7. METİN TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI

Bu bölümün sonunda,

- ✓ Arduino tümleşik geliştirme ortamının temel yapısını tanımlayabilecek,
- ✓ Arduino tümleşik geliştirme ortamının kullanım özelliklerini açıklayabilecek,
- ✓ Arduino UNO geliştirme kartının kullanım özelliklerini ve yapısını açıklayabilecek,
- ✓ Geliştirme yapılan bilgisayarla Arduino kartları arasında bağlantı oluşturulabilecek,
- ✓ Geliştirme yapılan bilgisayarla Arduino kartları arasında bağlantı ayarlarını yapabilecek,
- ✓ Arduino tümleşik geliştirme ortamının program yapısını açıklayabilecek,
- ✓ Arduino tümleşik geliştirme ortamının program yapısını oluşturan bileşenleri geliştirilen programa uygun şekilde kullanabilecek,
- ✓ Arduino tümleşik geliştirme ortamının değişken yapısının açıklayabilecek,
- ✓ Arduino tümleşik geliştirme ortamının değişken yapısını oluşturan bileşenleri geliştirilen programa uygun şekilde kullanabilecek,
- ✓ Arduino tümleşik geliştirme ortamının fonksiyon yapısının açıklayabilecek,
- ✓ Arduino tümleşik geliştirme ortamının fonksiyon yapısını oluşturan bileşenleri geliştirilen programa uygun şekilde kullanabilecek,
- ✓ Arduino tümleşik geliştirme ortamında kullanılan seri haberleşme protokollerini karşılaştırabilecek,
- ✓ Arduino kartları ile elektronik bileşenler arasında seri iletişim sağlayabilecek,
- ✓ Arduino kartları ile elektronik bileşenler arasında I2c veri yolu ile iletişim sağlayabilecek,
- ✓ Arduino kartları ile elektronik bileşenler arasında SPI veri yolu ile iletişim sağlayabilecek,
- ✓ Arduino kütüphanelerini program geliştirme projelerinde kullanabilecek,
- ✓ Arduino tümleşik geliştirme ortamında geliştirilen programları yeniden düzenlebilecek,
- ✓ Arduino tümleşik geliştirme ortamında geliştirilen programları yorumlayabilecek,
- ✓ Arduino tümleşik geliştirme ortamında program geliştirilebileceksiniz.

7.1. Metin Tabanlı Robot Programlama Yazılımları ve Ortamları

Robot programlamak amacıyla kullanılan pek çok programlama yazılımı ve ortamı bulunmaktadır. Robot programlama için kullanılan diller incelendiğinde geleneksel dillere robotik kontrolleri kolaylaştıran yapıların eklenmesiyle oluşturduğu görülür. Robot C ve Parallax Propeller C bu alanda dikkat çeken C dilinin robotik programlamaya uyarlanmış metin temelli sürümleridir. Burada metin tabanlı programlama yazılımı olarak Arduino IDE tercih edilmiştir. Arduino IDE robotik programlamada oldukça yaygın kullanılan tümleşik geliştirme ortamıdır.

7.2. Arduino Geliştirme Kartları

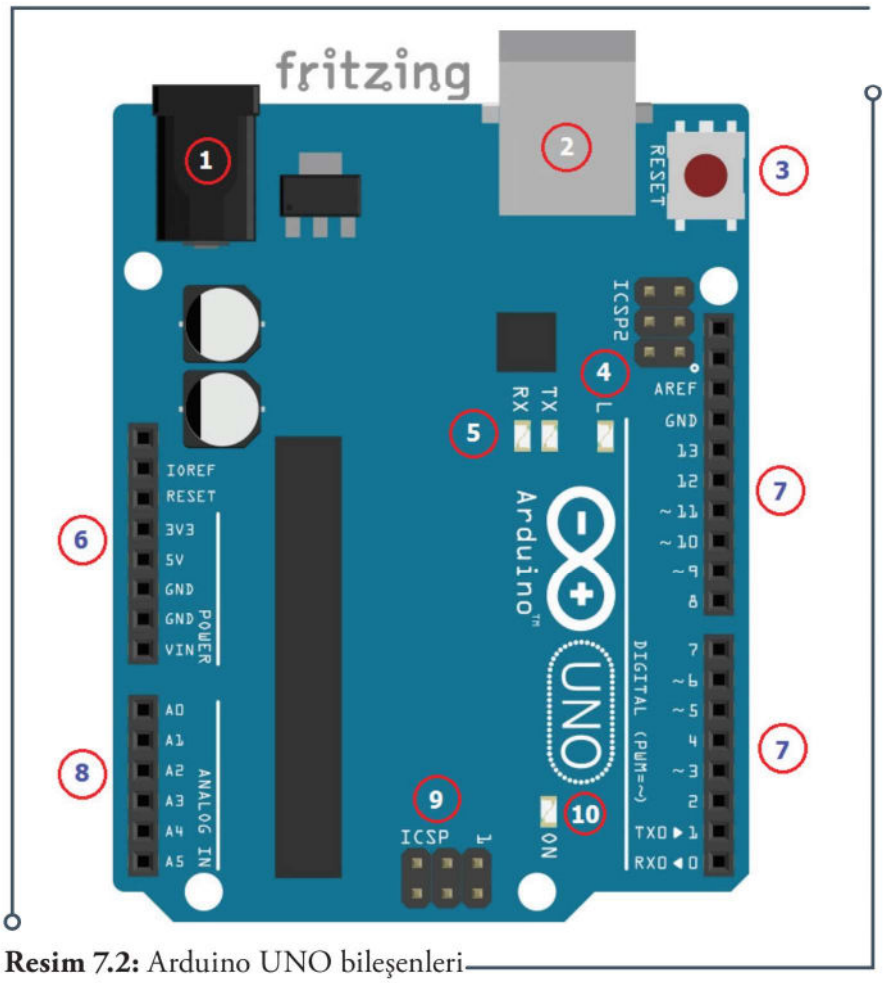
Arduino, ileri derecede elektronik ve mikrodenetleyici bilgisi gerektirmeden çok çeşitli projelerin uygulanabileceği açık kaynaklı, donanımında Atmel firması tarafından üretilen AVR mikrodenetleyici içeren bir elektronik geliştirme platformudur. Kolaylıkla analog ve dijital sinyaller alınarak işlenebilmektedir. Bununla birlikte Arduino ile birlikte kullanılacak devre elemanlarının çalışma mantığı ve bağlantı yapısının bilinmesi gerekmektedir. Algılayıcılardan gelen sinyalleri kullanarak, çevresiyle etkileşim içerisinde olan robotlar ve sistemler tasarlanabilmektedir. Tasarlanan projeye özgü olarak dış dünyaya hareket, ses, ışık gibi tepkiler oluşturulabilmektedir. Arduino'nun farklı ihtiyaçlara çözüm üretebilmek için tasarlanmış çeşitli kartları ve modülleri bulunmaktadır. Örneğin daha az donanımın yeterli olduğu projelerde Arduino Nano gibi modeller, çok sayıda giriş çıkış bağlantısına (pin) ihtiyaç duyulduğunda Arduino Mega gibi modeller kullanılmalıdır.

Resim 7.1: Arduino Uno

Birçok robotik uygulamada Arduino ya da Arduino uyumlu Atmel AVR mikrodenetleyici içeren kartlar kullanıldığı için Arduino robot programlamada, programlama öğretiminde de iyi bir seçenektir. Arduino kartlarla çok çeşitli kütüphaneleri yardımı ile kolaylıkla programlama yapılabilmektedir. Bu kitapta gerek konu anlatımında gerekse uygulamalarda çok yaygın olarak kullanılan Arduino UNO modeli tercih edilmiştir.

7.3. Arduino UNO Geliştirme Kartı

Arduino Uno Atmel Atmega 328P mikrodenetleyicisine sahip mikrodenetleyici karttır. Kart üzerinde temel olarak; 14 adet dijital giriş / çıkış pini (6 adeti PWM (Pulse Width Modulation-Darbe / Sinyal Genişlik Modülasyonu), 6 adet analog giriş pini, 16 MHz saat hızı için osilatör, bir adet USB bağlantısı, bir adet DC güç girişi, bir adet ICSP bağlantı başlığı ve bir adet reset düğmesi bulunmaktadır. 32 KB kapasiteli bir flash belleğe sahiptir. Uno'nun mikrodenetleyicisinde diğer modellerde olduğu gibi önceden hazırlanmış bir bootloader programı yüklüdür. Bu nedenle programlama için harici bir programlayıcıya ihtiyaç duyulmaz. Kartın kolaylıkla kullanılabilmesi, bileşenlerin kablo bağlantılarının rahatlıkla yapılabilmesi için pin soket yapısı kullanılmaktadır. Arduino Uno'nun temel bileşenleri aşağıda gösterilmektedir.



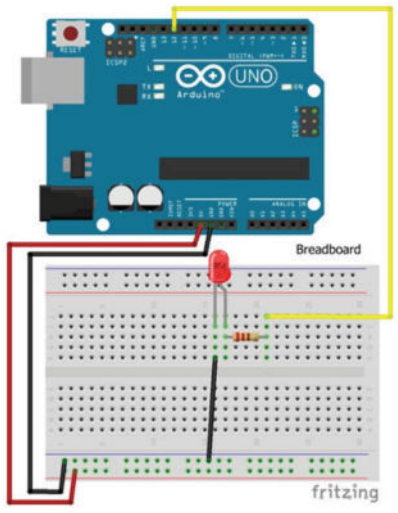
Resim 7.2: Arduino UNO bileşenleri

1. **Güç Girişi:** 7-12 Volt DC adaptör girişi.
2. **USB Bağlantı Konnektörü (USB Port):** UNO'ya program yüklemek ve bilgisayar ile haberleşmek için kullanılmaktadır.
3. **Reset Butonu:** Arduinoyu ve programı setup() fonksiyonundan itibaren yeniden başlatmak için kullanılmaktadır.
4. **LED (Light-Emitting Diodes):** 13 numaralı dijital pine bağlıdır. Programları test etmek için kullanılabilir.

5. **RX (Receiving) ve TX (Transmitting) LED'leri:** Seri haberleşme için kullanılan RX ve TX pininin durumunu gösteren LED'lerdir. Veri alışverişi olduğunda bu LED'ler yanar.
6. **Güç Bağlantıları:** Bu kısımda güç çıkışları yer almaktadır.
7. **Dijital Giriş /Çıkış Pinleri:** Yanında ~ işareti bulunan pinler aynı zamanda analog çıkış (PWM) almak için de kullanılabilir. Ayrıca analog-dijital çeviricinin referans giriş pini ve seri iletişim pinleri de (RX ve TX) buradadır.
8. **Anolog Giriş Pinleri:** 6 adet analog giriş pini bu bölümde bulunmaktadır.
9. **Kart Üzerinde Programlama (ICSP) Pinleri:** Atmega microdenetleyiciyi harici bir programlayıcı ile programlamak için kullanılan pinlerdir.
10. **Güç LED'i:** Kartın güç gösterge LED'idir.

Arduino UNO ile bilgisayar veya diğer cihazlar arasında seri iletişim yoluyla bağlantı kurulmaktadır. Tüm Arduino kartlarında en az bir seri bağlantı noktası (UART veya USART) bulunmaktadır. Seri bağlantı için dijital pin 0 (RX) ve pin 1 (TX) üzerinden bilgisayara USB portundan bağlanarak iletişim kurulur. Bu nedenle, USB bağlantısı kullanılırken dijital giriş veya çıkış için 0 ve 1 numaralı pinler kullanılamazlar. Çalışılan programa bağlı olarak seri iletişim işlemleri için Arduino ortamının dâhili seri monitörü kullanılır.

Arduino uygulamalarında kullanılacak elektronik bileşenler için breadboard (devre tahtası) kullanılması uygulamaların hızlı, kolay ve en önemlisi lehim yapmadan yapılmasına olanak tanıyacaktır. Breadboardların kenarında bulunan alanlar voltaj bağlantıları için enine bağlı, diğer alanlar ise dikine bağlıdır. Bu bağlantı noktalarını kullanarak LED, direnç ve benzeri elektronik bileşenlerin birbirine bağlanması oldukça kolaydır. Aşağıda örnekte Arduino UNO R3'ün 12 numaralı dijital pinine bağlı bir LED'in breadboard bağlantısı gösterilmektedir. Bağlantılar için breadboard kabloları kullanılmaktadır.



Resim 7.3: Arduino IDE'de breadboard kullanımı

Resim 7.4: Breadboard jumper kabloları

7.4. Arduino IDE (Tümleşik Geliştirme Ortamı-Integrated Development Environment)

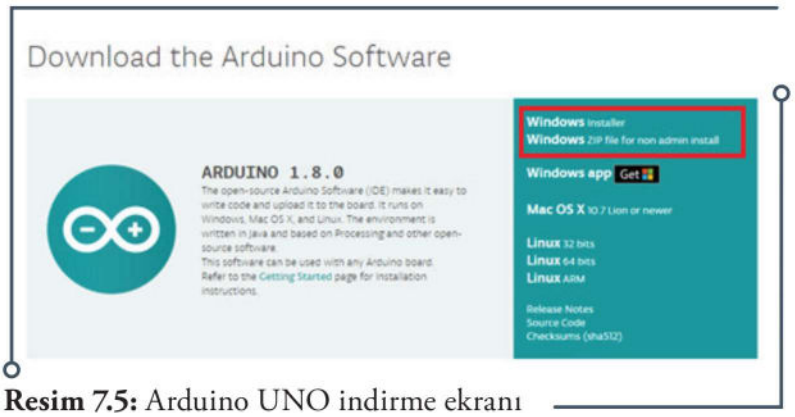
Arduino IDE; kod yazım editörü, tümleşik bir derleyici, yorumlayıcı ve hata ayıklayıcı olarak görev yapan, aynı zamanda derlenen programı karta yükleme işlemini de yapabilen, her platformda çalışabilen Java programlama dilinde yazılmış bir uygulamadır. Arduino tümleşik geliştirme ortamı (IDE); Arduino bootloader (Optiboot), Arduino kütüphaneleri, AVRDude (Arduino üzerindeki mikrodenetleyiciyi programlayan, derlenen kodları programlamak için kullanılan yazılım) ve derleyiciden (AVR-GCC) oluşmaktadır. Bu araçlar yazılımın derlenmesi, bağlanması, çalışmaya tümüyle hazır hale gelmesi ve daha birçok ek işi otomatik olarak yapabilmek amacıyla kullanılmaktadır.

Arduino yazılımı bir tümleşik geliştirme ortamı (IDE) ve mikrodenetleyici ve elektronik konusunda detaylı bilgi sahibi olmayı gerektirmeden herkesin programlama yapabilmesini sağlayan kütüphanelerden oluşmaktadır. Arduino kütüphaneleri, geliştirme ortamı ile birlikte gelmekte ve "libraries" klasörünün altında bulunmaktadır. IDE, Java dilinde yazılmıştır ve Processing adlı dilin ortamına dayanmaktadır. Kütüphaneler ise C ve C++ dillerinde yazılmıştır ve AVR-GCC ve AVR Libc ile derlenmiştir. Optiboot bileşeni Arduino 'nun bootloader bileşenidir. Bu bileşen, Arduino kartlarının üzerindeki mikrodenetleyicinin programlanmasını sağlayan bileşendir. Arduino C++ dili ile kolayca programlanabilmektedir. Kütüphaneler yardımıyla uygulama geliştirilmesi de oldukça kolay ve hızlıdır.

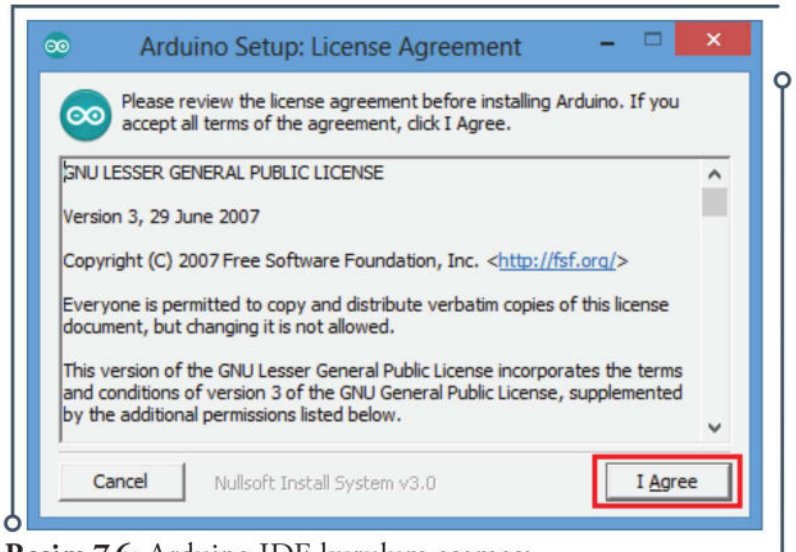
7.5. Arduino IDE Yazılımının Yüklenmesi

Arduino IDE yazılımının son versiyonu http://arduino.cc/en/main/software sitesinden kullanılacak işletim sistemi seçilerek ücretsiz olarak indirilebilmektedir. Örneğin Windows işletim sistemi için zip dosyası veya doğrudan exe dosyasından biri seçilerek indirilmelidir. Eğer zıp seçilmişse dosya ayıklandıktan sonra arduino.exe'ye tıklayarak programın kurulması yeterlidir. Kurulum işlemi tamamlandıktan sonra program kullanılmaya hazırdır. Önemli kurulum aşamaları aşağıda gösterilmiştir.

İlk aşamada lisans sözleşmesi onaylanmalıdır.

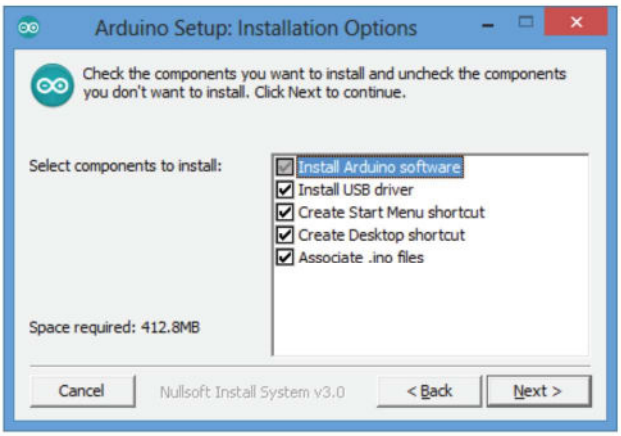


Resim 7.5: Arduino UNO indirme ekranı



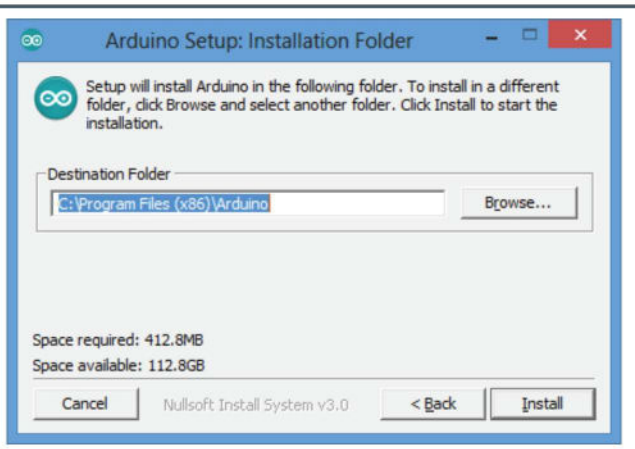
Resim 7.6: Arduino IDE kurulum aşaması

İkinci aşamada kurulum seçenekleri seçili değilse hepsi onaylanmalıdır.



Resim 7.7: Arduino IDE kurulum seçenekleri

Üçüncü aşamada kurulum klasörü seçilmelidir. İstenirse olduğu gibi bırakılabilir. Kurulum sırasında gerekli aygıt sürücüleri de kurulduğu için Arduino destekli bütün kartlar otomatik olarak tanınacaktır. Ayrıca tanıtılmasına gerek yoktur.

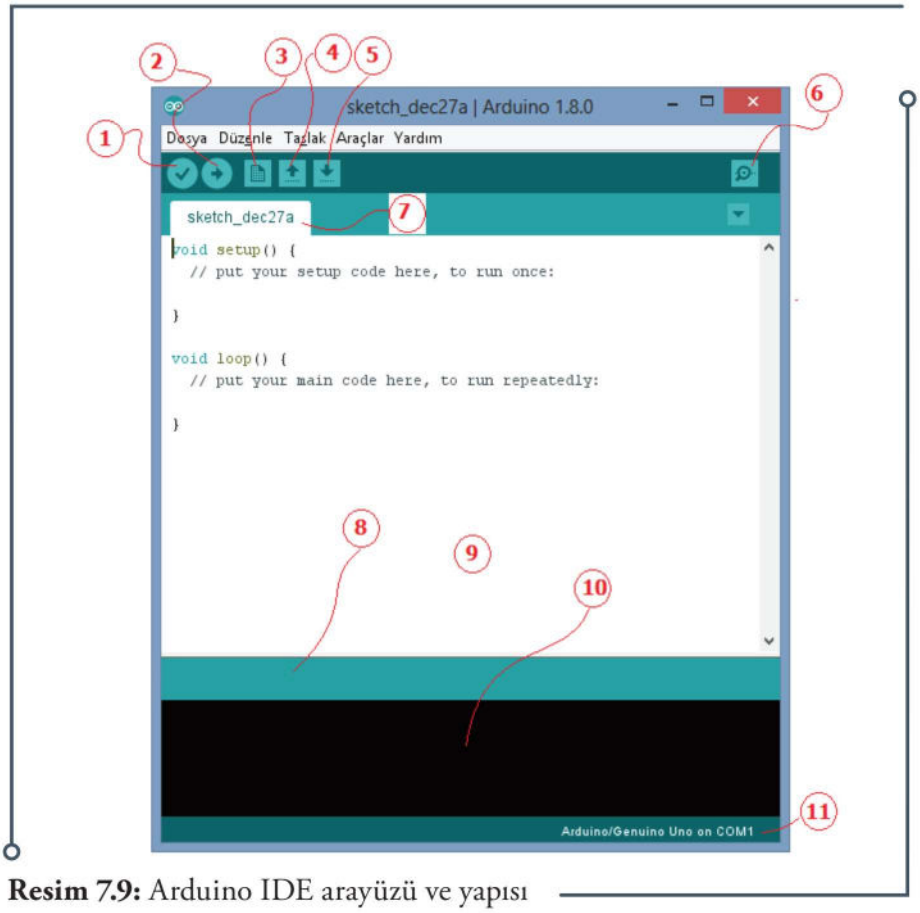


Resim 7.8: Arduino IDE kurulum klasörü

Program çalıştırıldığı zaman karşımıza aşağıdaki arayüz çıkmaktadır. Arayüz üzerinde bulunan sık kullanılan butonlar ve görevleri aşağıda açıklanmıştır.

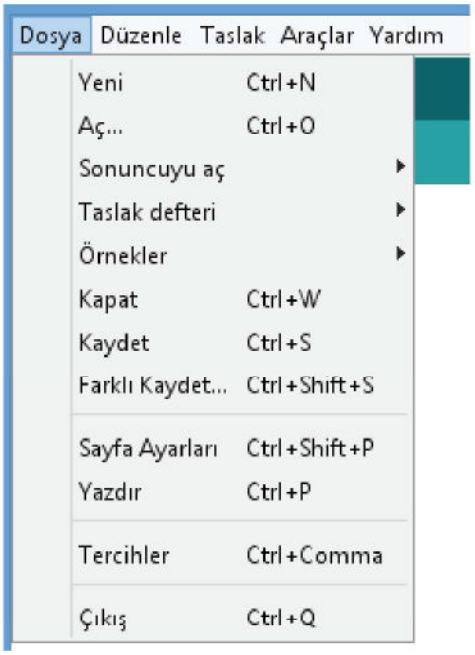
1. **Kontrol Et:** Yazılan kodları derler ve hataları bulur.
2. **Yükle:** Yazılan programı Arduino kartına yükler.
3. **Yeni:** Yeni çalışma sayfası açar.
4. **Aç:** Kayıtlı bir programı açar.
5. **Kaydet:** Yazılan programı kaydeder.
6. **Seri Port Ekranı:** Arduino ile seri iletişim yaparak ekran açar.
7. **Sketch:** Yazılan programın dosya ismini gösterir.
8. **Gösterge:** Yaptığı işlemin ilerleme durumunu gösterir.
9. **Boş alan:** Yazılacak program alanıdır.
10. **Rapor:** Derleme sonucu varsa yapılan hataları, yoksa programın yükleme sonrası mikrodenetleyicide kapladığı alanı gösterir.
11. **Gösterge:** Bilgisayara usb ile bağlanan Arduino'nun bağlandığı portu ve hangi Arduino modeli ile çalışılıyorsa onu gösterir.

Arayüzün üst kısmında temel bir menü çubuğu bulunmaktadır. Bu bölümde yer alan komutlarla da sık kullanılan butonların yaptığı işlevler yerine getirilebilmekte, buna ek olarak parça ayarlamaları, iletişim seçenekleri, ileri program ayarlamaları gibi işlevler de düzenlenebilmektedir.



Resim 7.9: Arduino IDE arayüzü ve yapısı

Dosya menüsünde temel komutlar yer almaktadır. Ayrıca "Örnekler" komutu ile program kütüphanesinde yer alan kodlar incelenebilmektedir. Bu bölümde çok sayıda örnek bulunmakta, gerekli bağlantılar yapılarak görsel olarak incelenebilmekte, küçük değişiklikler yaparak etkisi görülebilmektedir. Bu bölüm yeni başlayan kullanıcılar için büyük kolaylık sağlamaktadır.



Resim 7.10: Arduino IDE dosya menüsü

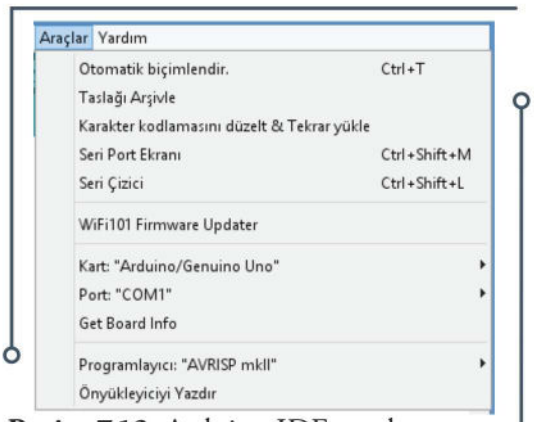
Düzenle menüsünde standart işlemlerin dışında "Forum için Kopyala" ve "HTML olarak Kopyala" seçenekleri kullanılarak internet ortamında, Arduino arayüzünde yazılan biçimde (renk- yazı tipi) paylaşabilmektedir.

Resim 7.11: Arduino IDE düzenle menüsü

Taslak menüsünde standart işlemlerin dışında "library ekle" sekmesini kullanarak istenilen kütüphane otomatik olarak çalışılan koda eklenebilmektedir. Ayrıca başka bir kütüphane eklemek için "Dosya Ekle" kullanılabilmektedir.

Resim 7.12: Arduino IDE taslak menüsü

Araçlar menüsünde standart işlemlerin dışında "Kart" sekmesini kullanarak programlanacak Arduino modelinin seçimi, "Port" kısmından kullanılan Arduino'nun bilgisayarın hangi portuna bağladığı seçilmelidir. Arduino'nun bağlı olduğu port ile arayüzde seçili olan portun aynı olması gereklidir. Arduino destekli robotla bağlantı bu şekilde yapılmalıdır.



Resim 7.13: Arduino IDE araçlar menüsü

7.6. Arduino Tümleşik Geliştirme Ortamının Temel Özellikleri

- ✓ Arduino IDE tümleşik geliştirme ortamında basitleştirilmiş C++ kullanılır.
- ✓ Arduino programları genellikle tanımlamalar, kurulum ve ana program bloğu olmak üzere üç bölümden oluşur.
- ✓ Program yazımı belirli kalıpta, bloklar halinde gerçekleştirilir.
- ✓ Program kodları renkli olarak gösterilir. Kodların bulunduğu yerlerde gri renkte olan yazılar kodun ne işe yaradığı hakkında bilgi vermek için kullanılır.
- ✓ Arduino'ya yüklenen programlar kaldırılana kadar Arduino içinde kalır. Yüklemeden sonra bağımsız olarak çalıştırılabilir.
- ✓ Bloklar, { } parantezleri ile oluşturulur.
- ✓ Komutlar aynı veya alt alta satırlara yazılabilir. Fakat programın anlaşılabilirliği açısından alt alta yazmak daha uygundur.
- ✓ Tüm komutlar noktalı virgül (;) ile biter. Fakat blok başlatan ifadelerden sonra noktalı virgül kullanılmaz.
- ✓ Programda kullanılan tüm değişkenler ve bilgi tipleri bildirilir.
- ✓ Programın başında kullanılacak kütüphaneler aktifleştirilir /çağrılır.
- ✓ Açıklamalar "//" ve "/* */ " (birden fazla satır için) ile yazılır.
- ✓ Türkçe karakter kullanılmamalıdır. Fakat açıklama satırları içerisinde (derleme işlemine dâhil edilmediğinden) kullanılabilir.
- ✓ Eşdeğer ifadeler #define ile atanır.
- ✓ Kütüphaneler #include ile çağrılır.

7.7. Arduino Tümleşik Geliştirme Ortamının Bölümleri

Arduino programları yapı, değişkenler (değişkenler ve sabitler) ve fonksiyonlar olmak üzere üç ana bölümden oluşmaktadır. Her bölümde kullanılan yapılar, operatörler, işlev ve fonksiyonlar aşağıda verilmektedir.

A. PROGRAM YAPISI		
void setup() void loop()	**3. Aritmetik Operatörler**	**6. İşaretçi Operatörler**
1. Kontrol Yapıları	= Atama İşleci + Toplama - Çıkarma * Çarpma / Bölme % Modulo	* Referan dışı operatör & Referans operatörü
if if/else for switch/case while do/while break continue return goto	**4. Karşılaştırma Operatörleri**	**7. Bitsel Operatörler**
2. Söz Dizimi	== (eşit eşit) != (eşit değil) < (küçük) > (büyük) <= (küçük eşit) >= (büyük eşit)	& (Bitsel Ve) | | (Bitsel Veya) ^ (Bitsel Xor) ~ (Bitsel Değil) << (Bitshift Sol) >> (Bitshift Sağ)
; Noktalı Virgül {} Süslü Parantez // Çift Slash /**/ Yıldızlı Slash #define #include	**5. Boolean Operatörleri**	**8. Birleşik Operatörler**
	&& (ve) || (veya) ! (değil)	++ (arttırma) -- (azaltma) +- (Birleşik artırma) -= (Birleşik çıkarma) *= (Birleşik çarpma) /= (Birleşik bölme) %= (Birleşik mod) &= (Bitsel Lojik Ve) |= (Bitsel Lojik Veya)

Tablo 7.1: Arduino IDE'nin program yapısı

B. DEĞİŞKENLER
1. Sabitler
HIGH | LOW
INPUT | OUTPUT | INPUT_PULLUP
LED BUILTIN
True | false
integer constants
floating point constants
2. Veri Tipleri
void
boolean
char
unsigned char
byte
int
unsigned int
word
long
unsigned long
short
dounle
string – char array
substring – object
array
3. Dönüşümler
char()
byte()
int()
word()
long()
float()
4. Değişken Kapsamları
static
volatile
const
5. Yardımcılar
PROGMEM
Sizeof()

C. FONKSİYONLAR	
1. Dijital Giriş Çıkışlar	**8. Rastgele Sayılar**
pinMode(pin,mod)	randomSeed()
digitalWrite(pin,değer)	random()
digitalRead(pin)	**9. Bit ve Bayt'lar**
2. Analog Giriş Çıkışlar	lowByte()
analogRead(pin,mod)	high(Byte()
analogWrite(pin,değer)	bitRead()
analogReference(tip)	bitWrite()
analogReadResolution ()	bitSet()
analogWriteResolution ()	bitClear()
3. Gelişmiş Giriş Çıkışlar	bit()
tone()	**10. Harici İnterruptlar (Kesmeler)**
noTone()	attachInterruption(interrupt, function, mode)
shiftOut()	detachInterrupt(interrupt)
shiftIn()	**11. İnterruptlar (Kesmeler)**
pulseIn()	interrupts()
4. Gecikmeler	noInterrupts()
delay(milisaniye)	**12. Seri Haberleşme**
unsigned long milis()	Serial.begin(hız)
delay(Microse conds (mikrosaniye)	int Serial.available()
5. Matematiksel İşlevler	int Serial.read()
min(x,y)	Serial.flush()
max(x,y)	Serial.print(data)
abs(x)	Serial.printIn(data)
constrain(x, a, b)	**13. Haberleşme Protokolleri**
map(value, fromLow fromHigh, toLow, toHigh)	I2C Veri Yolu
pow(base, esponent)	SPI Veri Yolu
sqrt(x)	
6. Trigonometri İşlevleri	
sin(raf)	
cos(rad)	
tan(rad)	
7. Karakterler	
isAlphaNumeric()	
isAlpha()	
isAscii()	
isWhiteSpace()	
isDigit()	
isGraph()	
isPrintable()	
isPunct()	
isSpace()	
isUpperCase()	
isHexaDecimalDigit()	

Tablo 7.2: Arduino IDE'de kullanılan değişken ve fonksiyon yapısı

7.8. Arduino Tümleşik Geliştirme Ortamının "Program" Yapısı

void setup()

void setup() fonksiyonu program yüklenip enerji verildikten veya tekrar başlatıldıktan sonra 1 defa çalışan fonksiyondur. "void se-

```
int buttonPin = 4; // butonu 4. pine tanımlandı
void setup()
{
  Serial.begin(9600); // Seri haberleşme başlatıldı
  pinMode(buttonPin, INPUT); // Pin modu tanımlandı
}
void loop()
{
  // ...
}
```

Resim 7.14: void setup() fonksiyonu örneği

tup()" ile başlayan satır, takip eden tırnak içindeki bölümde temel ayarların yapılacağını belirtmektedir. Bu fonksiyon içine pin modları, kütüphaneyi başlatma ve değişkenler yazılmaktadır. Burada yapılan ayarlamalarda hangi mikrodenetleyici pininin (veri bacağının) giriş (input-veri çekilen port-) ya da çıkış (output-veri gönderilen port-) olduğu belirtilmektedir. Örnekte (Resim 7.15) "void setup" içinde seri haberleşme başlatılmış ve pin modu tanımlanmıştır.

void loop()

Void loop() fonksiyonuna setup işleminden sonra eklenen ve mikrodenetleyici ya da Arduino'nun beslemesi devam ettiği sürece tekrarlanan komutlar yazılmaktadır. Buraya yazılan komutlar ile Arduino pinleri arasından karşılaştırma, ilişkilendirme, matematiksel işlemler vs. yapılmaktadır. Yazılan program burada sonsuz döngü içinde çalışmaktadır. Aşağıdaki örnekte tanımlanmış olan butuna basıldıysa seri ekrana "Basıldı", basılmadıysa "Basılmadı" yazan küçük bir program "void loop" içine yazılmıştır.

```
int buttonPin = 4; // butonu 4. pine tanımlandı

void setup()
{
  Serial.begin(9600); // Seri haberleşme başlatıldı
  pinMode(buttonPin, INPUT); // Pin modu tanımlandı
}

void loop()
{
  if (digitalRead(buttonPin) == HIGH) //okunan buton 1 ise
  Serial.write('Basıldı'); // Seri ekrana yaz
  else
  Serial.write('Basılmadı');
  delay(500);
}
```

Resim 7.15: void loop() fonksiyonu örneği

7.8.1. Kontrol Yapıları

#if

#if deyimi koşullu ifadeleri yürütmek için kullanılır. Örneğin belirlenen butona basıldıysa LED'i yak gibi durumlarda veya bir karşılaştırma operatörüyle birlikte karşılaştırmalarda kullanılır. Parantez içinde verilen sayı veya ifade ile belirlenen koşula ulaşılıp ulaşılamadığını test eder. Parantez içindeki ifadeler karşılanıyorsa, parantez içindeki ifadeler çalıştırılır. Değilse, program kod üzerinde atlanır. Aşağıda verilen örnekte Arduino UNO'nun 4 numaralı dijital pinine bağlı bir LED'in yanık kalma süresini kontrol eden #if deyimi uygulaması yer almaktadır. Arduino UNO'ya LED bağlantılarını yaptıktan sonra örneği dersin sitesinden (7.8.1.1 numaralı uygulama) kopyalayarak Arduino IDE'de test ediniz.

```
int buton = 4; // butonu 4. pine tanımladık
int led = 10; // ledi 10. pine tanımladık

void setup() {
pinMode(buton, INPUT); // 4. pin giriş oldu
pinMode(led, OUTPUT); // 10. pin çıkış oldu
}

void loop(){//sonsuz döngü
if (digitalRead(buton) == HIGH) { //okunan buton 1 ise
digitalWrite(led, HIGH); // ledi yak
}
}
```

Resim 7.16: #if örneği

```
int Led = 4; // Arduino 4 numaralı dijital pinine LED bağlanıyor
 // LED'in önüne 220/330 ohm değerinde bir direnç bağlanmalıdır

void setup()
 {
 pinMode(Led, OUTPUT);      // LED çıkış olarak tanımlanıyor
 }
 int gecikmeSuresi = 1000; // Başlangıç değeri 1000 olan bir değişken
tanımlanıyor

void loop()
 {
 gecikmeSuresi = gecikmeSuresi - 100; //Değişkenden her işlem için
100 değeri çıkartılıyor
 if(gecikmeSuresi <= 0)     // Eğer gecikme süresi sıfır veya daha
az ise
 {
 gecikmeSuresi = 1000;      //  Şart gerçekleşirse gecikme süresi
tekrar 1000 yapılıyor
 }
 digitalWrite(Led, HIGH);   // LED yanıyor
 delay(gecikmeSuresi);      // Gecikme süresi kadar bekleniyor
 digitalWrite(Led, LOW);    // LED söndürülüyor
 delay(gecikmeSuresi);      // Gecikme süresi kadar bekleniyor
}
```

#if/else

#if/else deyimi koşullu ifadeleri yürütmek için kullanılır. Temel kod akışı üzerinde daha fazla denetim sağlar. "if" eğer, "else" ise değil demektir. "if" ve "else" birlikte kullanılır. "else" tek başına kullanılamaz. Parantez içinde verilen sayı veya ifade ile belirlenen koşula ulaşılıyorsa bir eylem, ulaşılamıyorsa başka bir eylem yapılır. Aşağıda verilen örnekte Arduino UNO'nun 4 numaralı dijital pinine bağlı bir LED'in yanık kalma süresini kontrol eden if/else deyimi uygulaması yer almaktadır. #if için yapılan örneğe #else eklenerek oluşturulmuştur. Arduino UNO'ya LED bağlantılarını yaptıktan sonra örneği dersin sitesinden (7.8.1.2

```
int buton = 4; // butonu 4. pine tanımlandı
int led = 10; // ledi 10. pine tanımlandı

void setup() {
pinMode(buton, INPUT); // 4. pin giriş yapıldı
pinMode(led, OUTPUT); } // 10. pin çıkış yapıldı

void loop(){//sonsuz döngü
if (digitalRead(buton) == HIGH) { //okunan buton 1 ise
digitalWrite(led, HIGH);} // ledi yak
else { // değil ise
digitalWrite(led, LOW); }} // ledi söndür
```

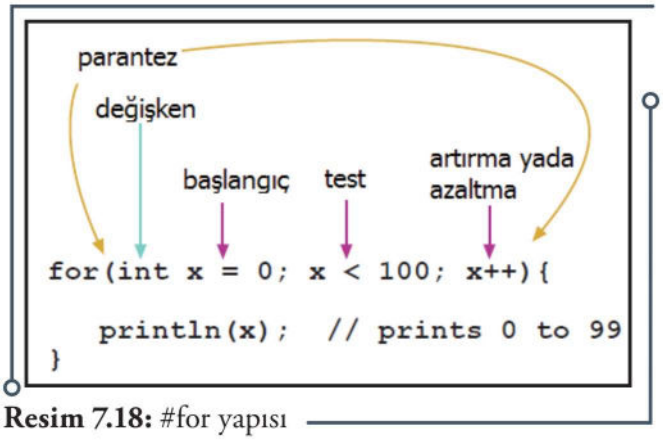
Resim 7.17: #if /else örneği

numaralı uygulama) kopyalayarak Arduino IDE'de test ediniz. İlk örnekle aralarındaki benzerlik ve farklılıkları inceleyiniz.

```
int Led = 4; // Arduino 4 numaralı dijital pinine LED bağlanıyor
 // LED'in önüne 220/330 ohm değerinde bir direnç bağlanmalıdır

void setup()
{
 pinMode(Led, OUTPUT);      // LED çıkış olarak tanımlanıyor
 }
 int gecikmeSuresi = 1000; //  Başlangıç  değeri  1000  olan  bir
değişken tanımlanıyor

void loop() {
 if(gecikmeSuresi <= 0)      // Eğer gecikme süresi sıfır veya daha az
ise
 {
 gecikmeSuresi = 1000;      //  Şart  gerçekleşirse  gecikme  süresi
tekrar 1000 yapılıyor
 }
 else                        // Değilse
 {
 gecikmeSuresi = gecikmeSuresi - 100; // Değişkenden her işlem için
100 değeri çıkartılıyor }
 digitalWrite(Led, HIGH);    // LED yanıyor
 delay(gecikmeSuresi);       // Gecikme süresi kadar bekleniyor
 digitalWrite(Led, LOW);     // LED söndürülüyor
 delay(gecikmeSuresi);       // Gecikme süresi kadar bekleniyor
}
```

#for

#for deyimi küme parantezi içine alınmış bir deyim bloğunu tekrarlamak için kullanılır. Döngüyü artırmak ve sonlandırmak için genellikle bir artış sayacı kullanılır. For ifadesi tekrar eden işlemler için yararlıdır ve genellikle veri / pin topluluğu üzerinde çalışmak için dizilerle birlikte kullanılır. For döngüsü üstbilgisinde başlatma, koşul ve artırma olmak üzere üç bölüm bulunmaktadır. İlk başlatma tam olarak bir kez olur. Döngüde her defasında koşul test edilir; eğer doğruysa, ifade bloğu ve artım yürütülür, koşul tekrar test edilir. Koşul yanlış olduğunda ise döngü sona erer. Yandaki örneği inceleyiniz. Aşağıda verilen diğer örnekte Arduino UNO'nun 4 numaralı dijital pinine bağlı bir



Resim 7.18: #for yapısı

LED'in birer saniye aralıklarla 10 defa yandıktan sonra döngüye tekrar başlayan #for deyimi uygulaması yer almaktadır. Arduino UNO'ya LED bağlantılarını yaptıktan sonra örneği dersin sitesinden (7.8.1.3 numaralı uygulama) kopyalayarak Arduino IDE'de test ediniz.

```
// PWM pini ile LED kontrolu
int PWMpin = 13; // LED 13. Pin'e tanımlandı
void setup()
{
  // setup gerekli değildir
}
void loop()
{
   for (int i=0; i <= 255; i++){
      analogWrite(PWMpin, i);
      delay(10);
   }
}
```

Resim 7.19: #for örneği

```
int Led = 4; // Arduino 4 numaralı dijital pinine LED bağlanıyor
 // LED'in önüne 220/330 ohm değerinde bir direnç bağlanmalıdır
void setup() {
 pinMode(Led, OUTPUT);        // LED çıkış olarak tanımlanıyor
 }
 int gecikmeSuresi = 1000;  //  Gecikme  süresi  1sn  olarak  bir
değişkenle tanımlanıyor

void loop() {
 for(int m = 0; m < 10; m++) // Döngü boyunca m değeri artırılıyor
 {
 digitalWrite(Led, HIGH);     // LED yanıyor
 delay (gecikmeSuresi);       // Gecikme süresi kadar (1 sn) bekleniyor
 digitalWrite(Led, LOW);      // LED söndürülüyor
 delay(gecikmeSuresi);        // Gecikme süresi kadar (1 sn) bekleniyor
 }
 delay(2000);                 // 2 sn bekleniyor
}
```

#switch/case

#switch/case bir ifadenin sabit değerlerinden birisiyle eşleşip eşleşmediğini test eden çok yönlü bir karar verme yapısıdır. Bir programda çok sayıda koşul kontrolü ve bunların sonucuna göre gerçekleştirilmesi gereken işlemler varsa kullanılır. Bir "switch" deyimi bir değişkenin değerini "case" ifadelerinde belirtilen değerlerle karşılaştırır. Değişkenin değeriyle eşleşen bir "case" ifadesi bulunursa, bu "case" ifadesinin kodu çalıştırılır. Bir switch/case yapısından çıkışı sağlamak ya da sonlandırmak için "break" ya da "return" kullanılmalıdır. "switch" ifadesinden hemen sonra gelen ifade parantez içinde yer almalı ve bir tamsayı veya değişken olmalıdır. "case" ifadesini izleyen ifadeler tamsayı türünde ifadeler olmalıdır, yani değişken içermemelidir. Aşağıdaki örneği inceleyiniz.

```
/* Arduino'nun A1 girişine bir potansiyometrenin
   çıkış pini girdiğimizi düşünelim */
void setup() {
Serial.begin(9600); //Seri haberleşme
}
void loop() {
int okunandeger = analogRead(A1);
//map fonksiyonu okuduğumuz değeri (0-1023 aralığını) 2 ye böler
//0-511 ise aralığa 0 değerini verir.
//512-1023 ise aralığa 1 değerini verir.
int aralik= map(okunandeger, 0, 1023, 0, 1);
switch (aralik) {
case 0: //Pot döndürülmesi 0-49% arasında ise
Serial.println("DEĞER DÜŞÜK"); //Seri port ekranına yazdır
break; // döngüden çıkış
case 1: // Pot döndürülmesi 50-100% arasında ise
Serial.println("DEĞER YÜKSEK"); //Seri port ekranına yazdır
break; } // döngüden çıkış
delay(1); // stabil çalışması için 1 milisaniye beklenir
}
```

Resim 7.20: #switch/case örneği

#while

#while döngüsü başlamadan önce koşul kontrol edilir. Belirtilen koşul doğru olduğu sürece, döngü içindeki işlemler gerçekleştirilir. Örnekte while döngüsü; LED'i yakacak, 1 saniye bekleyip LED'i söndürecek ve işlemi 1 arttıracaktır. While döngüsünde 10 kere aynı işlem yapıldıktan sonra döngüden çıkılacaktır. Aşağıda verilen örnekte Arduino UNO'nun 4 numaralı dijital pinine bağlı bir LED'in yanık kalma süresini kontrol eden #while deyimi uygulaması yer almaktadır. Arduino UNO'ya LED bağlantılarını yaptıktan sonra örneği dersin sitesinden (7.8.1.4 numaralı uygulama) kopyalayarak Arduino IDE'de test ediniz. #if örneğindeki uygulama ile aralarındaki benzerlik ve farklılıkları inceleyiniz.

```
int deneme=0; int led=0;
void setup() { }
void loop() { while (deneme<10)
digitalWrite(led, HIGH); // ledi yak
delay(1000); //1 saniye bekle
digitalWrite(led, LOW); // ledi söndür
deneme=deneme+1;
}
```

Resim 7.21: #while örneği

```
int Led = 4; // Arduino 4 numaralı dijital pinine LED bağlanıyor
 // LED'in önüne 220/330 ohm değerinde bir direnç bağlanmalıdır
void setup() {
 pinMode(Led, OUTPUT); }        // LED çıkış olarak tanımlanıyor
int gecikmeSuresi = 1000;       // Başlangıç değeri 1000 olan bir
değişken tanımlanıyor

void loop() {
 while (gecikmeSuresi > 0) {    // Gecikme süresi sıfırdan büyükse
 digitalWrite(Led, HIGH);       // LED yanıyor
 delay(gecikmeSuresi);          // Gecikme süresi kadar bekleniyor
 digitalWrite(Led, LOW);        // LED söndürülüyor
 delay(gecikmeSuresi);          // Gecikme süresi kadar bekleniyor
gecikmeSuresi = gecikmeSuresi - 100; // Değişkenden her işlem için
100 değeri çıkartılıyor
 }
 while (gecikmeSuresi < 1000) {  // Gecikme süresi 1000'den küçükse
gecikmeSuresi = gecikmeSuresi + 100; // Değişkenden her işlem için
100 değeri ekleniyor
 digitalWrite(Led, HIGH);       // LED yanıyor
 delay(gecikmeSuresi);          // Gecikme süresi kadar bekleniyor
 digitalWrite(Led, LOW);        // LED söndürülüyor
 delay(gecikmeSuresi);          // Gecikme süresi kadar bekleniyor
 }
}
```

#do/while

#do/while operatörü karşılaştırmalarda koşul içeriyorsa parantez içinde belirtilen "do" ifadesine göndererek tekrar hesaplama yaptırarak yeni değeri karşılaştırır. Örnekteki program "do" döngüsüne girecek sayıyı 2 arttıracak, seri ekranına yazdıracak "while" ile durumu kontrol edecektir. Sayı 10'dan büyük olduğu zaman döngüden çıkacaktır.

```
void setup() {
int sayi = 0;
Serial.begin(9600); // seri haberleşme
do { // 10 e kadar 2 şer sayma
sayi = sayi+ 2;
Serial.print("sayi = ");
Serial.println(sayi);
delay(500);
// 500ms bekleme
} while (sayi< 10);
}
void loop() {
}
```

Resim 7.22: #do/while örneği

#break

#break do, for ve while döngülerinde döngü çalışması bittiğinde döngü dışına çıkmak için kullanılmaktadır. Switch/case yapısında da kullanılır. Örnekte sensör değeri 200'den büyük olursa döngü dışına çıkılacaktır.

```
for (x = 0; x < 255; x ++)
{
    analogWrite(PWMpin, x);
    sens = analogRead(sensorPin);
    if (sens > 200){ // sensor çıkışı tespit ediliyor
       x = 0;
       break;
    }
    delay(50);
}
```

Resim 7.23: #break örneği

#continue

#continue do, for ve while döngülerinde bir satırın, işlem yapılmadan geçilmesini istediğimiz durumlarda kullanılmaktadır. Döngünün geri kalan kısmını kontrol etmeye devam eder.

```
for(x=0;x<255;x++){
 if(x>40&&x<100){
 continue; // x değeri 100 den büyük ve
 //100 küçükse atla
 }
 analogWrite(PWMpin, x);
 delay(50);
}
```

Resim 7.24: #continue örneği

#return

#return bir fonksiyon sonlandırılmak istenirse "return" ile döndürülecek değer belirtilmelidir. Returnun ikinci bir kullanım şekli de belli bir yerden sonra kodların çalışmasının istendiği durumlarda kullanılmasıdır. Yandaki örnekte okunan algılayıcı değeri 100'den büyük ise 1 değerine, 100'den küçük ise sıfır değerine döndürülmektedir. Aşağıda ise çalışması istenmeyen kodların nasıl yazılacağı gösterilmiştir.

```
int sensorkontrol () {
  if(analogRead(0) > 100) {
return 1 ;
}
  else
return 0;
}
```

Resim 7.25: #return örneği

```
void loop ( ) {
// çalışması isteten kodlar
return; //çalışması istenmeyen kodlar buraya yazılır
```

Resim 7.26: #return diğer kullanım örneği

#goto

#goto program akışını istenilen yöne yönlendirmek için kullanılmaktadır. Derin iç içe geçmiş döngülerde veya "if" mantık bloklarından belirli bir şartla ayrılma durumunda kullanışlı olmakta ve kodlamayı basitleştirmek için tercih edilmektedir.

```
void loop() {
int x = analogRead(1);
if (x < 100) {
goto git;
}
git:
delay(1000);
// çalışması istenilen kod}
```

Resim 7.27: #goto örneği

7.8.2. Söz Dizimi

; Noktalı Virgül

C programlama dilinde her satır programından sonra noktalı virgül konulması, sonlandırılması gerekmektedir. Noktalı virgül konulmadığı durumlarda derleme hatası oluşmaktadır.

```
void loop() {
int x = analogRead(1);
```

Resim 7.28: ;noktalı virgül örneği

{} Süslü Parantez

Fonksiyonlarda, döngülerde ve koşullu ifadelerin bildirilmesinde süslü parantez kullanılmaktadır. İç içe olan fonksiyonlarda en dıştaki süslü parantezin en baştaki fonksiyona ait olduğuna dikkat edilmeli, aksi takdirde derleme hatası ortaya çıkmaktadır.

```
void fonksiyonlarim()
{
//yapılacaklar }
while () {
//yapılacaklar }
do {
//yapılacaklar
}
for (x=0;x<=100;x++)
{
//yapılacaklar }
```

Resim 7.29: {} süslü parantez örneği

// Çift Slash

Program satırından, program başında veya herhangi bir yerde programın çalışma şekli hakkında açıklama yapmak isteniyorsa çift slash kullanılmalıdır. Çift slashtan sonra yazılanlar program kodu olarak dikkate alınmaz. Derleyici tarafından yok sayılır ve mikroişlemciye aktarılmaz. Örneği (Resim 7.30) inceleyiniz.

```
X = 5; // Bu, tek satırlı bir açıklamadır. Çift slash sonrasındaki her şey bir açıklamadır
       // satırın sonuna çift slash eklenmelidir.
/ * bu satırlı bir açıklamadır - tüm kod bloklarını açıklamak için kullanabilirsiniz
(gwb == 0) {  // Tek satırlık açıklama tek satır olarak kullanılabilir
X = 3;        / * Ancak başka satırlar kullanılacaksa - yukarıdaki şekilde geçersizdir.
Bu şekilde kullanılmalıdır. * /
}
// "kapanış" slahsını unutmayın
* /
```

Resim 7.30: / / Çift slash örneği

/**/ Yıldızlı Slash

/* Buraya yazılan her türlü satır, sütun dahil program olarak alınmamakta, açıklama olarak dikkate alınmaktadır. */ Yukarıdaki örneği inceleyiniz.

#define

#define ön işlemci komutu olup kullanılan bir isim yerine başka bir ismin kullanımını ve değişimini sağlamaktadır. Programın derlenmesinden önce programcının sabit bir değere isim vermesine izin vermektedir. "#" işaretinin kullanılması gereklidir. #define deyiminden sonra hiçbir noktalı virgül veya eşittir bulunmamalıdır. Yanda verilen örneğe göre, programda ledpin görüldüğü yere 12 rakamı yerleştirilmelidir.

```
#define ledpin 12
```

Resim 7.31: #define örneği

#include

#include Arduino için yazılmış kütüphanelere erişimi sağlamak için kullanılmaktadır. Diğer bir ifadeyle üzerinde çalışılan program dışındaki kütüphanelere erişmek için kullanılan fonksiyondur. Örneğin yapılan programda SPI kütüphanesini kullanmak ve onun içerisindeki komutlara ulaşmak istendiğinde program başı tanımlanması ("...") ya da (<...>) şekilde olmak üzere iki şekilde yapılmaktadır. #include deyiminden sonra hiçbir noktalı virgül veya eşittir bulunmamalıdır.

```
#include "SPI.h" //Tanımlama 1. tanımlama şekli
#include <SPI.h> //Tanımlama 2. tanımlama şekli
```

Resim 7.32: #include kullanım şekli

7.8.3. Aritmetik Operatörler

= Atama İşleci

C programlama dilindeki tek eşit işareti atama operatörü olarak adlandırılır. Bir denklem veya eşitlik gösterdiği cebir sınıfından farklı bir anlam taşır. Atama işleci, mikrodenetleyiciye eşittir işaretinin sağ tarafında bulunan herhangi bir değer veya ifadeyi değerlendirmesini ve eşit işaretin solundaki değişkende saklamasını söyler.

```
int algiDeg;              // algiDeg adını taşıyan bir tamsayı değişkeni
algiDeg = analogRead(1); // analog pin 1 in giriş gerilimi algiVal de saklanmaktadır
```

Resim 7.33: = Atama işleci kullanım örneği

+ Toplama, - Çıkarma, * Çarpma ve / Bölme

C'de programlama yaparken matematiksel işlemlerde aritmetik operatörler kullanılır. İnt ya da float (virgüllü) değerinden sonuçlar bulunabilir.

```
Y = y + 4;
X = x - 8;
I = j * 7;
R = r / 6;
```

Resim 7.34: 4 işlem örneği

% Mod

Bir tam sayı belirlenen bir bölüme ayrıldığında kalanını hesaplar. Belirli bir aralıktaki bir değişkeni tutmak için de kullanılmaktadır.

```
x = 7% 5; // X şimdi 2 içerir
x = 9% 5; // X şimdi 4 içerir
x = 5% 5; // X şimdi 0 içerir
x = 4% 5; // x şimdi 4 içerir
```

Resim 7.35: % Mod örneği

7.8.4. Karşılaştırma Operatörleri

== (eşit eşit), != (eşit değil), < (küçük), > (büyük), <= (küçük eşit), >= (büyük eşit)

Karşılaştırma operatörlerinin kullanımı aşağıda verilmiştir. Genellikle if içerisinde karşılaştırma yaparken kullanılan operatörlerdir. "if" karşılaştırma operatörüyle birlikte kullanıldığında, belli bir değerin üzerinde olup olmadığı test edilir. Parantez içindeki ifadeler doğruysa parantez içindeki ifadeler çalıştırılır. Doğru değilse program kod üzerinde atlanır.

```
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 eşit)
x >= y (x, y den büyük eşit)
```

Resim 7.36: Karşılaştırma operatörlerinin kullanım örneği

```
if (degisken> 100)
{
//değişken 100 den büyükse buraya girilir
}
```

Resim 7.37: Karşılaştırma operatörlerinin "if" içinde kullanım şekli

7.8.5. Boolean Operatörleri

&& (mantıksal ve)

Boolean ifadeleri genellikle "if" yapısının içerisinde ve belirli şart gerektiren durumlarda kullanılmaktadır. Yalnızca her iki işlem de doğruysa "if" şartı doğrudur. Bu durumda "if" içerisindeki komut çalıştırılır. Herhangi bir durum ya da ikisi de false, yanlış sonuç ise if yapısı çalıştırılmaz. Örnekte oku1 ve oku2 HIGH ise LED yanmaktadır.

```
void loop(){ //sonsuz döngü
int okul= digitalRead(1);
int oku2= digitalRead(2);
if (okul == HIGH && oku2 == HIGH) {
digitalWrite(led, HIGH); // ledi yak
}
}
```

Resim 7.38: && (mantıksal ve) kullanım örneği

|| (mantıksal veya)

Her iki işlemden herhangi birisi doğruluk şartını taşıyorsa if içerisindeki komut çalıştırılır. Örnekte oku1 veya oku2 HIGH ise led yanmaktadır.

```
void loop(){ //sonsuz döngü
int okul= digitalRead(1);
int oku2= digitalRead(2);
if (okul == HIGH || oku2 == HIGH) {
digitalWrite(led, HIGH); // ledi yak
}
}
```

Resim 7.39: || (mantıksal veya) kullanım örneği

! (mantıksal değil)

Değer olarak verilen ifadenin sıfır olma durumudur. İfadenin sıfır olma şartı sağlanıyor ise if yapısı çalıştırılmaktadır. Örnekte buton1'e basılmıyor ise LED'i söndürülmektedir.

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

Resim 7.40: ! (mantıksal değil) kullanım örneği

7.8.6. İşaretçi Operatörler

& Referans operatörü, * Referans dışı operatör

C programlama dilinde bazı veri yapılarını değiştirmek, bir değişkene başvuru yoluyla geçmek için bu işaretçilerin kullanılması kodun basitleştirilmesini sağlamaktadır. Örneğin bazı durumlarda bir değişkene bir şey yapılması ve diğer durumlarda da aynı şeyin başka bir değişkene yapılması gereken durumlarda kullanılmaktadır. Esasen iki farklı değişken üzerinde aynı işlemi gerçekleştiren iki kod sırası yerine, hangi değişkenin üzerinde çalışacağını seçen bir bit kodu bulunmaktadır. C / C ++ 'da değişken geçirmenin en temel biçimi değeri geçmektir ve bir değişken başka bir değişkene atandığında, aslında o değişkenin bir kopyası oluşturulduğu anlamına gelmektedir. Aşağıdaki örneği inceleyiniz.

```
int a = 1;
int b = a;
B + = 1;
// şimdi a == 2 ve b == 2
```

```
int a = 1;
int * b = & a;
* B + = 1;
// şimdi a == 2 (ve b'nin değeri, bir bellekteki konuma
// işaret eden bir işaretçidir)
```

Resim 7.41: & Referans ve * Referans dışı operatör kullanım örneği

7.8.7. Bitsel Operatörler

& (bitsel ve)

Bitsel operatörler hesaplamalarını değişkenlerin bit düzeyinde gerçekleştirir. Bitsel ve, işleme giren bitlerin ve'sini verirler. Yani her iki giriş biti de 1 ise, sonuç 1; aksi halde sonuç 0 olmaktadır. Örneği inceleyiniz.

```
0 0 1 1 operand1
0 1 0 1 operand2
----------
0 0 0 1 (operand1 & operand2) - sonuç
```

Resim 7.42: & (bitsel ve) kullanım örneği

```
int a = 92; // ikili olarak: 0000000001011100
int b = 101; // ikili olarak: 0000000001100101
int c = a & b; // sonuç: 0000000001000100, veya ondalık 68
```

Resim 7.43: & (bitsel ve) 2. kullanım örneği

| (bitsel veya)

Giriş bitlerinden biri veya her ikisi birden 1 ise iki bitlik bitsel veya 1, aksi halde sonuç 0 olmaktadır. Örneği inceleyiniz.

```
0 0 1 1 operand1
0 1 0 1 operand2
----------
0 1 1 1 (operand1 | operand2) - sonuç
```

Resim 7.44: | (bitsel veya) kullanım örneği

```
int a = 92; // ikili olarak: 0000000001011100
int b = 101; // ikili olarak: 0000000001100101
int c = a | B; // sonucu: 0000000001111101 veya ondalık sayı125
```

Resim 7.45: | (bitsel veya) 2. kullanım örneği

^ (bitsel xor)

Bu operatör, "bitsel ve" operatörüne çok benzemektedir. Yalnızca o pozisyon için her iki giriş biti 1 olduğunda verilen bir bit pozisyonu için 0 olarak değerlendirilir. Örnekleri inceleyiniz.

```
0 0 1 1 operand1
0 1 0 1 operand2
----------
0 1 1 0 (operand1 ^ operand2) - döndürülen sonuç
```

Resim 7.46: ^ (bitsel xor) kullanım örneği

```
int x = 12; // binary: 1100
int y = 10; // binary: 1010
int z = x ^ y; // binary: 0110 veya ondalık 6
```

Resim 7.47: ^ (bitsel xor) 2. kullanım örneği

~ (bitsel değil)

Bitsel değil operatörü sağdaki tek bir işlenene uygulanır. Bitsel değil uygulandığında her bit için tersi olmaktadır: 0 1 olur ve 1 0 olur. En yüksek bit 1 ise sayı negatif olarak yorumlanır. Örneği inceleyiniz.

```
0  1    operand1
----------
1  0   ~ operand1
```

Resim 7.48: ~ (bitsel değil) kullanım örneği

```
int a = 103;   // binary:  0000000001100111
int b = ~a;    // binary:  1111111110011000 = -104
```

Resim 7.49: ~ (bitsel değil) 2. kullanım örneği

<< (bitshift sol) ve >> (bitshift sağ)

Bu operatörler, sol işlenendeki bitlerin sağ işlenen tarafından belirtilen konum sayısına göre sola veya sağa kaydırılmasına neden olur.

```
int a = 5; // binary: 0000000000000101
int b = a << 3; // binary: 0000000000101000 veya ondalıklı 40
int c = b >> 3; // binary: 0000000000000101 veya başlanılan yere 5'e dönüş
```

Resim 7.50: << (bitshift sol) ve >> (bitshift sağ) kullanım örneği

7.8.8. Birleşik Operatörler

a++ (arttırma) ve -- (azaltma)

Bir değişkende arttırma veya azaltma yapılmasını sağlamaktadır. Aşağıdaki örnekleri inceleyiniz.

```
X = 2;
Y = ++ x; // şimdi x 3 içerir  // y 3 içerir
y = x--; // x tekrar 2 içerir, y 4'ü içerir
```

Resim 7.51: ++ (arttırma) ve -- (azaltma) kullanım örneği

```
X ++; // x değerini birer arttırır ve eski x değerini döndürür
++ x; // x değerini birer artırır ve x'in yeni değerini döndürür

X--; // x değerini birer azaltır ve eski x değerini döndürür
--x; // x değerini birer azaltır ve yeni x değerini döndürür
```

Resim 7.52: << (bitshift sol) ve >> (bitshift sağ) 2. kullanım örneği

+= (birleşik artırma), -= (birleşik çıkarma), *= (birleşik çarpma), /= (birleşik bölme) ve %= (birleşik mod)

Bu operatörler değişken üzerinde başka bir sabit veya değişkenle matematiksel işlem yapmak için kullanılmaktadır. Örneği inceleyiniz.

```
x = 2;
x += 4;       // x şimdi 6 içerir
x -= 3;       // x şimdi 3 içerir
x *= 10;      // x şimdi 30 içerir
x /= 2;       // x şimdi 15 içerir
x %= 5;       // x şimdi 0 içerir**

**Mod alınıyor. Mod alma x in y ile
bölümünden kalan sayıdır.
x=15%5; olduğunda x=0 olacaktır.
```

```
X + = y; // x = x + y ifadesine eşdeğerdir
X - = y; // x = x - y ifadesine eşdeğerdir
X * = y; // x = x * y ifadesine eşdeğerdir
X / = y; // x = x / y ifadesine eşdeğerdir
X% = y; //  x = x% y; ifadesine eşdeğerdir

X: herhangi bir değişken türü
Y: herhangi bir değişken türü veya sabit
```

Resim 7.53: += (birleşik artırma), -= (birleşik çıkarma), *= (birleşik çarpma), /= (birleşik bölme) ve %= (birleşik mod) kullanım örnekleri

&= (bitsel lojik ve)

Bitsel Lojik Ve değişkendeki belirli bitleri LOW (düşük) durumuna zorlamak için genellikle bir değişken ve bir sabitle kullanılır. Buna programlama kılavuzlarında "temizleme" veya "sıfırlama" bitleri denmektedir. Bitsel Lojik Ve bir değişkenin geri kalanını değiştirmeden bırakmak, değişkenin 0 ve 1 bitlerini sıfırlamak (sıfıra ayarlamak) için kullanılmaktadır.

```
1 0 1 0 1 0 1 0 değişken
1 1 1 1 1 1 0 0 maske
----------------------
1 0 1 0 1 0 0 0

Değişken değişmez
Bitler silindi. Aşağıdaki örneğe bakın

MyByte = B10101010;
MyByte & = B11111100 == B10101000;
```

Resim 7.54: &= (bitsel lojik ve) kullanım örneği

|= (bitsel lojik veya)

Bitsel Lojik Veya bir değişkende belirli bitlere "ayar" (1 olarak ayarlanır) yapmak için genellikle bir değişkenle ve bir sabitle kullanılmaktadır.

```
1 0 1 0 1 0 1 0 değişken
0 0 0 0 0 0 1 1 maske
----------------------
1 0 1 0 1 0 1 1

Değişken değişmez
                  Bit seti

Aşağıdaki örneğe bakın

MyByte = B10101010;
MyByte | = B00000011 == B10101011;
```

Resim 7.55: |= (bitsel lojik veya) kullanım örneği

7.9. Arduino Tümleşik Geliştirme Ortamının "Değişken" Yapısı

7.9.1. Sabitler

Sabitler Arduino dilinde önceden tanımlanmış ifadelerdir. Programların okunmasını kolaylaştırmak için kullanılırlar. Sabitler gruplar hâlinde sınıflandırılmaktadır.

HIGH | LOW

Okuma veya yazma yaparken dijital pine verilen aktif veya/pasif olma durumunu ifade eder. HIGH ile pin çıkışı aktif edilirken, LOW ile pin çıkışı pasif yapılmaktadır. Pim aktif hâle getirildiğinde (HIGH yapıldığında) 5 Volt ile çalışan kartlarda 3 volttan daha yüksek bir voltaj mevcutken 3.3 Volt ile çalışan kartlarda 2 volttan daha yüksek bir voltaj bulunmaktadır. LOW durumunda ise (LOW yapıldığında) 5 Volt ile çalışan boardlarda 3 volttan daha düşük bir voltaj mevcutken 3.3 Volt ile çalışan boardlarda 2 volttan daha düşük bir voltaj bulunmaktadır.

```
int led= 13;
digitalWrite(led, HIGH); // Ledi yakılıyor
digitalWrite(led, LOW);  // Ledi söndürülüyor
```

Resim 7.56: HIGH | LOW kullanım örneği

INPUT | OUTPUT

Temelde dijital pine verilen modunun giriş ya da çıkış olacağı belirlenmektedir. Dijital pinler INPUT, INPUT_PULLUP, veya OUTPUT olarak kullanılabilmektedir. Dijital pinlerin elektriksel davranışı pinMode() ile değiştirilebilmektedir.

```
int led=13;
int buton=4;
void setup()
pinMode(led, OUTPUT); // Çıkış olarak tanımlandı
pinMode(buton, INPUT); // Giriş olarak tanımlandı
```

Resim 7.57: INPUT | OUTPUT kullanım örneği

LED_BUILTIN

Arduino kartlarının bir çoğu, bir dirençle seri olarak bağlı bir LED'in bulunduğu bir pime sahiptir. Sabit LED_BUILTIN, yerleşik LED'in bağlandığı pinin numarasıdır. Genellikle bu LED dijital pin 13'e bağlanmıştır.

true | false

Arduino da doğruyu ve yanlışı belirtmek için kullanılan iki sabit mantıksal tanımlamadır. Yanlış false 0 (sıfır) olarak tanımlanır. Doğru ise true 1 olarak tanımlanır. Ancak doğru olarak tanımlamanın daha geniş bir anlamı vardır. Boolean anlamında, sıfır olmayan herhangi bir tam sayı doğrudur. Dolayısıyla -1, 2 ve -200 tümüyle Boolean anlamında doğru olarak tanımlanır.

```
int m = true; // m doğru
int n = false; // n yanlıs
int led=13; // led 13. pine tanımlandı
int buton=12; // buton 12. pine tanımlandı
void setup() { } //ana kurulum yapıldı
void loop() { // sonsuz döngü sağlandı
if (buton == m) { // butona basıldı mı? Evet ise
digitalWrite(led, HIGH); // led e 5v ver
delay(1000); //1 saniye bekle
digitalWrite(led, LOW); } //led söndür
}
}
```

Resim 7.58: True | False kullanım örneği

integer constants

Sayı sistemleri için kullanılırlar. Tamsayı sabitleri, 123 gibi doğrudan bir eskizde kullanılan rakamlardır. Varsayılan olarak bu rakamlar int olarak kabul edilir, ancak bunlar U ve L değiştiricileriyle değiştirilebilmektedir. Normalde tamsayı sabitleri taban 10 (ondalık) tamsayılar olarak kabul edilirler, ancak diğer tabanlara sayı girmek için özel gösterim (formatlayıcılar) kullanılabilmektedir.

Decimal kullandığımız 10luk sayı sistemidir.

101 == ((1 * 10^2) + (0 * 10^1) + 1)

Binary ikili sayı sistemidir. 0 ve 1 kullanılmaktadır.

B101 == ((1 * 2^2) + (0 * 2^1) + 1) decimal

Octal sekizlik sayı sistemidir. 0 dan 7 ye kadar sayılardan oluşmaktadır.

0101 == ((1 * 8^2) + (0 * 8^1) + 1) decimal

Hexadecimal on altılık sayı sistemidir. Sembollerden 10 tanesi rakamlarla (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), geri kalan 6 tanesi harflerle (A, B, C, D, E, F) temsil edilmektedir.

0x101 == ((1 * 16^2) + (0 * 16^1) + 1) ==257

Sayı Sistemi	Örnek	Formatı	Karakter
10'luk (decimal)	123	-	
2'lik (binary)	B1111011	"B"	8 bit (0-255)
8'lik (Octal)	0173	"0"	0-7 karakter
16'lık (hexadecimal)	0x7B	"0x"	0-9, A-F

Tablo 7.3: Integer Constants kullanım formatı

U & L

Sayı tanımlamaları varsayılan int olarak kabul edilmektedir. Başka bir veri türüne sahip olan sayıları belirtmek için U, L imzasız veri türü kullanılmaktadır.

- ✓ Sabiti imzalamamış bir veri biçimine zorlamak için bir 'u' veya 'U'. Örnek: 33u
- ✓ Sabiti uzun bir veri formatına zorlamak için bir 'l' veya 'L'. Örnek: 100000L
- ✓ Sabiti imzasız bir uzun sabit hâline getirmek için bir 'ul' veya 'UL'. Örnek: 32767ul

floating point constants

Tamsayı sabitlerine benzer şekilde, kayan nokta sabitleri kodu daha okunabilir hâle getirmek için kullanılmaktadır. 'E' ve 'e', geçerli üs göstergeleri olarak kabul edilir. Örnek

2.34E5=2.34 * 10^5 234000; 67e-12= 67.0 * 10^-12 .000000000067

7.9.2. Veri Tipleri

void

Void anahtar sözcüğü yalnızca işlev bildirimlerinde sadece fonksiyon tanımlanırken kullanılır. Bu işlevin çağrıldığı işleve herhangi bir bilgi döndürmemesi beklenmez. Yani fonksiyonun değer döndürmeyeceği anlamına gelir.

```
// Eylemler ve fonksiyonlar "kurulum" ve
// "döngü" içerisinde gerçekleştirilir.
// Ancak hiçbir bilgi yada program bildirilmez
void setup() {
  // ...
}
void loop() {
  // ...
}
```

Resim 7.59: void kullanım örneği

boolean

Bir boolean doğru veya yanlış olmak üzere iki değerden birini tutar. 0 veya 1 değerlerini "true" ve "false" olarak alır.

```
int LEDpin = 5;        // LED 5 nolu pine bağlanıyor
int switchPin = 13;    // anahtar 13 numaralı pine bağlanıyor
boolean running = false;
void setup()
{
  pinMode(LEDpin, OUTPUT);
  pinMode(switchPin, INPUT);
  digitalWrite(switchPin, HIGH);
}
void loop()
{
  if (digitalRead(switchPin) == LOW)
  {  // anahtara basıldığında normal olarak çalışıyor
    delay(100);                          // anahtar için gecikme sağlanıyor
    running = !running;                  // değişkenler arasında geçiş yapılıyor
    digitalWrite(LEDpin, running);       // LED yakılıyor
  }
}
```

Resim 7.60: boolean kullanım örneği

char

Bir karakter değeri saklayan (1 bayt bellek alan) veri tipidir. Karakter verisini tanımlamak için kullanılmaktadır. Karakter harfleri, tek tırnak işaretleriyle yazılır: 'A' (birden fazla karakter için çift tırnak kullanılır: "ABC"). Ancak karakterler sayı olarak saklanır.

```
char myChar = 'A';
char myChar = 65;   // Her ikiside eşdeğerdir
```

Resim 7.61: char kullanım örneği

unsigned char

1 baytlık belleği kaplayan işaretsiz bir veri türüdür. Bayt veri türü ile aynıdır. 0-255 arası değer alır.

```
unsigned char myChar = 230;
```

Resim 7.62: unsigned char kullanım örneği

byte

Bir bayt, 0'dan 255'e kadar 8 bitlik bir işaretsiz sayı verisi taşır. Binary olarak da işlem yapılabilir.

```
byte b = B10010;  // "B" binary biçimi (B10010 = 18 decimal)
```

Resim 7.63: byte kullanım örneği

int

Tamsayıları saklamak için kullanılan birincil veri türüdür. 16 bit işlemcilerde -32,768 ile 32,767 arası 32 bit işlemcilerde ise -2,147,483,648 ile 2,147,483,647 arasında değişen veri saklanabilir.

```
int ledPin = 13;

int var = val;
//var -int değişken adı
//val -o değişkene atanan değer
```

Resim 7.64: int kullanım örneği

unsigned int

Negatif sayıları saklamak yerine 16 bitlik işlemcilerde sadece 0 ile 65,535 arasında değişen 2 baytlık (16 bit) bir pozitif değeri saklar. 32 bit işlemcilerde 0 ila 4,294,967,295 arasında değişen 4 baytlık (32 bit) bir pozitifdeğeri saklar.

```
unsigned int ledPin = 13;

unsigned int var = val;
//var -işaretsiz int değişken adı
//val -o değişkene atanan değer
```

Resim 7.65: unsigned int kullanım örneği

word

16 bitlik işlemcisi bulunan kartlarda 16 bitlik bir işaretsiz sayı saklanır. 32 bitlik işlemcisi bulunan kartlarda 32 bitlik bir işaretsiz sayı saklanır.

```
word w = 10000;
```

Resim 7.66: word kullanım örneği

long

Sayı saklamak için genişletilmiş boyut değişkenleridir ve 32 bit (4 bayt), -2,147,483,648 ile 2,147,483,647 arasında değişen değeri saklar. Tamsayılar kullanılıyorsa sayıların en az biri "long" olmalı ve bir "L" tarafından takip edilmelidir.

```
long speedOfLight = 186000L;

long var = val;
// var -long değişken adı
// val -o değişkene atanan değer
```

Resim 7.67: long kullanım örneği

unsigned long

Unsigned long değişkenler sayı saklamak için genişletilmiş boyut değişkenleridir ve 32 bit (4 bayt) depolamaktadır. Standart "long"un aksine, unsigned long 0 ile 4,294,967,295 arasında değişen pozitif sayıları saklar, negatif sayıları saklamaz. unsigned long değişken adı = o değişkene atayan değer olarak yazılır.

```
unsigned long time;
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  Serial.print("Time: ");
  time = millis();
  // program başlatıldıktan sonra zamanı yazdırır
  Serial.println(time);
  // bir saniye bekleniyor
  delay(1000);
}
```

Resim 7.68: unsigned long kullanım örneği

short

Short, 16 bitlik (2 baytlık) bir veri türüdür. -32,768 ile 32,767 arasında değişen değeri saklar.

```
short ledPin = 13;

short var = val;
// var -short değişken adı
// val -o değişkene atanan değer
```

Resim 7.69: short kullanım örneği

float

Ondalıklı sayılar için kullanılan veri türüdür. Ondalıklı sayılar, tamsayılara göre daha yüksek çözünürlüğe sahip oldukları için genellikle analog ve sürekli değerleri yaklaştırmak için kullanılırlar. Ondalık sayıları 3.4028235E + 38 ile -3.4028235E + 38 arasında olabilirler. Bunlar 32 bit (4 bayt) bilgi olarak saklanırlar.

```
float myfloat;
float sensorCalbrate = 1.117;

float var = val;
// var -float değişken adı
// val -o değişkene atanan değer
```

Resim 7.70: float kullanım örneği

double

Double, ondalıklı sayılar için kullanılan veri türüdür. Burada hassasiyet 2 kat yüksektir. "double" uygulaması tam olarak "float" ile aynıdır. Uno ve diğer ATMEGA tabanlı kartlarda 4 bayt yer kaplar. Arduino Due'da "double" 8 bayt (64 bit) hassaslığa sahiptir.

string - char array

Yazı verisi depolamak için kullanılır. Karakter dizisidir. Metin dizeleri string ile gösterilir. Yandaki gösterimlerin tümü geçerli kullanım şekillerine örnektir.

```
char Str1 [15];
char Str2 [8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
char Str3 [8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\ 0'};
char Str4 [] = "arduino";
char Str5 [8] = "arduino";
char Str6 [15] = "arduino";
```

Resim 7.71: string - char array kullanım örneği

substring()

String, içindeki kelimeden kaç karakter alacağını belirtmek için kullanılır.

```
String cumle = "Robot Programlama Dersi";
if (cumle.substring(3,9) == "Programlama") {
//
}
```

Resim 7.72: substring() kullanım örneği

String – object

String sınıfı, metin dizelerini karakter dizilerinden daha karmaşık yollarla kullanılmasına ve değiştirilmesine olanak tanır. Dizeler bir araya getirebilir, onlara eklenebilir, alt dizeler aranıp değiştirilebilir. Basit bir karakter diziliminden daha fazla bellek kullanır ancak daha kullanışlıdır.

array

Bir dizi, bir dizin numarasıyla erişilen değişkenlerin bir toplamıdır. Diğer bir ifadeyle her birine indeks numarası ile ulaşılan aynı türdeki veri topluluğudur. Dizi, bilgisayar belleğinde aynı isim altında genellikle aynı tipten çok sayıda veriyi bir arada saklayan veri yapısıdır. Yandaki yöntemlerin tümü, bir diziyi oluşturmak (bildirmek) için kullanılabilecek geçerli biçimlerdir.

```
int myInts [6];
int myPins [] = {2, 4, 8, 3, 6};
int mySensVals [5] = {2, 4, -8, 3, 2};
char message [8] = "merhaba";

mySensVals [0] = 10; // Bir diziye değer atamak
x = mySensVals [4] ; // Bir diziden bir değer almak
```

Resim 7.73: array kullanım örneği

- ✓ Bir dizi "myInts"'de olduğu gibi dizi başlatılmadan tanımlanabilir. Burada 6 farklı "int (integer)" veri tipinde değer tanımlaması yapılmıştır.
- ✓ "myPins"'te açıkça bir boyut seçmeden bir dizi tanımlanmıştır. Bu durumda derleyici elemanları sayar ve uygun büyüklükte bir diziyi otomatik olarak oluşturur.
- ✓ "mySensVals" örneğinde olduğu gibi dizi başlatabilir ve boyutlandırabilir.
- ✓ char türündeki bir diziyi bildirirken, gerekli boş karakteri tutmak için fazladan bir eleman gerekmektedir.
- ✓ Tanımlamış değişkenlere erişim için köşeli parantez ile değişkenin sıra numarası belirtilmektedir. Yani başta tanımlanan yada boş bırakılan indis'i çağırma işlemi gerçekleştirilmektedir. Dikkat edilmesi gereken nokta indisin '0' dan başladığıdır. Yani örnekte yazdığımız 5 farklı değeri olan "mySensVals" kümemizde 4 numaralı elemana erişmek için köşeli parantez içinde [4] değilde [3] yazılmalıdır. Örnekte "mySensVals" kümemize 1. elaman olarak 10 değeri atanmıştır. Yine ayni diziden 4 numaralı değer (2) alınmıştır.

```
int i;
for (i = 0; i < 5; i = i + 1) {
  Serial.println(myPins[i]);
}
```

Resim 7.74: array'ın For döngüsünde kullanım örneği

Dizilerin for döngülerinde kullanımı için yukarıdaki şekli inceleyip, aşağıdaki örneği uygulayınız. Aşağıda verilen örnekte Arduino UNO'nun 2, 3, 4 ve 5 numaralı dijital pinlerine bağlı dört LED'in sırayla bir saniye aralıklarla yanıp sırayla sönmesini kontrol eden array uygulaması yer almaktadır. Arduino UNO'ya LED bağlantılarını yaptıktan sonra örneği dersin sitesinden (7.8.1.5 numaralı uygulama) kopyalayarak Arduino IDE'de test ediniz.

```
const int LEDsayisi = 4; // Dizide kaç eleman olacağı tanımlanıyor
const int pinLEDS[LEDsayisi] = {2,3,4,5}; // LED'ler 2, 3, 4, ve 5
nolu pinlere bağlanıyor

void setup()
  {
  for(int m = 0; m < LEDsayisi; m++)  // m değişkeni LED sayısı kadar
artırılıyor
  {
  pinMode(pinLEDS[m], OUTPUT);        //   LED'lerin   bağlı   olduğu
pinler çıkış olarak ayarlanıyor
  }
 }

void loop()
  {
  for(int m = 0; m < LEDsayisi; m++)  // m değişkeni LED sayısı kadar
artırılıyor
  {
  digitalWrite(pinLEDS[m], HIGH);     // LED'le sırayla yanıyor
  delay(1000);                        // 1 sn bekleniyor
  }

  for(int m = LEDsayisi - 1; m >= 0; m--) // m değişkeni LED sayısı
kadar azaltılıyor
  {
  digitalWrite(pinLEDS[m], LOW);      // LED'le sırayla sondürülüyor
  delay(1000);                        // 1 sn bekleniyor
  }
 }
```

7.9.3. Dönüşümler

char()

Herhangi bir değeri char veri türüne dönüştürür.

```
char myChar = 'A';
char myChar = 65; // Eşdeğeri
```

Resim 7.75: char() kullanım örneği

byte()

Herhangi bir değeri byte veri türüne dönüştürür.

```
byte b = B10010; // "B" Binary biçimi
(B10010 = 18 ondalık biçimi)
```

Resim 7.76: byte() kullanım örneği

int()

Herhangi bir değeri int (integer-tam sayı) veri türüne dönüştürür.

```
int led= 4;
```

Resim 7.77: int() kullanım örneği

word()

Herhangi bir değeri word veri türüne dönüştürür ya da iki bayt bir kelime oluşturur. Kelime oluşturuken word (h, l) şeklinde ifade edilir. Burada H: sözcüğün üst sırası (en soldaki), L: sözcüğün en düşük (en sağdaki) baytını ifade eder.

```
word w = 10000;
```

Resim 7.78: word() kullanım örneği

long()

Herhangi bir değeri long veri türüne dönüştürür.

```
long sensor = 253000L;
```

Resim 7.79: long() kullanım örneği

float

Herhangi bir değeri float veri türüne dönüştürür.

```
float myfloat;
float sensorCalbrate = 1.123;
```

Resim 7.80: float() kullanım örneği

7.9.4. Değişken Kapsamları

static

Statik olarak tanımlanan değişkenler yalnızca bir işleve çağrıldığında (o işleve özel) ilk kez oluşturulur ve başlatılırlar. Bu değişkenler sonra silinmeyip bellekte tutulmaktadır ve daha sonra kullanılmak istendiğinde yeniden oluşturulmadan bellekten çağırılmaktadır.

```
static int sayi=1;
```

Resim 7.81: static kullanım örneği

volatile

Volatile ile tanımlanan değişkenler Arduino mikro kontrolorunun ram bölgesine kaydedilir. Kullanılmak istendiğinde doğrudan ram bellekten okunur. Volatile kullanıldığında değişkenin değerini interrupt ile değiştirmek gerekmektedir.

```
int pin = 13;
volatile int state = LOW;
```

Resim 7.82: volatile kullanım örneği

const

Const değişken türü diğer tüm değişkenler gibi kullanılır. Oluşturulan değişkenler sabitlenirler ve değerleri daha sonradan değiştirilemezler.

```
const float pi = 3.14;
const float fi= 1.61;
```

Resim 7.83: const kullanım örneği

7.9.5. Yardımcılar

sizeof()

Sizeof işleci, herhangi bir tipteki değişken türünün bayt sayısını (değişkenin kaç bayt olduğunu) verir veya bir dizinin işgal ettiği toplam bayt sayısını döndürür. Yandaki program bir seferde bir karakterlik bir metin dizisi yazmaktadır. Metnin ifadesini değiştirerek deneyiniz.

```
char mesaj[] = "Robot Programlama";
int i;
void setup(){
Serial.begin(9600); }
void loop() {
for (i = 0; i < sizeof(mesaj) - 1; i++){
Serial.print(i, DEC);
Serial.print(" = ");
Serial.write(mesaj[i]);
Serial.println();
}
delay(5000); }
```

Resim 7.84: sizeof() kullanım örneği

PROGMEM

Bilgiyi SRAM bellek yerine Flash bellekte depolamak için kullanılmaktadır. Programda uzun char yazıldığında sorun çıkarabilmektedir. Bu nedenle çok uzun metinlerin Flash belleğe kaydedilmesi gerekmektedir. PROGMEM değişken değiştiricidir, yalnızca pgmspace.h'de tanımlanan veri türleriyle birlikte kullanılmalıdır. Yani kullanılabilmesi için programa #include <avr/pgmspace.h> kütüphanesi eklenmelidir.

```
#include <avr/pgmspace.h>
const dataType variableName[] PROGMEM = {};   // uygun kullanım şekli
const PROGMEM  dataType  variableName[] = {}; // uygun kullanım şekli
```

Resim 7.85: PROGMEM kullanım örneği

7.10. Arduino Tümleşik Geliştirme Ortamının "Fonksiyon" Yapısı

7.10.1. Dijital Giriş Çıkışlar

pinMode()

Belirtilen pinin giriş (INPUT) yada çıkış (OUTPUT) olacağının tanımlandığı komuttur. A0, A1 vb. olarak adlandırılan Analog giriş pinleri dijital pimler olarak kullanılabilir. Aşağıdaki yazım şekillerinin hepsi kullanılanilir.

```
int ledPin = 13;            // LED digital pin 13'e bağlandı
void setup()
{
  pinMode(ledPin, OUTPUT); // digital pin çıkış olarak tanımlandı
  pinMode(5, OUTPUT);      // Çıkış olarak tanımladı
  pinMode(6, INPUT);       // Giriş olarak tanımladı
}
```

Resim 7.86: pinMode() kullanım örneği

digitalWrite()

Belirtilen pinin aktif (HIGH) yada pasif (LOW) olacağının tanımlandığı komuttur. A0, A1 vb. olarak adlandırılan Analog giriş pinleri dijital pimler olarak kullanılabilir. Aşağıdaki örnek ve yazım şekillerinin hepsi kullanılanilir.

```
int ledPin = 13;           // LED digital pin 13'e bağlandı
void setup()
{
  pinMode(ledPin, OUTPUT); // Digital pin çıkış olarak tanımlandı
void loop()
{
  digitalWrite(ledPin, HIGH); // LED aktif
  delay(1000);                // 1 sn bekleniyor
  digitalWrite(ledPin, LOW);  // LED pasif
  delay(1000);                // 1 sn bekleniyor
}
digitalWrite(13, HIGH); //13. pin aktif (bu şekilde de gösterilebilir)
digitalWrite(13, LOW);  //13. pin pasif (bu şekilde de gösterilebilir)
```

Resim 7.87: digitalWrite() kullanım örneği

digitalRead()

Belirtilen bir dijital pinden, aktif (HIGH) yada pasif (LOW) değerini okur. A0, A1 vb. olarak adlandırılan Analog giriş pinleri dijital pimler olarak kullanılabilir. Aşağıdaki örneği hazırlayınız.

```
int buton = 7; // Buton dijital 7'ye tanımlandı
void setup() {
Serial.begin(9600); //Seri haberleşme hızı
pinMode(buton, INPUT); // Buton pini giriş yapıldı
}
void loop() {
int butondurumu = digitalRead(buton); //Giriş pini okundu
Serial.println(butondurumu); // Seri ekrana yazıldı
delay(250);
}
```

Resim 7.87: digitalRead() kullanım örneği

7.10.2. Analog Giriş Çıkışlar

analogReference()

Analog giriş için kullanılan referans voltajını (yani giriş aralığının üstü olarak kullanılan değeri) konfigüre etmek için kullanılır. Seçenekler şunlardır:

- ✓ **DEFAULT**: Arduino bordlarında çalışma voltajına göre varsayılan 5 volt ya da 3,3 volt olarak belirlenmiştir.

- ✓ **INTERNAL**: Atmega168 ya da Atmega 328'de 1,1 volt, ATmega8'de 2,56 volt geçerlidir. Mega katlar hariçtir.
- ✓ **INTERNAL1V1**: Sadece Arduino Mega 1,1 volt olarak belirlenmiştir.
- ✓ **INTERNAL2V56**: Sadece Arduino Mega 5,56 volt olarak belirlenmiştir.
- ✓ **EXTERNAL**: Kartın üzerinde bulunan AREF pin (sadece 0 - 5V) referans olarak kullanılabilir.

analogRead()

Belirtilen analog pimden değeri okumak için kullanılır. Arduino Uno'da 6 adet, Mini ve Nano'da 8 adet ve Mega'da 16 adet 10 bit analog sinyali dijitale dönüştüren çevirici bulunmaktadır. Okuma işlemi analog bir girişi dijitale çevirerek yapılmaktadır. Bu 0 ile 5 volt arasındaki giriş voltajlarını 0 ile 1023 arasında tam sayı değerlerine bölünmesi anlamına gelmektedir. Böylece 5 volt / 1024 ünite veya birim başına 0,009 volt (4,9 mV) okuma arasında bir çözünürlük sağlamaktadır. Giriş aralığı ve çözünürlük, analogReference() kullanılarak değiştirilebilmektedir. Yandaki örnekte orta ucu analog pine (A1) takılmış bir potansiyometrenin çevrilmesi sonucu elde edilen voltaj seri ekrandan okunmaktadır.

```
void setup() {
Serial.begin(9600);
}
void loop() {
int sensor = analogRead(A1);
float volt = sensor * (5.0 / 1023.0);
Serial.println(volt);
```

Resim 7.88: analogRead() kullanım örneği

analogWrite() -PWM

Bir pine bir analog değer (PWM) yazar. Bir LED'i farklı parlaklıklarda aydınlatmak veya çeşitli hızlarda bir motoru sürmek için kullanılabilir. Belirtilen görev döngüsünde kararlı bir kare dalga oluşturulur. Çoğu pim üzerindeki PWM sinyalinin frekansı yaklaşık 490 Hz'dir. Uno ve benzeri kartlarda 5 ve 6 numaralı pinlerin frekansı ise yaklaşık 980 Hz'dir. Aşağıdaki örnekte orta ucu analog pine (A1) takılmış bir potansiyometrenin çevrilmesi ile 5 numaralı pine takılmış bir LED'in parlaklığı kontrol edilmektedir.

- ✓ Arduino Uno'da bu komut 3, 5, 6, 9, 10. pinler için kullanılabilir.
- ✓ Arduino Mega'da 11, 2-13 ve 44-46. pinler için kullanılabilir.
- ✓ Arduino Due 2 den 13. pine kadar kullanılabilir.

```
int ledPin = 5;       // LED pin 5'e bağlandı
int analogPin = 3;    // Potansiyometre analog pin 3'e bağlandı
int okunan = 0;       // Okunan değeri saklamak için değişken tanımlandı
void setup()
{
  pinMode(ledPin, OUTPUT);   // Çıkış pini ayarlandı
}
void loop()
{
  okunan = analogRead(analogPin);  // Giriş pini okundu
  analogWrite(ledPin, okunan / 4); // AnalogRead değeri 0 ile 1023 arasında
                                   // AnalogWrite değeri 0 ile 255 arasında
}
```

Resim 7.88: analogWrite() -PWM kullanım örneği

analogReadResolution() ve analogWriteResolution()

AnalogReadResolution() ve analogWriteResolution() Arduino Due ve Zero için Analog API'nin bir uzantısıdır. AnalogRead() tarafından döndürülen değerin boyutunu (bit olarak) ayarlar. AVR tabanlı kartlarda geriye dönük uyumluluk için varsayılan olarak 10 bit (0-1023 arasındaki değerlerler döndürür). AnalogWriteResolution(), analogWrite() işlevinin çözünürlüğünü ayarlar. AVR tabanlı kartlarla geriye dönük uyumluluk için varsayılan 8 bit'dir (0-255 arasındaki değerler). Arduino Due ve Zero 12-bit çözünürlüğünde oldukları için 0 ile 4095 arasında çözünürlük sağlamaktadır.

7.10.3. Gelişmiş Giriş Çıkışlar

tone()

Kare dalga üretilmesine olanak sağlar. İstenilen pin istenilen frekansa ayarlanabilmektedir. Verilen sinyal %50 görev döngüsüne sahiptir. Görev döngüsü 1 periyot boyunca HIGH ve LOW kalma süresidir. Zil sesleri çalmak için pin bir piezo ziline veya başka bir hoparlöre bağlanabilir.Bir seferde yalnızca bir ton üretilebilir. Farklı bir pinde zaten bir ton çalıyorsa, tone() çağrısının etkisi olmayacaktır. Yandaki örneği uygulayınız.

```
int hoparlorPin = 12;
int notaSayisi = 2;
int A = 440;
int B = 494;
int notalar[] = {A, B_};
void setup()
{
for (int i = 0; i < notaSayisi; i++)
{
tone(hoparlorPin, notalar[i]);
delay(500);
noTone(hoparlorPin);
delay(20);
}
noTone(hoparlorPin);
}
void loop()
{
}
```

Resim 7.90: tone() ve no tone() kullanım örneği

noTone()

tone() ile üretilen kare dalgayı sonlandırmaya yarar. Hiçbir ton üretilmediğinde hiçbir etkisi olmaz.

shiftOut()

Her seferinde bir bayt veri kaydırır. En büyük (en soldaki) veya en küçük (en sağdaki) önemli bitten başlanır. Her bit bir veri pinine yazılır ve daha sonra bit bulunduğunu belirtmek için bir saat darbesi bir saat pinine uygulanır. Bu şekilde 8 bitlik veri tek bir seri giriş pininden girilir ve işlem sonucunda seri olarak girilen veri paralel çıkışlardan alınır. Bu işlem için özel işlemci (74HC595 Shift Register Entegresi) kullanılır. Görevi Arduino'da az sayıda dijital çıkış pini kullanarak çok sayıda veriyi kontrol etmeyi sağlamaktır.

```
// MSBFIRST için
int data = 500;
// HIGH baytların dışarı kaydırılması
shiftOut(dataPin, clock, MSBFIRST, (data >> 8));
// LOW baytların dışarı kaydırılması
shiftOut(dataPin, clock, MSBFIRST, data);

// LSBFIRST için
data = 500;
// LOW baytların dışarı kaydırılması
shiftOut(dataPin, clock, LSBFIRST, data);
// HIGH baytların dışarı kaydırılması
shiftOut(dataPin, clock, LSBFIRST, (data >> 8));
```

Resim 7.91: shiftOut() kullanım örneği

shiftOut(dataPin, clockPin, bitOrder, value) şeklinde yazılır. dataPin: Data pinini belirler. clockPin: Clock pinini belirler. bitOrder: Bitleri kaydırmak için hangi sıra ile başlanacağını belirler (MSBFIRST: En büyük bitten başlanacağını, LSBFIRST: En küçük bitten başlanacağını belirler). Value: Kaydırılacak olan verilerdir (bayt).

shiftIn()

Bir seferde bir bit veri kaydırır. En çok (en soldaki) veya en az (en sağdaki) önemli bitten başlanır. Her bit için saat pimi HIGH alınır, sonraki bit veri satırından okunur ve saat pimi LOW alınır. Bu bir yazılım uygulamasıdır. Arduino donanım uygulaması daha hızlıdır ama sadece belirli pinler üzerinde çalışan bir SPI kütüphanesiyle sağlanmaktadır. Görevi Arduino'da az sayıda dijital giriş pini kullanarak çok sayıda veriyi kontrol etmeyi sağlamaktır.

byte incoming = shiftIn(dataPin, clockPin, bitOrder) şeklinde yazılır. dataPin: Data pinini belirler. clockPin: Clock pinini belirler. bitOrder: Bitleri kaydırmak için hangi sıra ile başlanacağını belirler (MSBFIRST: En büyük bitten başlanacağını, LSBFIRST: En küçük bitten başlanacağını belirler).

```
int data = 0;  // PIN for data
int clock = 1; // PIN for clock
int latch = 2; // PIN for latch
int value = 0;
void setup() {
 pinMode(data INPUT);
 pinMode(clock, OUTPUT);
 pinMode(latch, OUTPUT);
 Serial.begin(9600);
}
void loop() {
 digitalWrite(latch, LOW);
 // Kayıttan 8 bit (bir bayt) veri okunuyor
 value = shiftIn(data, clock, LSBFIRST, 8);
 digitalWrite(latch, HIGH);
 Serial.println(value);
 delay(1000);
}
```

Resim 7.92: shiftIn() kullanım örneği

pulseIn()

Bir pin üzerinde bir sinyalin (HIGH veya LOW) olduğunu belirlemek için kullanılır. Örneğin değer HIGH ise, pulseIn() pinin HIGH konumuna gelmesini bekler, zamanlamaya başlar, sonra pinin LOW olmasını bekler ve zamanlamayı durdurur. Sinyalin uzunluğu mikrosaniye cinsinden döndürülmektedir.

```
int pin = 7;
unsigned long zaman;
void setup()
{
pinMode(pin, INPUT);
}
void loop()
{
zaman = pulseIn(pin, HIGH);
}
```

Resim 7.93: pulseIn() kullanım örneği

7.10.4. Gecikmeler

millis()

milis, program başladıktan sonra geçen milisaniye sayısını belirlemek için kullanılır. Yaklaşık 50 gün sonra bu süre sıfırlanmaktadır. Komutun kullanımında herhangi bir parametre bulunmamaktadır. Yandaki örneği inceleyiniz.

```
unsigned long sure;
void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.print("Süre: ");
  sure = millis();
  Serial.println(sure);
  delay(1000);
}
```

Resim 7.94: millis() kullanım örneği

micros()

micros, program başladıktan sonra geçen mikrosaniye sayısını belirlemek için kullanılır. Yaklaşık 70 dakika sonra bu süre sıfırlanmaktadır. Komutun kullanımında herhangi bir parametre bulunmamaktadır. Yandaki örneği inceleyiniz. Not: Bir milisaniyede 1.000 mikrosaniye ve 1 saniyede 1.000.000 mikrosaniye vardır.

```
unsigned long sure;
void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.print("Süre: ");
  sure = micros();
  Serial.println(sure);
  delay(1000);
}
```

Resim 7.95: micros() kullanım örneği

delay()

delay program akışını milisaniye cinsinden duraklatmak için kullanılır. Örneğin 1 saniyelik gecikme ihtiyacı için delay(1000) şeklinde ifade edilir. Gecikme fonksiyonu süresince hiçbir algılayıcı, matematiksel hesaplama ya da pin okuması devam edemez. Bu nedenle 10'un milisaniyesinden daha uzun süren olayların zamanlaması için delay() kullanılmasına uygun değildir. Yanda verilen yanıp sonen LED örneğini inceleyiniz.

```
int ledPin = 13;                  // LED pin 13'e bağlı
void setup()
{
  pinMode(ledPin, OUTPUT);        // Pin çıkış olarak ayarlandı
}
void loop()
{
  digitalWrite(ledPin, HIGH);     // LED açık
  delay(1000);                    // 1 sn bekliyor
  digitalWrite(ledPin, LOW);      // LED kapalı
  delay(1000);                    // 1 sn bekliyor
}
```

Resim 7.96: delay() kullanım örneği

delayMicroseconds()

Programı parametre olarak belirtilen süre boyunca (mikro saniye olarak) duraklatmak için kullanılır. Şu anda doğru bir gecikme için üretilen en büyük değer 16383'tür. Bu işlev 3 mikro saniye ve üzeri aralıklarla çok doğru çalışır. Birkaç bin mikrosaniyeden daha uzun olan gecikmeler için bunun yerine delay() kullanılır. Aşağıdaki örneği inceleyiniz.

```
int outPin = 8;                 // Pin 8 belirlendi
void setup()
{
  pinMode(outPin, OUTPUT);      // Pin çıkış olarak ayarlandı
}
void loop()
{
  digitalWrite(outPin, HIGH);   // Pin açık
  delayMicroseconds(50);        // 50 mikrosanye duraklatıldı
  digitalWrite(outPin, LOW);    // pin kapalı
  delayMicroseconds(50);        // 50 mikosaniye duraklatıldı
}
```

Resim 7.97: delayMicroseconds() kullanım örneği

7.10.5. Matematiksel İşlevler

min()

X ve y sayısal değerlerinden en küçük olanını seçmek için kullanılır. min() genellikle bir değişken aralığının alt ucunu sınırlamak içindir. Aşağıdaki örneği inceleyiniz.

max()

X ve y sayısal değerlerinden en büyük olanını seçmek için kullanılır. max()genellikle bir değişken aralığın üst ucunu sınırlamak içindir. Aşağıdaki örneği inceleyiniz.

```
enkucuk=min(30,40); // En küçük 30 olarak belirlenir
enbuyuk=max(30,40); // En büyük 40 olarak belirlenir
sure = min(sure, 20); // Hiçbir zaman 20'yi geçmez
sure = max(sure, 20); // En az 20 olur
```

Resim 7.98: min() ve max() kullanım örneği

abs()

Bir sayının mutlak değerini hesaplamak için kullanılır. Sayı sıfırdan küçükse pozitif değerini, sıfırdan büyükse aynı sayıyı verir.

```
int m= -10;
int n= 10;
veri=abs(m); // Veri 10 olarak belirlenir
veri=abs(n); // Veri 10 olarak belirlenir
```

Resim 7.99: abs() kullanım örneği

constrain()

Bir sayıyı belirli aralıklarla sınırlamak için kullanılır. constrain(x,a,b) şeklinde tüm veri türleri için kullanılır. x: herhangi türdeki bir sayı, a: alt aralık, b: üst aralık. Örnekte sensör değeri 10 ile 150 arasında sınırlandırılmıştır. Sensör değeri 10'dan küçükse çıktı 10, 150'den büyükse 150 olur. 10 ile 150 arasında bir değer ise o değer çıktı olarak kullanılır.

```
sensor = constrain(sensor, 10, 150);
// Sensor değeri 10 ile 150 arasında sınırlandırılmış
```

Resim 7.100: constrain() kullanım örneği

map()

Sensor değerini belli aralıkta tutarak istenilen aralığa dönüştürmek için kullanılır. Bu amaçla gerekirse bir dizi aralığı tersine çevirmek için de kullanılabilir. Map() işlevi tamsayı matematiği kullanır, böylece kesirler üretmez, kesirli kalanlar kesilir ve yuvarlamaz veya ortalamalanmaz. map(value, fromLow, fromHigh, toLow, toHigh) şeklinde ifade edilir.

- ✓ **value:** Sensorden gelen değer
- ✓ **fromLow:** Değerin geçerli aralığının alt sınırı
- ✓ **fromHigh:** Değerin geçerli aralığının üst sınırı
- ✓ **toLow:** Değerin hedef aralığının alt sınırı
- ✓ **toHigh:** Dğerin hedef aralığının üst sınırı

```
void setup() {}
void loop()
{
  int sensor = analogRead(0);
  sensor = map(sensor, 0, 1023, 0, 255);
  analogWrite(9, sensor);
}
```

Resim 7.101: map() kullanım örneği

Yanda verilen örnekte sensörden okunan değer 0 ile 1023 arasındadir fakat 0 ile 255 arasında bir değere dönüştürülmektedir.

pow()

Bir sayının üstsel kuvvetini almak için kullanılır. Bir sayıyı kesirli bir kuvvete yükseltmek için de kullanılabilir. Değerlerin veya eğrilerin üstel haritalamasını üretmek için yararlı bir işlevdir. pow(base, exponent) şeklinde ifade edilir. Base: Taban sayısı, exponent: Üstsel kuvvet. Yanda "for döngüsü" boyunca bir değişkenin katlanarak kuvvetini alan bir örnek verilmiştir.

```
for (int i = 0; i < 10; i++) {
// For döngüsü devam ettiği için veri katlanarak büyür
veri = pow(2, i);
}
```

Resim 7.102: pow() kullanım örneği

sqrt()

Bir sayının karekökünü hesaplamak için kullanılır. Sayı herhangi bir veri türünde olabilir.

```
islem=sqrt(65536)// Sonuç 265'dır
```

Resim 7.103: sqrt() kullanım örneği

7.10.6. Trigonometri İşlevleri

Trigonometrik işlemlerin yaptırılabilmesi için "math.h" kütüphanesinin kullanılması gereklidir. Kütüphane #include "math.h" şeklinde kullanıma alınır.

sin()

Bir açının sinüsünü radyan cinsinden hesaplar. Sonuç -1 ile 1 arasında olacaktır.

cos()

Bir açının kosinusünü radyan cinsinden hesaplar. Sonuç -1 ile 1 arasında olacaktır.

tan()

Bir açının tanjantını radyan cinsinden hesaplar. Sonuç negatif sonsuzluk ile pozitif sonsuzluk arasında olacaktır.

```
#include "math.h"
islem=sin(65); // Sonuç 0,826828679 olacaktır
islem=cos(65); // Sonuç -0,562453851 olacaktır
islem=tan(65); // Sonuç -1,470038258 olacaktır
```

Resim 7.104: Trigonometrik işlev örnekleri

7.10.7. Karakterler

Bir karakterin ne tür bir karakter olduğunu kontrol etmek için kullanılmaktadır. Aşağıda verilen uygulama örneği dersin sitesinde 7.10.7 numarasıyla yer almaktadır. Uygulamayı indirerek test ediniz.

isAlphaNumeric(): Bir karakterin alphanumeric olup olmadığını kontrol eder.

isAlpha(): Bir karakterin alpha olup olmadığını kontrol eder.

isAscii(): Bir karakterin Ascii tablosundaki değerini verir.

isWhiteSpace(): Bir karakterin bir boşluk olup olmadığını kontrol eder.

isControl(): Bir karakterin kontrol karakteri olup olmadığını kontrol eder.

isDigit(): Bir karakterin dijital karakter olup olmadığını kontrol eder.

isGraph(): Bir karakterin grafik karakter olup olmadığını kontrol eder.

isLowerCase(): Bir karakterin küçük harfli bir karakter olup olmadığını kontrol eder.

isPrintable(): Bir karakterin yazdırılabilir bir karakter olup olmadığını kontrol eder.

isPunct(): Bir karakterin noktalama işareti olup olmadığını kontrol eder.

isSpace(): Bir karakterin boşluk olup olmadığını kontrol eder.

isUpperCase(): Bir karakterin büyük harfli bir karakter olup olmadığını kontrol eder.

isHexaDecimalDigit(): Bir karakterin geçerli heksedesimal (onaltılık) rakam olup olmadığını kontrol eder. Yukarıdaki örnek program gönderilen herhangi bir karakterin karakter analizini yapmaktadır.

7.10.8. Rastgele Sayılar

randomSeed()

randomSeed() rastgele sayı üretimini rastgele sıralamayla rastgele bir noktadan başlatır. Bu dizi çok uzun ve rastgele iken daima aynıdır. Üretilen bir dizi sıranın farklı olması istendiğinde rastgele sayı üreteci, bağlantısız bir pin üzerinden (analogRead() gibi) oldukça rastgele bir girdi ile başlatılabilir. Rastgele sayı üretilirken bir Seed değeri alınır. Bu algoritmalarla uzun bir sayı listesi hesaplanır. Seed belirtilmezse şu anki zaman alınır ve sayılar hep aynı sırada rastgele olarak üretilir. Yandaki örnek 0 ile 249 arasında rastgele sayı üretmektedir.

```
long rasgelesayi;
void setup(){
Serial.begin(9600);
randomSeed (analogRead (1));
}
void loop(){
rasgelesayi = random(250);
Serial.println(rasgelesayi);
delay(100);
}
```

Resim 7.106: randomSeed() kullanım örneği

```
Serial.println();}
void loop() {
if (Serial.available() > 0) {
int thisChar = Serial.read();
Serial.print("Gonderilen: \'");
Serial.write(thisChar);
Serial.print("\' ASCII Degeri: ");
Serial.println(thisChar);
if (isAlphaNumeric(thisChar)) {
  Serial.println("Alphanumerik Degeri");
}
if (isAlpha(thisChar)) {
  Serial.println("Bu Alfabetik!");
}
if (isAscii(thisChar)) {
Serial.println("Bu ASCII karakter");
}
if (isWhitespace(thisChar)) {
  Serial.println("Bu Bos Karakter");
}
if (isControl(thisChar)) {
  Serial.println("Bu kontrol karakteri");
}
if (isDigit(thisChar)) {
  Serial.println("Bu Dijital Karakter");
}
if (isGraph(thisChar)) {
Serial.println("Bosluk Degil Yazdirilabilir Karakter");
}
if (isLowerCase(thisChar)) {
  Serial.println("Kucuk Harf");
}
if (isPrintable(thisChar)) {
  Serial.println("Yazdirilabilir");
}
if (isPunct(thisChar)) {
  Serial.println("Noktalama Isareti");
}
if (isSpace(thisChar)) {
  Serial.println("Bosluk Karakteri");
}
if (isUpperCase(thisChar)) {
  Serial.println("Buyuk Harf");
}
if (isHexadecimalDigit(thisChar)) {
Serial.println("Gecerli Heksadesimal Dijit Var");
}
Serial.println();
Serial.println("Baska Bir Karakter Girin:");
Serial.println();
}
```

Resim 7.105: Karakterlerin kullanım örneği

random()

Minimum ve maksimum olarak belirtilen sayılar arasında rastgele sayılar üretmek için kullanılır. Üretilen bir dizi sıranın farklı olması istendiğinde rastgele sayı üreteci, bağlantısız bir pin üzerinden (analogRead() gibi) oldukça rastgele bir girdi ile başlatılabilir. Yandaki örnek 10 ile 39 arasında rastgele sayı üretmektedir.

```
long rasgelesayi;
void setup() {
Serial.begin(9600); }
void loop(){
rasgelesayi = random(10,40);
Serial.println(rasgelesayi);
delay(100);
}
```

Resim 7.107: random() kullanım örneği

7.10.9. Bit ve Bayt'lar

lowByte(): Bir değişkenin (örneğin bir sözcük) düşük sıralı (en sağdaki) baytı için kullanılır.

highByte(): Bir kelimenin üst sıra (en soldaki) baytını (veya daha büyük bir veri türünün en düşük ikinci baytını) ayıklamak için kullanılır.

bitRead(): Herhangi bir bit'i okumak için kullanılır.

bitWrite(): Bir sayısal değişken bit yazmak için kullanılır.

bitSet(): Bir sayısal değişkenin bit'ini ayarlamak için kullanılır.

bitClear(): Bir sayısal değişkenden bit silmek için kullanılır.

bit(): Belirtilen bit değerini hesaplamak için kullanılır.

7.10.10. İnterruptlar (Kesmeler)

Arduino üzerinde bir programı yürütürken yürümekte olan programın duraklatıp arada başka bir programın çalıştırılması için kesmeler kullanılır. Arada çalıştırılan program komutları yerine getirildikten sonra ana program kaldığı yerden devam eder. Arduino'da farklı görevlerde kullanılmak üzere çeşitli Interrupt'lar (kesmeler) bulunur. Zaman kesmesi (timer interrupt) ve dış kesmeler (external İnterrupt) en yaygın kullanılan Arduino kesmeleridir.

interrupts()

Kesmeler, bazı önemli görevlerin arka planda yapılmasını sağlamak için kullanılır.

noInterrupts()

Kesmeleri devre dışı bırakmak için kullanılır, (interrupts() komu ile yeniden etkinleştirilebilir). Yanda kullanım şekli gösterilmektedir.

```
void setup() {}
void loop()
{
  noInterrupts(); // Kritik zamana duyarlı kod burada
  interrupts();   // Diğer kodlar burada
}
```

Resim 7.108: interrupts() kullanım şekli

7.10.11. Harici Interruptlar (Kesmeler)

attachInterrupt()

Arduinonun belli bir pinine gelen sinyalle belli bir fonksiyonun ya da önceden belirlenmiş bir alt programın çalıştırılmasını sağlamak için kullanılır. Kullanılan Arduinonun türüne göre dijital pinlerden bazıları attachInterrupt pini olarak da işlev görmektedir. Aşağıdaki tabloda farklı Arduino modellerine göre kullanılabilir attachInterrupt pinleri listelenmiştir.

Kart Türleri	Dijital Pin Interrupt						
	Int.0	Int.1	Int.2	Int.3	Int.4	Int.5	...
Uno, Nano, Mini ve diğer 328 temelli kartlar	2	3					
Mega, MEga2560, MegaADK	2	3	18	19	20	21	
Micro, Leonardo, diğer 32u4 temelli kartlar	0	1	2	3	7		
Zero	4. hariç tüm pinler						
Due	Tüm dijital pinler						
MKR1000 Rev.1	0, 1, 4, 5, 6, 7, 8, 9, A1, A2 interrupt numarası = pin numarası						

Tablo 7.4: Arduino modellerine göre kullanılabilir attachInterrupt pinleri

Kullanımı attachInterrupt(pin, fonksiyon, mod) şeklindedir. **Pin:** Kullanılan interrupt pinidir. **Fonksiyon:** Interrupt tetiklendiğinde yapılacak işlemlerin içinde bulunduğu fonksiyonunu (Void loop'un altına yazılmalıdır), **Mod:** Kesmeye ne zaman girileceğini ifade eder. Kullanılan modlar şu şekidedir:

LOW: Dijital pindeki gerilim 0 olduğunda kesmeye girmesini sağlar.

CHANGE: Belirli dijital pinde oluşacak her gerilim değişiminde kesmeye girmesini sağlar. Yani pindeki gerilim 0'dan 5'e yükseldiğinde veya 5'ten 0'a düştüğünde kesmeye girer.

RISING: Yükselen kenar olduğunda kesmeye girmesini sağlar. Yani dijital pindeki gerilim 0'dan 5 volta çıktığında kesmeye girer.

FALLING: Düşen kenar olduğunda kesmeye girmesini sağlar. Yani dijital pindeki gerilim 5'ten 0 volta çıktığında kesmeye girer.

HIGH: Arduino Due, Zero ve MKR1000'da Dijital pindeki gerilim 5 volt olduğunda kesmeye girmesini sağlar.

```
int ledPin = 12;
boolean ledDurumu = LOW;
int butonPin = 3;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(butonPin,INPUT);
  attachInterrupt(1, yanson, CHANGE);
}
void loop() {
}
void yanson() {
  ledDurumu = !ledDurumu;
  digitalWrite(ledPin, ledDurumu);
}
```

Resim 7.109: attachInterrupt() kullanım örneği

Yukarıdaki örnekte bir Led, buton ile kontrol edilmektedir. Bunun için 1 adet interrupt, CHANGE modunda kullanılmıştır.

detachInterrupt()

Verilen kesmeyi kapatmak için kullanılır. Devredışı bırakılacak kesmenin pin numarası parantez içine girilmelidir

Timer Interrupt

Zaman kesmesi (timer interrupt), belirli süre aralıklarında belirli görevlerin yapılabilmesi için kullanılır. Örneğin bir Led'in saniyede bir yakıp söndürülmesi işleminde. Bu işlem için loop fonksiyonunun kullanılması yerine, zaman kesmesinin kullanılması ile kesme her saniyede bir Arduino'ya haber vererek, LED'in yakılıp söndürülmesini sağlamaktadır. Timer interruptu kullanmak için hazır kütüphane olan <TimerOne.h> kütüphanesi kullanılmalıdır. Yandaki örnek bir LED'i 1'er saniye aralıklarla yakıp söndürmektedir.

```
#include <TimerOne.h>
void setup()
{
pinMode(13, OUTPUT);
Timer1.initialize(100000); // 1 saniyede tetikleniyor
Timer1.attachInterrupt( kontrolet ); }
void loop() {
void kontrolet()}
{
digitalWrite(13);
digitalRead(13) ^ 1 );
}
```

Resim 7.110: Timer Interrupt() kullanım örneği

7.11. Arduino Tümleşik Geliştirme Ortamında Seri Haberleşme

Arduino kartlarında seri iletişim TX / RX pinleri ve USB aracılığıyla gerçekleştirilir.

Seri bağlantı Arduino kartı ile bir bilgisayar veya diğer cihazlar arasındaki iletişim için kullanılır. Tüm Arduino kartları, en az bir seri porta (aynı zamanda UART veya USART olarak da bilinir) sahiptir. Dijital pin 0 (RX) ve 1 (TX) üzerinden ve ayrıca bilgisayar üzerinden (USB) aracılığıyla iletişim kurulabilir. Bu nedenle USB kullanıldığı sürece, dijital giriş veya çıkış için 0 ve 1 numaralı pimler kullanılamazlar. Bir Arduino kartıyla iletişim kurmak için Arduino ortamının dahili seri monitörü de kullanılabilir. Bu amaçla araç çubuğundaki seri monitör düğmesi tıklanarak ve "begin()" çağrısında kullanılan aynı baud hızı seçilmelidir. Birçok seri haberleşme fonksiyonu bulunmaktadır. Burada temel olanlara yer verilmiştir.

Serial.begin()

Bilgisayar ile Arduino arasında seri iletişimi başlatmak için kullanılır. Seri veri iletimi için veri hızı saniyedeki bit sayısı (baud) cinsinden ayarlanır. Veri kaybı yaşanmaması için Arduino üzerinde ayarlanan baud oranı ile bilgisayar üzerinde ayarlanan baud oranı aynı olmalıdır. 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, ve 115200 baud oranlarından biri alınan ve gönderilen verileri hızı olarak seçilebilir.

if (Serial): Belirtilen seri portun hazır olup olmadığını gösterir.

Serial.available(): Seri bağlantı noktasından okumak için kullanılabilir bayt (karakter) sayısını öğrenmeyi sağlar.

Serial.availableForWrite(): Yazma işlemini engellemeden seri arabelleğe yazılabilecek bayt (karakter) sayısını öğrenmek için kullanılır.

```
void setup() {
  Serial.begin(9600); // Seri iletim başlatılır ve portun açılmasını beklenir
  while (!Serial) {    // Seri portun (USB) bağlanması beklenir.
    ;
  }
}
void loop() {
 // Program noormal olarak devam eder
}
```

Resim 7.111: if(Serial)ve Serial.begin kullanım şekli

```
int gelenByte = 0;   // Gelen seri veri için değişken tanımlanıyor
void setup() {
  Serial.begin(9600); // Seri port 9600 bps veri hızına ayarlanıyor
}
void loop() {
  if (Serial.available() > 0) {    // Eğer veri alınmışsa
  gelenByte = Serial.read();       // Gelen bayt okunuyor
  Serial.print("Aldım: ");         // Seri ekrana "Aldım: " yazılıyor
  Serial.println(gelenByte, DEC); // Alınan bayt seri ekranda gösteriliyor
}
}
```

Resim 7.112: Serial.available(), Serial.read(), Serial.print() ve Serial.printIn() kullanım örneği

Serial.begin(): Seri veri iletimi için veri hızını saniyedeki bit sayısı (baud) cinsinden ayarlamak için kullanılır.

Serial.end(): Seri haberleşmeyi devre dışı bırakmak için kullanılır. Fakat genel giriş ve çıkış işlemleri için RX ve TX pinlerinin kullanılmasına izin verir.

Serial.findUntil(): Verilen uzunluktaki hedef dizgi bulunana kadar seri arabelleğinden veri okumak için kullanılır.

Serial.flush(): Giden seri iletim verilerinin tamamlanmasını beklemek için kullanılır.

Serial.parseFloat(): ilk geçerli kayan nokta sayısını seri arabelleğinde döndürmek için kullanılır.

Serial.parseInt(): Gelen seri akıştan bir sonraki geçerli tam sayıyı aramak için kullanılır.

Serial.peek(): Gelen seri verilerin bir sonraki baytını (karakterini) dahili seri arabelleğinden kaldırmadan döndürmek için kullanılır.

Serial.print(): Veriyi seri porta okunabilir ASCII metni olarak yazdırmak için kullanılır.

Serial.println(): Veriyi seri porta okunabilir ASCII metni olarak yazdırmak için kullanılır. Fakat sonuna satır sonu ekleyerek alt satıra geçmesi sağlanır.

Serial.read(): Gelen seri iletim verilerini okumak için kullanılır.

Serial.readBytes(): Karakterleri seri porttan bir arabelleğe okumak için kullanılır. Arabelleğe yerleştirilen karakter sayısını döndürür.

Serial.readBytesUntil(): Seri arabellekteki karakterleri bir diziye okumak için kullanılır. Arabelleğe okunan karakter sayısını döndürür.

Serial.readString(): Seri arabellekteki karakterleri bir dizeye okumak için kullanılır.

Serial.readStringUntil(): Seri arabellekteki karakterleri bir dizeye okumak için kullanılır.

Serial.setTimeout(): Serial.readBytesUntil (), Serial.readBytes (), Serial.parseInt () veya Serial.parseFloat () işlevlerini kullanırken seri veri beklenecek maksimum milisaniye değerini ayarlamak için kullanılır. Varsayılan süre 1000 milisaniyedir.

Serial.write(): İkili verileri seri bağlantı noktasına yazmak için kullanılır. Bu veriler bir bayt veya bayt serisi olarak gönderilir.

```
void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.write(45); // Değeri 45 olan bir bayt yazılıyor
// "Merhaba" dizesini gönderir ve dizgenin uzunluğunu döndürür
  int bytesSent = Serial.write("Merhaba");
```

Resim 7.113: Serial.write() kullanım şekli

serialEvent(): Veri mevcut olduğunda çağırılması için kullanılır. Bu verileri okumak için Serial.read() kullanılır.

7.12. Arduino Tümleşik Geliştirme Ortamında Seri Haberleşme Protokolleri

Dijital sistemlerde kablolu seri haberleşme ile ilgili birçok standart protokol bulunmaktadır. SPI ve I²C protokolleri bunlara örnek olarak verilebilir. Arduino kartı, diğer Arduino kartlarla veya algılayıcılarla (sensörlerle) haberleşmek için bu haberleşme protokollerini kullanmaktadır. Bu protokollerin kullandıkları uç sayıları, ulaşabilecekleri maksimum hızları ve kullanım şekilleri birbirinden farklıdır.

7.12.1. I²C Veri Yolu

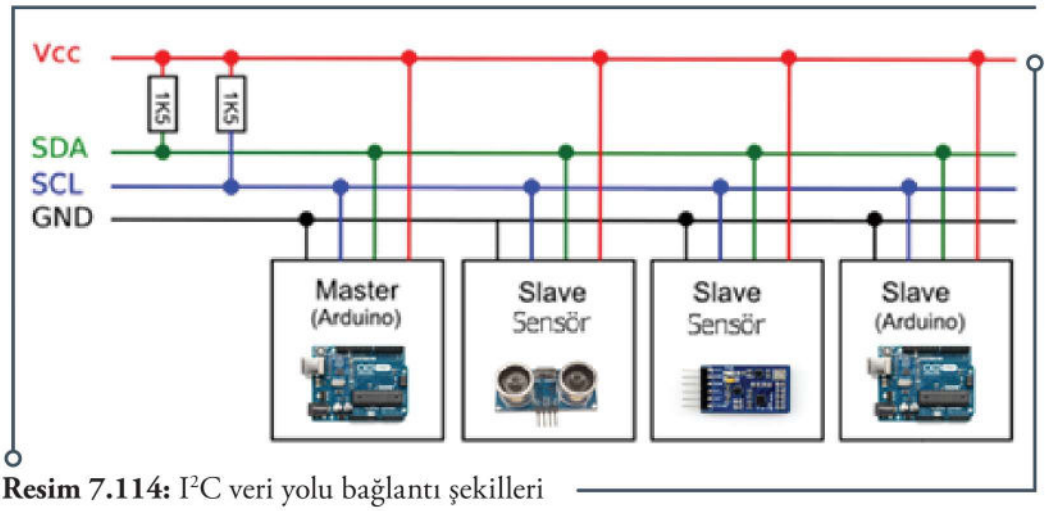
I²C (Inter-Integrated Circuit), oldukça hızlı veri aktarımına olanak tanıyan seri haberleşme türlerindendir. Bir arada çalışan, belirli aralıklarla birbiriyle haberleşen, çeşitli çevresel cihazların çok az harici donanım gereksinimiyle haberleşmelerini sağlar. Uzun mesafeli haberleşmelerde tercih edilmez. Genellikle kısa mesafeli ve düşük veri aktarım hızının yeterli olduğu yerlerde kullanılır. Haberleşme senkron (eş zamanlı) olarak gerçekleştirilir. Haberleşme için toprak hattı dışında SDA (SerialDAta) ve SCL (SerialCLock) olmak üzere iki hat bulunmaktadır. Haberleşme hızı 400kbps'ye kadar çıkabilmektedir.

I²C ile birden fazla cihaz adresleme planı içerisinde bulunduğu için kolayca haberleşebilmektedir. SDA ve SCL pinleri, kullanılan Arduino türüne göre değişiklik göstermektedir. Arduino türlerine göre SDA ve SCL pinleri aşağıdaki tabloda gösterilmiştir.

Arduino Kart Türü	SDA Pini	SCL Pini
Arduino Uno	A4	A5
Arduino Mega	20	21
Arduino Leonardo	2	3
Arduino Due	20	21
Arduino Nano	A4	A5

Tablo 7.5: Arduino türlerine göre SDA ve SCL pinleri

I²C haberleşmesinde, haberleşmeyi kontrol eden master cihazı bulunmalıdır. Haberleşmenin gerçekleşebilmesi için haberleşme hattına en az bir adet de slave (köle) cihaz bağlanmalıdır. Hatta bağlanan birden fazla slave cihazlardan hangisinin veri aktaracağına, master cihaz karar vermektedir. Böylece hat sayısında bir değişiklik olmadan birden fazla cihazla haberleşmenin yapılması sağlanır. Bağlantı şekilleri aşağıdaki tabloda gösterilmiştir. Haberleşme için #include ile <Wire.h> kütüphanesinin eklenmesi gereklidir. Haberleşmenin tüm hat boyunca hatasız bir şekilde sağlanabilmesi için SDA ve SCL hatları, pull-up dirençlerle VCC hattına bağlanmalıdır.



Resim 7.114: I²C veri yolu bağlantı şekilleri

Aşağıdaki örnekte, robotik uygulamalarda yaygın olarak kullanılan MPU6050 3 eksenli gyro ve 3 eksen açısal ivme ölçer sensörünün I²C bağlantısı yapılarak test edilmesi sağlanmaktadır. Arduino Uno'da sda ve scl I²C pinleri sırayla A4 ve A5 pini olduğundan, sensör üzerindeki sda ve scl bağlantılarının o pinlere yapılması gerekmektedir. Bunların dışında güç pinlerinin bağlanması yeterlidir. Uygulama dersin sitesinde 7.12.1. numarasıyla yer almaktadır. Uygulamayı indirerek test ediniz. Gerekli kütüphaneyi https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050 adresinden indirebilirsiniz.

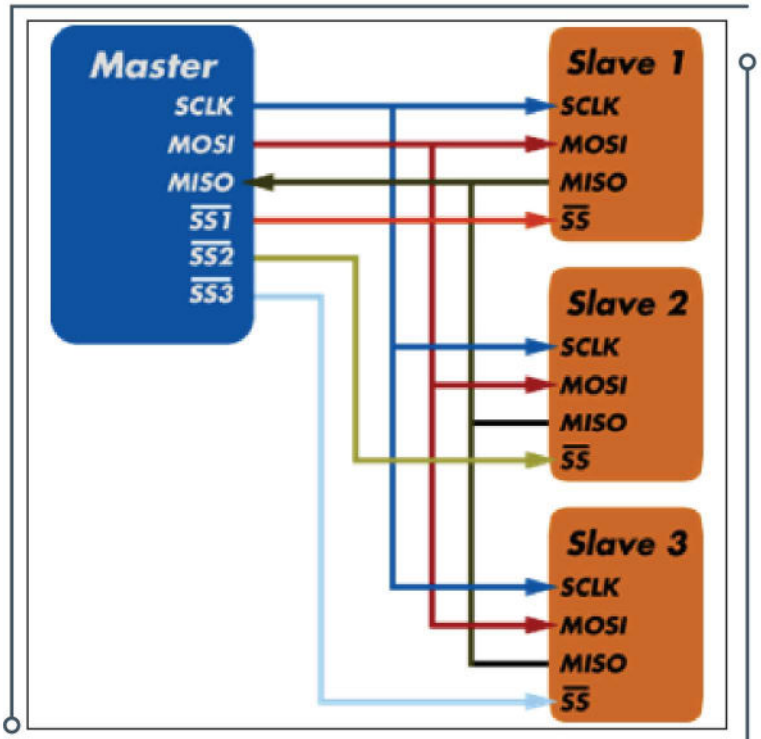
```
#include "I2Cdev.h"  // I2C kütüphanesi ekleniyor
#include "MPU6050.h" // Mpu6050 kütüphanesi ekleniyor
#include "Wire.h"    // Seri bağlantı kütüphanesi ekleniyor
MPU6050 accelgyro;   // Mpu6050 sensör tanımlanıyor
int16_t ax, ay, az;  // İvmeölçer tanımlanıyor
int16_t gx, gy, gz;  // Gyro sensör tanımlanıyor

void setup() {
Wire.begin();                          // Seri bağlantı kütüphanesi başlatılıyor
Serial.begin(9600);                    // Seri iletişim hızı tanımlanıyor
Serial.println("MPU6050 Başlatılıyor"); // Seri ekrana yazdırılıyor
accelgyro.initialize();                // Sensör başlatılıyor
Serial.println(accelgyro.testConnection() ? "MPU6050 Başarılı" : "MPU6050 Başarısız");
}

void loop() {
accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); // İvme ve gyro değerlerini okunuyor

// Açısal ivmeleri ve gyro değerlerini seri ekrana yazdırılıyor
  Serial.print("a/g:\t");
  Serial.print(ax); Serial.print("\t");
  Serial.print(ay); Serial.print("\t");
  Serial.print(az); Serial.print("\t");
  Serial.print(gx); Serial.print("\t");
  Serial.print(gy); Serial.print("\t");
  Serial.println(gz);
  delay (1000); // 1 sn bekletiliyor
}
```

Resim 7.115: I²C veri yolu uygulama örneği

7.12.2. SPI Veri Yolu

SPI (Seri Çevresel Arayüz - Serial Peripheral Interface), Arduino'nun desteklediği senkron (veri alma ve gönderme işleminin eş zamanlı olarak gerçekleştirildiği) seri haberleşme protokolüdür. Veri iletimi tek yönlü olarak sağlanmaktadır. Özellik ve kullanım olarak I²C ile oldukça benzerlik göstermektedir. Bir Arduino'nun diğer Arduino veya sensörlerle kısa mesafede haberleşmesini sağlamak için kullanılmaktadır. SPI protokolünde de I²C'de olduğu gibi bir adet Master cihaz bulunmaktadır. Bu cihaz hatta bağlı diğer çevresel cihazları kontrol etmektedir. SPI bağlantısı için 4 adet pin kullanılmaktadır.

Master ve çevresel cihazlara bağlanan MOSI (Master Out Slave In),



Resim 7.116: SPI veri yolu bağlantı şekilleri

MISO (Master In Slave Out) ve SCLK (Serial Clock) olmak üzere üç adet SPI hattı bulunmaktadır. MOSI: Master cihazdan yollanan verilerin çevresel cihazlara aktarıldığı hattır. MISO: Çevresel cihazlardan (slave) yollanan verilerin master cihaza aktarıldığı hattır. SCLK: SPI haberleşmesinde senkronu sağlayan saat sinyalinin bulunduğu hattır. Saat sinyali master cihaz tarafından üretilmektedir.

SPI protokolünde I²C'den farklı olarak veri hatları tek yönlüdür. Ayrıca çevresel cihazların (slave) adreslerinin olmasına gerek yoktur. Her çevresel cihazın seçim pini bulunur. Bu pine, SS (Slave Select) denir. Bu hattın sayısı kullanılan çevresel cihazların sayısı kadardır. Her cihaz için master cihazından ayrı SS hattı çıkar. SS hattı LOW (0 volt) düzeyinde olan çevresel cihaz, master cihaz ile iletişime başlar. Verilerin gönderilmesi ve alınması clock sinyali ile senkronize bir şekilde gerçekleşir. Veriler byte'lar halinde gönderilir / alınır. Yukarıdaki tabloda 3 adet slave çevresel aygıtın yer aldığı örnek bir SPI haberleşme hattı gösterilmiştir. Bu hatların bağlandığı SPI pinleri Arduino türüne göre değişiklik göstermektedir. Arduino türlerine göre değişen bu pinler aşağıdaki tabloda verilmiştir.

Arduino Kart Türü	MOSI	MISO	SCK	SS (Slave)	SS (Master)
Arduino Uno	11 veya ICSP4	12 veya ICSP1	13 veya ICSP3	10	-
Arduino Mega	51 veya ICSP4	50 veya ICSP1	52 veya ICSP3	53	-
Arduino Leonardo	ICSP-4	ICSP-1	ICSP-3	-	-
Arduino Due	ICSP-4	ICSP-1	ICSP-3	-	-

Tablo 7.6: Arduino türlerine göre kullanılan pinler

SPI veri yolu kullanarak haberleşmek için #include ile <SPI.h> kütüphanesinin uygulamaya eklenmesi gereklidir. Kütüphanenin uygulamaya eklenmesiyle kullanılabilecek fonksiyonlardan bazıları şunlardır:

SPI.begin(): SPI veri yolu haberleşmesini başlatmak için kullanılır.

SPI.end (): SPI veri yolunu devre dışı bırakmak için kullanılır.

SPI.beginTransaction(): Tanımlanan SPI ayarlarını kullanarak SPI veri yolunu başlatmak için kullnılır.

SPI.endTransaction(): Diğer kitaplıkların SPI veri yolunu kullanmasına izin vermek amacıyla mevcut kitaplığın SPI veri yolunu kullanmasını durdurmak için kullanılır.

SPI.usingInterrupt(): Program bir kesme işlemi içinde SPI işlemlerini gerçekleştirecekse, kesme numarası veya adını SPI kitaplığına kaydetmek için kullanılır.

SPI.setClockDivider(): SPI veri yolu haberleşmesinin saatini sistem saatine göre ayarlamak için kullanılır. Standart SPI saati "SPI_CLOCK_DIV4" olarak ayarlanmıştır. Fonksiyonun alabileceği diğer değişkenler; SPI_CLOCK_DIV8, SPI_CLOCK_DIV16, SPI_CLOCK_DIV32, SPI_CLOCK_DIV64, SPI_CLOCK_DIV128'dir.

SPI.transfer(): SPI hattına eşzamanlı olarak veri göndermek veya veri almak için kullanılır.

Aşağıdaki örnekte birbirine SPI pinleri ile bağlı 2 Arduino UNO arasından SPI protokolu ile yapılan veri alışverişinin master ve slav kodları yeralmaktadır. Uygulama, dersin sitesinde 7.12.2. numarasıyla yer almaktadır. Uygulamayı indirerek test ediniz.

Master Kodu

```
#include <SPI.h>
void setup (void) {
   Serial.begin(115200); // Baud hızı 115200 olarak ayarlanıyor
   digitalWrite(SS, HIGH); // Slave Select devre dışı bırakılıyor
   SPI.begin ();
   SPI.setClockDivider(SPI_CLOCK_DIV8);// Sistem saati 8'e bölünüyor
}
void loop (void) {
   char c;
   digitalWrite(SS, LOW); // Slave Select etkinleştiriliyor
   // Test dizesi gönderiliyor
   for (const char * p = "Merhaba, Dünya!\r" ; c = *p; p++) {
      SPI.transfer (c);
      Serial.print(c);
   }
   digitalWrite(SS, HIGH); // Slave Select devre dışı bırakılıyor
   delay(2000);
}
```

Resim 7.117: SPI veri yolu uygulaması master örneği

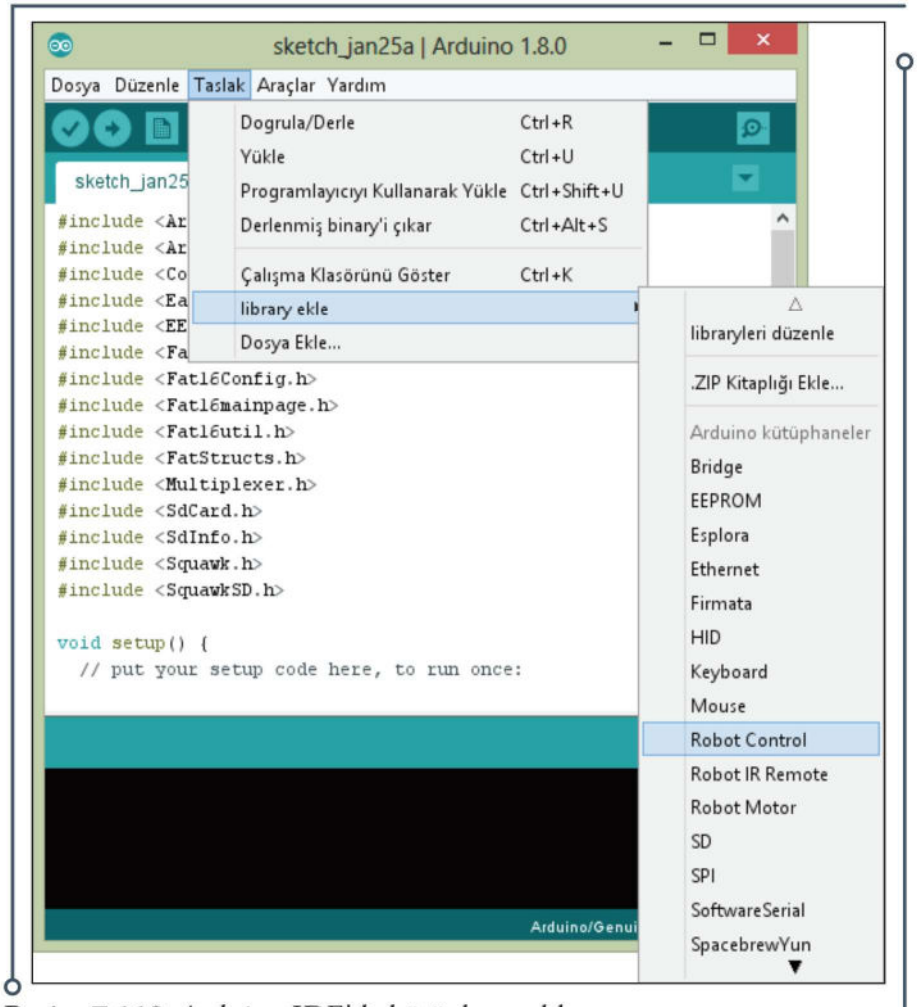
Slave Kodu

```
#include <SPI.h>
char buff [50];
volatile byte indx;
volatile boolean process;
void setup (void) {
   Serial.begin (115200); // Seri iletim Master ile aynı olmalı
   pinMode(MISO, OUTPUT); // Çıktı olarak ayarlanacak şekilde master'ı gönderiliyor
   SPCR |= _BV(SPE); // Köle modunda SPI'yı açılıyor
   indx = 0; // Arabellek boş
   process = false;
   SPI.attachInterrupt(); // Kesme açılıyor
}
ISR (SPI_STC_vect) // SPI kesme rutini
   byte c = SPDR; // SPI Veri Kaydı'ndan bayt okunuyor
   if (indx < sizeof buff) {
      buff [indx++] = c; // Veriler dizinin önbellekteki bir sonraki dizininde saklanıyor
      if (c == '\r') // Sözcüğün sonu kontrol ediliyor
      process = true;
   }
}
void loop (void) {
   if (process) {
      process = false; // İşlem sıfırlanıyor
      Serial.println (buff); // Dizi seri monitörde yazdırılıyor
      indx= 0; // Resetlenerek sıfırlanıyor
   }
}
```

Resim 7.118: SPI veri yolu uygulaması slave örneği

7.13. Kütüphaneler

Arduino kütüphaneleri belirli görevleri yerine getirecek bileşen bilgilerini içeren yapılardır ve bu yapıları sayesinde yapılacak işlemlere daha kısa yoldan ve komut karmaşası olmadan ulaşılmasını sağlarlar. Diğer bir ifadeyle bu kütüphaneler sayesinde mikrodenetleyiciler ve kullanılan bileşenler ayrıntılı olarak bilinmese de kolayca programlanabilmektedir. Arduino projelerini oluşturan elektronik bileşenler, çeşitli sensörler, motorlar, LCD ekranlar, butonlar gibi çok fazla sayıda ek elemana ihtiyaç duymaktadır. Bu elemanlardan bir çoğu ise kendi içerisinde başlı başına bir yapıya, çalışma örgüsüne sahiptirler. Bu elemenların Arduino ile kullanımı, o eleman tarafından yapılacak iş için gerekli kodlamanın da programcı tarafından yapılmasını gerektirmektedir. Bu gerekliliğin sağlanmasını, bu elemanlar ile Arduino programlamasınının birleştirilmesi ve aralarında köprü görevi görecek yapıların oluşturulması görevi kütüphaneler tarafından gerçekleştirilmektedir. Kütüphanesi olmayan bileşenler için kütüphaneler kullanıcı tarafından oluşturabileceği gibi genellikle kullanılan bileşenler, bileşen üretici firmalar tarafından hazırlanırlar veya Arduino IDE ile birlikte hali hazırda yüklü olarak gelirler. Kullanılacak bileşenin ihtiyaç duyduğu kütüphane, hazırlanan programa bir komut aracılığıyla eklenerek bu bileşen kolayca kullanıma hazır hale getirilmektedir. Kütüphaneler içeriğinde belli başlı ek komutlar ve değişkenler içerirler. Programa eklenen bir kütüphane ile bu kütüphane içeriğinde bulunan komutları ve değişkenler hazırlanan program üzerinde kullanıma hazır hale gelmektedir.



Resim 7.119: Arduino IDE'de kütüphane ekleme

Belli başlı güncel kütüphaneler Arduino IDE ile yüklü halde gelmektedir. Bunlar IDE penceresinde

Taslak>library ekle sekmesinde açılan listeden istenilen seçilerek programa eklenmektedir. Burada yüklü gelen tüm kütüphaneler de görülebilmektedir. Ancak kullanılacak kütüphane farklı bir kaynaktan temin ediliyorsa kütüphane dosyası, Arduino'nun kurulu olduğu dosya konumu altında **libraries** klasörü içerisine kopyalanmalıdır. Ayrıca yapılacak programın en başına yazılacak **#include <KütüphaneAdı.h>** şeklinde bir komutla da çalışılan programa eklenmelidir. Burada standart kütüphane Arduino Robot kütüphanesi kısaca açıklanmıştır.

7.13.1. Arduino Standart Kütüphaneleri

EEPROM: Kalıcı hafızaya veri yazmak ve okumak için kullanılmaktadır.

Ethernet / Ethernet 2: Arduino Ethernet Shield, Arduino Ethernet Shield 2 ve Arduino Leonardo ETH kullanarak internete bağlanmak için kullanılmaktadır.

Firmata: Standart bir seri protokol kullanılarak bilgisayardaki uygulamalarla iletişim kurmak için kullanılmaktadır.

GSM: GSM Shield ile bir GSM / GPRS ağına bağlanmak için kullanılmaktadır.

LiquidCrystal: Likit kristal ekranları (LCD) kontrol etmek için kullanılmaktadır.

SD: SD kartları okumak ve yazmak için kullanılmaktadır.

Servo: Servo motorları kontrol etmek için kullanılmaktadır.

SPI: Seri Çevresel Arabirim (SPI) kullanan cihazlar ile iletişim kurmak için kullanılmaktadır.

SoftwareSerial: Herhangi bir dijital pin üzerinden seri haberleşme yapmak için kullanılmaktadır.

Step: Step motorlar kontrol etmek için kullanılmaktadır.

TFT: Arduino TFT ekranda metin, resim ve şekiller çizmek için kullanılmaktadır.

Wi-Fi: Arduino üzerinde Wi-Fi Shield kullanarak internete bağlanmak için kullanılmaktadır.

Wire: İki Tel Arabirimi (TWI / I^2C veri yolu üzerindeki cihaz veya sensörlerden) veri almak ve göndermek için kullanılmaktadır.

7.13.2. Arduino Robot Kütüphanesi

Robot kütüphanesi Arduino IDE 1.0.5 ve sonrası ile birlikte gelmektedir. Kütüphane, Arduino tarafından satışı yapılan Arduino Robot fonksiyonlarına kolayca erişmek için tasarlanmıştır. Robota Stok Kartı Yazılımı yüklendiğinde, robotu oluşturan dâhili algılayıcıları ve aktüatörleri için destek sağlamaktadır. Robotun ve ana kontrol kartı ve motor kontrolünden oluşan iki temel bileşen için gerekli araçları sunar. Her kart ayrı bir programlanabilir işlemciye sahiptir ve robot kütüphanesiyle kolayca kullanılabilir. Örneğin kütüphane; kontrol kartındaki çeşitli algılayıcılar ve çevre birimleri ile arabirim oluşturulmasına, motor hızı ve yönünün kontrol edilmesine, I^2C bağlantı noktasının kontrol edilmesine vb. olanak sağlamaktadır. Bunlar için oluşturulan RobotControl sınıfı fonksiyonları robot kontrol kartına ve kontrol kartındaki tüm giriş çıkışlara (I/O) ve motorlara komut verilmesine olanak sağlamaktadır. RobotMotor sınıfı ise programcının motor kontrolü için kendi bellenimini (firmware) oluşturmasına olanak tanımaktadır.

7.14. Düşünelim / Araştıralım / Uygulayalım

Burada verilen uygulamaların çözümleri dersin web sayfasında uygulama numaralarıyla yer almaktadır. Öncelikle uygulamaları kendiniz yapmaya çalışınız. İlk olarak uygulamaya ait akış diyagramlarını hazırlayarak başlayınız. Sonra uygun bileşenleri biraraya getirerek gerekli bağlantıları hazırlayınız. Son olarak da kod yazmaya geçiniz. Zorlandığınız durumlarda çözüm ve kontrol için örneklere bakmanız daha uygun bir yöntem olacaktır. Hazırlanmış örnekleri de inceleyerek, üzerinde değişiklikler yaparak etkilerini gözlemleyiniz. Farklı çözüm yöntemleri düşünrek araştırma yapınız.

Uygulamalarda kullanılan elektronik devre elemanları ve algılayıcılar ile ilgili ayrıntılı bilgilerin teknik dökümanlarına (datasheet) bakılarak gözden geçirilmesinde yarar bulunmaktadır. Bu dökümanlara İnternet üzerinde yapılacak basit bir aramayla ulaşılabilmektedir. Dökümanları inceleyerek bileşenlerin çalışma yapısı, elektriksel özelikleri ve mikrodenetleyici kartlarla kullanım şekillerini, bağlantı pinleri ve yapılarını öğrenebilirsiniz.

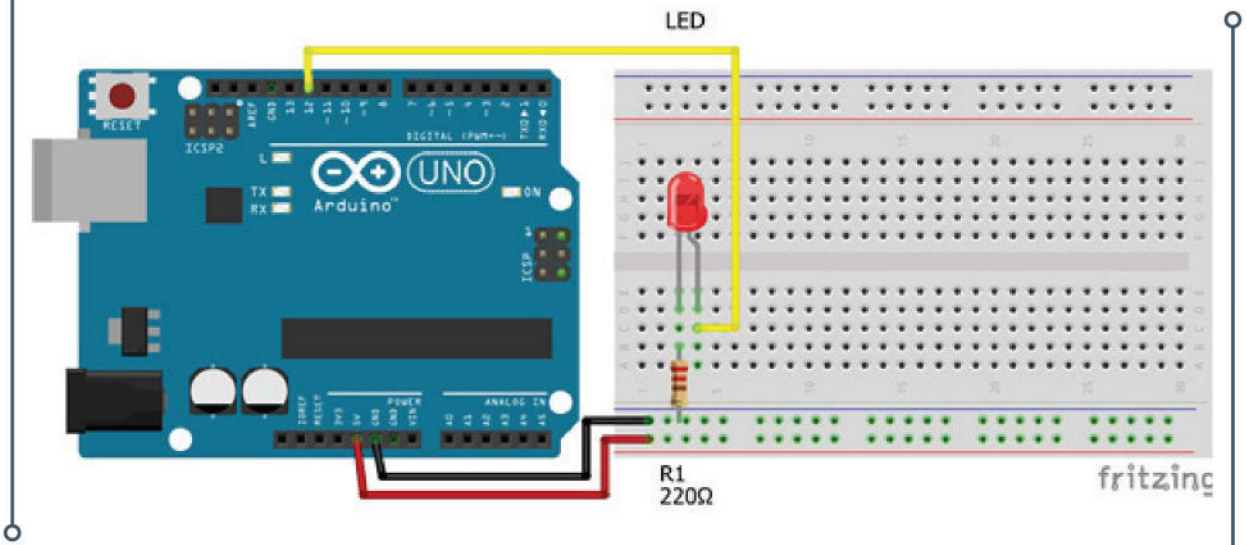
Eğer uygulamalarda kullanılan algılayıcıların arduino ile haberleşmesinde kütüphane kullanılacaksa veya bu zorunluysa uygun kütüphanenin çalışmaya eklenmesi gerektiği unutulmamalıdır. Zira eklenen kütüphane dosyaları algılayıcıdan gelen farklı yapıdaki bilginin okunması görevini üstlenerek kullanımı kolaylaştırmaktadir. Bu kütüphaneler uzmanlar tarafından hazırlanmakta ve algılayıcılardan gelen bilgiyi anlamlı hale getiren kod parçacıklarından meydana gelmektedir. Algılayıcılara ait Arduino kütüphane dosyaslarına da İnternette yapılan kısa bir arama ile ulaşılabilmektedir. İndirilen kütüphane dosyaları Arduino IDE programına ait program dosya kütüphanesine (libraries) kopyalanması gerekmektedir.

Burada yer alan örnek uygulamalarda Arduino UNO R3 versiyonu kullanılmıştır. Aksi belirtilmeyen uygulamalarda besleme voltajları Arduino UNO'nun 5 Volt (+) ve GND (-) pinlerinden bağlanmalıdır. Diğer bağlantılar uygulamada açıklanmış ve Breadboard çizimi üzerinde gösterilmiştir. Temel bileşenler dışında elektronik devre elemanı olarak yalnızca direnç kullanılmıştır. Direnç elektronik devrelerde akımı sınırlayan ve gerilimi bölen, iki uçlu bir elemandır. R harfi ile sembolik olarak gösterilir ve birimi Ohm (Ω)'dur. Örnek uygulamalarda LED'lerin ve gerekli olan diğer bileşenlerin korunması için kullanılmıştır.

Yapılan örnek uygulamalarda; Arduino IDE programı ile gelen hazır örneklerden, İnternette yer alan açık kaynak uygulamalardan ve algılayıcılar için hazırlanan örnek programlardan da yararlanılmıştır. Bütün uygulamalar hazırlanarak test edilmiştir.

7.14.1. LED Yakma Uygulaması

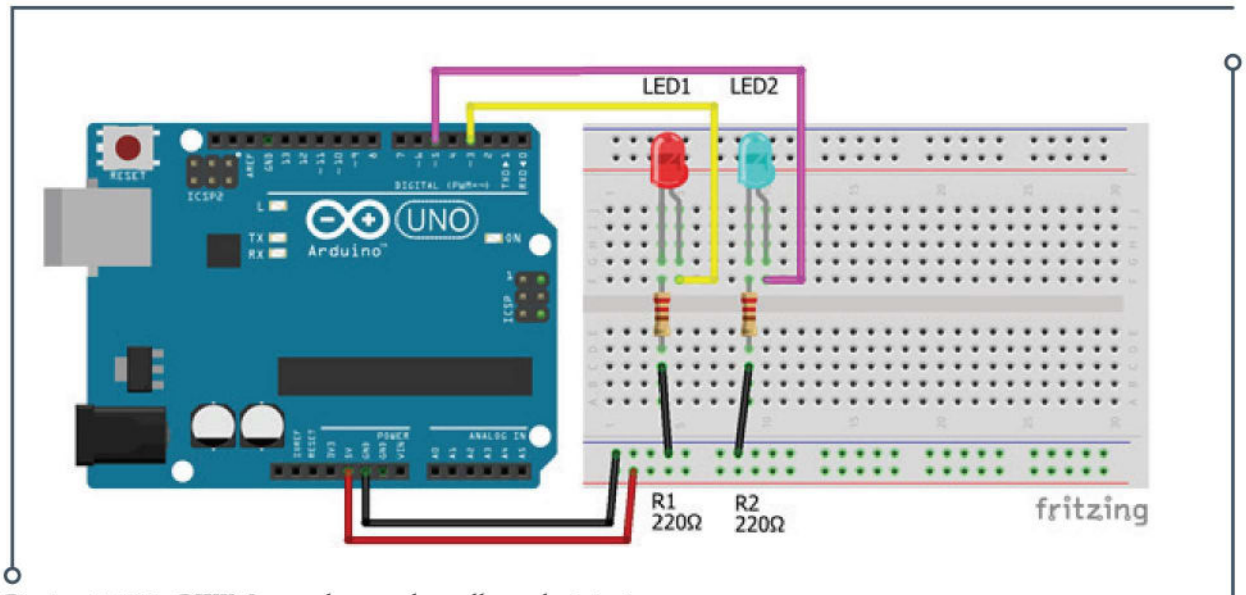
Arduino'nun 12 numaralı dijital pinlerine bağlı bir LED'in 1 sn aralıklarla yanıp sönmesini sağlayacak bir uygulama hazırlayınız. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 220 ohm (Ω) direnç kullanılmıştır.



Resim 7.120: LED yakma uygulaması breadboard çizimi

7.14.2. PWM LED Uygulaması

Arduino'nun 3 ve 5 numaralı pwm pinlerine bağlı 2 LED'in çalıştığı sürece ışık seviyesinin artarak yanması sağlayan bir uygulama hazırlayınız. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 2 adet 220 Ω direnç kullanılmıştır.

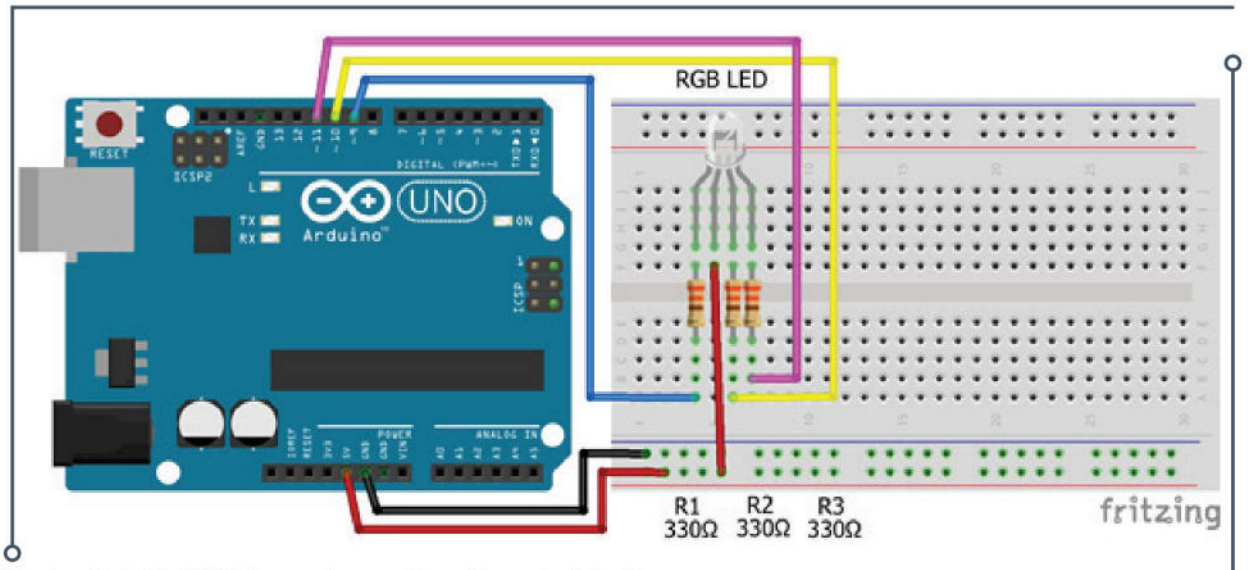


Resim 7.121: PWM uygulaması breadboard çizimi

7.14.3. RGB LED Uygulaması

RGB LED'in kırmızı Led pinini Arduino'nun 9, yeşil Led pininin 10 ve mavi Led pinini 11 numaralı pwm pinlerine bağlayarak RGB LED'in kırmızı, yeşil, mavi, sarı, camgöbeği ve beyaz renklerde veriler

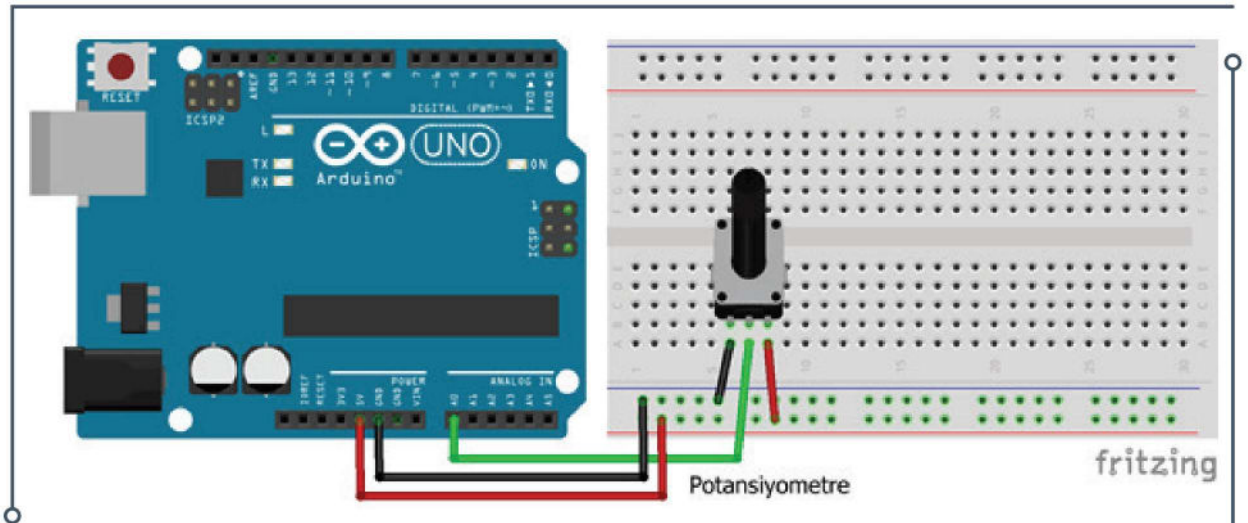
sırayla yanması sağlayacak bir uygulama hazırlayınız. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir RGB LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 3 adet 330 Ω direnç kullanılmıştır.



Resim 7.122: PWM uygulaması breadboard çizimi

7.14.4. Potansiyometre Uygulaması

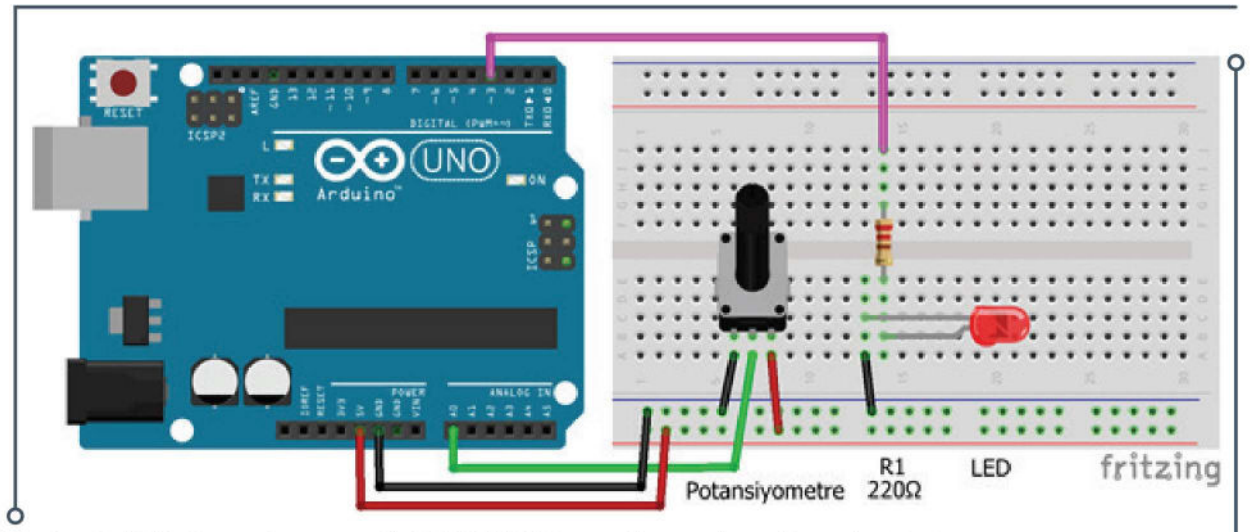
Arduino'nun A1 numaralı analog pinine bağlı bir potansiyometre kullanılarak değer okuma uygulaması hazırlayınız. Potansiyometrenin hareketi ile değişen direnç değerleri Arduino IDE seri port ekranında okunmalıdır. Potansiyometrenin Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. Örnek uygulamada 100 kiloohm (kΩ) potansiyometre kullanılmıştır.



Resim 7.123: Potansiyometre uygulaması breadboard çizimi

7.14.5. Potansiyometre ile PWM LED Uygulaması

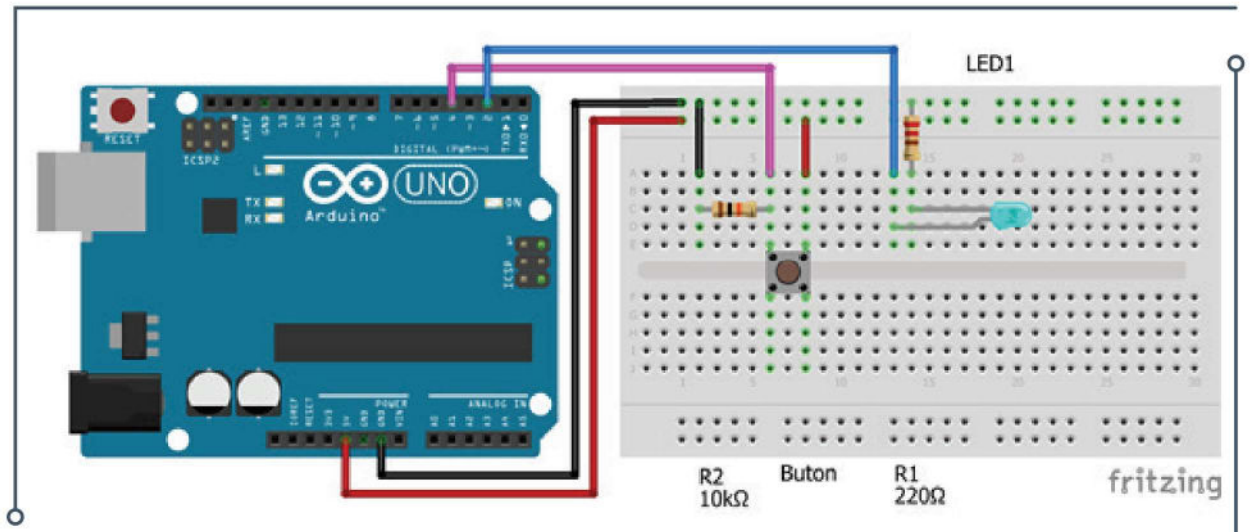
Arduino'nun A0 numaralı analog pinine bağlı bir potansiyometre kullanılarak, Arduino'nun 3 numaralı PWM pinine bağlı bir LED'in ışık seviyesinin kontrolülü sağlayacak bir uygulama hazırlayınız. Potansiyometrenin hareketi ile ışığın parlaklığı ayarlanabilmelidir. Potansiyometre ve LED'in Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 220 Ω direnç ve 100 kΩ potansiyometre kullanılmıştır.



Resim 7.124: Potansiyometre ile PWM LED uygulaması breadboard çizimi

7.14.6. Buton ile LED Yakma Uygulaması

Arduino'nun 4 numaralı dijital pinlerine bağlı bir buton kullanılarak 2 numaralı dijital pinine bağlı LED'in kontrol edilmesini sağlayacak bir uygulama hazırlayınız. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada LED için 220 Ω, buton için 10 kΩ direnç kullanılmıştır.

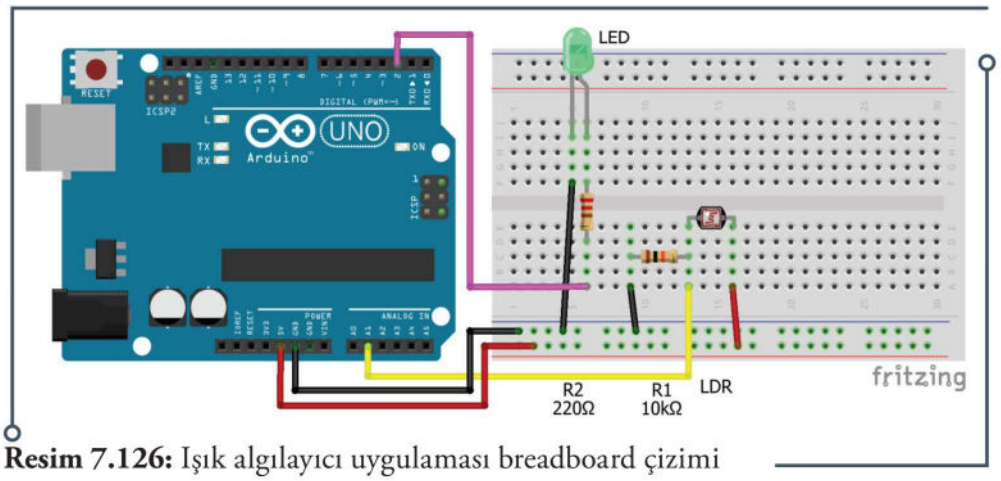


Resim 7.125: Buton uygulaması breadboard çizimi

7.14.7. Işık Algılayıcı Uygulaması

Sinyal çıkış pini Arduino'nun A1 numaralı analog pinine bağlı bir ışık algılayıcısı (LDR) kullanarak ışık algıladığı zaman, ışığın şiddetine göre 2 numaralı dijital pine bağlı LED'i sürekli olarak yakıp söndüren bir uygulama hazırlayınız. Işık şiddetine göre değişen değerler Arduino IDE seri port ekranında okunmalıdır.

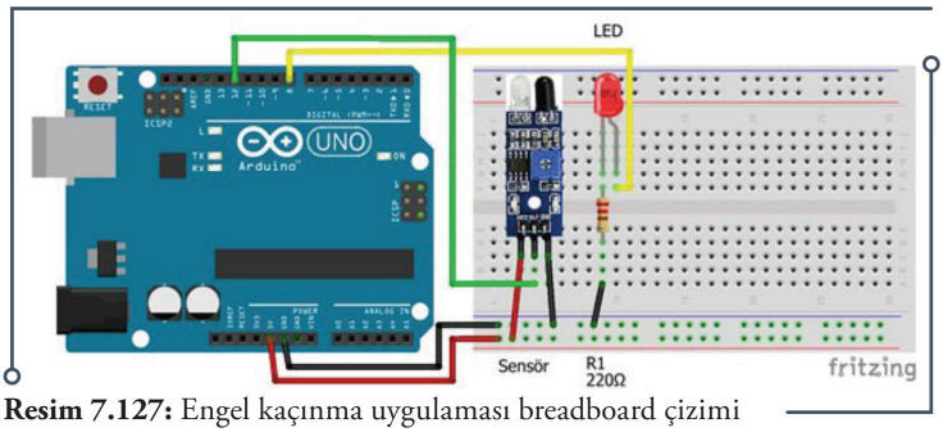
Aynı devre üzerinde yapacağınız ikinci uygulamada ise LED'in karanlıkta yanmasını aydınlıkta sönmesini sağlayınız. LDR algılayıcının algıladığı ışık şiddetini belirleyerek, okunan değer sizin belirleyeceğiniz değerden küçük olunca LED'in yanmasını değilse (büyükse) sönmesini sağlayınız. Işık algılayıcının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada LED için 220 Ω, LDR için 10 kΩ direnç kullanılmıştır.



Resim 7.126: Işık algılayıcı uygulaması breadboard çizimi

7.14.8. Engel Kaçınma Uygulaması

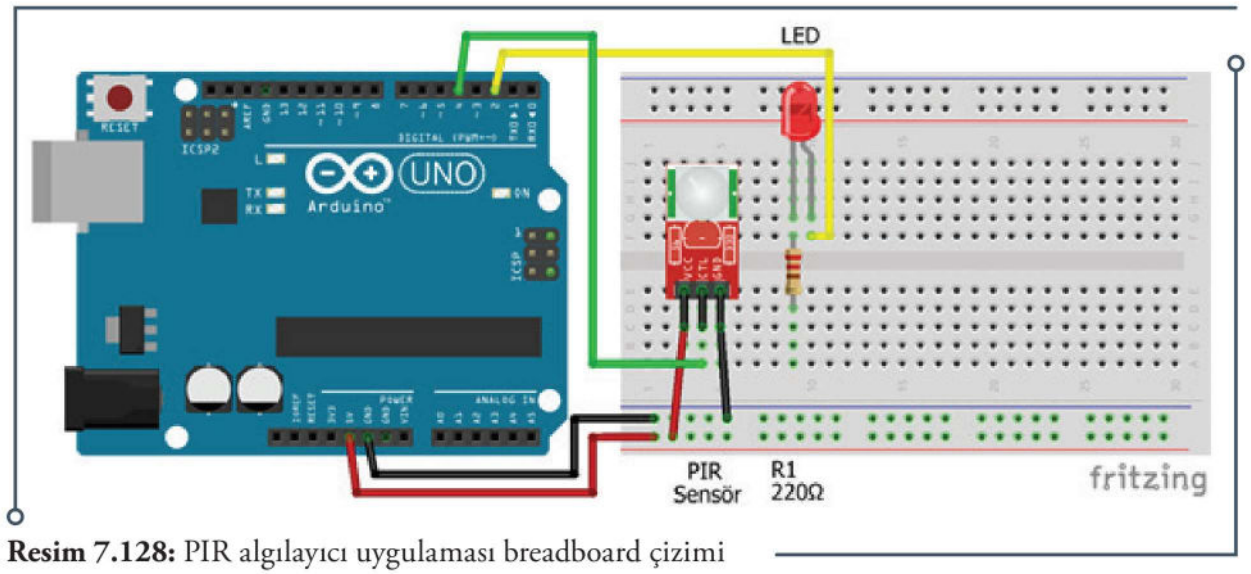
Sinyal çıkışı Arduino'nun 12 numaralı pinine bağlı bir engel kaçınma algılayıcısı kullanılarak engeli algıladığı zaman 4 numaralı dijital pine bağlı LED'i yakan bir uygulama hazırlayınız. Engel kaçınma algılayıcısının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada LED için 220 Ω direnç kullanılmıştır.



Resim 7.127: Engel kaçınma uygulaması breadboard çizimi

7.14.9. PIR Algılayıcı Uygulaması

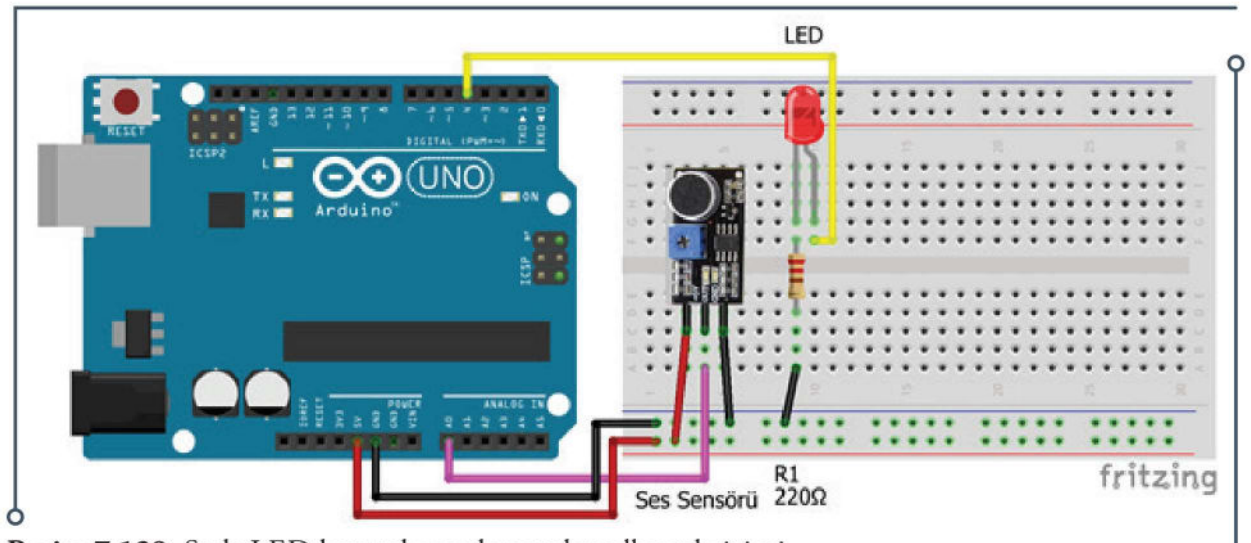
Sinyal çıkış pini Arduino'nun 2 numaralı dijital pinine bağlı bir PIR algılayıcısı kullanarak canlı algıladığı zaman 4 numaralı dijital pine bağlı LED'i yakan bir uygulama hazırlayınız. PIR algılayıcının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 220 Ω direnç kullanılmıştır.



Resim 7.128: PIR algılayıcı uygulaması breadboard çizimi

7.14.10. Sesle LED Kontrol Uygulaması

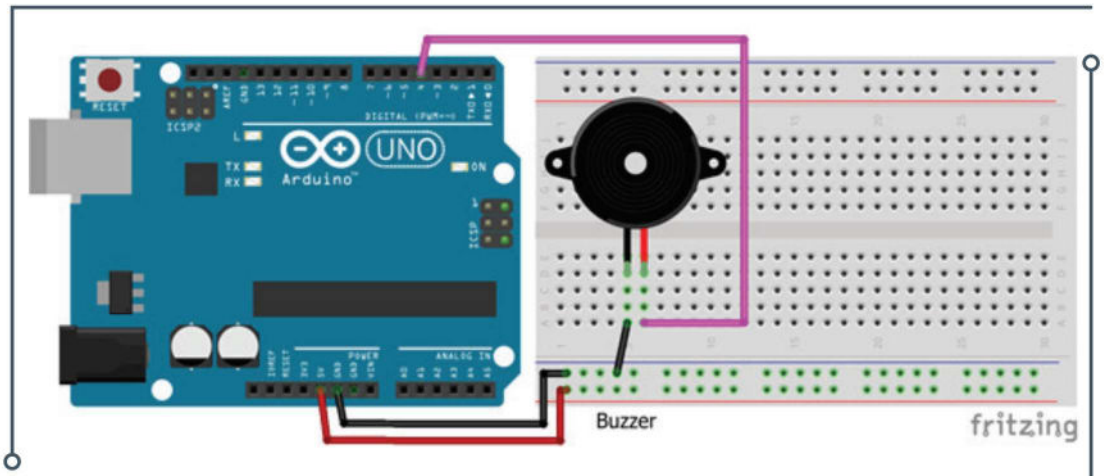
Sınyal çıkışı Arduino'nun A0 numaralı analog pinine bağlı bir ses algılayıcısı kullanarak ses seviyesine göre 4 numaralı dijital pine bağlı LED'i yakıp söndüren ses ile LED kontrol uygulaması hazırlayınız. Ses algılayıcısının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 220 Ω direnç kullanılmıştır.



Resim 7.129: Sesle LED kontrol uygulaması breadboard çizimi

7.14.11. Buzzer Uygulaması

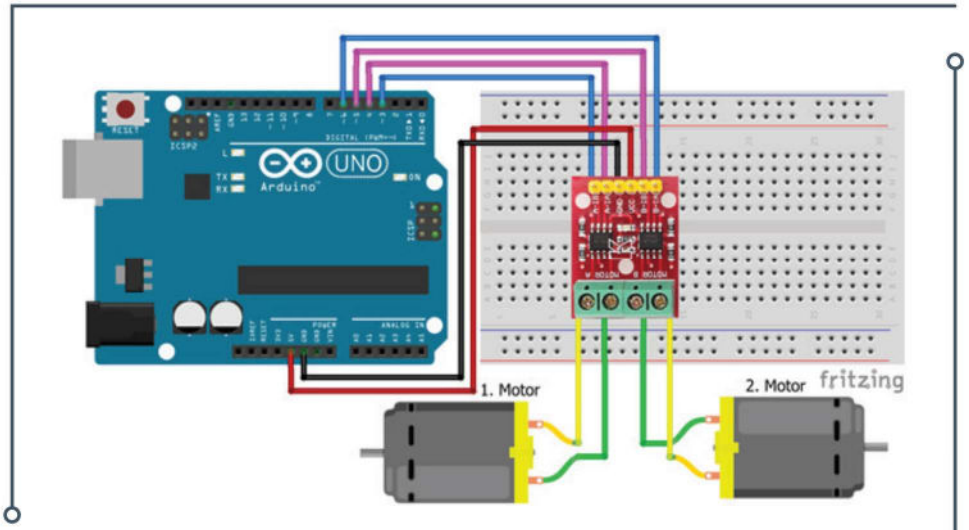
Arduino'nun 4 numaralı dijital pinine bağlı bir buzzer kullanılarak 2 farklı tonla alarm sesi üretmeyi sağlayacak bir uygulama hazırlayınız. Buzzer'in Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmiştir. Örnek uygulamada standart bir buzzer kullanılmıştır.



Resim 7.130: Buzzer uygulaması breadboard çizimi

7.14.12. DC Motor Kontrol Uygulaması

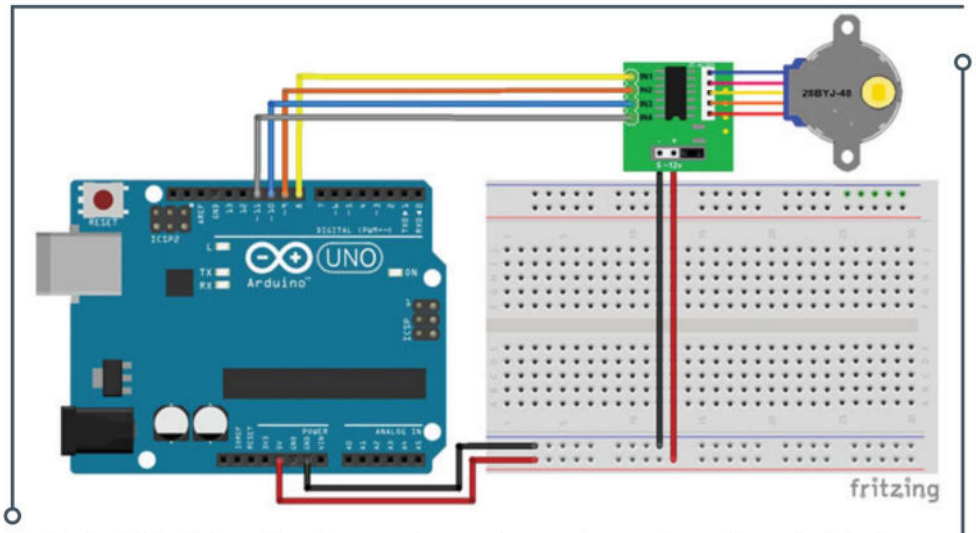
Motor sürücüye bağlı bir çift motoru ileri, geri, sağa ve sola hareket ettirip durduran bir motor kontrol uygulaması hazırlayınız. Bu amaçla farklı motor sürücü kartları kullanılabilir. Uygun voltajda (en fazla 6 volt) herhangi bir dc motor kullanılabilir. Daha güçlü motor kullanılacaksa Arduino UNO'nun zarar görmemesi için harici bir güç kaynağı ile motorların beslenmesini sağlayınız. Bu amaç için hazırlanan örnekte L9110 motor sürücü kartı kullanılmıştır. Sürücünün A-IA girişi Arduino'nun 3 numaralı dijital pinine, sürücünün A-IB girişi Arduino'nun 4 numaralı dijital pinine, sürücünün B-IA girişi Arduino'nun 5 numaralı dijital pinine, sürücünün B-IB girişi Arduino'nun 6 numaralı dijital pinine bağlanmalıdır. Motor sürücünün Arduino ve motorlarla bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir.



Resim 7.131: DC motor kontrol uygulaması breadboard çizimi

7.14.13. Adım (Step) Motor Kontrol Uygulaması

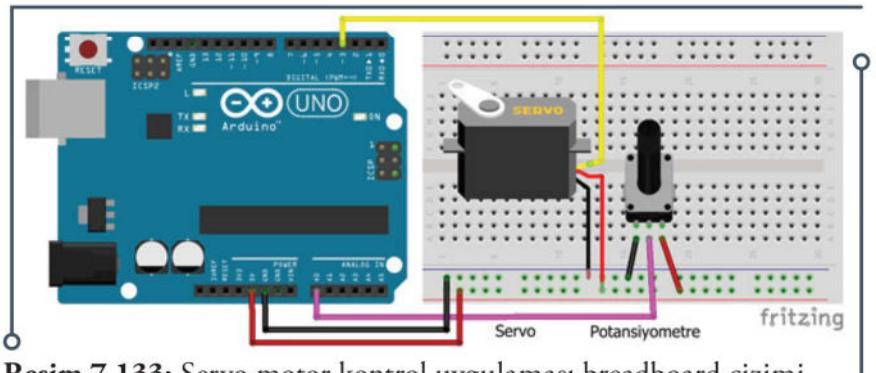
Adım motor sürücüye bağlı bir adım motorun; saat yönünde ve saat yönünün tersinde, verilecek olan adım sayısı kadar adım atmasını sağlayacak adım motor kontrol uygulaması hazırlayınız. Bu amaç için hazırlanan örnekte 28 BYJ-48 redüktörlü adım motor ve ULN2003A adım motor sürücü kartı kullanılmıştır. Daha güçlü adım motor kullanılacaksa Arduino UNO'nun zarar görmemesi için harici bir güç kaynağı ile adım motorun beslenmesini sağlayınız. Motor sürücünün 1N1 pini Arduino'nun 8 numaralı dijital pinine, 1N2 pini Arduino'nun 9 numaralı dijital pinine, 1N3 pini Arduino'nun 10 numaralı dijital pinine ve 1N4 pini Arduino'nun 11 numaralı dijital pinine bağlanmıştır. Adım motor bağlantısı için kart üzerinde konnektör bulunmaktadır. Adım motor sürücünün motor ve Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir.



Resim 7.132: Adım (Step) motor kontrol uygulaması breadboard çizimi

7.14.14. Servo Motor Kontrol Uygulaması

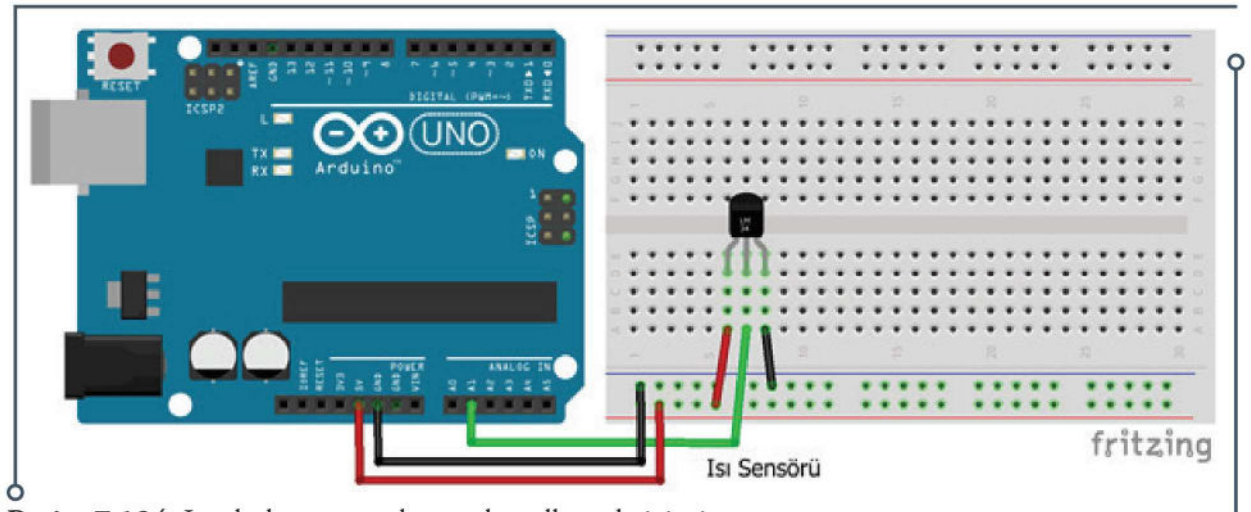
Arduino'nun A0 numaralı analog pinine bağlı bir potansiyometre kullanılarak 0 ile 180 derece arasında hareket eden bir servo motor kontrol uygulaması hazırlayınız. Potansiyometrenin değerini değiştirerek servo motorun 0 ile 180 derece arasında pozisyon alması gerekmektedir. Potansiyometreden okunan analog değerin 0 ile 1023 arasında olduğunu ve map yöntemiyle 0 ile 180 derece arasında sınırlandırılması gerektiğini dikkate alınız. Servo motorun ve potansiyometrenin Arduino ile bağlantısı yukarıdaki breadboard çizimi üzerinde gösterilmektedir. Örnek uygulamada 9 gramlık standart bir servo motor ile 100 kΩ potansiyometre kullanılmıştır.



Resim 7.133: Servo motor kontrol uygulaması breadboard çizimi

7.14.15. Isı Algılayıcısı Uygulaması

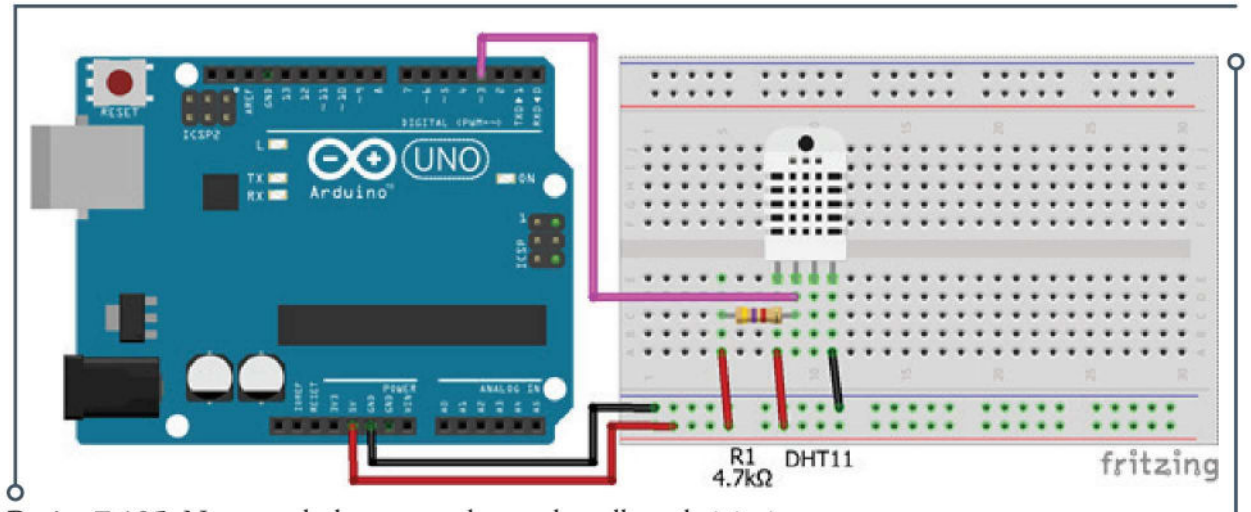
Sensör ucu Arduino'nun A1 numaralı analog pinine bağlı bir ısı algılayıcısı kullanılarak ortam ısısını ölçmeyi sağlayacak bir uygulama hazırlayınız. Ortam ısısı bilgisi Arduino IDE seri port ekranından okunmalıdır. Bu amaç için hazırlanan örnekte LM34 ısı algılayıcısı kullanılmıştır. Isı algılayıcısının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmiştir.



Resim 7.134: Isı algılayıcı uygulaması breadboard çizimi

7.14.16. Nem-Isı Algılayıcısı Uygulaması

Sensör ucu Arduino'nun 3 numaralı dijital pinine bağlı bir nem-ısı algılayıcısı kullanılarak içinde bulunulan ortamın nem ve ısısını ölçmeyi sağlayacak bir uygulama hazırlayınız. Uygulamada algılayıcının doğru şekilde çalışması için 4.7k'lık bir direncin çizimde gösterildiği gibi sensör ucu ile besleme voltajı (5 Volt) arasına bağlanması gerekmektedir. Ortamın nem ve ısı bilgisi Arduino IDE seri port ekranından okunmalıdır. Bu amaç için hazırlanan örnek uygulamada 4.7 kΩ direç ve DHT11 nem-ısı algılayıcısı ile buna ait https://github.com/RobTillaart/Arduino/tree/master/libraries/DHTlib adresinde bulunan kütüphane kullanılmıştır. Nem-ısı algılayıcısının Arduino ile bağlantısı yukarıdaki breadboard çizimi üzerinde gösterilmiştir.

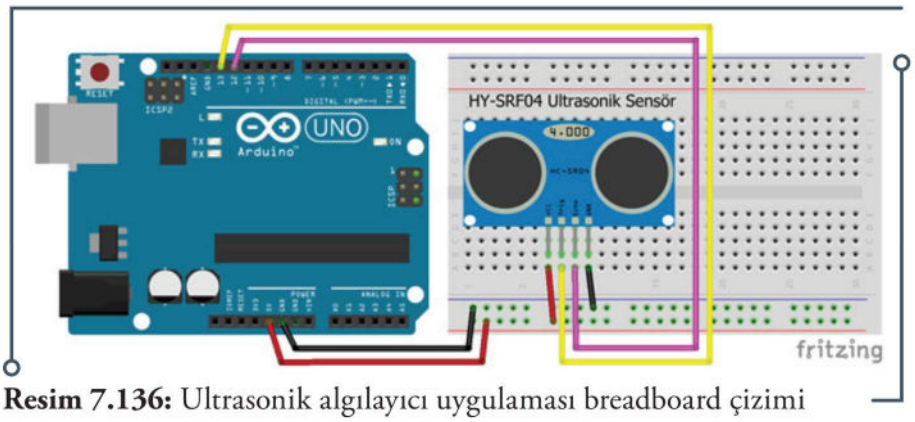


Resim 7.135: Nem-ısı algılayıcı uygulaması breadboard çizimi

7.14.17. Ultrasonik Algılayıcı Uygulaması

HY-SRF04 veya HY-SRF05 ultrasonik algılayıcı kullanılarak bir mesafe ölçme uygulaması hazırlayınız. Bu amaç için hazırlanan örnek uygulamada HY-SRF04 model ultrasonik algılayıcı kullanılmıştır. Ultrasonik algılayıcının tetik (trig) pini Arduino'nun 13 numaralı, okuma (echo) pini 12 numaralı dijital pinlerine bağlanmıştır. Ölçme 2 ile 200 cm aralığı için sınırlandırılmalı ve mesafe bilgisi Arduino IDE seri port ekranından okunmalıdır. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir.

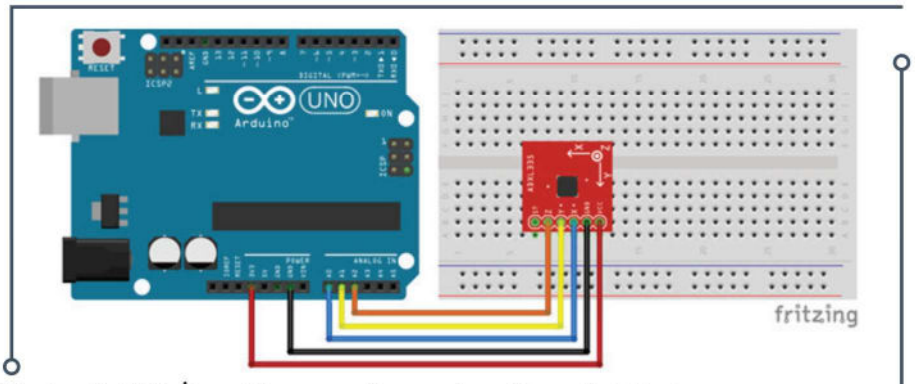
Aynı devre üzerinde yapacağınız ikinci mesafe ölçme uygulamasında ise aynı ultrasonik algılayıcının tetik (trig) ve okuma (echo) pini birleştirilerek, Arduino'nun 12 numaralı dijital pinine bağlanmalıdır. Bunun dışındaki bağlantılar aynı şekilde kalmalıdır. Uygulamada mesafe bilgisi Arduino IDE seri port ekranından okunmalı ve http://playground.arduino.cc/Code/NewPing adresinde bulunan "NewPing" kütüphanesi kullanılmalıdır.



Resim 7.136: Ultrasonik algılayıcı uygulaması breadboard çizimi

7.14.18. İvmeölçer Uygulaması

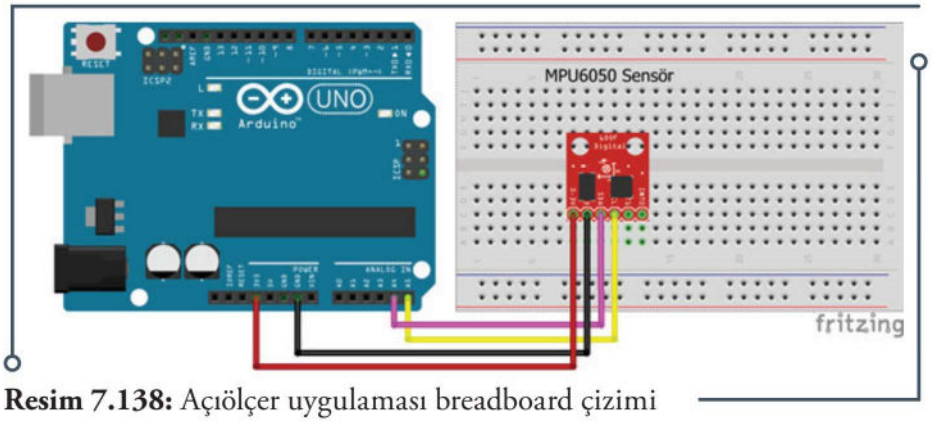
İvme ölçerin X ekseni sınyal çıkışı Arduino'nun A0, Y ekseni sınyal çıkışı Arduino'nun A1 ve Z ekseni sınyal çıkışı Arduino'nun A2 numaralı analog pinlerine bağlı ivmeölçer algılayıcısı kullanarak 3 eksenli ivme ölçme uygulaması hazırlayınız. Bu amaçla farklı ivmeölçer agılayıcıları kullanılabilir. Bu amaç için hazırlanan örnek uygulamada ADXL335 ivmeölçer algılayıcısı kullanılmıştır. Okunan 3 eksen ivme bilgisi Arduino IDE seri port ekranından okunmalıdır. İvmeölçer algılayıcısının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. Kullanılan algılayıcı 1.8 ile 3.6 Volt arasındaki doğru akım ile çalıştığı için algılayıcının Vcc girişi Arduino'nun 3.3 Volt pinine bağlanmalıdır. 5 Volt ile çalışan bir algılayıcı kullanıyorsanız Arduino'nun 5 Volt pinine bağlayınız.



Resim 7.137: İvmeölçer uygulaması breadboard çizimi

7.14.19. Açıölçer Uygulaması

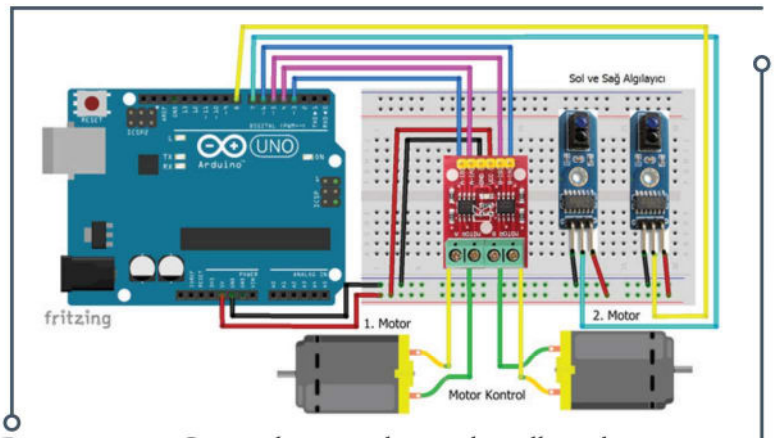
Sinyal çıkışları Arduino'nun A4 (SDA) ve A5 (SCL) numaralı analog pinlerine bağlı gyro ve ivmeölçer algılayıcısı kullanarak bir açıölçer uygulaması hazırlayınız. Bu amaçla farklı gyro algılayıcılar kullanılabilir. Bu amaç için hazırlanan örnekte MPU-6050 algılayıcı kartı kullanılmıştır. Bu üzerinde 3 eksenli bir gyro ve 3 eksenli bir açısal ivmeölçer bulunan 6 eksenli bir IMU algılayıcı kartıdır. Okunan açı bilgisi Arduino IDE seri port ekranında gösterilmelidir. Algılayıcının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. Bu kartın bazı çeşitleri 1.8 ile 3.6 Volt arasındaki doğru akım ile çalışmaktadır. Bu özellikteki kartların güç girişi Arduino'nun 3.3 Volt pinine bağlanmalıdır.



Resim 7.138: Açıölçer uygulaması breadboard çizimi

7.14.20. Çizgi İzleyen Robot Uygulaması

DC Motor kontrol uygulamasında kullandığınız motor ve motor sürücüsüne çizgi takip algılayıcılarını da ekleyerek bir çizgi takip uygulaması hazırlayınız. Bu amaçla farklı motor, motor sürücü kartı ve çizgi takip algılayıcıları kullanılabilir. Uygulama için motorlara tekerlek takarak ve bütün sistemi bir şase üzerine yerleştirerek hareketli bir robot haline dönüştürmeniz gereklidir. Bu uygulamayı gurup halinde hazırlamanız daha uygun olacaktır. Hazırlanan örnekte 6 voltla çalışan redüktörlü iki adet dc motor, L9110 motor sürücü kartı ve iki adet TCRT5000 çizgi takip algılayıcısı kullanılmıştır. Motor sürücünün A-IA girişi Arduino'nun 3 numaralı dijital pinine, sürücünün A-IB girişi Arduino'nun 4 numaralı dijital pinine, sürücünün B-IA girişi Arduino'nun 5 numaralı dijital pinine, sürücünün B-IB girişi Arduino'nun 6 numaralı dijital pinine bağlanmıştır. Sol çizgi algılayıcının sinyal pini 7, sağ çizgi algılayıcının sinyal pini Arduinonun 8 numaralı dijital pinine bağlanmıştır. Algılayıcılar ile motor sürücünün Arduino ve motorlarla bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir.



Resim 7.139: Çizgi takip uygulaması breadboard çizimi

7.15. DEĞERLENDİRME SORULARI

1. **Arduino IDE ortamında hangi programlama dili kullanılır?**
   a) C
   b) C++
   c) C #
   d) Pascal
   e) Python

2. **Arduino IDE yapısı ve temel özellikleri açısından aşağıda verilen ifadelerden hangisi doğrudur?**
   a) Arduino'ya yüklenen programlar elektriği kapatılana kadar Arduino içinde kalır. Yüklemeden sonra bağımsız olarak çalıştırılabilir
   b) Tüm komutlar virgül (,) ile biter. Fakat blok başlatan ifadelerden sonra virgül kullanılmaz
   c) Programda kullanılan bazı değişkenler ve bilgi tipleri bildirilmelidir
   d) Programın başında kullanılacak kütüphaneler aktifleştirilmeli /çağrılmalıdır
   e) Açıklamalar "\\" ve "* *\ " (birden fazla satır için) ile yazılabilir

3. **Arduino uygulamalarında kullanılacak elektronik bileşenler için aşağıdakilerden hangisinin kullanılması uygulamaların hızlı, kolay ve en önemlisi lehim yapmadan yapılmasına olanak tanıyacaktır?**
   a) Breadboard (Devre Tahtası)
   b) Delikli pertinaks
   c) Baskılı devre
   d) Konnektör
   e) Jumper

4. **Arduino IDE için aşağıda verilen ifadelerden hangisi tam olarak doğru değildir?**
   a) Kod yazım editörüdür
   b) Tümleşik bir derleyicidir
   c) Programlama dilidir
   d) Yorumlayıcı ve hata ayıklayıcıdır
   e) Tümleşik geliştirme ortamıdır

5. **Arduino IDE ortamında mikrodenetleyici ya da Arduino'nun beslemesi devam ettiği sürece tekrarlanan komutlar hangi fonksiyon içinde yer alır?**
   a) void setup()
   b) digitalWrite()
   c) serial.begin()
   d) digitalRead()
   e) void loop()

6. **Arduino IDE ortamında program yüklenilip enerji verildikten veya tekrar başlatıldıktan sonra 1 defa çalışan fonksiyon aşağıdakilerden hangisidir?**
   a) digitalRead()
   b) digitalWrite()
   c) serial.begin()
   d) void setup()
   e) void loop()

7. **Arduino UNO, USB ile bilgisayara bağlı olduğu sürece giriş veya çıkış işlemleri için hangi numaralı pinler kullanılamazlar?**
   a) A0 ve A1
   b) 0 ve 1
   c) 3 ve 5
   d) A2 ve A3
   e) Güç pinleri

8. **Arduino UNO'nun belli bir pinine gelen sinyalle belli bir fonksiyonun ya da önceden belirlenmiş bir alt programın çalıştırılmasını sağlamak için hangi numaralı pinler kullanılır?**
   a) 2 ve 3
   b) 4 ve 5
   c) 6 ve 7
   d) 8 ve 9
   e) 10 ve 11

9. **I^2C veri yolu kullanan bir algılayıcı Arduino UNO'nun hangi numaralı SDA ve SCL pinlerine bağlanmalıdır?**
   a) A0 ve A1
   b) A2 ve A3
   c) A4 ve A5
   d) 0 ve 1
   e) 2 ve 3

10. **SPI veri yolu kullanan bir algılayıcı Arduino UNO'nun hangi numaralı MOSI ve MISO pinlerine bağlanmalıdır?**
   a) 3 ve 4
   b) 5 ve 6
   c) 7 ve 8
   d) 9 ve 10
   e) 11 ve 12

KAYNAKÇA

Adafruit (2017). Sensors. [Çevrim-içi: https://www.adafruit.com/category/35, Erişim tarihi: 03.03.2017].

Adafruit (2017). Sensors. [Çevrim-içi: https://learn.adafruit.com/category/sensors, Erişim tarihi: 03.03.2017].

Akademikport (2017). AkademikPort Arduino Başlangıç Projeleri. [Çevrim-içi: http://kitap.akademikport.com/AkademikPort-Arduino-Baslangic-Projeleri.pdf?, Erişim tarihi: 04.01.2017].

Arduino (2017). What is Arduino? [Çevrim-içi: https://www.arduino.cc/en/Guide/Introduction, Erişim tarihi: 03.01.2017].

Arduino (2017). Arduino IDE. [Çevrim-içi: https://www.arduino.cc/en/main/software, Erişim tarihi: 03.01.2017].

Arduino (2017). Language Reference. [Çevrim-içi: https://www.arduino.cc/en/Reference/HomePage, Erişim tarihi: 04.01.2017].

Arduino (2017). Arduino Libraries. [Çevrim-içi: https://github.com/arduino-libraries, Erişim tarihi: 04.01.2017].

Banzi, M., & Shiloh, M. (2014). Getting started with Arduino: the open source electronics prototyping platform. Maker Media, Inc.

Baykara, M. (2017). Mikroişlemciler ve Programlama Dersi- ARDUINO. [Çevrim-içi: http://muhammetbaykara.com/wp-content/uploads/2017/04/arduino-uygulamalar%C4%B1.pdf, Erişim tarihi: 10.01.2017].

Benson, C. (2012). Basics: What Types of Mobile Robots are There? [Çevrim-içi: http://www.robotshop.com/blog/en/what-types-of-mobile-robots-are-there-3652, Erişim tarihi: 05.01.2017].

Boxall, J. (2013). Arduino Workshop: A Hands-On introduction with 65 projects. No Starch Press.

Çobanoğlu, B. (2016). Arduino Programlama. [Çevrim-içi: http://cobanoglu.wikispaces.com/file/view/Arduino_Cobanoglu.pdf/495731668/Arduino_Cobanoglu.pdf, Erişim tarihi: 12.01.2017].

Demir, U. (2015). Arduino Programlama Kitabı. [Çevrim-içi: https://ugrdmr.wordpress.com/2016/02/01/arduino-programlama-kitabi-cikti-2/, Erişim tarihi: 12.01.2017].

EBA (2017). Arduino Temel Atölye Uygulamaları. [Çevrim-içi: http://img.eba.gov.tr/659/8f6/e5b/21b/932/024/027/881/4f3/997/77f/4e5/eac/ff2/001/6598f6e5b21b9320240278814f399777f4e5eacff2001.pdf, Erişim tarihi: 06.02.2017].

Electronicsteacher (2017). Robotics - Types of Robots. [Çevrim-içi: http://www.electronicsteacher.com/robotics/type-of-robots.php, Erişim tarihi: 09.01.2017].

Evans, B. (2011). Beginning Arduino Programming. Apress.

Firmata (2017). Firmata. [Çevrim-içi: https://github.com/firmata/arduino, Erişim tarihi: 16.01.2017].

Font, D. B., García, C. S., & de Mántaras, R. L. (2003). A Multiagent Approach to Qualitative Navigation in Robotics. Universitat Politècnica de Catalunya. [Çevrim-içi: http://eia.udg.es/~busquets/thesis/thesis_html/node34.html, Erişim tarihi: 06.01.2017].

Fritzing (2017). Fritzing. [Çevrim-içi: http://fritzing.org/home/, Erişim tarihi: 05.04.2017].

Geleceğiyazanlar (2017). Arduino Dünyası. [Çevrim-içi: https://gelecegiyazanlar.turkcell.com.tr/konu/arduino, Erişim tarihi: 18.01.2017].

Harold, T. (2011). Practical Arduino Engineering.Technology In Actıon.

Herrmann, M. (2015). IVR: A Brief Outlook to Robot Architectures. [Çevrim-içi: http://homepages.inf.ed.ac.uk/mherrman/IVRINVERTED/pdfs/architectures.pdf, Erişim tarihi: 11.01.2017].

İskefiyeli, M. (2017). Sakarya Üniversitesi Bilgisayar Mühendisliği Gömülü Sistemler Deney Föyü. [Çevrim-içi: http://content.lms.sabis.sakarya.edu.tr/Uploads/50142/35704/gomsisdeneyler01.pdf,

http://content.lms.sabis.sakarya.edu.tr/Uploads/50142/35706/gomsisdeneyler02.pdf, Erişim tarihi: 14.04.2017].

Karvinen, T., & Karvinen, K. (2011). Make: Arduino Bots and Gadgets Six Embedded Projects with Open Source Hardware and Software (Learning by Discovery). Make Books-Imprint of: O'Reilly Media.

Kelly, J. F., & Timmis, H. (2012). Arduino Adventures: Escape from Gemini Station. Apress.

Makeblock (2017). Robot Kits. [Çevrim-içi: http://www.makeblock.com/robot-kit-series-STEM, Erişim tarihi: 08.03.2017].

Makeblock (2017).mBlock. [Çevrim-içi: http://learn.makeblock.com/en/software/, Erişim tarihi: 08.03.2017].

mBlock (2017). Program Robots / Arduino. [Çevrim-içi: http://www.mblock.cc/, Erişim tarihi: 08.03.2017].

McRoberts, M. (2013). Beginning Arduino. Apress.

Monk, S. (2013). 30 Arduino projects for the evil genius. McGraw-Hill Professional.

Odendahl, R. (2015). Robotics Notes – Paradigms. [Çevrim-içi: http://www.cs.oswego.edu/~odendahl/coursework/csc338/notes/paradigms/overview.html, Erişim tarihi: 05.01.2017].

Olsson, T. (2012). Arduino wearables. Apress.

Oxer, J., & Blemings, H. (2011). Practical Arduino: cool projects for open source hardware. Apress.

Pololu (2017). Elektronics. [Çevrim-içi: https://www.pololu.com/category/6/electronics, Erişim tarihi: 13.02.2017].

Premeaux, E., & Evans, B. (2012). Arduino Projects to Save the World. Apress.

Qureshi, F., Terzopoulos, D., & Gillett, R. (2004). The cognitive controller: a hybrid, deliberative/reactive control architecture for autonomous robots. Innovations in Applied Artificial Intelligence, 1102-1111. [Çevrim-içi: https://www.researchgate.net/profile/Ross_Gillett/publication/221048199_The_Cognitive_Controller_A_Hybrid_DeliberativeReactive_Control_Architecture_for_Autonomous_Robots/links/00b7d5368dd3057ca6000000.pdf, Erişim tarihi: 05.01.2017].

Reis, L., P. (2007). Robot Software Architectures. [Çevrim-içi: http://paginas.fe.up.pt/~lpreis/ir2007_08/documents/IR0708-3-AgentArchitectures.pdf, Erişim tarihi: 05.01.2017].

Robots and Androids (2017). Robots and Androids. [Çevrim-içi: http://www.robots-and-androids.com/, Erişim tarihi: 15.02.2017].

Robotpark (2017). All Types of Robots - By Locomotion. [Çevrim-içi: http://www.robotpark.com/All-Types-Of-Robots, Erişim tarihi: 13.01.2017].

Robot Wiki (2017). Robot Types. [Çevrim-içi: http://robotics.wikia.com/wiki/Robot_types, Erişim tarihi: 12.01.2017].

Sevinç, H. (2017). Temel Elektronik Arduino Eğitimi. [Çevrim-içi: http://www.ardimed.com/upload/post/datasheet/datasheet_2_564773fb1dcd2.pdf, Erişim tarihi: 14.03.2017].

Smith, A. G. (2011). Introduction to Arduino. Reference book, September, 30, 1-172.

SparkFun (2017). Sensors. [Çevrim-içi: https://www.sparkfun.com/categories/23, Erişim tarihi: 14.03.2017].

Thomas, D. (2002). Robot Architectures. [Çevrim-içi: http://cs.brown.edu/~tld/courses/cs148/02/architectures.html, Erişim tarihi: 09.01.2017].

Thrun, S. (2001). Is Robotics Going Statistics? The Field of Probabilistic Robotics. Communications of the ACM, 45 (3), 1-8. [Çevrim-içi: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.79.8332&rep=rep1&type=pdf, Erişim tarihi: 03.01.2017].

TTKB (2016). Ortaöğretim Bilgisayar Bilimi Dersi (Kur 1, Kur 2) Öğretim Programı. [Çevrim-içi: http://ttkb.meb.gov.tr/www/ogretim-programlari/icerik/72#, Erişim tarihi: 10.10.2016].

Warren, J. D., Adams, J., & Molle, H. (2011). Arduino Robotics. Collection: Technology in Action.

Zöhra, B. (2015). Arduino ile Sensor Uygulamaları. [Çevrim-içi: https://www.academia.edu/16909759/ARDUINO_%C4%B0LE_SENSOR_UYGULAMALARI?auto=download, Erişim tarihi: 05.04.2017].