Operatörler

- Matematiksel, mantıksal veya daha farklı işlemleri gerçekleştirmek için kullanılan simgelerdir.
- **Aritmetik Operatörler**: Sayısal türden değişkenler üzerinde işlem yapan operatörlerdir.

Operatör	Açıklama
+	Toplama işlemi için kullanılır.
-	Çıkarma işlemi için kullanılır.
*	Çarpma işlemi için kullanılır.
/	Bölme işlemi için kullanılır.
%	Mod Alma (Kalanı Bulma) için kullanılır.
=	Değer atama için kullanılır.
++	Değişkenin değerini bir arttırmak için kullanılır.
--	Değişkenin değerini bir azaltmak için kullanılır.

```
int sayi1, sayi2;
sayi1=10;
sayi2=3;
int toplam=sayi1+sayi2;
int fark=sayi1-sayi2;
int carpim=sayi1*sayi2;
float bolum=sayi1/sayi2
int kalan=sayi1%sayi2;
sayi++;
sayi--;
```

Operatörler

- **Birleşik Operatörler**: Aritmetik operatörlerin atama operatörü ile birlikte kullanıldığım ikili operatörlerdir.

Operatör	Açıklama
+=	Toplama işlemi yap sonra atama yap
-=	Fark işlemi yap sonucu ata
*=	Çarpma işlemi yap sonra atama yap
/=	Bölme işlemi yap atama yap
%=	Kalanı bul atama yap

```
int a=20,b=30;
a+=b;    // a=a+b;
a*=b;    // a=a*b;
a/=b;    // a=a/b;
b%=a;    // b=b%a;
```

- **Karşılaştırma Operatörleri**: İfadelerin karşılaştırılması için kullanılan operatörlerdir.

Operatör	Açıklama
==	Eşit mi?
!=	Eşit Değil mi ?
>	Büyük mü?
<	Küçük mü?
>=	Büyük eşit mi?
<=	Küçük Eşit mi?

```
int a=0,b=-5;
a==0
a!=0
b>=a
a<=b
```

Operatörler

- **Üçlü Operatör**: Bir koşul ifadesine bağlı olarak iki farklı işlemin gerçekleştirilebildiği deyimdir.

```
(Koşul)? Değer1: Değer2;
int LED=(T<0):girisPin++:girisPin--;
```

- **Mantıksal Operatörler**: Koşul ifadeleri üzerinde mantıksal işlemler yapan operatörlerdir.

Operatör	Açıklama
&& (AND, Ve)	İki önermenin true olması durumunda true, diğer durumlarda false üretir.
|| (OR, Veya)	Önermelerden birinin true olması durumunda true, diğer durumlarda false üretir.
! (NOT, Değil)	Önermenin değilini alır. True ise false, false ise true üretir.

```
int a=7;
int b=-4;
bool sonuc1=a>0 && b>0;    // Sonuç false olacaktır. Çünkü ikinci
                           // şartın sonucu false'tir.
bool sonuc2= a>0 || b>0;
bool sonuc3=a!>0;
```

Operatörler

- **Bitsel Operatörler**: Değerler üzerinde bit düzeyinde işlem yapan operatörlerdir.
- & (AND, Ve) Operatörü: Sayısal değerler üzerinde bit düzeyinde AND (ve) işlemi yapan operatördür.

```
int a=6;     // ikili:  0110
int b=5;     //  ikili:  0101
int c=a&b;   //  ikili sonuç=0100, onlu : 4
```

- I (OR, Veya) Operatörü: Sayısal değerler üzerinde bit düzeyinde OR(Veya) işlemi yapar.

```
int a=6;     // ikili:  0110
int b=5;     //  ikili:  0101
int c=a | b;  //  ikili sonuç=0111, onlu : 7
```

Operatörler

- ^(XOR, Özel Veya) Operatörü: Bu operatör bitlerin farklı olması durumunda 1, aynı olmaları durumunda ise 0 değeri üretir.

```
int a=6;       // ikili:  0110
int b=5;       // ikili:  0101
int c=a ^ b;   //  ikili sonuç=0011, onlu : 3
```

- ~(NOT, Değil) Operatörü: Değil operatörü ~ ile gösterilir ve uygulandığı bit ifadesinin tersini alır.

```
int a=6;      // ikili:  0110
int b=~a;    //  ikili sonuç=1001, onlu : 9
```

Operatörler

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

void loop()
{
  int a=6;
  int b=5;
  int ve=a & b;
  int veya=a|b;
  int oveya=a^b;
  Serial.println(ve,DEC);
  Serial.println(veya,DEC);
  Serial.println(oveya,DEC);
  Serial.println(ve,BIN);
  Serial.println(veya,BIN);
  Serial.println(oveya,BIN);
}
```

Operatörler

- **Bitsel Kaydırma Operatörleri**: Bitsel kaydırma operatörleri bitsel olarak ifade edilen sayıların bitlerini sağa(>>) veya sola (<<) kaydıran operatörlerdir.

```
byte a=7; // ikili değer: 0000 0111
byte b=a>>1; // Sağa bir kere kaydır.
// İkili 0000 0011, onlu 3 olur. Sayı ikiye bölündü.

byte c=a<<1; // Sola bir kere kaydır.
// İkili 0000 1110 Onlu 14 olur. Sayı iki ile çarpıldı.
```

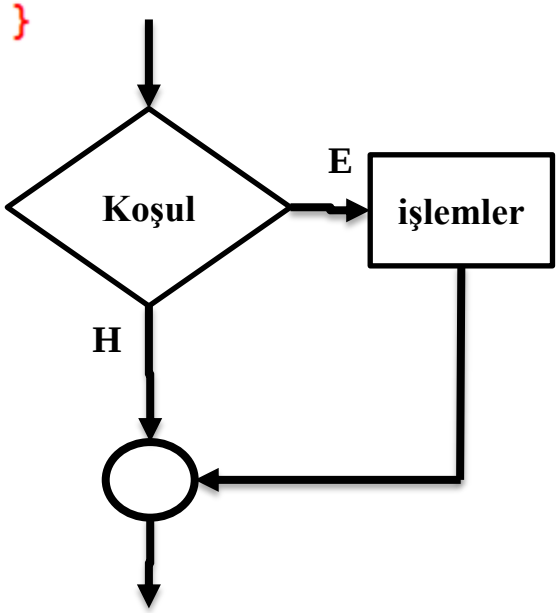
Operatörler

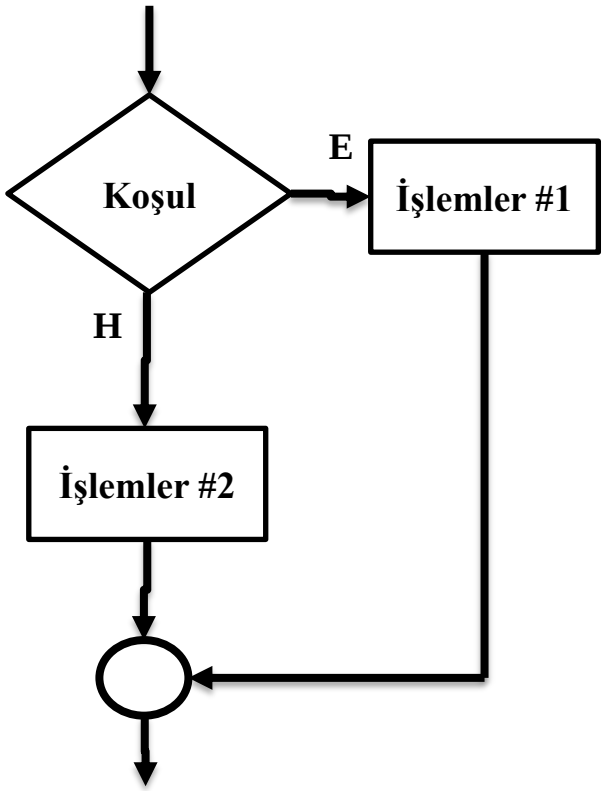
```
void setup()
{
Serial.begin(9600);
}

void loop()
{
int a=7;
int saga=a>>1;
int sola=a<<1;
Serial.println(sag,BIN);
Serial.println(sol,BIN);
Serial.println(sag,DEC);
Serial.println(sol,DEC);
delay(100);
}
```

Kontrol Yapıları

- Bu deyimler, koşullu işlem yapan deyimlerdir. if - else tek bir karşılaştırma deyimi olup else kullanımı isteğe bağlıdır. Eğer bu koşul olumlu ise if den sonraki bölüm yürütülür ve else den sonraki bölüm atlanır. Koşul olumsuz ise if den sonraki bölüm atlanır ve eğer varsa, else den sonraki bölümdeki işlemler gerçekleştirilir. if deyiminin else olmadan ve else ile birlikte kullanımı şu şekildedir:

```
if(koşul){
    ...
    deyimler;
    ...
}
```

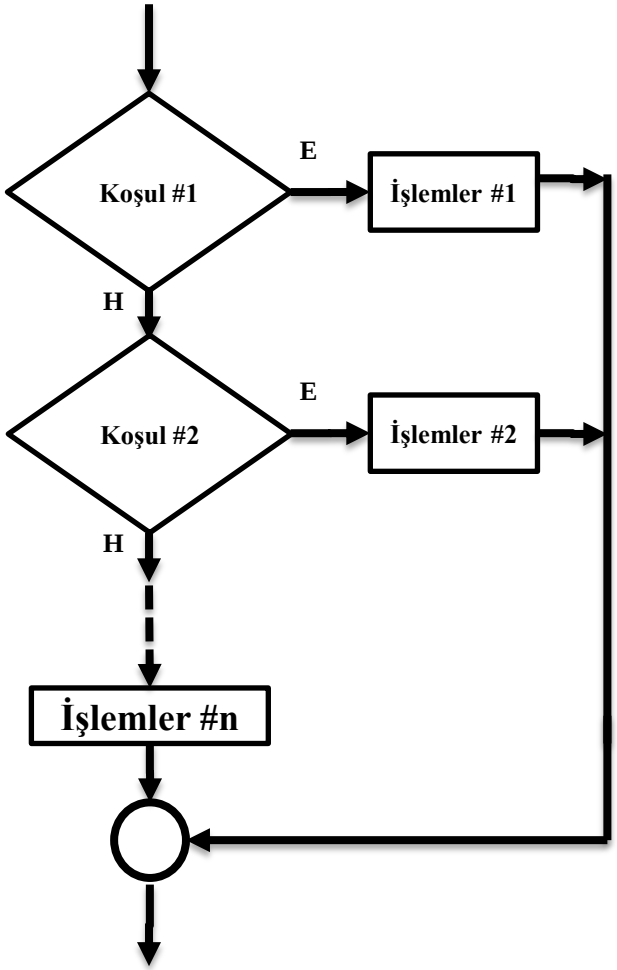





```
if(koşul){
    ...
    deyimler;
    ...
} else {
    ...
    deyimler;
    ...
}
```

Kontrol Yapıları

- Eğer koşullarınızın sayısı ikiden çok ise aşağıdaki gibi **if-elseif-else** yapısı kullanılır.




```
if(koşul_1) {
    ...
    deyimler; (küme_1)
    ...
} else if(koşul_2) {
    ...
    deyimler; (küme_2)
    ...
}
.
.
.
else if(koşul_n-1) {
    ...
    deyimler; (küme_n-1)
    ...
}
else {
    ...
    deyimler; (küme_n)
    ...
}
```

Kontrol Yapıları

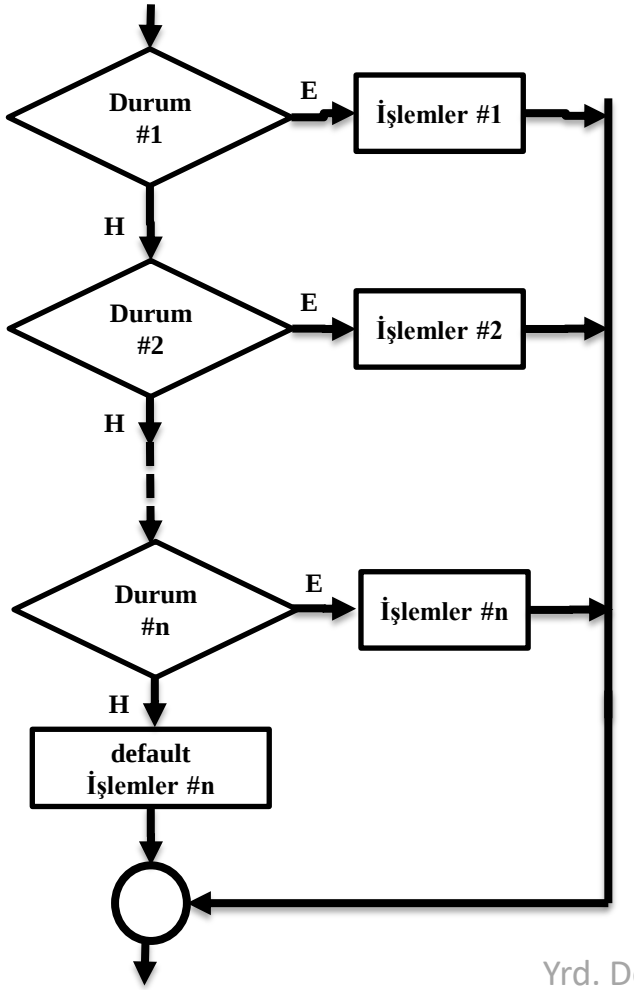
```
if(a%2==0)
{
Serial.println("Çift Sayı");
}
else
{
Serial.println("Tek Sayı");
}
```

```
if(a>0 && b<0)
{
Serial.println("Şartlar Sağlıyor")
}
else
{
Serial.println("Şartlar sağlanmıyor")
}
```

```
if(a==0)
{
Serial.println("Sıfır");
}
else if(a==1)
{
Serial.println("Bir");
}
else if(a==2)
{
Serial.println("İki");
}
else if(a==3)
{
Serial.println("Üç");
}
….
….
else
{
Serial.println("Hiç biri");
}
```

switch

- Bir değişkenin içeriğine bakarak, programın akışını bir çok seçenekten birine yönlendirir. Seçenekler case (durum) deyimi ile belirlenir ve ilgili komutlar işleme konur. Bütün durumların aksi söz konusu olduğunda default deyimi altındaki deyimler çalıştırılır. Genel yazım biçimi:

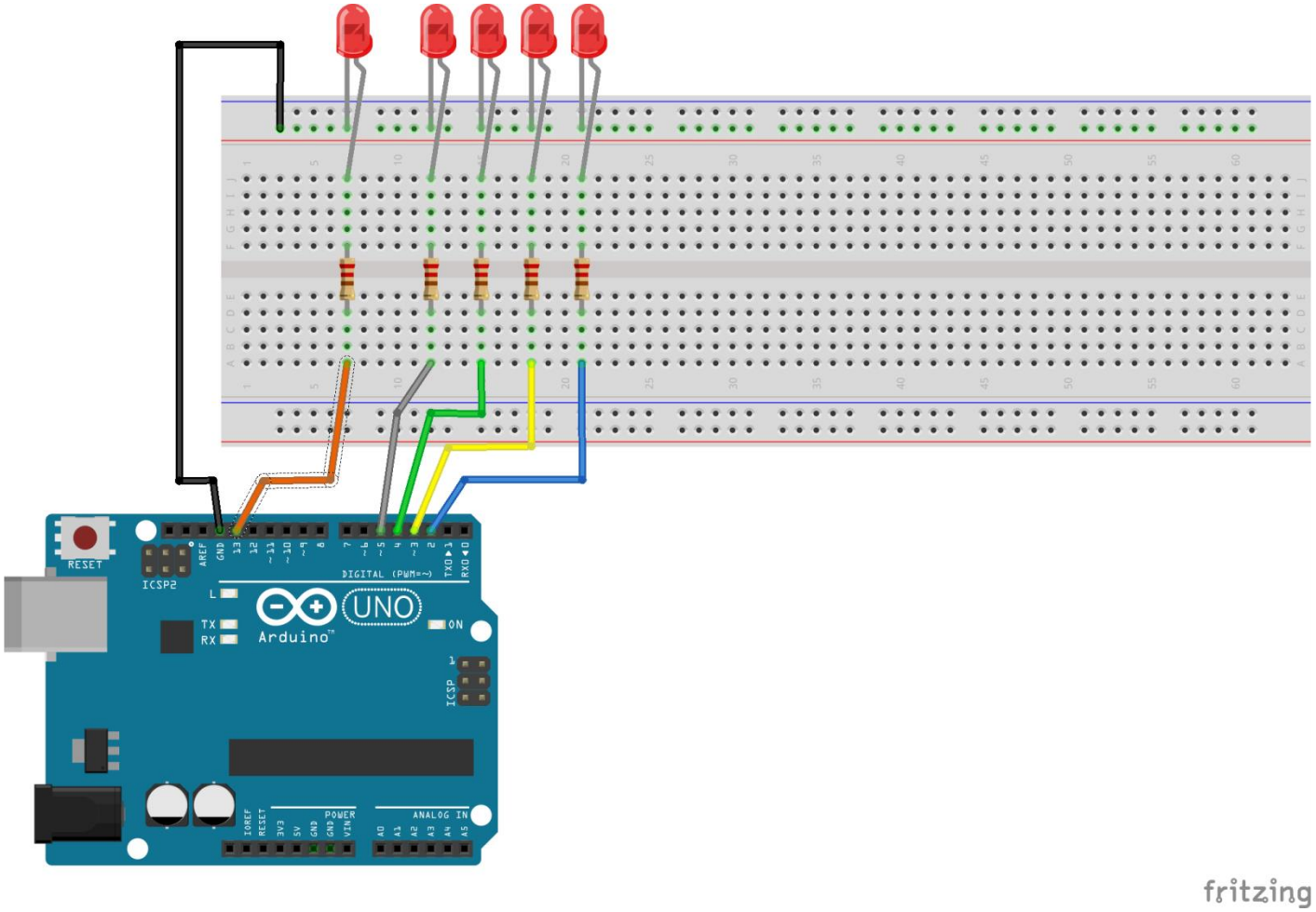



```
switch(değişken) {
    case sabit1:
        ...
        deyimler;
        ...
    case sabit2:
        ...
        deyimler;
        ...
    .
    .
    .
    case sabitn:
        ...
        deyimler;
        ...
    default:
        ...
        hata deyimleri veya varsayılan deyimler;
        ...
}
```

switch

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

void loop() {
  int dijitalpin = Serial.parseInt();
  switch(dijitalpin)  {
    case 1:
    digitalWrite(2, HIGH);
    break;
    case 2:
    digitalWrite(3, HIGH);
    break;
    case 3:
    digitalWrite(4, HIGH);
    case 4:
    digitalWrite(5, HIGH);
    break;
    default:
    digitalWrite(LED_BUILTIN, HIGH);
    break
    }
}
```

switch


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5
fritzing