Arduino Uygulama Çalışması-2

Yumurta Zamanlayıcısı

Özet :

Değişken isimleri belirlenirken dikkat edilmesi gereken hususlar;

Değişken isimleri;

- Türkçe karakter içeremezler.
- Rakamlarla başlayamazlar
- Büyük – küçük harf duyarlıdırlar.
- Programlama dilinin anahtar kelimeleri isim olarak kullanılamaz.

Veri Tipleri

- ✓ Tam sayı tipler: (unsigned /signed) char, int, long
- ✓ Kayar Noktalı Tipler: float, double
- ✓ Karakter: char, string
- ✓ Mantıksal : bool (true veya false)

Diziler

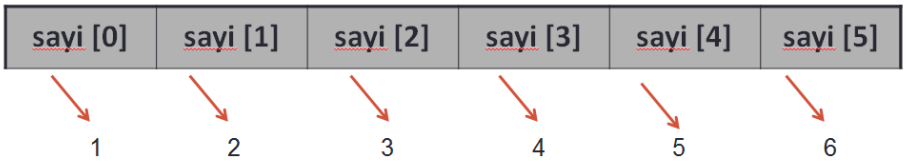
Aynı tipte birden fazla değişken gerekli olduğunda diziler kullanılabilir.

```
int dizi[10];                // 10 elemanlı bir int dizisi
float ondalikDizi[5];        // 5 elemanlı bir float dizisi
char karDizi[] = {'A','R','D','U','I','N','O'};
int sayiDizisi[] = {1,2,3,4,5,6,7};
```

Diziler-Bellek Yerleşimi

```
int sayi[6] = {1,2,3,4,5,6};
```

Değişkenler RAM de derleyicinin belirlediği bellek bölgelerine yerleşirler. Dizideki her bir int tipinden dolayı bellekte 16 bitlik yer kaplar. Yerleşim yeri RAM deki herhangi bir bölge olabilir. Adres bölgesi kulanılan değişken sayısına göre değişir.



Değişken Tanımlamalarına Bazı Farklı Örnekler

- Karakterler tırnak içinde tanımlanırlar.
  char a='a'; // a değişkenine 'a' karakteri atandı.
- Mantıksal değişkenler sadece iki değer alabilirler, false(0) ve true(1) olmak üzere.
  bool kontrol=false;
- String değişkenler karakter dizileridir en basit anlamıyla kelimeleri ve cümleleri tutmak için kullanılırlar.
  String sifre="ben124"; // string ifadeler çift tırnak içersinde yazılırlar.

Değişkenlerin Faaliyet Alanları(scope)

Değişkenler program içerisinde geçerli oldukları alanlara göre global veya lokal değişkenler olarak ikiye ayrılabilirler.

```
/* Global değişkenlerin tanımlı olduğu bölge*/
void setup() {
 /* Yerel değişkenlerin tanımlı olduğu bölge*/
}
void loop() {
 /* Yerel değişkenlerin tanımlı olduğu bölge*/
}
```

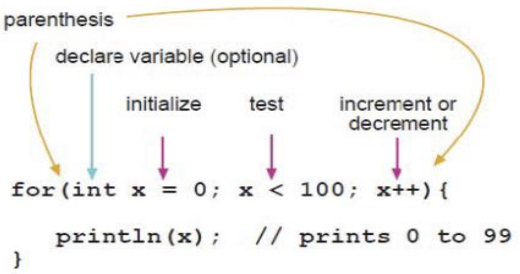
Değişkenlerin Ömürleri

- ✓ Değişkenler program içerisinde belirli bölgelerde tanımlı olup daha sonra yok edilebilirler.
- ✓ Global değişkenler bütün program boyunca tanımlıdırlar, yok edilmezler.
- ✓ Otomatik değişkenler tanımlı oldukları blok boyunca yaşayıp bloktan({…} arası) çıkınca yok edilirler.

Döngüler: Tekrar Eden İfadeler

Programlama esnasında bir çok defa tekrar edilen ifadeler bir döngü içersine alınabilirler. Tekrarlama işlemi döngü blokları arasında istenilen sayı kadar yapılabilir.

for döngüsü :




```
int ledler[8] = {13, 12, 11, 10, 9, 8, 7, 6};

void setup() {
     for (int x = 0; x < 8; x++) {
        pinMode(ledler[x], OUTPUT);
      }
} // Bu döngü ifadesi  {…} blokları arasında 8 defa
  // döner ve ledler dizisindei pinleri çıkış olarak
  // kurar.
```

While döngüsü :

Verilen mantıksal önerme sonucu doğru(true) olduğu müddetçe işlemler döngü(loop) halinde iki blok ('{' ve '}') arasında sırayla gerçekleştirilir. Önerme sonucu yanlış(false) olduğunda blok dışına çıkılır.

```
while (mantıksal önerme)
{
    // her bir adımda gerçekleştirilecek işlemler
}
```

```
int i=0;
while (i < 100)
{
    // her bir adımda gerçekleştirilecek işlemler
    Serial.println("Arduino");
    i++;            //i =i+1 anlamındadır i değişkenini 1 arttırır.
}
```

```
while(true){
        //sonsuz döngü oluşur ve işlemler sürekli tekrar edilir
        //….
}
```

do while Döngüsü: while ile aynı işleve sahiptir sadece mantıksal sınama bloğun altında yapılır. Blok içindeki işlemlerden sonra sınama yapılmış olur. Böylece bloktaki işlemler en az bir kere işletilmiş olur.

```
int i=0;
do
{
    // her bir adımda gerçekleştirilecek işlemler
    Serial.println("Arduino");
    i++;            //i =i+1 anlamındadır i değişkenini 1 arttırır.
} while(i < 100);
```

Döngüler: break ve continue ifadesi

Döngü içerisindeyken **break** ifadesi bulunuyorsa döngüden çıkılır. break ifadesini belirli bir şart oluştuğunda döngüyü sonlandırmak için kullanabiliriz.

Döngü içerisindeyken **continue** ifadesinden sonraki işlemler atlanarak döngü tekrar başa döndürülür. continue ifadesini döngüdeyken istisnai durumlar oluşturmak için kullanabiliriz.

```
while (x < 16) {
  if (x < 8) {
    digitalWrite(ledler[x], HIGH);
    delay(sure);
  }

  //if(x==10) continue;// donguyu geriye başlangıca döndürür
  if(x==10) break;   // döngüden çıkar.
  x++;
}
```

Kontrol Yapıları

Programımızın akışını değiştirmek, belirli şartlar sağlandığında belirli işlemler yaptırmak için kontrol yapıları kullanılır. "Belirli şartları" mantık önermelerle ifade ediyoruz. Programlarımıza karar verme yeteneği katarlar.

Mantık Önermeleri

a == b	a eşittir b
a != b	a eşit değildir b
a > b	a küçüktür b
a < b	a büyüktür b
a <= b	a küçük - eşittir b
a >= b	a büyük - eşittir b

Mantık önermelerin sonucu DOĞRU (TRUE / 1) veya YANLIŞ (FALSE / 0) olabilir.

Mantık Operatörleri

Temel mantık operatörleri AND (ve), OR (veya), NOT (değil) 'dir. TRUE (1) ve FALSE (0) 'dır.

Operatör	Sembol
✓ AND	&&
✓ OR	||
✓ NOT	!

Mantık Önermeleri AND :

İFADE - 1	İFADE - 2	İFADE1 && İFADE 2
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

AND operatörü uygulanan iki ifadeden her ikisinin de sonucu TRUE ise sonuç TRUE olur!

Mantık Önermeleri OR :

İFADE - 1	İFADE - 2	İFADE1 || İFADE 2
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

OR operatörü uygulanan iki ifadeden EN AZ birisinin sonucu TRUE ise sonuç TRUE olur!

Mantık Önermeleri NOT:

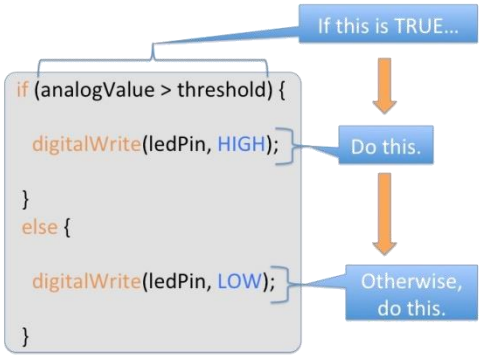
İFADE - 1	!İFADE-1
TRUE	FALSE
FALSE	TRUE

Bir ifadeye NOT operatörü uygulanırsa sonuc ifadenin sonucun DEĞİLİ (tersi) olur!

Dallanma : Karar Verme

Programın herhangi bir yerinde belirli bir şarta bağlı olarak bir işlem yaptırmak istiyorsak bu ifadeleri kullanırız. Bunlar en temel karar yapılarıdır. Programlarımıza karar verme yeteneği katarlar. Örneğin "Hava kararınca lambayı yak" gibi. Bu yapılarla program akışını değiştirebiliriz. Nitelik ve nicelikleri karşılaştırabiliriz.

***Şartsal İfadeler:* if()**




```
if ( degisken1 == 3)
{
    // degisken1 3'e eşitse işletilecek bölüm
    ....
}
else // eşit değil se
{
    // degisken1 3'e eşit değilse işletilecek bölüm
    ....
}
```

Bir diğer kullanımı şu şekildedir ;

```
if ( degisken1 == 3)
{
        // degisken1 3'e eşitse işletilecek bölüm
```

```
        ….
}
else if ( degisken1 == 4)
{
        // degisken1 4'e eşit değilse işletilecek bölüm
        ….
}
else    // yukarıdaki if ifadelerindeki mantıksal sonuç false(yanlış-0) ise
{
        …
}
```

if iadesi kendi başına else olmadan da kullanılabilir;

```
if(x>=10) break;
```

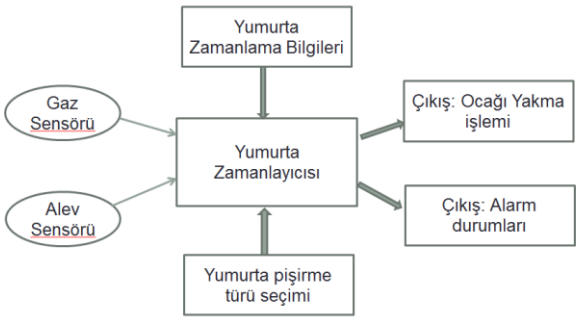
veya

```
if ( degisken1 == 3)
{
        // degisken1 3'e eşitse yapılacak işlemler…
}
```

UYGULAMA : Yumurta Zamanlayıcısı

Amaç: Ocağa koyduğumuz bir yumurtanın otomatik olarak pişirilip istediğimiz yumurta pişme türüne göre pişme zamanının ayarlanması.

Bir yumurta kaynatılıp pişirileceği zaman ocakta kalma süresine göre rafadan (az pişmiş), orta ve çok pişmiş olarak pişirilebilir. Ev hanımları veya aşçılar bunlarla ilgili zamanlamayı tecrübelerine dayanarak belirler.



Yapacağımız çalışmada yumurtanın pişebilmesi için gerekli şartların olup olmadığının da denetlenmesi ve alarm durumlarını gösterilmesi gerçek hayatta uygulanabilirlik açısından önemlidir. Bundan dolayı sistemimize şimdilik iki adet sensor eklenmiştir. Gösterge olarak ta 8 adet led kullanılacaktır. Eklenen sensor programsal olarak değiştirilerek etkileri gözlenmeye çalışılacaktır. Donanımsal olarak şimdilik konulmayacaktır.

Kodlama işlemi:

```
//Yumurta Zamanlayıcısı
// Yumurta pişme türü secimi
const char RAFADAN = 'R';
const char ORTA_PISMIS = 'O';
const char COK_PISMIS = 'C';
// yumurta pişme süreleri
int rafadan_sure = 3000;
int orta_sure = 5000;
int cok_sure = 7500;
```

```
//Pişme zaman durumunu ve alarm durumunu göstermek için.
int ledler[8] = {13, 12, 11, 10, 9, 8, 7, 6};
int led_index = 0;

char secim = 0;   //secimi tutan değişken

bool gaz_sensoru = true; //Gaz yoksa false varsa true değerini alır,
bool alev_sensoru = true; // Alev yoksa false, varsa true değerini alır.
bool basla = true;        // işleme başlama
long zaman_deg = 0;
```

```
void setup() {
  for (int x = 0; x < 8; x++) {
    pinMode(ledler[x], OUTPUT);
  }
  secim = ORTA_PISMIS;
  gaz_sensoru = true;
  alev_sensoru=true;
  basla=true;
}

void loop() {

  if (basla && gaz_sensoru && alev_sensoru)
  {
    if (secim == ORTA_PISMIS)
    {
      if (zaman_deg > orta_sure)
      {
        delay(1000);
        for (int x = 0; x < 8; x++) digitalWrite(ledler[x], LOW);
        basla = false;
        zaman_deg = 0;
      }
      if (zaman_deg >= ((orta_sure / 8)*led_index)) {
        led_index++;
      }
    }

    digitalWrite(ledler[led_index-1], HIGH);

  } else if (!gaz_sensoru || !alev_sensoru) //alarm durumu gaz veya alev yoksa
  {
      for (int x = 0; x < 8; x++) digitalWrite(ledler[x], HIGH);
      delay(100);
      for (int x = 0; x < 8; x++) digitalWrite(ledler[x], LOW);
     delay(100);
  }

  zaman_deg++;  // zaman_deg değikeninin her 1 mS de 1 arttır...
  delay(1);// 1 mS(mili saniye) aralıklarla işlem yap...

}//
```

1- Diğer yumurta pişirme türlerini kodda uygun yere ekleyerek çalışmalarını gözlemleyiniz.
2- Led gösterge için alternatif çalışma şekilleri belirleyerek deneyiniz.
3- Çalışma anında yumurta pişirme devam ederken alarm durumunu programsal olarak oluşturarak çalışmayı gözlemleyiniz.
4- Neler öğrendiğinizi kendi cümlelerinizle ifade ederek buranın alt kısmına yazınız.