int b = 6;  
    int toplam = fonksiyonarti(a, b);  
    float bolme = fonksiyonbolme(a, b);  
    Serial.print(" toplam = ");  
    Serial.println(toplam);  
    Serial.print(" Bolme = ");  
    Serial.println(bolme);  
    while (1);  
}
```

Fonksiyon Türleri

1. Değer almayan, parametre döndürmeyen fonksiyonlar

Bu fonksiyonlar void ile başlar ve parantez içerisinde **değer almadığı** için boş bırakılır.

Sağ taraftaki örnekte ledi yakıp söndürme ile ilgili bir program yazılmıştır.
Ledin yarımsaniye aralıklarla yanıp sönmesi için;

void ledyak() adında bir fonksiyon oluşturulmuştur. Devreye bağlı butona her basıldığında fonksiyon **ledyak();** komutu ile çağrılp ledin yanıp sönmesi sağlanmıştır.

```
#define led 2
#define buton 3
void setup() {
    pinMode(led, OUTPUT);
    pinMode(buton, INPUT);
}
void loop() {
    if(digitalRead(buton)==HIGH) {
        ledyak();
    }
    else {
        digitalWrite(led, LOW);
    }
}
void ledyak() {
    digitalWrite(led, HIGH);
    delay(500);
    digitalWrite(led, LOW);
    delay(500);
}
```

Fonksiyon Türleri

2. Değer alan, parametre döndürmeyen fonksiyonlar

Bu fonksiyonlar parametre döndürmediği için void ile başlar ve parantez içerisine değişkenler tanımlanır.

Sağ taraftaki örnekte ledi yakıp söndürme ile ilgili bir program yazılmıştır.

void zamanayarliled() adında bir fonksiyon oluşturulmuştur.

Parantez içerisinde sırasıyla **a** ve **zaman** adında değişkenler tanımlanmıştır.

Ledin kaç kere ve hangi aralıklarla yanıp soneceği **zamanayarliled(5, 600);**

fonksiyonu içerisinde yazılan değerlerle belirtilmiştir. Burada 5 sayısı **a** değişkenine, 600 sayısı ise **zaman** değişkenine atanmıştır. Programda butona basıldığında ledi 5 defa 600 ms aralıklarla yakıp söndürecektr.

```
#define led 2
#define buton 3
void setup() {
pinMode(led, OUTPUT);
pinMode(buton, INPUT);}
void loop() {
}
if(digitalRead(buton) == HIGH){
    zamanayarliled(5, 600); }
else {
    digitalWrite(led, LOW); }
}
void zamanayarliled( int a, int zaman){
for(int i=0; i<a; i++){
digitalWrite(led, HIGH);
delay(zaman);
digitalWrite(led, LOW);
delay(zaman);
}}
```

Fonksiyon Türleri

3. Değer almayan, parametre döndüren fonksiyonlar

Bu fonksiyonlar parametre döndürdüğü için void kullanılmaz. Fonksiyon int ile başlar.

Sağ taraftaki örnekte devreye bağlı potansiyometrenin değerinin seri akrandan okunması sağlanmıştır.

int pot() adında bir fonksiyon oluşturulmuştur. Devreye bağlı butona her basıldığından fonksiyon **pot();** komutu ile çağrılp fonksiyonun değeri seri ekrandan okunur.

Return komutu parametre döndüren fonksiyonlarda kullanılır.

```
byte potpin=A0;
#define buton 2

void setup() {
    Serial.begin(9600);
    pinMode(buton, INPUT);
}

void loop() {
    if(digitalRead(buton)==HIGH) {
        Serial.print(pot());
    }
}

int pot(){
    byte potdeger = analogRead(potpin);
    delay(500);
    return potdeger;
}
```

millis()

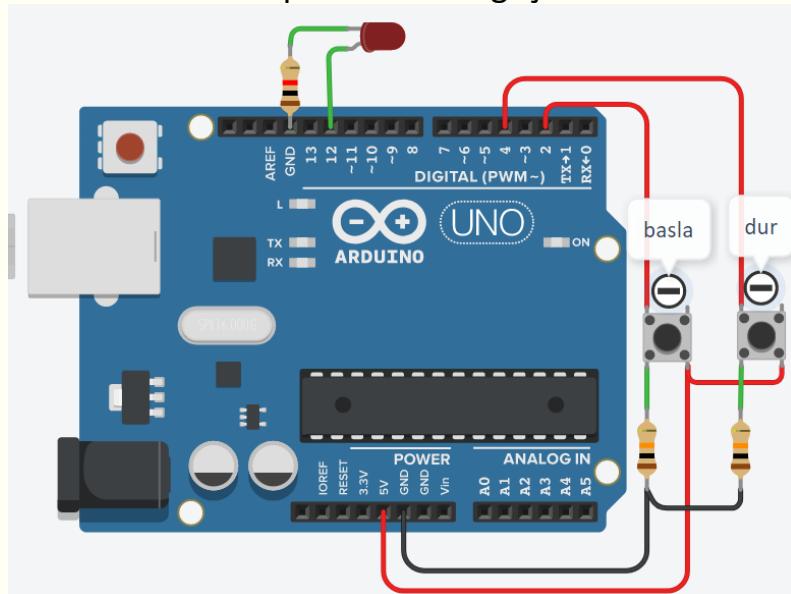
Millis bir fonksiyondur. Kullandığınız zaman unsigned long veri tipinde bir sayı döndürür. Her bir milisaniyede artan bir sayaç olarak çalışır.

Arduino' nun enerjisini kesildiğinde bu sayaç sıfırlanacak ve enerji aldığında tekrar sıfırdan başlayacaktır. Bunun dışında unsigned long int' in max değeri olan 4294967295 değeri geçildiğinde de sayaç sıfırlanacak ve sıfırdan yeniden saymaya başlayacaktır.

```
unsigned long eskiZaman=0;  
unsigned long yeniZaman;  
int LEDdurumu = 0;  
void setup() {  
    pinMode(13,OUTPUT);  
    Serial.begin(9600); }  
void loop() {  
    yeniZaman = millis();  
    /* bir önceki turdan itibaren 1000 milisaniye geçmiş mi  
     * yani yeniZaman ile eskiZaman farkı 1000den büyük mü */  
    if(yeniZaman-eskiZaman > 1000) {  
        if(LEDdurumu == 1){  
            digitalWrite(13,LOW);  
            LEDdurumu = 0; }  
        else{  
            digitalWrite(13,HIGH);  
            LEDdurumu = 1; } /* Eski zaman değeri yeni zaman değeri  
             * ile güncelleneiyor */  
        eskiZaman = yeniZaman; } }
```

Uygulama

Millis() komutu ile basla butonuna basıldığında 4 saniye süreyle yanın led uygulaması yapılmıştır. Led aktif edildikten sonra herhangi bir zamanda stop butonuna basılırsa led pasif duruma geçecektir.



```
unsigned long zaman;

#define 4 dur

#define 2 basla

void setup(){

  pinMode(13, OUTPUT);

  pinMode(dur, INPUT);

  pinMode(basla, INPUT);

}

void loop(){

  if(digitalRead(basla)) zaman=millis();

  if(digitalRead(dur))    zaman=0;

  bool durum=(millis()-zaman>=2000||zaman==0 ? false:true);

  digitalWrite(13,durum ? HIGH : LOW);

  if(!durum) zaman=0;

}
```

Tek Boyutlu Diziler

Belli miktardaki verileri toplu olarak tutmak için kullanılıyor.

```
int ogrenciler [10]={15,20,10,30,60,70,50,40,85,90};  
          ↘   ↗  
          Öğrenci sayısı  
          ↗  
          Öğrencilerin aldığı notlar  
  
void setup()  
{  
    Serial.begin(9600);  
}  
void loop()  
{  
    Serial.println(ogrenciler[0]);  
}  
// 1. sıradaki öğrenci 15 puan aldığı için ekrana 15  
sayısı yazdırılacak
```

Tek Boyutlu Diziler

Bir önceki örnekte 2 numaralı öğrencini notuna değiştirmek isteseydik;

```
int ogrenciler [10]={15,20,10,30,60,70,50,40,85,90};

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

void loop()
{
    ogrenciler[1]=100;
    Serial.println(ogrenciler[1]);
}

// 2. sıradaki öğrencinin notu 100 olarak değiştirilmiş oldu.
```

Tek Boyutlu Diziler

```
int ogrenciler [10]={15,20,10,30,60,70,50,40,85,90};  
sayısı ile dizinin boyutunu belirttik.
```

```
int ogrenciler []={15,20,10,30,60,70,50,40,85,90};  
bos bırakılırsa dizede belirttiğimiz karakter kadar yer açılacak
```

Eğer oluşturduğumuz diziye kaç adet değer alacağını belirtirsek ve belirttiğimiz sayıdan daha az değer girdiğimizde, geri kalan dizi değeri sıfır olarak belirlenir.

```
int ogrenciler [6]={10,5,30,40};  
// şeklinde bir dizi tanımladığımızda  
Serial.println(ögrenciler[5]);  
// komutunun çıktısı ne olur?
```

Tek Boyutlu Diziler

Tanımladığımız bir değişkene,
oluşturduğumuz dizinin elemanını
tanımlayabiliriz.

```
int ogrenciler [10]={15,20,10,30,60,70,50,40,85,90};  
int x;  
  
void setup()  
{  
    Serial.begin(9600);  
}  
  
void loop()  
{  
    x=ogrenciler[3];  
  
    // 4. sıradaki öğrencinin notu x değişkenine atanmış oldu  
  
    Serial.println(x);  
}
```

Tek Boyutlu Diziler

Bir dizi elemanına bir değişkenin değeri tanımlanabilir.

```
int ogrenciler [10]={15,20,10,30,60,70,50,40,85,90};  
int x=95;  
  
void setup()  
{  
    Serial.begin(9600);  
}  
  
void loop()  
{  
    ogrenciler[3]=x;  
  
    // x değişkeninin değeri 4. sıradaki öğrencinin notuna atandı  
  
    Serial.println(ogrenciler[3]);  
}
```

Karakter Dizileri String

Karakterleri **char** veri tipi ile tanımlarız. Harflerden oluşan bir dizi tanımlamak istersek ;

```
char alfabe [6]={'a','b','c','d','e','f','g';

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

void loop()
{
    Serial.println(alfabe[0]);
    //ekrana a harfi yazdırılır.
}
```

String

Eğer tanımladığımız dizideki elemanlardan birini başka bir dizi karakteriyle değiştirmek istersek;

```
char alfabe [6]={'a','b','c','d','e','f','g'};  
  
char yeniharf='x';  
  
void setup()  
{  
    Serial.begin(9600);  
}  
  
void loop()  
{  
    alfabe[2]=yeniharf;  
    Serial.println(alfabe[2]);  
    //alfabe dizisinin 3. harfi x ile değiştirildi  
}
```

Uygulama

Alfabe dizisinin karakterlerini 500ms
aralıklarla, sırayla ve sadece bir defa
ekrana yazdırınız