

Bölüm Rehberi

Arduino Programlama: Değişken, Koşul, Döngü ve Fonksiyon

Arduino'da veri türleri ve aralıkları, sabit ve dizi, if ve switch, for ve while döngüleri, fonksiyon yazma ve seri monitörle hata ayıklama; çözümlü örnekler.

Bölüm Rehberi

Arduino Programlama: Değişken, Koşul, Döngü ve Fonksiyon

bote.web.tr

BELGE BİLGİLERİ

Belge kodu	BT-070
Yazı no	70
Sürüm	1.0
Sürüm tarihi	4 Ekim 2026
İçerik izi	9565 B426 4929
Yayımlayan	bote.web.tr

GÜNCELLEK KONTROLÜ

Bu belgeyi kullanmadan veya paylaşmadan önce güncelliğinden emin olunuz. QR kodu okutun ya da adrese girin: elinizdeki sürümün güncel olup olmadığını ve sonradan düzeltilen bir hata bulunup bulunmadığını gösterir.

bote.web.tr/d/bt-070/1-0

Hatalar düzeltildikçe yeni sürüm yayımlanır; bütün değişiklikler sitede sürüm geçmişinde listelenir.

Bu belge tamamen ücretsizdir. Size satmaya çalışanlara itibar etmeyiniz.



Güncelliği kontrol et

İÇİNDEKİLER

- | | |
|--|--|
| 1. Arduino programının iskeleti: setup() ve loop() | 11. Hepsi bir arada: sırayla yanan LED'ler |
| 2. Değişken ve veri türleri | 12. Öğrencilerin sık yaptığı hatalar |
| 3. Tam sayı bölmesi, kalan ve taşıma | 13. Alıştırmalar |
| 4. Sabitler ve kapsam | 14. Cevaplar |
| 5. Diziler | 15. Terim tablosu |
| 6. Koşullar: if, else if ve else | 16. Sonuç |
| 7. switch case | 17. Özet |
| 8. Döngüler: for, while ve do...while | 18. Sık sorulan sorular |
| 9. Fonksiyon yazmak | 19. Kaynaklar |
| 10. Hata ayıklama: seri monitör | 20. Atıf |

Bu yazı, [Fiziksel Programlama dersi](#) rehberinin ikinci derin yazısıdır ve Arduino'da program yazmanın temel yapı taşlarını anlatır: değişken ve veri türleri, sabitler ve diziler, koşullar, döngüler ve fonksiyonlar. Her başlıkta sözdizimi, Arduino UNO'ya özgü sınırlar ve çözümlü örnekler verilir; ardından seri monitörle hata ayıklama, sık yapılan hatalar ve cevaplı alıştırmalar gelir. Pinlerden okuma ve pinlere yazma fonksiyonlarını (digitalWrite, analogRead gibi) [Arduino Dijital ve Analog Giriş Çıkış](#) yazısında anlattık; bu yazı o fonksiyonların bir programın içinde nasıl düzenlendiğini anlatır.

Arduino programının iskeleti: setup() ve loop()

Arduino'da yazılan programa sketch denir. Her sketch'te bulunması zorunlu iki fonksiyon vardır, setup() ve loop(); öteki bütün fonksiyonlar bu ikisinin süslü parantezlerinin dışında yazılır.^[1]

- **setup()** sketch başladığında çağrılır. Kart her açıldığında ya da sıfırlandığında yalnız bir kez çalışır; değişkenlere ilk değer vermek, pin modlarını ayarlamak ve kütüphaneleri başlatmak için kullanılır.^[2]
- **loop()** adının söylediğini yapar: art arda, durmadan tekrar eder ve programın değişip tepki vermesini sağlar.^[3]
- İki fonksiyonun önündeki **void**, fonksiyonun çağrıldığı yere bir değer döndürmeyeceğini gösterir.^[4]

```
int sayac = 0;           // fonksiyonların dışında: global değişken
```

```
void setup() {  
    Serial.begin(9600);    // yalnız bir kez çalışır  
}
```

```
void loop() {  
    sayac = sayac + 1;      // her turda 1 artar  
    Serial.println(sayac);  // seri monitöre 1, 2, 3, ... yazar  
    delay(1000);           // 1 saniye bekler  
}
```

Arduino'nun int sayfasındaki örnek de aynı işi yapar: 0 değeriyle başlayan bir tam sayı değişkeni her turda 1 artar ve seri monitörde görüntülenir.^[5]

Değişken ve veri türleri

Değişken, programın bir değeri saklamak için kullandığı adlandırılmış bellek alanıdır. Tanımlanırken önce veri türü, sonra ad yazılır; istenirse ilk değer de verilir: `int sayac = 0;` gibi. Veri türü, değişkenin bellekte kaç bayt yer kaplayacağını ve hangi aralıkta değer alabileceğini belirler. Arduino UNO'da sık kullanılan türler şunlardır:

Tür	UNO'da boyut	Değer aralığı
bool	1 bayt	true ya da false ^[6]
byte	8 bit	0 ile 255 ^[7]
char	En az 8 bit	Tek karakter ^[8]
int	16 bit	-32.768 ile 32.767 ^[5]
unsigned int	16 bit	0 ile 65.535 ^[9]
long	32 bit	-2.147.483.648 ile 2.147.483.647 ^[10]
unsigned long	32 bit	0 ile 4.294.967.295 ^[11]
float	32 bit	$\pm 3,4028235 \times 10^{38}$ ^[12]

Tablodaki türler hakkında bilinmesi gerekenler:

- **int** Arduino'nun temel tam sayı türüdür. UNO ve öteki ATmega tabanlı kartlarda 16 bittir; Due ve SAMD tabanlı kartlarda ise 32 bittir ve -2.147.483.648 ile 2.147.483.647 arasında değer alır.^[5] Yani aynı program farklı kartta farklı sınırla çalışabilir.
- int negatif sayıları **ikiye tümlleme** yöntemiyle saklar; en yüksek bit, sayının negatif olduğunu gösteren işaret bitidir.^[5] İkiye tümlmeyi [Sayı Sistemleri](#) yazısında anlattık.
- **unsigned int** int ile aynı 2 baytı kullanır ama negatif sayı tutmaz; bu yüzden aralığı 0 ile 65.535'tir.^[9]
- **char** bir karakter saklar ve karakter tek tırnakla yazılır ('A'). Karakterler sayı olarak (ASCII kodu) tutulduğu için 'A' + 1 işleminin değeri 66'dır; büyük A harfinin ASCII değeri 65'tir.^[8] Birden çok karakterden oluşan metin çift tırnakla yazılır.^[8]
- **float** ondalıklı sayılar içindir ve toplam 6-7 basamak duyarlıdır; bu sayı virgülden sonraki basamakları değil, bütün basamakları kapsar. UNO'da double da float ile aynı boyuttadır, daha fazla duyarlılık vermez.^[12]
- Kayan noktalı sayılar tam değildir ve karşılaştırıldıklarında beklenmedik sonuç verebilir; iki float'ı == ile karşılaştırmak yerine farklarının mutlak değerinin küçük bir sayıdan küçük olup olmadığına bakılır. Float işlemleri tam sayı işlemlerinden çok daha yavaştır.^[12]

Tür seçerken kısa bir kural: sayaçlar için int yeter, 0-255 arasındaki değerler byte'a sığar, açık ya da kapalı bilgisi bool'da tutulur. millis() fonksiyonu, kartın programı çalıştırmaya başlamasından bu yana geçen milisaniyeyi unsigned long türünde döndürür; bu değerle int gibi daha küçük bir türde işlem yapmak mantık hatalarına yol açar.^[13]

Tam sayı bölmesi, kalan ve taşma

İki tam sayı bölündüğünde sonuç da tam sayıdır ve küsurat atılır. Arduino'nun float sayfasındaki örnekte $x = 1$ iken $y = x / 2$ işleminin sonucu 0'dır; 0,5 elde etmek için (float)x / 2.0 yazmak gerekir.^[12] Kayan noktalı işlem için sayıya ondalık nokta eklenir (2 yerine 2.0); eklenmezse sayı int olarak işlenir.^[12]

İşlenenlerden biri float ise kayan noktalı işlem yapılır; ama sonuç int bir değişkene atanırsa yalnız tam kısmı saklanır. 55.5 / 6.6 işleminin sonucu yaklaşık 8,409'dur; int değişkende 8 kalır.^[14] Float'tan int'e çevirmede yuvarlama değil kesme yapılır: 2.9 int'e atanınca 2 olur. Yuvarlamak için 0.5 eklenir ya da round() kullanılır.^[12]

Kalan işleci %, bir tam sayının ötekine bölümünden kalanı verir: $7 \% 5 = 2$, $9 \% 5 = 4$, $5 \% 5 = 0$. Bir değişkeni belli bir aralıkta tutmak için kullanışlıdır; $i = (i + 1) \% 10$ ifadesi i'yi 0 ile 9 arasında döndürür. % float sayılarla çalışmaz ve ilk sayı negatifse sonuç da negatif çıkar: $-4 \% 5 = -4$.^[15]

Taşma. Bir değişken türünün sınırını aşarsa taşar (overflow). İşaretli türlerde taşmanın sonucu öngörülemez ve bundan kaçınılmalıdır; tipik belirti değişkenin en büyük değerinden en küçüğüne "dönmesidir", ama her zaman böyle olmaz.^[5] İşaretsiz türlerde ise değer sınırı aşınca 0'a döner, 0'ın altına inince en büyük değere çıkar: unsigned int x = 0 iken $x = x - 1$ işleminden sonra x 65.535 olur.^[9]

Örnek 1 (bir saat kaç milisaniye?). Bir saat $60 \times 60 \times 1000 = 3.600.000$ milisaniyedir. Bu sayı 16 bitlik int'in en büyük değeri olan 32.767'yi çok aşar.^[5] Arduino'nun belgesine göre tam sayılarla işlem yaparken değerlerden en az biri long olmalıdır; bunun için sayının sonuna L eklenir ya da long türünde bir değişken kullanılır.^[10]

```
unsigned long yanlis = 60 * 60 * 1000;    // bütün sayılar int: işlem taşar
unsigned long dogru  = 60L * 60 * 1000;    // 60L long: sonuç 3600000
```

Örnek 2 (ortalama alırken taşma). Bir sensörü birkaç kez okuyup ortalamasını almak yaygın bir yöntemdir. UNO'da analogRead 0 ile 1023 arasında değer döndürür.^[16] int bir toplamda en büyük değer olan 1023 en çok $32.767 / 1023 \approx 32$ kez toplanabilir: $32 \times 1023 = 32.736$ sınırın altındadır, $33 \times 1023 = 33.759$ ise sınırı aşar. Daha çok okuma toplanacaksa toplam long türünde tutulur.

Sabitler ve kapsam

const anahtar sözcüğü değişkeni salt okunur yapar: değişken, kendi türündeki öteki değişkenler gibi kullanılabilir ama değeri değiştirilemez. const değişkene değer atamaya çalışmak derleme hatası verir.^[17] Arduino'nun belgesine göre sabit tanımlamak için const, #define'a tercih edilir.^[17] Pin numaraları program boyunca değişmediği için genellikle const ile tanımlanır:

```
const int ledPin = 13;
const float pi = 3.14;
// pi = 7;    // derleme hatası: sabite değer atanamaz
```

Kapsam (scope), bir değişkenin programın hangi bölümünden görülebildiğidir.^[18]

- Bir fonksiyonun dışında, örneğin setup() ve loop()'un üstünde tanımlanan değişken **globaldir**; her fonksiyon onu görür.^[18]
- Bir fonksiyonun içinde tanımlanan değişken **yereldir**; yalnız o fonksiyonda görülür. for döngüsünün başlığında tanımlanan değişken ise yalnız döngünün süslü parantezleri içinde kullanılabilir.^[18]

- Yerel değişkenler, program büyüdükçe bir fonksiyonun başka bir fonksiyonun değişkenini yanlışlıkla değiştirmesini önler.^[18]
- Yerel değişkenler fonksiyon her çağrıldığında yeniden oluşturulur ve fonksiyon bitince yok olur. Değerini çağrılar arasında koruması gereken yerel değişken **static** ile tanımlanır; static değişken yalnız fonksiyon ilk çağrıldığında oluşturulur ve ilk değerini alır.^[19]

Son madde sık bir hatanın nedenidir: loop() içinde int sayac = 0; diye tanımlanan sayaç her turda yeniden 0'dan başlar. Sayacın artarak sürmesi için ya global tanımlanır ya da static int sayac = 0; yazılır.

Diziler

Dizi (array), bir indis numarasıyla erişilen değişkenler topluluğudur. Arduino'nun belgesine göre dizi şu biçimlerde tanımlanabilir:^[20]

```
int degerler[6];           // 6 elemanlı, ilk değer verilmemiş
int pinler[] = {2, 4, 8, 3, 6, 4}; // boyutu derleyici sayar: 6
int olcumler[5] = {2, 4, -8, 3, 2}; // 5 elemanlı, ilk değerli
char mesaj[6] = "hello";   // 5 harf ve 1 sonlandırma karakteri
```

- Diziler **0'dan numaralanır**: yukarıdaki olcumler dizisinde olcumler[0] 2'ye, olcumler[1] 4'e eşittir. 10 elemanlı bir dizide son eleman 9 numaralı indistedir.^[20]
- char dizisinde, metnin sonundaki sonlandırma (null) karakteri için bir eleman fazladan yer ayrılır.^[20]
- C++ derleyicisi dizinin sınırları dışına çıkılıp çıkılmadığını denetlemez. 10 elemanlı dizide [10] geçersizdir ve başka amaçla kullanılan bellekten rastgele bir değer okur; böyle bir yere yazmak programın çökmesine ya da hatalı çalışmasına yol açabilir ve bu hatayı bulmak zordur.^[20]
- Diziler çoğunlukla for döngüsüyle işlenir: döngü sayacı, dizinin indisi olarak kullanılır.^[20]

Koşullar: if, else if ve else

if bir koşulu sınar; koşul doğruysa ardından gelen komutları çalıştırır. Koşul, doğru (true) ya da yanlış (false) olabilen bir ifadedir ve karşılaştırma işleçleriyle yazılır.^[21]

İşleç	Anlamı
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 ya da eşit
x >= y	x, y'den büyük ya da eşit

Tablodaki işleçler Arduino'nun if sayfasından alınmıştır.^[21]

En sık hata, karşılaştırma için tek eşittir kullanmaktır. if (x = 10) yazıldığında x'e 10 atanır; 10 sıfırdan farklı olduğu için koşul her zaman doğru sayılır. Karşılaştırma için çift eşittir yazılır: if (x == 10).^[21] if'ten sonra süslü parantez konmazsa yalnız ilk komut koşula bağlı olur; birden çok komut için süslü parantez gerekir.^[21]

else if ve else. else, if'teki koşul yanlış çıktığında çalışır. else'ten sonra yeni bir if gelebilir; koşullar sırayla denir, ilk doğru koşulun bloğu çalışır ve program bütün if/else yapısının sonrasına geçer. Hiçbir koşul doğru değilse, varsa son else bloğu çalışır.^[22] Arduino'nun belgesindeki sıcaklık örneği:^[22]

```
if (sicaklik >= 70) {  
    // Tehlike: sistemi kapat  
}  
else if (sicaklik >= 60) {    // 60 <= sicaklik < 70  
    // Uyarı: kullanıcının dikkati gerekli  
}  
else {                      // sicaklik < 60  
    // Güvenli: olağan işlere devam  
}
```

Sıra önemlidir. Koşullar ters yazılıysaydı (önce >= 60, sonra >= 70), 75 derecede ilk koşul doğru çıkacağı için yalnız uyarı bloğu çalışır, tehlike bloğuna hiç ulaşılmazdı.

Birden çok koşul. İki koşulun ikisinin de doğru olması isteniyorsa mantıksal VE işleci && kullanılır: if (digitalRead(2) == HIGH && digitalRead(3) == HIGH) yalnız iki anahtar da HIGH okunduğunda çalışır. Mantıksal VE (&&), bit düzeyindeki VE işleci & ile karıştırılmamalıdır.^[23] VE, VEYA ve DEĞİL işlemlerinin mantığını [Mantık Kapıları ve Boole Cebiri](#) yazısında anlattık.

switch case

switch case de if gibi programın akışını yönetir: bir değişkenin değerini case satırlarındaki değerlerle karşılaştırır ve eşleşen durumun kodunu çalıştırır. break, switch yapısından çıkar ve genellikle her durumun sonuna konur. break konmazsa program, bir break'e ya da switch'in sonuna ulaşana kadar sonraki durumların kodunu da çalıştırır.^[24]

- Karşılaştırılan değişken byte, char, int, long gibi bir tam sayı türünde olmalıdır; case değerleri sabittir. Yalnız iki durum varsa bool da kullanılabilir; negatif değerler de kullanılabilir.^[24]
- default, hiçbir durum eşleşmediğinde çalışır ve isteğe bağlıdır.^[24]

```
switch (mod) {  
    case 1:  
        digitalWrite(ledPin, HIGH);    // mod 1: LED yanık  
        break;  
    case 2:  
        digitalWrite(ledPin, LOW);     // mod 2: LED sönmüş  
        break;  
    default:  
        Serial.println("Bilinmeyen mod");  
        break;  
}
```

Döngüler: for, while ve do...while

for süslü parantez içindeki bir komut bloğunu tekrarlamak için kullanılır; döngüyü yürütüp bitirmek için genellikle bir sayaç artırılır.^[25] Başlığı üç parçalıdır:

```
for (başlangıç; koşul; artırma) {  
    // komutlar  
}
```

- Başlangıç önce ve yalnız bir kez çalışır.^[25]
- Koşul her turda sınanır; doğruysa blok, ardından artırma çalışır ve koşul yeniden sınanır. Koşul yanlış olunca döngü biter.^[25]

Örnek 3 (LED'i yavaşça parlaklaştırmak). Arduino'nun for sayfasındaki örnek, 10. pine bağlı LED'in parlaklığını 0'dan 255'e artırır.^[25]

```
const int pwmPin = 10;  
  
void loop() {  
    for (int i = 0; i <= 255; i++) {  
        analogWrite(pwmPin, i);  
        delay(10);  
    }  
}
```

Döngü $i = 0$ 'dan $i = 255$ 'e kadar 256 kez döner; her turda 10 ms beklendiği için bir parlaklaşma yaklaşık $256 \times 10 = 2.560$ ms, yani 2,56 saniye sürer. $i++$ ifadesi i 'yi 1 artırır.

while parantez içindeki koşul yanlış olana kadar döner. Koşuldaki değişkeni bir şeyin değiştirmesi gerekir, yoksa döngü hiç bitmez; bu değişiklik koddaki bir sayaç da olabilir, bir sensörün okunması gibi dış bir koşul da.^[26]

```
int sayi = 1;  
while (sayi < 100) {  
    sayi = sayi * 2;    // 2, 4, 8, ..., 128  
}  
// döngü 7 tur döner ve sayi = 128 olur
```

do...while while gibi çalışır; tek farkı koşulun döngünün sonunda sınanmasıdır. Bu yüzden do...while bloğu en az bir kez çalışır.^[27]

```
int i = 0;  
do {  
    delay(50);          // sensörlerin oturmasını bekle  
    i++;  
} while (i < 100);    // 100 kez tekrarla
```

break, for, while ya da do...while döngüsünden normal koşulu beklemeden çıkmak için kullanılır; switch case'ten çıkmak için de aynı sözcük kullanılır. Örneğin bir sensör eşik değerini aştığında döngü break ile yarıda kesilir.^[28]

Döngü	Koşul ne zaman sınanır?	Tipik kullanım
for	Her turun başında	Tekrar sayısı belli işler
while	Her turun başında	Bir koşul değişene kadar sürme
do...while	Her turun sonunda	En az bir kez yapılması gereken işler

Tablodaki sınama zamanları Arduino'nun for, while ve do...while sayfalarından alınmıştır. ^[25, 26, 27] loop() zaten art arda tekrar ettiği için her turda bir kez yapılacak bir iş için ayrıca döngü yazmak gerekmez. ^[3]

Fonksiyon yazmak

Fonksiyon, belirli bir işi yapan ve iş bitince çağrıldığı yere dönen kod parçasıdır. Fonksiyon yazmanın tipik nedeni, aynı işin programda birkaç kez yapılmasıdır. Arduino'daki fonksiyonlar, BASIC gibi dillerdeki alt yordamların (subroutine) işini görür ve genişletir. ^[1] Arduino'nun belgesine göre fonksiyonlar programcının düzenli kalmasını sağlar, bir işi tek yerde topladığı için o işin bir kez düşünülüp bir kez hata ayıklanmasına imkân verir, kodu yeniden kullanılır kılarak sketch'i küçültür ve genellikle okunabilirliği artırır. ^[1]

Fonksiyon, setup() ve loop() gibi başka bir fonksiyonun içinde değil dışında tanımlanır; loop()'un üstünde ya da altında olabilir. ^[1]

```
int carp(int x, int y) {    // dönüş türü, ad, parametreler
    int sonuc;
    sonuc = x * y;
    return sonuc;          // sonucu çağırana döndürür
}

void loop() {
    int k = carp(2, 3);     // k = 6
}
```

Bu örnek, Arduino'nun belgesindeki iki sayıyı çarpan fonksiyonun Türkçe adlandırılmış hâlidir. ^[1] Fonksiyonun parçaları:

- **Dönüş türü** (int): fonksiyonun döndüreceği değerin türü. Değer döndürmeyen fonksiyonun dönüş türü **void**'dir. ^[4]
- **Parametreler** (int x, int y): fonksiyona çağırılırken verilen değerler. Fonksiyon, beklediği veri türündeki parametrelerle çağırılır. ^[1]
- **return**: fonksiyonu bitirir ve istenirse çağırana fonksiyona bir değer döndürür. ^[29]
- Parametre almayan ve değer döndürmeyen bir fonksiyon çağırılırken de parantezler ve noktalı virgül yazılır: ledYak(); gibi. ^[1]

Örnek 4 (sensörü beş kez okuyup ortalama almak). Arduino'nun belgesindeki ikinci örnek, analog 0 pinindeki sensörü beş kez okur, ortalamasını alır, değeri 4'e bölerek 0-255 aralığına indirir ve tersini (255'ten farkını) döndürür. ^[1]

```
int sensoru0ku() {
    int deger = 0;
```

```

for (int i = 0; i < 5; i++) {
    deger = deger + analogRead(A0);
}
deger = deger / 5;          // ortalama
deger = deger / 4;          // 0-255 aralığına indir
return 255 - deger;         // ters çevir
}

```

Beş okumanın hepsi 600 ise toplam 3.000, ortalama 600 olur; $600 / 4 = 150$ ve fonksiyon $255 - 150 = 105$ döndürür. Beş okumanın toplamı en çok $5 \times 1023 = 5.115$ olduğu için int'e sığar.

Örnek 5 (fonksiyonda taşma). `int kare(int x) { return x * x; }` fonksiyonu 181 için 32.761 döndürür. 182 için sonuç 33.124 olmalıdır, ama bu değer int'in sınırı olan 32.767'yi aştığı için taşar.^[5] Büyük değerler için dönüş türü ve parametre long yapılır: `long kare(long x)`.

Hata ayıklama: seri monitör

Hata ayıklama (debugging), programın beklenenden farklı davranmasının nedenini bulup gidermektir. Hatalar iki türdür. Bir kısmını derleyici yakalar; örneğin `const` bir değişkene değer atamak derleme hatası verir.^[17] Ötekiler derlenir ve çalışır ama yanlış sonuç verir; if ($x = 10$) bunun tipik örneğidir.^[21] İkinci türü bulmanın en yaygın yolu seri monitördür: seri monitör hata ayıklamak, bir fikri denemek ve kartla doğrudan haberleşmek için kullanılır ve Arduino IDE 2'de editörün alt kısmında açılır.^[30]

1. `setup()` içinde **`Serial.begin(9600)`** ile seri iletişim başlatılır. 9600, saniyede aktarılacak en çok bit sayısını gösteren baud hızıdır.^[30] Seri monitörle haberleşirken seri monitörün hız menüsünde bulunan değerlerden biri kullanılır.^[31] Seri monitörün hız menüsünde de programdakiyle aynı değer seçilmelidir.
2. Programın önemli noktalarında **`Serial.print`** ve **`Serial.println`** ile değişkenlerin değeri yazdırılır. `Serial.print` veriyi insanın okuyabileceği metin olarak gönderir; `Serial.println` aynısını yapar ve sonuna satır başı ile yeni satır karakteri ekler.^[32, 33]
3. Değerin yanına ne olduğunu anlatan bir etiket yazdırmak, seri monitördeki satırları ayırt etmeyi kolaylaştırır.

```

void loop() {
    int ham = analogRead(A0);
    int parlaklik = ham / 4;
    Serial.print("ham: ");
    Serial.print(ham);
    Serial.print(" parlaklik: ");
    Serial.println(parlaklik);
    analogWrite(9, parlaklik);
    delay(200);
}

```

Yazdırma biçimiyle ilgili iki ayrıntı:

- Ondalıkli sayılar varsayılan olarak iki basamakla yazılır: `Serial.print(1.23456) "1.23"`, `Serial.print(1.23456, 4) "1.2346"` yazar.^[32]

- Tam sayılarda ikinci parametre tabanı seçer: `Serial.print(78, BIN)` "1001110", `Serial.print(78, HEX)` "4E", `Serial.print(78, OCT)` "116" yazar.^[32]

Kodun bir bölümünü devre dışı bırakmak. `return`, hatalı olabilecek uzun bir bölümü yorum satırına çevirmeden denemek için de kullanılabilir: `loop()` içindeki `return`; satırından sonraki kod hiç çalışmaz.^[29]

Hepsi bir arada: sırayla yanan LED'ler

Aşağıdaki program dizi, sabit, döngü ve fonksiyonu birlikte kullanır: 3, 5, 6 ve 9. pinlere bağlı dört LED'i sırayla 200 ms yakar. LED'lerin her biri, [önceki yazıda](#) anlattığımız gibi seri bir dirençle bağlanır.

```
const int ledler[] = {3, 5, 6, 9};    // 4 LED, 4 pin
const int ledSayisi = 4;

void setup() {
    for (int i = 0; i < ledSayisi; i++) {
        pinMode(ledler[i], OUTPUT);    // her pini çıkış yap
    }
}

void ledYak(int pin, int sure) {       // değer döndürmez: void
    digitalWrite(pin, HIGH);
    delay(sure);
    digitalWrite(pin, LOW);
}

void loop() {
    for (int i = 0; i < ledSayisi; i++) {
        ledYak(ledler[i], 200);        // LED'ler sırayla
    }
}
```

- Pin dizisi sabit olarak tanımlanmıştır; dizi sabit olacaksa `#define` değil `const` kullanılır.^[17]
- Döngü sayacı `i`, dizinin indisi olarak kullanılır; `i < ledSayisi` koşulu `i`'yi 0 ile 3 arasında tutar ve `ledler[4]`'e hiç erişilmez.^[20]
- `ledYak` bir değer döndürmediği için `void` ile tanımlanmıştır.^[4] Bir tur $4 \times 200 = 800$ ms sürer.

Öğrencilerin sık yaptığı hatalar

1. **Karşılaştırmada tek eşittir.** `if (x = 10)` koşulu her zaman doğrudur ve `x`'in değerini de değiştirir; doğrusu `if (x == 10)`.^[21]
2. **switch'te break'i unutmak.** `break` konmayan durumdan sonra sonraki durumların kodu da çalışır.^[24]
3. **Dizinin dışına çıkmak.** 5 elemanlı dizide geçerli indisler 0-4'tür; `for (int i = 0; i <= 5; i++)` döngüsü olmayan [5] elemanına erişir ve derleyici bunu yakalamaz.^[20]
4. **Tam sayı bölmesini unutmak.** $1 / 2$ işleminin sonucu 0'dır; ondalıklı sonuç için 2.0 yazılır.^[12]

5. **Taşmayı gözden kaçırmak.** int UNO'da 32.767'de biter; büyük toplamalar ve çarpımlar için long, millis() için unsigned long kullanılır. ^[5, 13]
6. **byte sayaçla 255'e kadar saymak.** byte 0-255 aralığındadır ve işaretsiz türler sınırı aşınca 0'a döner; bu yüzden for (byte i = 0; i <= 255; i++) döngüsü hiç bitmez. ^[7, 9]
7. **Sonsuz while döngüsü.** Koşuldaki değişken döngü içinde değişmezse while hiç bitmez. ^[26]
8. **Yerel sayacın sıfırlanması.** loop() içinde tanımlanan yerel değişken her çağrıda yeniden oluşturulur; değerini koruması için global ya da static tanımlanır. ^[19]
9. **Fonksiyonu başka bir fonksiyonun içinde tanımlamak.** Fonksiyonlar setup() ve loop()'un dışında yazılır. ^[1]
10. **Serial.begin'i unutmak.** Kartın bilgisayarla seri haberleşmesi için setup() içinde Serial.begin ile baud hızı ayarlanır. ^[30]

Alıştırmalar

1. int a = 7 / 2; float b = 7 / 2; float c = 7 / 2.0; satırlarından sonra a, b ve c'nin değeri nedir?
2. unsigned int x = 65535; x = x + 1; satırlarından sonra x kaç olur?
3. for (int i = 2; i < 11; i = i + 3) döngüsü kaç kez döner, i hangi değerleri alır?
4. Bir switch yapısında case 1 "A", case 2 "B", case 3 "C" yazdırıyor; yalnız case 3'ün sonunda break var, default ise "D" yazdırıyor. mod = 2 iken seri monitöre ne yazılır?
5. Sıcaklık programında önce if (sicaklik >= 60) uyarı, sonra else if (sicaklik >= 70) tehlike yazılmış. sicaklik = 75 iken hangi blok çalışır? Hata nerede?
6. int bir toplam değişkeninde analogRead'in en büyük değeri olan 1023 en çok kaç kez taşmadan toplanabilir?
7. int kare(int x) { return x * x; } fonksiyonu UNO'da en büyük hangi pozitif x için doğru sonuç verir?
8. millis() değeri unsigned long'un sınırı olan 4.294.967.295 milisaniyeyi aşınca sıfıra döner. Bu yaklaşık kaç gündür?
9. int pinler[] = {3, 5, 6, 9, 10, 11}; dizisinin son elemanının indisi nedir? pinler[6] okunursa ne olur?
10. int buton = digitalRead(2); if (buton == HIGH) digitalWrite(13, HIGH); else digitalWrite(13, LOW); programında LED, butona basılmasa da yanık kalıyor. Neden?

Cevaplar

1. a = **3** (tam sayı bölmesi). b = **3.0**: 7 / 2 önce tam sayı olarak 3 hesaplanır, sonra float'a atanır. c = **3.5**: 2.0 ondalıklı olduğu için kayan noktalı bölme yapılır. ^[12]
2. x = **0**. İşaretsiz türler en büyük değeri aşınca 0'a döner. ^[9]
3. **3 kez**; i = 2, 5, 8. i = 11 olunca koşul (11 < 11) yanlış olur ve döngü biter. ^[25]
4. **BC**. mod = 2 eşleşir ve "B" yazılır; case 2'nin sonunda break olmadığı için program case 3'e geçip "C" yazar ve oradaki break'te switch'ten çıkar. ^[24]
5. **Uyarı bloğu** çalışır: 75 >= 60 doğru olduğu için ilk koşulun bloğu seçilir ve kalan koşullar denenmez. ^[22] Tehlike bloğuna hiç ulaşılmaz; daha dar koşul (>= 70) önce yazılmalıdır.
6. 32.767 / 1023 = 32,03 olduğu için en çok **32** kez (32 × 1023 = 32.736). 33 okuma 33.759 eder ve int'in sınırını aşar. ^[5]
7. **181**. 181 × 181 = 32.761, sınırın altındadır; 182 × 182 = 33.124 ise 32.767'yi aşar. ^[5]

8. $4.294.967.295 / 1000 / 86.400 \approx 49,7$ gün; Arduino'nun belgesi bunu yaklaşık 50 gün olarak verir. ^[13]
9. Dizinin 6 elemanı vardır, son indis 5'tir. pinler[6] geçersizdir; başka amaçla kullanılan bellekten rastgele bir değer okunur ve derleyici bunu denetlemez. ^[20]
10. Koşulda tek eşittir var: buton = HIGH, değişkene HIGH değerini atar ve koşul her zaman doğru olur. Doğrusu if (buton == HIGH). ^[21]

Terim tablosu

Türkçe	İngilizce	Açıklama
Değişken	Variable	Adı olan, değer saklayan bellek alanı
Veri türü	Data type	Değişkenin boyutu ve değer aralığı
Sabit	Constant (const)	Değeri değiştirilemeyen değişken
Dizi	Array	İndisle erişilen değişkenler topluluğu
İndis	Index	Dizideki elemanın 0'dan başlayan numarası
Kapsam	Scope	Değişkenin görülebildiği program bölümü
Global değişken	Global variable	Her fonksiyonun gördüğü değişken
Yerel değişken	Local variable	Yalnız tanımlandığı fonksiyonda görülen değişken
Koşul	Condition	Doğru ya da yanlış olan ifade
Döngü	Loop	Tekrarlanan komut bloğu
Fonksiyon	Function	Belirli bir işi yapan, çağrılabilen kod parçası
Parametre	Parameter	Fonksiyona çağrılırken verilen değer
Dönüş değeri	Return value	Fonksiyonun çağırana verdiği sonuç
Taşma	Overflow	Değerin, türünün sınırını aşması
Hata ayıklama	Debugging	Hatanın nedenini bulup giderme
Baud hızı	Baud rate	Seri iletişimde saniyedeki bit sayısı

Sonuç

Arduino programlamanın temeli birkaç kurala dayanır: doğru veri türünü seçmek (UNO'da int 16 bittir), sabitleri const ile tanımlamak, dizilerin 0'dan başladığını unutmamak, karşılaştırmada == kullanmak, döngünün nasıl biteceğini bilmek ve tekrarlanan işi fonksiyona taşımak. Bu kurallar giriş ve çıkış fonksiyonlarıyla birleşince butonla mod değiştiren, sensörü okuyup ortalama alan ya da LED'leri sırayla yakan programlar yazılabilir. Program beklendiği gibi çalışmadığında ilk adım, değerleri seri monitöre yazdırıp neyin nerede değiştiğini görmektir.

Özet

- Veri türü**, değişkenin bellekte kaç bayt yer kaplayacağını ve hangi aralıkta değer alacağını belirler. Arduino UNO'da int 16 bittir ve -32.768 ile 32.767 arasında değer alır; daha büyük sayılar için long, millis() süresi için unsigned long kullanılır.

- İki tam sayının bölümü tam sayıdır: $1 / 2$ işleminin sonucu 0'dır. Ondalık sonuç için sayılardan biri ondalıklı yazılır (2 yerine 2.0). Türünün sınırını aşan değişken taşar: işaretli türler 0'a döner, işaretli türlerde sonuç öngörülemez.
- **const** ile tanımlanan değişken salt okunurdur. Diziler 0'dan numaralanır: 10 elemanlı dizinin son elemanı [9]'dur ve derleyici dizinin dışına çıkılmasını denetlemez.
- **Koşul:** if, else if ve else sırayla denenir, ilk doğru koşulun bloğu çalışır. Karşılaştırmada == kullanılır; tek = atama yapar ve koşulu hep doğru kılar. switch case'te her durumun sonuna break konur.
- **Döngü ve fonksiyon:** for sayaçlı tekrar, while koşul doğru kaldıkça tekrar içindir; do...while en az bir kez çalışır. Fonksiyon aynı işi tek yerde toplar ve setup() ile loop() dışında tanımlanır. Hata ayıklamada değerler Serial.println ile seri monitöre yazdırılır.

Sık sorulan sorular

Arduino'da değişken nasıl tanımlanır?

Değişken, önce veri türü sonra adı yazılarak tanımlanır; istenirse ilk değer de verilir: `int sayac = 0;` gibi. Bir fonksiyonun dışında tanımlanan değişken globaldir ve her fonksiyon onu görür. `setup()` ya da `loop()` içinde tanımlanan değişken yereldir ve yalnız o fonksiyonda kullanılabilir. Pin numarası gibi program boyunca değişmeyecek değerler `const` ile sabit olarak tanımlanır.

Arduino UNO'da int kaç bit, hangi değerleri alır?

Arduino UNO ve öteki ATmega tabanlı kartlarda int 16 bittir (2 bayt) ve -32.768 ile 32.767 arasında değer alır. Negatif değer gerekmiyorsa unsigned int 0 ile 65.535 arasını tutar. Daha büyük sayılar için 32 bitlik long (-2.147.483.648 ile 2.147.483.647) ya da unsigned long (0 ile 4.294.967.295) kullanılır. Due ve SAMD tabanlı kartlarda ise int 32 bittir.

Arduino'da if ile switch case arasındaki fark nedir?

if, doğru ya da yanlış olabilen herhangi bir koşulu sınar ($x > 120$ gibi); else if ile birbirini dışlayan koşullar sıralanır ve ilk doğru koşulun bloğu çalışır. switch case ise tek bir tam sayı değişkenini sabit değerlerle karşılaştırır ve eşleşen durumun kodunu çalıştırır. switch'te bir durumun sonuna break konmazsa program sonraki durumların kodunu da çalıştırır.

for ile while döngüsü arasındaki fark nedir?

for döngüsü başlangıç, koşul ve artırma adımlarını tek satırda toplar; tekrar sayısı belli olan işler (bir dizinin her elemanı, 0'dan 255'e parlaklık) için uygundur. while yalnız koşulu sınar ve koşul yanlış olana kadar sürer; koşuldaki değişkeni bir şey değiştirmezse döngü hiç bitmez. do...while koşulu sonda sınıdığı için en az bir kez çalışır.

Arduino'da fonksiyon nasıl yazılır?

Fonksiyon `setup()` ve `loop()` dışında yazılır: önce dönüş türü, sonra adı, parantez içinde parametreleri ve süslü parantez içinde gövdesi. Değer döndüren fonksiyon sonucu `return` ile verir; örneğin `int carp(int x, int y) { return x * y; }` çağrıldığı yere iki sayının çarpımını döndürür. Değer döndürmeyen fonksiyonun dönüş türü void'dir. Parametresiz bir fonksiyon çağrılırken de parantezler ve noktalı virgül yazılır.

Arduino'da hata ayıklama nasıl yapılır?

En yaygın yol, değişkenlerin değerini seri monitöre yazdırmaktır. `setup()` içinde `Serial.begin(9600)` ile seri iletişim başlatılır, programın önemli noktalarında `Serial.print` ve `Serial.println` ile değerler gönderilir ve Arduino IDE'nin seri monitörü açılır; seri monitörün hız menüsünde de aynı değer (9600) seçilir. Böylece bir koşulun neden çalışmadığı ya da bir döngünün kaç kez döndüğü görülür.

Kaynaklar

- 1 Arduino (2022). [Using Functions in a Sketch](#). Arduino Documentation.
- 2 Arduino (2024). [setup\(\)](#). Arduino Language Reference.
- 3 Arduino (2024). [loop\(\)](#). Arduino Language Reference.
- 4 Arduino (2024). [void](#). Arduino Language Reference.
- 5 Arduino (2024). [int](#). Arduino Language Reference.
- 6 Arduino (2024). [bool](#). Arduino Language Reference.
- 7 Arduino (2024). [byte](#). Arduino Language Reference.
- 8 Arduino (2024). [char](#). Arduino Language Reference.
- 9 Arduino (2024). [unsigned int](#). Arduino Language Reference.
- 10 Arduino (2024). [long](#). Arduino Language Reference.
- 11 Arduino (2024). [unsigned long](#). Arduino Language Reference.
- 12 Arduino (2024). [float](#). Arduino Language Reference.
- 13 Arduino (2025). [millis\(\)](#). Arduino Language Reference.
- 14 Arduino (2024). [/ \(division\)](#). Arduino Language Reference.
- 15 Arduino (2024). [% \(remainder\)](#). Arduino Language Reference.
- 16 Arduino (2025). [analogRead\(\)](#). Arduino Language Reference.
- 17 Arduino (2024). [const](#). Arduino Language Reference.
- 18 Arduino (2024). [Variable Scope](#). Arduino Language Reference.
- 19 Arduino (2024). [static](#). Arduino Language Reference.
- 20 Arduino (2024). [array](#). Arduino Language Reference.
- 21 Arduino (2024). [if](#). Arduino Language Reference.
- 22 Arduino (2024). [if...else](#). Arduino Language Reference.
- 23 Arduino (2024). [&& \(logical and\)](#). Arduino Language Reference.
- 24 Arduino (2024). [switch...case](#). Arduino Language Reference.
- 25 Arduino (2024). [for](#). Arduino Language Reference.
- 26 Arduino (2024). [while](#). Arduino Language Reference.
- 27 Arduino (2024). [do...while](#). Arduino Language Reference.
- 28 Arduino (2024). [break](#). Arduino Language Reference.
- 29 Arduino (2024). [return](#). Arduino Language Reference.
- 30 Söderby, K. (2024). [Using the Serial Monitor tool](#). Arduino Documentation (Arduino IDE 2).
- 31 Arduino (2025). [Serial.begin\(\)](#). Arduino Language Reference.
- 32 Arduino (2025). [Serial.print\(\)](#). Arduino Language Reference.
- 33 Arduino (2025). [Serial.println\(\)](#). Arduino Language Reference.

Atıf

BÖTE Editör Ekibi. (2026, 4 Ekim). Arduino programlama: değişken, koşul, döngü ve fonksiyon. *BÖTE Blog*. <https://bote.web.tr/blog/arduino-degisken-kosul-dongu-fonksiyon>

Bu belge hakkında. Bu PDF, bote.web.tr'deki [Arduino Programlama: Değişken, Koşul, Döngü ve Fonksiyon](#) yazısının BT-070 kodlu belgesinin 1.0 sürümüdür.

Sürüm geçmişi ve değişiklikler: bote.web.tr/d/bt-070. Güncel sürümü ve bütün yazıların PDF'lerini bote.web.tr/pdf adresinde bulabilirsiniz. Elinizdeki dosyanın bizim yayımladığımız sürümle birebir aynı olup olmadığını bote.web.tr/d adresinde denetleyebilirsiniz; dosya cihazınızdan dışarı gönderilmez.

© 2026 bote.web.tr · Bilgisayar ve Öğretim Teknolojileri Eğitimi



PDF merkezi

BÖTE bölümü, eğitim fakültesi ve öğretmenlik üzerine ücretsiz, kaynaklı yazılar. Bu belgenin güncel hâli ve çok daha fazlası sitede.

Blog yazıları

Eğitim teknolojileri, kuramlar ve öğretmenlik

bote.web.tr/blog

PDF merkezi

Güncel PDF'ler, sürüm geçmişi ve doğrulama

bote.web.tr/pdf

BÖTE hakkında

Bölümün tarihi, amacı ve fakültedeki yeri

bote.web.tr/about

Müfredat

Dört yıllık ders programı ve dersler

bote.web.tr/curriculum

Mezunlar ve meslek

Mezunların çalıştığı alanlar ve unvanlar

bote.web.tr/graduation

Belge doğrulama

PDF'inizin güncel ve özgün olduğunu denetleyin

bote.web.tr/d



Üniversite topluluklarını, etkinlikleri ve şirketleri tek uygulamada buluşturan ücretsiz kampüs platformu.

- **Dijital portföy**
Katıldığınız etkinlikleri ve sertifikalarını uniconnectly.com/@kullaniciadi adresinde herkese açık sergile
- **Sertifikalar**
Platformda verilen sertifikalar doğrulanmış işaretli, edu.tr e-postan onaylı
- **Topluluklar ve etkinlikler**
Üniversitelerdeki toplulukları ve takvimi tek ekranda gör, başvurulmuş etkinliklere katıl
- **Doğrulanabilir katılım**
QR ile giriş yaptığınız etkinlikler portföyünde doğrulanmış listelenir
- **Anlık fırsatlar**
Staj, iş ve burs duyuruları akışına düşer
- **Üye kartı indirimleri**
Takip ettiğiniz toplulukların anlaşmalı işletme indirimlerinden yararlan



Ücretsiz keşfet

QR kodu okutun ya da uniconnectly.com adresine girin.



Diğer yapımlarımız

UniConnectly gibi bunlar da bizim yapımlarımız. Bu belgeyi ücretsiz sunuyoruz; yapımlarımızı keşfetmeniz ve çevrenizle paylaşmanız, bu çalışmalarını sürdürmemiz için en büyük destektir.

ahmetcelen.com.tr

Üniversite ve kamu sınavlarına ücretsiz matematik: konu anlatımları ve PDF'leri, çıkmış sorular ve sınav geri sayımları.

ahmetcelen.com.tr



Hayvanseverlerin sosyal medyası ve veteriner akıllı klinik yönetim uygulaması.

veterito.com