Arduino Kod yapısı ve Programlama Dili

Arduino programlama dili C ve Java dillerinden türetilmiş olan ve C dilinin temel özelliklerini taşıyan bir dildir. Burada Arduino IDE metinsel kod editörü ile Arduino programlama dilinde kullanılan belirli kodlardan bahsedilecektir.

Arduino kod blok yapısı

#include

Arduino kod yapısında ilk önce kullanılacak kütüphaneler belirtilir. Kütüphane yüklemek için ***include*** (dahil et) komutu kullanılır. Kullanılacak kütüphanenin ismi ya tırnak içinde **" "** ya da **< >** arasında belirtilmelidir.

#include <kütüphane_ismi.h> veya *#include "kütüphane_ismi.h"*

#define

Ön işlemci komutudur. Değişken ismi gibi davranır.

#define ledpin 13 // ledpin ismi görülen yerde 13 rakamını kullanır.

Açıklama satırları

C dilindeki gibi açıklama satırları kullanılır. Açıklama satırları programdaki önemli hususları not etmek, programa geri dönüldüğünde hatırlatma yapmak veya programı bir başka kişi okuyacaksa programı okuyacak başka bir kişiye bilgilendirme yapmak amacıyla kullanılır.
Tek satırdan oluşan bir açıklama satırı için

```
// (çift slaş)
```

kullanılır. Birden fazla satırlık açıklama yapılması gerekiyorsa o zaman

```
/* (slaş + asterix)

*/ (asterix + slaş)
```

şeklinde kullanım yapılmalıdır.

Örnek:
Tek satırlık açıklama satırı için örnek:

```
// x değişkeni sensörden gelen veri için kullanılmıştır
```

Birden fazla açıklama satırı için örnek:

```
/* bu uygulama
Flaşör uygulaması içermektedir
Bu uygulama sırayla yanıp sönen LED'ler için
kullanılan bir uygulamadır */
```

Sabitler

1- INPUT ve OUTPUT

INPUT, Arduino'ya veri girişi olacağı anlamındadır. Bu bir sensörden gelen veri için ya da bir button ile tetikleme yapmak için kullanılır. OUTPUT ise Arduino'nun veri çıkışı yapacağı anlamındadır. Bu bir LED' in enerjilendirilmesi olabilir.

Dikkat edilirse yapılacak olan çalışmaya göre bu kısımlar doğru şekilde kurulmalıdır. Örneğin bir HCSR04 sensöründen gelen veri INPUT olarak tanımlanmalıdır. Aksi halde yani çıkış olarak belirlenirse sensörden veri alınamaz.

2- HIGH ve LOW

Okuma ve yazma yapılırken kullanılırlar. HIGH olarak etiketlenen pine 5 V yani mantık 1 durumu uygulanır. LOW olarak etiketlenen pine 0 V yani mantık 0 durumu uygulanır.

kod blok yapısı

1- Void setup

Void yapısı dış dünyaya veri göndermeyen bir yapıdır. Fakat devamından anlaşılacağı üzere ***void setup***, Arduino ile yapılacak olan çalışmada "kurulumların" yapıldığı bölümdür. ***Bu bölüm her Arduino uygulamasında mutlaka bulunmak zorundadır!***

Bu bölümde Arduino'nun hangi pininin kullanılacağı ve bu pinin ***INPUT*** mu yoksa ***OUTPUT*** mu olarak kullanılacağı ve seri haberleşme kullanılacaksa ***Serial.begin*** ile seri haberleşmenin tanımlaması yapılır. Bu bölüm süslü parantezle açılır ***{*** ve kapanır ***}***.

Kod satırının bitişi ***;*** noktalı virgül ile belirtilir. Bu, kod satırının bittiği anlamına gelir.

Örnek:

```
void setup ()
{
      pinMode(13, OUTPUT);
      pinMode(7, INPUT);
      Serial.begin(9600);
}
```

Yukarıda verilen tanımlamada 13 nolu pin, ***OUTPUT*** ve 7 nolu pin ***INPUT*** olarak kurulmuştur. Ayrıca ***Serial.begin(9600)*** ile LCD ekran kullanılacağı belirtilmiştir. Burası kurulum bölümüdür. 7 ve 13 nolu pinler sırasıyla ***INPUT*** ve ***OUTPUT*** olarak belirlenmiştir ve seri haberleşme kullanılacağı belirtilmiştir.

2- Void loop

Mutlaka olması gereken diğer bir kod bloğudur. Bu bölümde Arduino'nun tekrarlı olarak gerçekleştireceği görevler yani asıl program belirtilir. Bu bölüm sonsuz bir döngüdür ve enerji olduğu müddetçe tekrarlanır. ***Bu bölüm her Arduino uygulamasında mutlaka bulunmak zorundadır!***

Bu bölümde yazılacak olan kodlar, yukarıdan aşağıya doğru okunur. Son kod satırı okunup gerçekleştirildikten sonra başa döner ve tekrar tekrar sonsuz döngü şekilde burada belirtilen işlemler gerçekleştirilir. Bu bölüm süslü parantezle açılır *{* ve kapanır *}*. Kod satırının bitişi *;* noktalı virgül ile belirtilir.

Örnek:

```
void loop ()
{
      digitalWrite(13, HIGH);
      delay(1000);
      digitalWrite(13, LOW);
      delay(1000);
}
```

Yukarıda verilen örnekte 13 nolu pin HIGH yapılmaktadır yani 5 V' luk bir gerilim uygulanmaktadır. Bu işlem 1000 ms yani 1 saniye gerçekleştirilmektedir. Daha sonra aynı pin, LOW yapılmakta yani bu pine uygulanan gerilim 0 V seviyesine çekilmektedir ve bu işlem de 1000 ms yani 1 s gerçekleştirilmektedir.

Sonsuz bir döngü ile ***1000 ms 13 nolu pine 5 V*** ve ***1000 ms 13 nolu pini 0 V*** seviyesinde tutmaktadır.

Yukarıda parça parça verilerek açıklanan programın tamamı aşağıdaki gibidir:

```
/*
Bu uygulamada kütüphane kullanmaya gerek yoktur.
O yüzden #include ile kütüphane dahil edilmemiştir.
*/

void setup ()
{
      pinMode(13, OUTPUT);
      pinMode(7, INPUT);
      Serial.begin(9600);  // aşağıda açıklanacaktır
}

void loop ()
{
      digitalWrite(13, HIGH);
      delay(1000);
      digitalWrite(13, LOW);
      delay(1000);
}
```

Arduino komutları

I- Digital komutlar

1- pinMode

Bu komut, Arduino pinlerinin durumunu yani pinin giriş (INPUT) ya da çıkış (OUTPUT) olarak belirlenmesini sağlar. Örneğin bir sensörden veri alınacaksa ya da button ile tetikleme yapılacaksa INPUT olarak seçilir ya da örneğin bir LED'in enerjilendirilmesi sağlanacaksa OUTPUT olarak belirlenir.

Bu komut pin numarası verildikten sonra INPUT ya da OUTPUT parametrelerini alır.

Örnek:

```
pinMode(13, INPUT)  // 13 nolu pin giriş pini olarak ayarlanır
pinMode(13, OUTPUT) // 13 nolu pin çıkış pini olarak ayarlanır
```

2- digitalWrite

OUTPUT yani çıkış olarak belirtilen pine enerji göndermek ya da gönderilen enerjiyi kesmek amacıyla kullanılır. Pin numarası belirtildikten sonra ***HIGH (yüksek)*** ya da ***LOW (düşük)*** parametrelerini alır.

Digital olarak ikilik sayı sistemi kullanıldığından 0 ve 1 mantık durumları vardır. 0 mantık durumu LOW ve 1 mantık durumu HIGH durumudur. LOW durumunda pine verilen enerji kesilir ve HIGH durumunda pine enerji verilir.

3- digitalRead

Giriş olarak tanımlanan pin üzerinden okuma yapılmasını sağlar. Bu bir sensör ya da button olduğundan bir değişkene atanarak sensörden gelen veri değişkenin değeri olarak kullanılabilir.

II- Geciktirme komutları

1- delay

İki kod arasında belirli bir süre duraklama olması amacıyla kullanılır. ***delay*** kelimesi ***gecikme*** anlamındadır ve ms (milisaniye) birimiyle kullanılır. Örneğin iki komut arasında 2 saniyelik bir gecikme olması isteniyorsa ***delay(2000)*** yazılmalıdır. Böylece ***2000 ms*** yani ***2 saniyelik*** bir gecikme sağlanmış olur.

Örnek:

```
digitalWrite(13, HIGH) // 13 nolu pine 5 V gönderir
delay(1000)
digitalWrite(13, LOW) // 13 nolu pinin gerilimini 0 V yapar
delay(1000)
```

2- delayMicroseconds

Programı içine sayı olarak yazılan parametre değeri kadar fakat mikrosaniye yani saniyenin milyonda milisaniyenin binde biri oranında duraklatır.

Örnek:

```
delayMicroseconds(50);       // pauses for 50 microseconds
```

3- *unsigned long millis*

Arduino programı yüklenilip enerji verildikten sonra süre saymaya başlayan komuttur. Maksimum 50 gün sayılabilir bu süre sonunda sıfırlanır.

III- Analog komutlar

1- analogWrite

Çıkış olarak ayarlanan analog pine PWM sinyali gönderilmesini sağlar. Örneğin bir motor için m olarak tanımlanan pine 100 PWM değerinin atanmasını sağlar.

Örnek:

```
analogWrite(m, 100)  // m ile tanımlı motora 100 PWM değeri gönderir
```

2- analogRead

Arduino UNO' da 6 adet 10 bitlik ADC vardır. Yani analog giriş digitale çevrilerek okunur. Analog pine bağlı olan ve giriş pini olarak tanımlanmış potansiyometre gibi giriş elemanlarının 0 ile 5 volt arasındaki gerilimlerini ölçmemize olanak tanır.

Analog okunan değer ADC (Analog Digital Converter) ile 1024 farklı değerde (0 ile 1023 arası) bir ikilik sayıya dönüştürülür. 0 V → 0 ve 5 V →1023' e karşı gelir. Dolayısıyla birim başına 0.0049 V yani 4.9 mV düşmektedir.

Örnek:

A1 analog pinine bağlı olan potansiyometrenin gerilimini ***potDeger*** adlı değişkene atayalım. Bu değer 0 ile 1024 arasındadır.

```
potDeger = analogRead(A1);
```

IV- Seri haberleşme komutları

1- Serial.begin

Bilgisayar ile Arduino arasında seri iletişimi başlatmak için void setup() fonksiyonu içinde tanımlanması gereken bir komuttur. 9600 değeri alınan ve gönderilen veri hızını belirler bazı sensör ya da haberleştirilen ekipman için bu değer değişebilir. Bu bilgiye sensörün datasheet (veri sayfasından) ulaşılmalıdır.

2- İnt Serial.available

Veri gelmeye başladığında seri portu okumaya açar.

3- Serial.print

Parametre olarak verilen değişkenin değerini seriport ekranına yazdırır.

V- Karşılaştırma Operatörleri

==	eşit eşit
!=	eşit değil
<	küçük
>	büyük
<=	küçük eşittir
>=	büyük eşittir

Genellikle if içerisinde karşılaştırma yaparken kullanılan operatörlerdir.

VI- Boolean Operatörleri

&& Ve

Her iki işlem de doğru ise if şartı sağlanır ve if içerisindeki komut çalıştırılır. Herhangi bir durum ya da ikisi de yanlış sonuç ise if yapısına girilmez.

|| Veya

Her iki işlemden herhangi birisi doğruluk şartını taşıyorsa if yapısının içine girer.

! Değil

Değer olarak verilen ifadenin sıfır olma durumudur. İfadenin sıfır olma şartı sağlanıyor ise if yapısının içine girer.

Örnek:

```
if (!buton1) {
digitalWrite(led, LOW);          // ledi söndür
}
```

VII- Birleşik Operatörler

++***x*** veya ***x***++ kullanımı, ***x=x+1*** ile aynıdır yani amaç sayıyı 1 arttırmaktır.

--*x* veya ***x*--** kullanımı, ***x=x-1*** ile aynıdır yani amaç sayıyı 1 azaltmaktır.

+=
-=
*=
/=
%=

```
x += y;          // x = x + y;
x -= y;          // x = x - y;
x *= y;          // x = x * y;
x /= y;          // x = x / y;
x %= y;          // x = x % y; //mod alma
```

Mod alma x in y ile bölümünden kalan sayıdır. ***x=100%3*** için x =1 yani 100 sayısının 3' bölümünden kalan 1'dir olarak döner

Map komutu

Sensor değerini belli aralıkta tutarak istenilen aralığa dönüştürülmesini sağlar. Sensörden okunan değer 0 ile 1023 arasında iken gerekli olan 0 ile 255 arası değer için

```
int sonuc=map(sensor, 0, 1023, 0, 255);
```

kodlanması gerekir.

Arduino'da koşullu durumlar ve döngüler

if

Eğer butona basıldıysa ledi yak vb. durumlarda veya karşılaştırmalarda kullanılır.

if/else

if else bir koşullu ifadeyi yürütmek için kullanılır. If eğer demektir else, değil ise demektir. if ve else birlikte kullanılır. Programlama dilinde else tek başına kullanılamaz.

switch/case

switch/case bir ifadenin sabit değerlerinden birisiyle eşleşip eşleşmediğini test eden çok yönlü bir karar verme yapısıdır. Bir switch/case yapısından çıkışı sağlamak ya da sonlandırmak için break ya da return kullanılır.

while

Sonsuz döngüdür. ***ifade***, doğru olduğu durumlarda iki süslü parantez içindeki ifadelerin sonsuz döngü olarak çalıştırılmasını sağlar.

```
while (ifade)
{
komutlar
}
```

do/while

do-while döngüsü öncelikle do parantez içindeki işlemi yapar ve while ile yapılan işlemi kontrol eder.

Break

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

continue

Continue do, for, while döngülerinde bir satırın, işlem yapılmadan geçilmesi istenildiği durumlarda kullanılır.

return

Return bir fonksiyon sonlandırılmak istenirse return ile döndürülecek değer belirtilir. Return'un ikinci bir kullanımı da belli bir yerden sonra kodlar çalışmasın istenirse return kullanılır.

for

Programlama dillerinde en sık kullanılan döngülerden biridir. For döngüsü uygulanması istenen işlemin kaç kez tekrar edileceğinin belli olduğu durumlarda kullanılır. Daha net bir ifade ile for döngüsü sayesinde işlemin kaç kez tekrar edileceğini, başlangıç/bitiş değerlerini ve değerler arasındaki artış/azalış miktarı kontrol edilir.

Switch – case örnek

```
// Pin tanımlamaları
const int buttonPin = 2;    // Butonun bağlı olduğu pin
const int ledPin1 = 3;      // LED1'in bağlı olduğu pin
const int ledPin2 = 4;      // LED2'in bağlı olduğu pin
const int ledPin3 = 5;      // LED3'ün bağlı olduğu pin

// Değişken tanımlamaları
int buttonState = 0;        // Butonun durumunu saklamak için değişken
int mode = 0;               // LED modunu belirleyecek değişken

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(ledPin3, OUTPUT);
}

void loop() {
  // Buton durumunu oku
  buttonState = digitalRead(buttonPin);

  // Butona basıldığında
  if (buttonState == HIGH) {
    // Mod değişkenini güncelle
    mode++;
    if (mode > 2) {
      mode = 0;  // Modu sıfırla (0, 1, 2 döngüsü)
    }
    delay(200);  // Butona tekrar basmadan önce kısa bir gecikme
  }
  // LED durumunu belirle
  switch (mode) {
    case 0:
      digitalWrite(ledPin1, HIGH);
      digitalWrite(ledPin2, LOW);
      digitalWrite(ledPin3, LOW);
      break;
    case 1:
      digitalWrite(ledPin1, LOW);
      digitalWrite(ledPin2, HIGH);
      digitalWrite(ledPin3, LOW);
      break;
    case 2:
      digitalWrite(ledPin1, LOW);
      digitalWrite(ledPin2, LOW);
      digitalWrite(ledPin3, HIGH);
      break;
  }
}
```

Bu örnekte:

- `buttonPin` pin numarasını belirler. Buton, bu pine bağlıdır ve butona basıldığında pin HIGH seviyesine gelir.
- `ledPin1`, `ledPin2`, ve `ledPin3` LED'lerin bağlı olduğu pinlerdir.
- `mode` değişkeni, hangi LED'in yanacağını belirler.

Kodun çalışma mantığı:

1. `loop()` fonksiyonu içinde, butona basıldığında `mode` değişkeninin değeri artırılır ve 0 ile 2 arasında döner.
2. `switch` ifadesi kullanılarak `mode` değişkeninin değerine göre hangi LED'in yanacağı belirlenir.

Butona her basıldığında, `mode` değişkeni artar ve farklı LED'ler yanar.

While örnek

```
// Pin tanımlamaları
const int buttonPin = 2;   // Butonun bağlı olduğu pin
const int ledPin = 13;     // LED'in bağlı olduğu pin

// Değişken tanımlamaları
int buttonState = 0;       // Butonun durumunu saklamak için değişken

void setup() {
  // Pin modlarını ayarla
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
}
void loop() {
  // Buton durumunu oku
  buttonState = digitalRead(buttonPin);

  // Butona basıldığında
  if (buttonState == HIGH) {
    // LED yanıp sönmeye başlar
    while (digitalRead(buttonPin) == HIGH) {
      // LED'i yak
      digitalWrite(ledPin, HIGH);
      delay(500); // 500 ms bekle
      // LED'i söndür
      digitalWrite(ledPin, LOW);
      delay(500); // 500 ms bekle
    }
    // Buton bırakıldığında LED'i kapat
    digitalWrite(ledPin, LOW);
  }
}
```

Bu örnekte

- **Pin Tanımlamaları:**
  - `buttonPin` butonun bağlı olduğu pin.
  - `ledPin` LED'in bağlı olduğu pin.
- **Setup Fonksiyonu:**
  - Buton ve LED pinlerinin modları belirlenir (`INPUT` ve `OUTPUT`).
- **Loop Fonksiyonu:**
  - `digitalRead(buttonPin)` kullanılarak butonun durumu okunur.
  - Eğer butona basılmışsa (`buttonState == HIGH`), `while` döngüsüne girilir.
  - `while` döngüsü, butona basıldıkça (`digitalRead(buttonPin) == HIGH`) çalışır. İçinde LED yanıp söner.
  - Butona basılmayı bıraktığınızda (`digitalRead(buttonPin) == LOW`), döngüden çıkılır ve LED kapatılır.

Bu örnek, `while` döngüsünün nasıl çalıştığını ve bir koşul sağlandığı sürece döngünün nasıl devam ettiğini göstermektedir.

do – while örnek

```
// Pin tanımlamaları
const int buttonPin = 2;   // Butonun bağlı olduğu pin
const int ledPin = 13;     // LED'in bağlı olduğu pin

void setup() {
  // Pin modlarını ayarla
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // LED'in yanıp sönmesini sağla
  do {
    // LED'i yak
    digitalWrite(ledPin, HIGH);
    delay(500); // 500 ms bekle

    // LED'i söndür
    digitalWrite(ledPin, LOW);
    delay(500); // 500 ms bekle
  }
  // Butona basılmadığı sürece döngüyü devam ettir
  while (digitalRead(buttonPin) == LOW);

  // Butona basıldıysa LED'i kapat
  digitalWrite(ledPin, LOW);

  // Bir süre bekle (butona basılmayı beklemek için)
  delay(1000); // 1 saniye bekle
}
```

Kodun Açıklaması:

- **Pin Tanımlamaları:**
  - `buttonPin`: Butonun bağlı olduğu pin.
  - `ledPin`: LED'in bağlı olduğu pin.
- **Setup Fonksiyonu:**
  - `pinMode` fonksiyonları ile buton ve LED pinlerinin modlarını ayarlarız (`INPUT` ve `OUTPUT`).
- **Loop Fonksiyonu:**
  - `do while` döngüsü, LED'in yanıp sönmesini sağlar.
  - LED, `digitalWrite(ledPin, HIGH)` komutu ile yanar, sonra `delay(500)` ile 500 ms bekler.
  - LED, `digitalWrite(ledPin, LOW)` komutu ile söner, ardından `delay(500)` ile 500 ms bekler.
  - `do while` döngüsü, butona basılmadığı sürece devam eder (`while (digitalRead(buttonPin) == LOW)`).
  - Butona basıldığında (`digitalRead(buttonPin) == HIGH`), döngüden çıkılır ve LED kapanır.
  - Ardından, butona basılmayı beklemek için 1 saniyelik bir gecikme eklenir.

Bu örnek, `do while` döngüsünün bir koşul sağlanana kadar kod bloğunu en az bir kez çalıştırma işlevini gösterir. `do while` döngüsü, belirli bir kod parçasının en az bir kez çalışmasını garanti eder ve ardından koşulu kontrol eder.