Robot ve Robot Mimarisi

Robotlar, kendi kendine (otonom) veya önceden programlanmış görevleri yerine getirebilen elektromekanik araçlardır.

Robot Mimarisinde İlkeler

Robot mimarisinde uzun bir süre boyunca yaygın olarak kabul edilen ilkeller sense, plan ve act arasındaki ilişkilere dayalı olarak açıklanmıştır. Sense, algılayıcılardan bilgi almayı ve diğer bileşenler için "çıktı" üretmeyi sağlamaktadır. Plan, algılayıcılardan veya diğer işlevsel bileşenlerden alınan tüm bilgileri kullanarak, gerçekleştirecek görevler üretmeyi, hareket planı yapmayı sağlar. Act, görevleri yerine getiren işlevsel bileşenlerin, hareket biçimini sağlar. Bu ilkelere dayalı olarak geliştirilen Hiyerarşik (Deliberative Kontrol) Mimari, Tepkisel (Reactive Kontrol) Mimari ve Karma (Hibrit Kontrol) Mimari yaygın olarak robot tasarımlarında kullanılır.

Robot Kontrol Yöntemleri

Robotun hangi durumda ne yapacağına, ne tepki göstereceğine karar verme işlemine robot kontrolü adı verilmektedir. Robot kontrol sistemleri farklı araç ve programlardan oluşmaktadır.

1. Tepkisel (Reactive) Kontrol:
2. Bilinçli (Deliberative) Kontrol:
3. Karma (Hibrit) Kontrol:
4. Davranışsal (Behavioral) Kontrol:

Kullanılan Uygulama Alanlarına Göre Robotlar

- Endüstriyel Robotlar:
- Ev Robotları:
- Tıbbi Robotlar
- Servis Robotları
- Askerî Robotlar
- Eğlence Robotları
- Uzay Robotları
- Hobi ve Yarışma Robotları
- Sanal Robotlar

Hareket Mekaniğine Göre Robotlar

- Sabit Robotlar
- Tekerlekli Robotlar
- Paletli Robotlar
- Ayaklı Robotlar
- Yüzen Robotlar
- Uçan Robotlar
- Yılan Robotlar
- Yumuşak Elastik Robotlar
- Mobil Küresel Robotlar (Robotik Toplar)
- Hibrit Robotlar
- Sürü Robotları
- Modüler Robotlar
- Mikro Robotlar
- Nano Robotlar
- Beam Robotlar

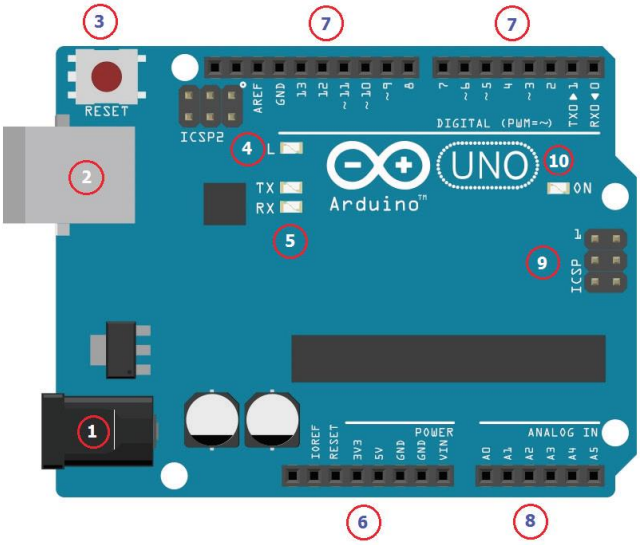
Eğitsel Amaçlı Robotlar

- Blok (LEGO Benzeri) Tabanlı Robot Montaj Setleri
- Düşük Maliyetli Programlanabilir Robotik Kol Setleri
- Düşük Maliyetli Minimum Özelliklerde Mobil Robot Tasarım Kitleri
- Açık Kaynaklı Düşük Maliyetli Mobil Robot Platformları
- Düşük Maliyetli, Tam Monte Edilmiş Mobil Robotlar
- Modüler Eğitsel Robot Kitleri
- Açık Kaynaklı Minyatür Sürü Robotlar

Arduino Nedir?

Arduino, ileri derecede elektronik ve mikro denetleyici bilgisi gerektirmeden çok çeşitli projelerin uygulanabileceği acık kaynaklı, donanımında Atmel firması tarafından üretilen AVR mikro denetleyici içeren bir elektronik geliştirme platformudur.

Arduino UNO Geliştirme Kartı



1. Güç Girişi: 7-12 Volt DC adaptor girişi.

2. USB Bağlantı Konnektörü (USB Port): UNO'ya program yuklemek ve bilgisayar ile haberleşmek icin 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 icin kullanılabilir.

5. RX (Receiving) ve TX (Transmitting) LED'leri: Seri haberleşme icin kullanılan RX ve TX pininin durumunu gosteren 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 icin de kullanılabilir. Ayrıca analog-dijital

çeviricinin referans giriş pini ve seri iletişim pinleri de (RX ve TX) buradadır.

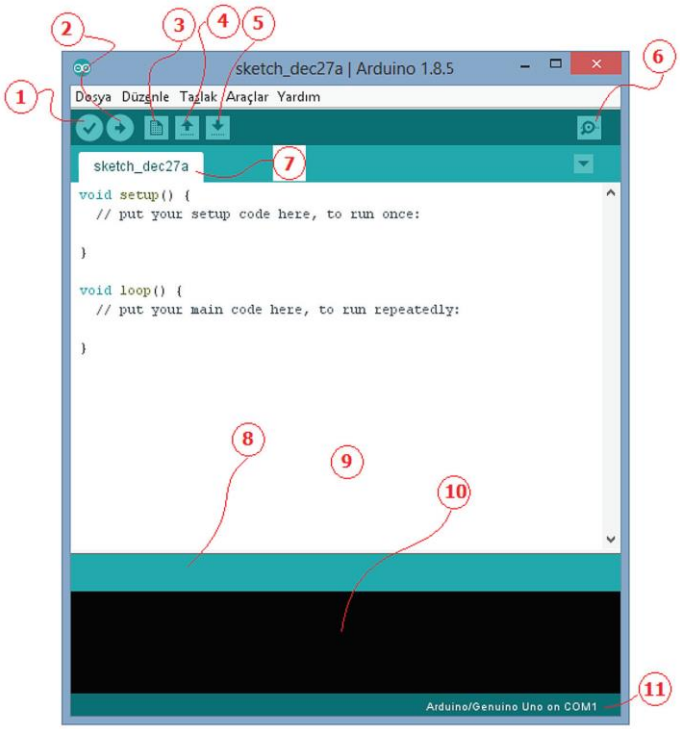
8. Anolog Giriş Pinleri: 6 adet analog giriş pini bu bölümde bulunmaktadır.

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 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.



1. Kontrol Et: Yazılan kodları derler ve hataları bulur.

2. Yükle: Yazılan programı Arduino kartına yükler.

3. Yeni: Yeni çalışma sayfası acar.

4. Aç: Kayıtlı bir programı acar.

5. Kaydet: Yazılan programı kaydeder.

6. Seri Port Ekranı: Arduino ile seri iletişim yaparak ekran acar.

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 mikro denetleyicide kapladığı alanı gösterir.

11. Gösterge: Bilgisayara usb ile bağlanan Arduino'nun bağlandığı portu ve hangi Arduino modeli ile çalışılıyorsa onu gösterir.

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 bolumden 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 icin kullanılır.
- Arduino'ya yüklenen programlar kaldırılana kadar Arduino içinde kalır. Yüklemeden sonra bağımsız
- olarak çalıştırılabilir.
- Bloklar, { } parantezleri (süslü parantez) ile oluşturulur.
- Komutlar aynı veya alt alta satırlara yazılabilir. Fakat programın anlaşıla bilirliği acısından 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 dahil edilmediğinden) kullanılabilir.
- Eşdeğer ifadeler #define ile atanır.
- Kütüphaneler #include ile çağrılır.
-

```
/* HIGH | LOW  */

int led= 10;
digitalWrite(Led, HIGH); // Led!i yak
digitalWrite(Led, LOW);  // Led'i söndür
```

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.

```
/* INPUT | OUTPUT  */

int Led = 10;
int buton = 4

void setup ()
  pinMode(Led, OUTPUT); // Led çıkış olarak tanımlandı
  pinMode(Led, INPUT);  // Buton giriş olarak tanımlandı
```

INPUT | OUTPUT

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

INPUT: Belirtilen pini giriş olarak ayarlamaktadır.

OUTPUT: Belirtilen pini cıkış olarak ayarlamaktadır.

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

void loop() : Void loop() fonksiyonuna setup işleminden sonra eklenen ve mikro denetleyici ya da Arduino'nun beslemesi devam ettiği surece tekrarlanan komutlar yazılmaktadır. Buraya yazılan komutlar ile Arduino pinleri arasından karşılaştırma, ilişkilendirme, matematiksel işlemler vs. yapılmaktadır. Yazılan program burada sonsuz döngü içinde çalışmaktadır.

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 kullanılabilir.

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

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.

analogWrite() –PWM : Bir pine bir analog değer (PWM) yazar. Bir LED'i farklı parlaklıklarda aydınlatmak veya çeşitli hızlarda bir motoru sürmek için kullanılabilir.

```
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

GEDİZ FEN LİSESİ BİLGİSAYAR BİLİMİ KUR 2 DERSİ

HALİL İBRAHİM KOCAGÖZ

BİLİŞİM TEKNOLOJİLERİ ÖĞRETMENİ

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.

Serial.begin() : Bilgisayar ile Arduino arasında seri iletişimi başlatmak için kullanılır. Seri veri iletimi için veri hızı saniyedeki bit sayısı (baud) cinsinden ayarlanır.

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.

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.

#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.

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** == (eşit eşit) != (eşit değil) < (küçük) > (büyük) <= (küçük eşit) >= (büyük eşit)	**7. Bitsel Operatörler** & (Bitsel Ve) | (Bitsel Veya) ^ (Bitsel Xor) ~ (Bitsel Değil) << (Bitshift Sol) >> (Bitshift Sağ)
2. Söz Dizimi	**5. Boolean Operatörleri**	**8. Birleşik Operatörler**
; Noktalı Virgül {} Süslü Parantez // Çift Slash /**/ Yıldızlı Slash #define #include	&& (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'de söz dizimi, operatör ve değişkenler