ORTAÖGRETİM

# BİLGİSAYAR BİLİMİ

KUR-2 (öğretmen ders notu)

KUR-2 (öğretmen ders notu)

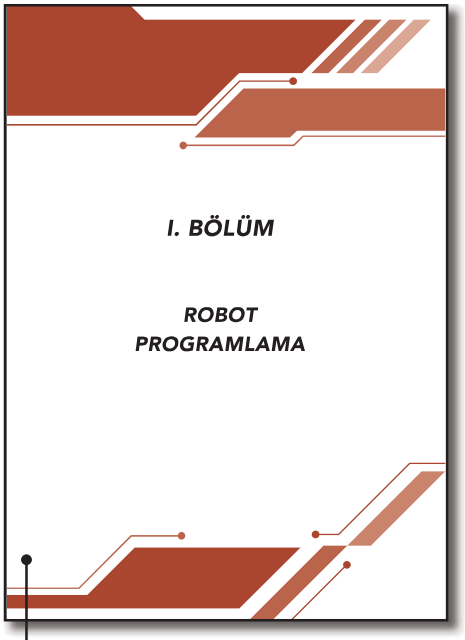
# İÇİNDEKİLER

## I. BÖLÜM

**II. BÖLÜM**

## III. BÖLÜM

# ORGANİZASYON ŞEMASI

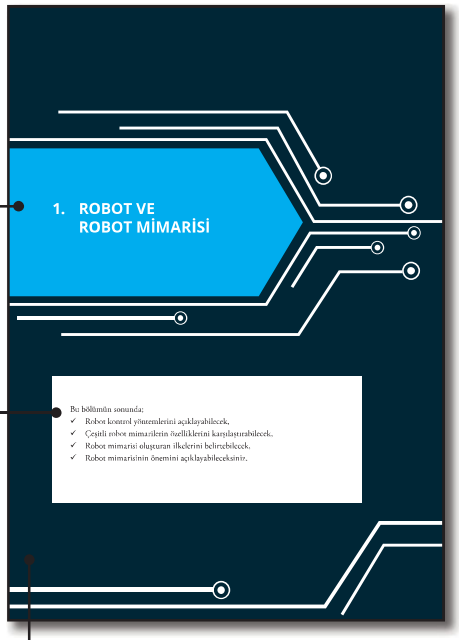



## BÖLÜM KAPAKLARI

Kitap üç ana bölümden oluşmaktadır. Bu bölümler görselde yer alan kapaklar ile başlamaktadır.

## ÜNİTE ADI VE NUMARASI

Ünitenin adı ve numarasını gösterir.




## EDİNİMLER

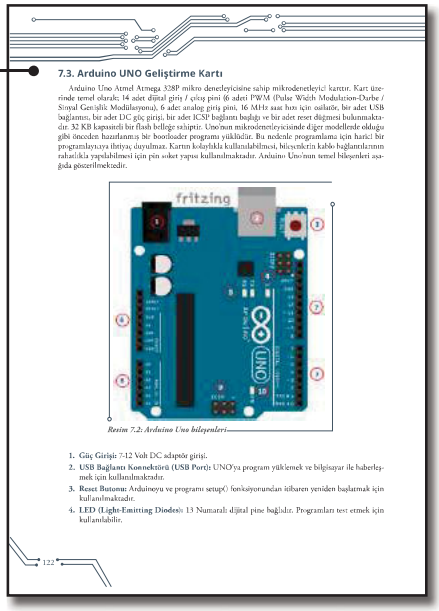
Ünite sonunda elde edilecek bilgi ve becerilerin listesi yer almaktadır.

## ÜNİTE KAPAKLARI

Ünitenin içeriğini ifade eden bir görsel ile her üniteye özgü renk şablonu yer almaktadır.

## KONU BAŞLIKLARI

İçindekiler bölümündeki sırayla listelenen konuların başlıkları yer almaktadır.


7.3. Arduino UNO Geliştirme Kartı

Arduino Uno Atmel Atmega 328P mikro denetleyicisine sahip mikrodenetleyici kartıdır. Kart üzerinde temel olarak; 14 adet dijital giriş / çıkış pini (6 adet 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, bilgisayarla 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 Portu):** 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.

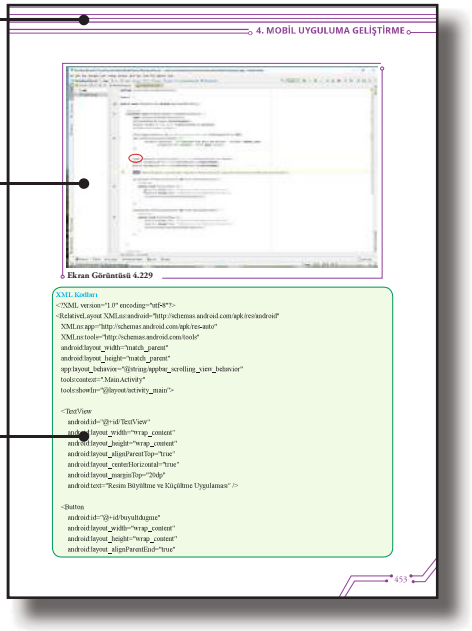
122

## ÜNİTE BANTLARI

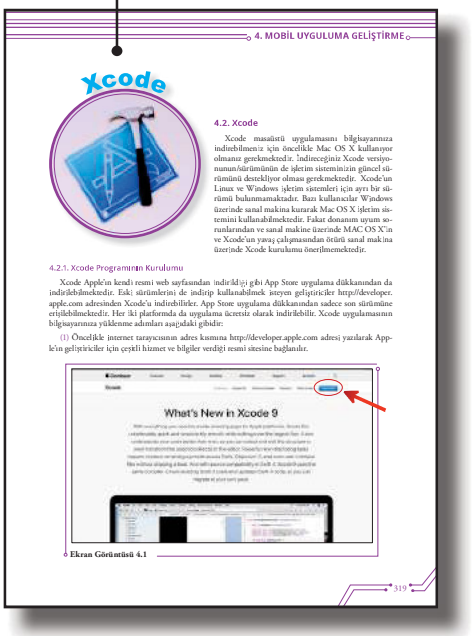
Her ünitede bu bölüm kendine özgü renge sahiptir.

## EKRAN GÖRÜNTÜLERİ

## UYGULAMA KODLARI




III. Bölümde Xcode programı ile yapılan uygulamaların yeraldığı bölümdür.




III. Bölümde Android Stüdyo programı ile yapılan uygulamaların yeraldığı bölümdür.

# I. BÖLÜM

# ROBOT PROGRAMLAMA

# 1. ROBOT VE ROBOT MİMARİSİ

Bu bölümün sonunda,

- ✓ Robot kontrol yöntemlerini açıklayabilecek,
- ✓ Çeşitli robot mimarilerin özelliklerini karşılaştırabilecek,
- ✓ Robot mimarisi oluşturan ilkelerini belirtebilecek,
- ✓ Robot mimarisinin önemini açıklayabileceksiniz.

## 1.1. Robot ve Robot Mimarisi

Robotlar, kendi kendine (otonom) veya önceden programlanmış görevleri yerine getirebilen elektromekanik araçlardır. Bunu yapabilmeleri için çevrelerini algılayabilmeleri, bilgi alabilmeleri ve bu bilgileri işleyerek tepkide bulunmaları, genellikle anlamlı bir amaç için kullanabilmeleri gerekmektedir. Bu açıdan değerlendirdiğimiz de robotun; işlem yapma, işlemin sonucunu belirleme ve karar verme yeteneği bulunmalıdır. Bu özelliklerin bulunduğu elektromekanik bir araç robot özelliğini kazanmaktadır. Robotlar bunları yaparken doğrudan bir operatörün kontrolünde çalışabildikleri gibi bağımsız olarak bir bilgisayar programının kontrolünde de çalışabilirler. Kontrol için farklı sistem ve yöntemler bir arada etkileşimli olarak kullanılabilmektedir. Robotları kontrol etmek için kullanılan sistem ve yöntemler temelde robot mimarilerini oluşturmaktadır.

## 1.2. 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. Aynı şekilde kontrol sistemleri için farklı kontrol yöntemleri kullanılmaktadır. Kullanılan kontrol yöntemleri şunlardır:

1. **Tepkisel (Reactive) Kontrol:** Etki tepki prensibiyle çalışan kontrol yöntemidir. Bu kontrol yöntemi uyaran-cevap ikililerinden oluşan kurallar içerir. Bu kontrol yöntemi "algılama" ve "hareket etme" modelini taban almıştır. Daha önce yapılan işlemleri hafızada tutmadığı gibi belirli bir hafızası da yoktur. Ne yapacağını düşünmediği için çok hızlıdır. Tepkisel kontrolü robotlar öğrenemez (kurallarını değiştirmez) ve ileriye yönelik plan yapamaz.

2. **Bilinçli (Deliberative) Kontrol:** Önce ayrıntılı olarak düşünen, sonra bu düşünce sonucuna göre hareket eden kontrol yöntemidir. Bu kontrol yöntemi "algılama", "planlama" ve "hareket etme" modelini taban almıştır. Planlama-araştırma gerektiği için ve araştırma da zaman aldığından bu kontrol yöntemi yavaştır. Bilinç kontrolü robotlarda düşünme ve hareket etme peş peşe gerçekleştirilir.

3. **Karma (Hibrit) Kontrol:** Düşünme ve hareket işleminin paralel olarak yürütüldüğü kontrol yöntemidir. Tepkisel ve bilinçli kontrol yöntemlerinin birleşmesinden oluşmaktadır.

4. **Davranışsal (Behavioral) Kontrol:** Karma kontrole alternatif olarak sunulmuştur. Tepkisel ve bilinçli hareket özelliklerine sahiptir.

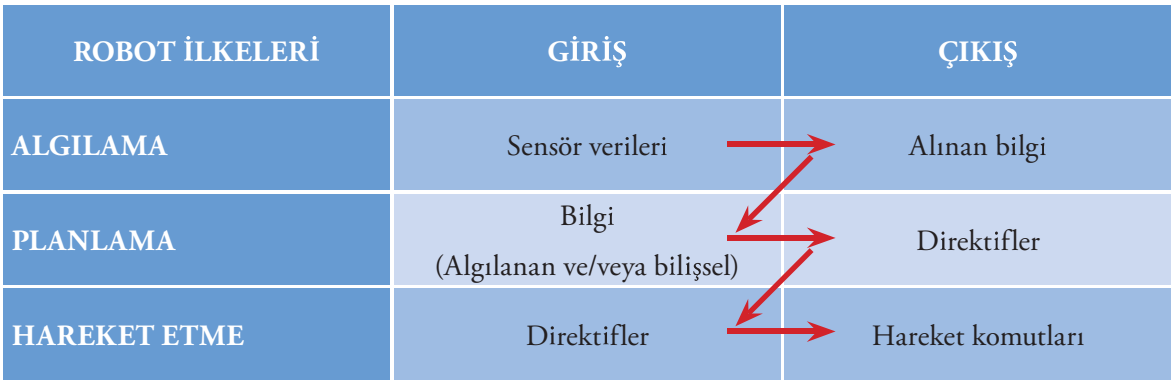
Robot mimarileri bu kontrol yöntemlerinin uygulanmasındaki farklı görüş ve tartışmalardan ortaya çıkmış, belirli dönemlerde belirli mimarilere sahip robotlar üretilmiştir. Robotik anlama (Sense-algılama), planlama (Plan) ve hareket etme (Act-eylem) arasındaki ilişkiler ve algılayıcılar tarafından üretilen duyusal verilerin robotik sistem tarafından işlenmesindeki ve değerlendirilmesindeki farklar çeşitli mimarilerin ortaya çıkmasına neden olmuştur.

## 1.3. 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 bileşen-

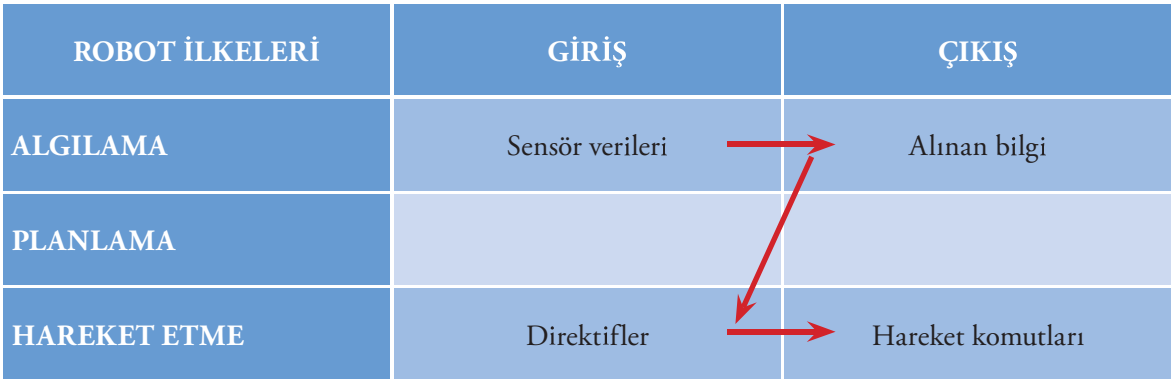
ler için "çıktı" üretmeyi sağlamaktadır. **Plan,** algılayıcılardan veya diğer işlevsel bileşenlerden alınan tüm bilgileri kullanarak, gerçekleştirecek görevler üretmeyi, hareket planı yapmayı sağlar. **Act,** görevleri yerine getiren işlevsel bileşenlerin, hareket biçimini sağlar. Bu ilkelere dayalı olarak geliştirilen Hiyerarşik (Deliberative Kontrol) Mimari, Tepkisel (Reactive Kontrol) Mimari ve Karma (Hibrit Kontrol) Mimari yaygın olarak robot tasarımlarında kullanılır.

Hiyerarşik mimaride; "algılama", "planlama" ve "hareket etme" peş peşe gelen bir süreç olup herhangi bir robotik eylem için çevre algılanmalı, buna dayalı yapılacaklar planlanmalı ve bundan sonra harekete geçilmelidir. Her adımda robot, sonraki hamlesini planlamalıdır. Bilgiler ardışık olarak işlendiği için bileşenlerinin herhangi birindeki başarısızlık bütün sistemi etkilemektedir. Bu tür mimarilerin en önemli dezavantajı performanslarının düşüklüğüdür. Bu model robotun çalışmakta olduğu çevrenin değişmediği sabit durumlar örneğin endüstriyel ortamlar oldukça uygundur.




**Şekil 1.1:** Hiyerarşik mimari

Tepkisel mimaride; "algılama" ve "hareket etme" eş zamanlı olarak gerçekleştirilir. Algılamaya karşılık hareket üretilmektedir. Burada bir planlama süreci bulunmamaktadır. Aşağıdaki şekilde tepkisel mimaride ilkelerin ilişkileri gösterilmiştir.



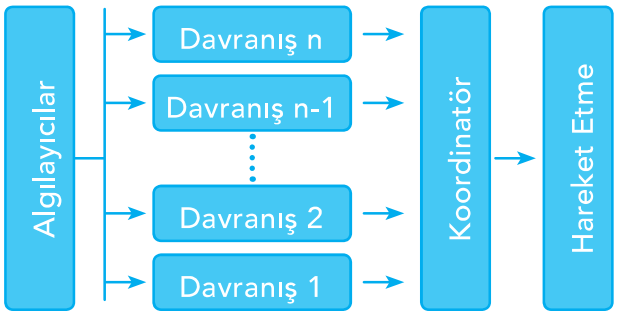

**Şekil 1.2:** Tepkisel mimari

Karma mimaride, 'algılama' ve 'hareket etme' eş zamanlı olarak gerçekleştirilirken planlama da yapılmaktadır. Aşağıdaki şekilde karma mimaride ilkelerin ilişkileri gösterilmiştir.

| ROBOT İLKELERİ | GİRİŞ | ÇIKIŞ |
|---|---|---|
| PLANLAMA | Bilgi (Algılanan ve/veya bilişsel) | Direktifler |
| ALGILAMA HAREKET ETME | Sensör verileri | Hareket komutları |

**Şekil 1.3:** Karma mimari

Geçmiş yıllarda robot mimarileri arasındaki temel fark daha planlamacı veya daha fazla tepkisel olup olmadığına dayanırdı. Daha sonra davranışsal mimariler öne çıkmıştır. Davranışsal mimaride; robotun çevresiyle ilgili durumlar için programlanmasına gerek yoktur. Çevresiyle ilgili bütün bilgiler algılayıcıları aracılığıyla kendisine ulaşmaktadır. Algılayıcılarından elde ettiği bu bilgileri yavaş yavaş yakın çevresindeki değişikliklere göre hareketlerini düzeltmek için kullanmaktadır. Bu yöntemle robot karşılaşabileceği her türlü durumla başa çıkabilecek bir tepkisel davranış sağlamaktadır. Aşağıdaki şekilde davranışsal mimaride ilkelerin ilişkileri gösterilmiştir.




**Şekil 1.4:** Davranışsal mimari

Günümüzde ise Olasılıksal (Probabilistic) Robotik olarak da adlandırılan istatistiksel robotik alan; robotların öngörülemeyen, belirsizlik içeren ortam ve olaylara maruz kaldığı durumlarda istenilen robotik kontrol ve davranışları yapmasını sağlar. Daha önce karşılaşmadığı ortamlarda etkin bir şekilde çalışabilen robotların geliştirilmesini amaçlanmaktadır. Bu nedenle bir robotun, tanımlanan istatistiksel fonksiyonlara dayalı belirli bir hareket ya da eylemi için, en olabilecek sonuçları geliştirmesi ve sonra da bunlardan en uygun olanı uygulayabilmesi gerekmektedir.

Robot mimarisini bir örnek üzerinde anlamaya çalışalım. Robot yönetim sistemi (pilot), robot görüş (vizyon) sistemi ve robot yönlendirme (navigasyon) sistemi olmak üzere üç sistem tarafından kontrol edilen bir robot düşünelim. Aslında bu üç sistem, robotumuzu kontrol etmek için kullandığımız mimariyi oluşturmaktadır.

Robot yönetim sistemi; robotun yönetimini sağlayan, örneğin bir engele çarpmayı önlemek için robotun hareket yönünü değiştiren, robotu hedef bölgeye doğru götürmek için kullanılan sistemdir. Robot görüş sistemi; belirli bir alanındaki bilinen yer işaretlerini, engelleri tanıması veya yenilerini bulması için kullanılan sistemdir. Robot yönlendirme sistemi ise robotun yerini ve hareket yönünü belirlemek için kullanılan sistemdir. Robot kontrolünü sağlamada bu üç sistem işbirliği içinde hareket etmek zorundadır. Örneğin navigasyon sistemiyle robot; hedefe doğru ilerlerken, bulunduğu ortamın belirli bir

alanındaki bilinen yer işaretlerini tanımak veya yenilerini bulmak için vizyon sistemine, aynı zamanda hedef bölgeye doğru ulaşmak için de pilot sistemine ihtiyaç duymaktadır. Pilot sistem bir engeli önlemek için robotun hareket yönünü değiştirmelidir. Üstelik pilot, önünde bir engel bulunup bulunmadığını kontrol etmek için kameraya ihtiyaç duyabilir. Aynı zamanda navigasyon sisteminin, bilinen yer işaretlerini tanıyarak robotun yerini belirlemek için arkasına bakması da gerekebilir. Bu nedenle, farklı sistemler arasındaki bu etkileşimleri sağlamak için bazı koordinasyon mekanizmalarına ihtiyaç duyulmaktadır. Kullanılacak mekanizma ve robotik sistemin mevcut kaynakları, bu etkileşimlerden elde edilen birleşimle birlikte robotun hedefine ulaşmasını sağlamak zorundadır. Bu nedenle robot mimarileri her zaman birer paradigma olarak ele alınmıştır.

## 1.4. Düşünelim/Tartışalım

Yer yer bataklıkların ve ağaçların bulunduğu bir araziyi geçmek zorunda olan bir robotumuzun olduğunu düşünelim. Robotumuzun bu araziyi bataklığa düşmeden, ağaçlara çarpmadan geçebilmesi istenmektedir. Buna göre:

a) Bu robot nasıl bir yönetim sistemine sahip olmalıdır? Niçin?

b) Robotta hangi kontrol araçlarının bulunması istersiniz? Bu araçları niçin tercih ettiniz? Tartışınız.

## 1.5. Değerlendirme Soruları

1. **Bir robotun, kendi kendine (otonom) veya önceden programlanmış görevleri yerine getirebilmesi için aşağıda belirtilen özelliklerden hangisine sahip olması yeterlidir?**

   a) Çevresini algılayabilme yeteneğinin bulunması yeterlidir.

   b) Bulundukları ortamdan bilgi alabilmeleri ve bu bilgileri işleyerek tepkide bulunabilmeleri yeterlidir.

   c) İşlem yapma, işlemin sonucunu belirleme ve karar verme yeteneği bulunmalıdır.

   d) Aldıkları bilgileri genellikle anlamlı bir amaç için kullanabilmeleri yeterlidir.

   e) Bir operatörden bağımsız olarak işlem yapma yeteneklerinin bulunması yeterlidir.

2. **Robotları kontrol etmek için kullanılan farklı sistem ve yöntemler aşağıdakilerden hangisini oluşturmaktadır?**

   a) Robot kontrol teknolojilerini

   b) Robot kontrol yöntemlerini

   c) Robot kontrol sistemlerini

   d) Robot mimarisini

   e) Robot paradigmalarını

3. **Uyaran-cevap ikililerinden oluşan kurallar içeren robot kontrol yöntemi aşağıdakilerden hangisidir?**
   a) Davranışsal (Behavioral) Kontrol
   b) Tepkisel (Reactive) Kontrol
   c) Karma (Hibrit) Kontrol
   d) Bilinçli (Deliberative) Kontrol
   e) Olasılıksal Kontrol
4. **Aşağıdakilerden hangisi karma kontrole alternatif olarak sunulan robot kontrol yöntemidir?**
   a) Davranışsal (Behavioral) Kontrol
   b) Tepkisel (Reactive) Kontrol
   c) Karma (Hibrit) Kontrol
   d) Bilinçli (Deliberative) Kontrol
   e) Olasılıksal Kontrol

5. **Aşağıdakilerden hangisi önce ayrıntılı olarak düşünen, sonra bu düşünce sonucuna göre hareket eden kontrol yöntemidir?**
   a) Davranışsal (Behavioral) Kontrol
   b) Tepkisel (Reactive) Kontrol
   c) Karma (Hibrit) Kontrol
   d) Bilinçli (Deliberative) Kontrol
   e) Olasılıksal Kontrol

6. **Düşünme ve hareket işleminin paralel olarak yürütüldüğü kontrol yöntemi aşağıdakilerden hangisidir?**
   a) Davranışsal (Behavioral) Kontrol
   b) Tepkisel (Reactive) Kontrol
   c) Karma (Hibrit) Kontrol
   d) Bilinçli (Deliberative) Kontrol
   e) Olasılıksal Kontrol

7. **Robotun çalışmakta olduğu çevrenin değişmediği sabit durumlar (örneğin endüstriyel robotlar) için oldukça uygun olan robot mimarisi aşağıdakilerden hangisidir?**
   a) Hiyerarşik Mimari
   b) Tepkisel Mimari
   c) Karma Mimari
   d) Davranışsal Mimari
   e) Olasılıksal Robotik

**8. Daha önce karşılaşmadığı ortamlarda etkin bir şekilde çalışabilen robotların geliştirilmesini amaçlayan robotik alanı aşağıdakilerden hangisidir?**

a) Hiyerarşik Mimari
b) Tepkisel Mimari
c) Karma Mimari
d) Davranışsal Mimari
e) Olasılıksal Robotik

**9. Aşağıdakilerden hangisi robotun çevresiyle ilgili durumlar için programlanmasına gerek olmadığını savunan mimaridir?**

a) Hiyerarşik Mimari
b) Tepkisel Mimari
c) Karma Mimari
d) Davranışsal Mimari
e) Olasılıksal Robotik

**10. Çeşitli robot mimarilerin ortaya çıkmasının nedenini aşağıda verilen görüşlerden hangisi daha güçlü olarak açıklamaktadır?**

a) Robot mimarileri robot kontrol yöntemlerindeki farklılıklardan ortaya çıkmıştır.
b) Belirli dönemlerde belirli mimarilere sahip robotların üretilmesi sonucu ortaya çıkmıştır.
c) Robotik anlama (Sense-algılama), planlama (Plan) ve hareket etme (Act-eylem) arasındaki ilişkilerin yorumlanma şeklinden ortaya çıkmıştır.
d) Robotik algılayıcılar tarafından üretilen bilişsel verilerin robotik sistem tarafından işlenmesindeki ve değerlendirilmesindeki farklardan ortaya çıkmıştır.
e) Robotların işlem yapma yeteneği, işlemin sonucunu belirleme yeteneği ve karar verme yeteneği arasındaki farklılıklardan ortaya çıkmıştır.

# 2. ROBOT TÜRLERİ VE EĞİTSEL AMAÇLI ROBOTLAR

Bu bölümün sonunda,

- ✓ Kullanılan uygulama alanlarına göre robot türlerine örnek gösterebilecek,
- ✓ Hareket mekaniğine göre robot türlerini açıklayabilecek,
- ✓ Eğitsel amaçlı robot türlerinin özelliklerini özetleyebilecek,
- ✓ Robot türlerini sınıflandırabilecek,
- ✓ Robot türlerini karşılaştırabileceksiniz.

## 2.1. Robot Türleri ve Eğitsel Amaçlı Robotlar

Günümüzde, robotlar pek çok alanda çok farklı görevler üstlenmekte ve robotlara devredilen işlerin sayısı sürekli olarak artmaktadır. Bu nedenle pek çok farklı ölçüte göre robotlar sınıflandırılabilmektedir (Benson, 2012; Robotpark, 2017; Robot Wiki, 2017). Örneğin hareketli (mobil) veya sabit olmasına göre, kullanılan alanlara göre, hareketin cinsine göre. Sınıflandırmada temel ölçüt robot için "Ne yapar?" ve "Bunu nasıl yapar?" sorularına verilen yanıt olmalıdır. Genellikle elde edilen yanıtlar robotları; uygulamaya göre robotlar ve hareket mekaniğine göre robotlar olmak üzere iki temel sınıfa ayırmaktadır. Özellikleri ve yapısı nedeniyle herhangi bir tür içerisine girmeyen veya bir tür içerisine henüz dâhil edilmeyen robotlar dikkate alınmamıştır. Eğitsel amaçlı robotlar ise özellikleri nedeniyle ayrı kategoride incelenmiştir.

## 2.2. Kullanılan Uygulama Alanlarına Göre Robotlar

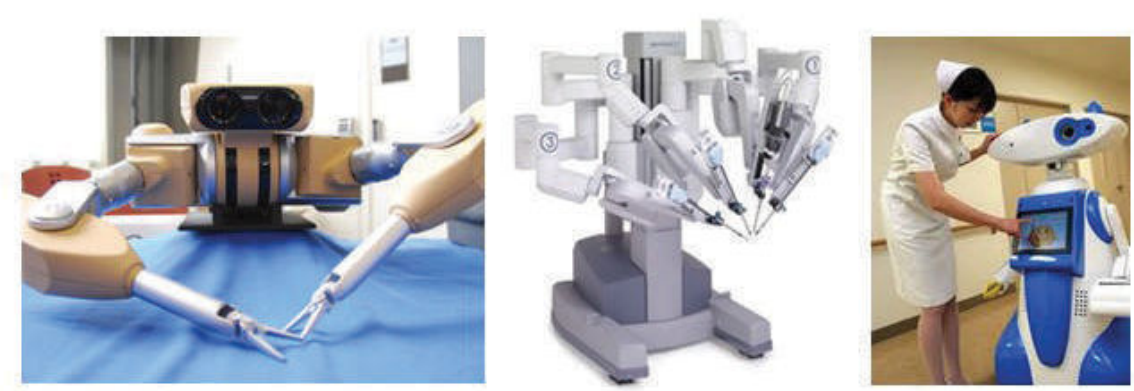
**Endüstriyel Robotlar:** Herhangi bir endüstriyel üretim ortamında kullanılan robotlardır. Endüstriyel robotların en önemli özelliği kollara sahip olmasıdır. Genellikle kaynak, birleştirme, boyama, eşya ve araç üretimi, montaj ve kontrol uygulamalarında kullanılmaktadır. Bu uygulamalar için gerekli olan malzeme taşıma, malzeme yükleme, kesme, tutma, yerleştirme, şekil verme, değiştirme, yüzey kaplama, silindirik ve düzlem yüzey taşlama gibi imalat işlemleri bu kollar aracılığıyla gerçekleştirilmektedir.

**Resim 2.1:** Endüstriyel robotlar

**Ev Robotları:** Evde kullanılmak için geliştirilmiş robotlardır. Elektrikli süpürge, havuz temizleyici, bahçe süpürgeleri, oluk temizliği ve diğer ev ve bahçe işlerini yapabilen robotları içerir. Bu ortamda kullanılan ayrıca, bazı gözetim ve Telepresence robotlar ev robotları olarak kabul edilebilir. Telepresence robotlar insanların fiilî olarak bulunmaması gereken nükleer, kimyasal felaketler gibi senaryolarda, sağlık alanında, askerî casusluk gibi birçok görevde kullanılması öngörülmüş insan kontrolünde çalışan robotlardır. Ürün satışı ve reklam, tur rehberi, gece bekçisi, fabrika müfettişi ve sağlık danışmanlığı için de kullanılmaktadır. Bir uzaktan eğitim sınıfında, bir telepresence robotu, sınıf içi eğitmen olabilir, sınıfın etrafında dolaşabilir ve öğrencilerle yüz yüze etkileşimde bulunabilirler. Kablosuz internet bağlantısı olan uzaktan kumandalı ve tekerlekli yapıdadırlar. Genellikle, bu robotlar video ve ses yetenekleri sağlamak için bir tablet kullanırlar.


**Resim 2.2:** Ev robotları

**Tıbbi Robotlar:** İlaç üretiminde ve dağıtımında, tıbbi kurumlarda, hastanelerde malzeme taşımak, doktorlara yardımcı olmak için kullanılan robotlardır. Bu robotların ilk ve en önemlisini cerrahi robotlar oluşturur. Cerrahi operasyonlarda doktorların en önemli yardımcısı konumundadır.



**Resim 2.3:** Tıbbî robotlar

**Servis Robotları:** Kullanım şekli açısından diğer türlere girmeyen robotlardır. Bu robotlar özerk üretim faaliyetlerinde kullanılmaz. İnsan tarafından yapılan tehlikeli ve zor işlerde insana yardımcı olması için geliştirilmiştir. İnsan refahını sağlamaya dönük her tür yararlı hizmeti gerçekleştirmek için tam veya yarı hizmet desteği veren robotlardır.


**Resim 2.4:** Servis robotları

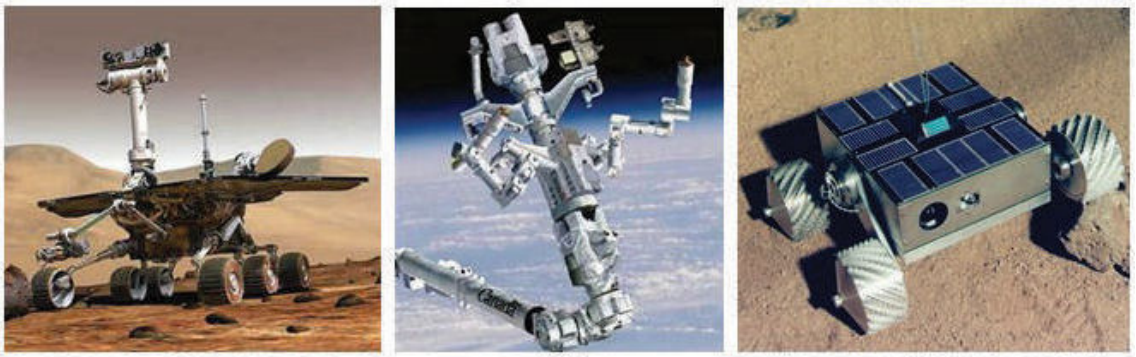
**Askerî Robotlar:** Askerî kullanım için geliştirilmiş robotlardır. Bomba imha robotları, farklı ulaşım robotları, robotik keşif uçağı bu tipte robotlardır. Genellikle başlangıçta askerî amaçlar için oluşturulan bu robotlar kolluk, arama kurtarma ve diğer ilgili alanlarda da kullanılabilmektedir.

**Resim 2.5:** Askerî robotlar

**Eğlence Robotları:** Bunlar herhangi bir hizmette kullanılmayıp çoğunlukla eğlence ve oyun arkadaşlığı için tercih edilen robotlardır. Bu robotlar çok geniş bir yelpazede yer almaktadır. AIBO, Poo-Chi gibi robotik köpekler ve hayvanlar, ses tanıma ve yürüme gibi bazı gelişmiş özellikleri sahip QRIO, Robosapien gibi insansı oyuncak robotlar, hareket simülatörleri olarak kullanılan belden robot kolları gibi fonksiyonel robotlar da bu kategoride değerlendirilmektedir.

**Resim 2.6:** Eğlence robotları

**Uzay Robotları:** Uzayda kullanılmak için üretilen robotlardır. Bu tür robotlar Uluslararası Uzay İstasyonu'nda, Mars'ın keşfinde ve diğer uzay görevlerinde kullanılmaktadır. Bu anlamda uzay sondaları da birer robottur.

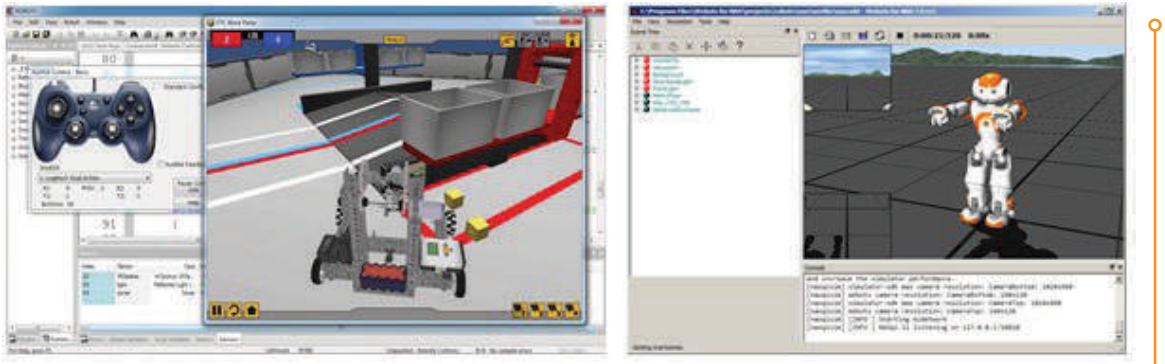


**Resim 2.7:** Uzay robotları

**Hobi ve Yarışma Robotları:** Kişisel olarak yapılan robotlardır. Çizgi takipçileri, sumo-botlar, uçan robotlar gibi sadece eğlence ve herhangi bir görevi yerine getirme konusunda yarışmak için yapılan robotlar bu kategoride değerlendirilmektedir. Birçok ulusal ve uluslararası yarışma bu amaçla gerçekleştirilmektedir.

**Resim 2.8:** Hobi ve yarışma robotları

**Sanal Robotlar:** Sanal robotlar gerçek hayatta fiziksel olarak bulunmayan robotlardır. Sanal robotların yapı taşları bilgisayar programlarıdır. Sanal robotlar, gerçek bir robot simülasyonunu ya da sadece tekrarlanan bir görevi gerçekleştirebilirler. İnternet üzerinde kullanabileceğiniz sohbet robotları, çağrı merkezleri için müşteri temsilcisi robotları gibi pek çok örneği kullanılmaktadır. Robot simülatörleri kullanılarak maliyet ve zaman tasarrufu sağlanmaktadır.



**Resim 2.9:** Sanal robotlar

## 2.3. Hareket Mekaniğine Göre Robotlar

**Sabit Robotlar:** Sabit robotlar sürekli tekrarlayan görevlerini pozisyonlarını değiştirmeden yapan robotlardır. Robotun sabit olması ile anlatılmak istenen robotun temelinin sabit olmasıdır. Yoksa robotun kolları hareket hâlindedir. Çoğu sabit robotlar sanayi ortamlarında imalat ve montaj sektöründe kullanılmaktadır. Bu türün içine Kartezyen / Portal robotlar, Silindirik robotlar, Küresel robotlar, SCARA robotlar, Belden robotlar (robotik kollar) ve Paralel robotlar girmektedir.

**Resim 2.10:** Sabit robotlar

**Tekerlekli Robotlar:** Tekerlekli robotlar pozisyonlarını tekerlekleri ile değiştirebilen mobil robotlardır. Tekerlekli hareketi mekanik olarak sağlamak üretim açısından kolay ve düşük maliyetlidir. Aynı zamanda tekerlekli hareketin kontrolü diğer mobil robotlara oranla daha kolaydır. Bu nedenle tekerlekli robotlar en sık karşılaşılan mobil robot tiplerindendir. Bu robot sınıfı kendi içerisinde çoğunlukla tekerlek sayısına göre sınıflandırılır. Bu türün içerisinde tek tekerlekli robotlar, mobil top robotlar, iki tekerlekli robotlar, üç ve daha fazla tekerlekli robotlar, çok tekerlekli robotlar bulunmaktadır. Bu robotlar düz alanlarda çok etkili olup arazi koşullarında pek yararlı olamaz.

**Resim 2.11:** Tekerlekli robotlar

**Paletli Robotlar:** Paletli robotlar tekerlekli olmasalar da çalışma prensibi açısından tekerlekli robotlara çok benzer olarak çalışır. Bu robotlar hareket etmek için tekerlekleri yerine tanklar gibi paletlerini kullanır. Bu hareket yöntemi düzensiz, yumuşak, kaygan, karlı ya da çamurlu zeminlerde tekerlekli robotlara göre daha fazla avantaj sağlamaktadır. Paletler yerle temas alanını genişlettiği için robotun ağırlığı daha geniş bir yüzeye dağılmakta bu tür zeminlere saplanmasını engellemektedir. Bu nedenle paletli robotlar tekerlekli robotlara göre daha fazla ağırlık taşıyabilir.

**Resim 2.12:** Paletli robotlar

**Ayaklı Robotlar:** Ayaklı robotlar da tekerlekli robotlar gibi mobil robotlardandır ancak hareket yöntemleri ve teknolojisi çoğunlukla tekerlekli robotlara daha üstün ve karmaşıktır. Ayaklı robotlar gelişmiş robotlardır. Hareketlerini sağlamak için ayaklarından faydalanırlar ve tekerlekli robotlara göre sorunlu olan pek çok zeminde hareket edebilirler. Bu tip robotlarda denge en önemli unsurdur. Bu robotların üretim ve kontrolü daha karmaşık ve maliyeti tekerlekli robotlara göre daha yüksektir. Bu tür içerisinde tek ayaklı robotlar, iki ayaklı robotlar (insansı –humanoid-robotlar), üç ayaklı robotlar, dört ayaklı robotlar, altı ayaklı robotlar ve çok ayaklı robotlar sayılabilir. Günümüzde birçok kurum ve üniversite tarafından araştırılan ve geliştirilen robot tipidir.

**Resim 2.13:** Ayaklı robotlar

**Yüzen Robotlar:** Yüzen robotlar, suda hareket edebilen robotlardır. Bu robotlar balıklar gibi yüzgeçlerini kullanarak su içerisinde manevra yapabilmektedir. Genellikle uzaktan kumandayla kontrol edilmekle birlikte otonom olarak da hareket edebilmektedir. Bunlar deniz kaynakları ve balık türleriyle ilgili araştırma ve incelemelerde, su altı arkeolojik keşiflerinde, su altı fotoğrafçılığı, su altı haritacılığı, petrol platformlarını denetleme, inceleme ve olası hasarların tespitinde kullanılmak için tasarlanmış deneysel robotlardır.

**Resim 2.14:** Yüzen robotlar

**Uçan Robotlar:** Uçan robotlar; kanat, pervane ya da balonları ile havada asılı kalarak ve manevra yaparak hareketlerini sağlayan hareket eden robotlardır. Bu robotlara örnek olarak uçak benzeri kanatlı robotlar, kuş/böcek benzeri kanatlı robotlar, pervaneli multikopterler, insansız hava araçları ve balonlu robotlar verilebilir. Bu robotlar doğal afetlerde arama-kurtarma, araştırma, bilgi edinme görevlerinde, insanlar tarafından yapılması gereken tehlikeli görevlerin yerine getirilmesinde, mal ve ürünlerin dağıtımında ve gözetiminde, tarımsal alanların kontrolünde, eğlence ve hobi amacıyla pek çok alanda kullanılmaktadır.

**Resim 2.15:** Uçan robotlar

**Yılan Robotlar:** Bu robotlar sahip oldukları hareket yetenekleri ile her tür ortamda çok yönlü olarak kullanılabilmektedir. Duvarlar ve boşluklar arasında dolaşabilmeleri, arama ve kurtarma faaliyetlerinde bilgi almak için çok uygun yapıda olmaları bu robotların geliştirilme nedenlerini oluşturmaktadır.

**Resim 2.16:** Yılan robotlar

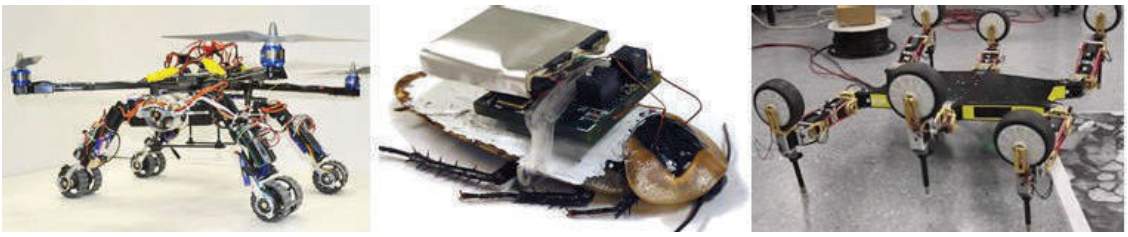
**Yumuşak Elastik Robotlar:** Hareket organları ve yapıları esnek robotlardır. Genellikle gövdeleri silikondan, diğer organları (el, kol vs.) ise elektrik akımıyla uyarıldığında boyut veya şekilde değişiklik yapan bir tür plastikten - elektroaktif polimer - üretilmiş robotlardır. Bu tür, robotik alanında kendilerine yeni yeni yer bulan robotlardır. Bu robotların esin kaynakları genellikle kalamar ve toprak solucanı gibi hayvanlardır. Kendilerine özgü davranışlara sahiptirler.

**Resim 2.17:** Yumuşak elastik robotlar

**Mobil Küresel Robotlar (Robotik Toplar):** Bu robotlar görünüş olarak topa benzeyen robotlardır. Kar veya kum gibi zeminlerde tekerlekli robotlara göre daha fazla performans gösterdikleri, ayrıca düşme riski daha az olduğu için tercih edilmektedir. Daha çok bilimsel araştırmalarda, tehlikeli ve zor arazi koşullarında (gezegen keşiflerinde) kullanım için tasarlanmakla birlikte oyun amacıyla geliştirilen çok fazla çeşidi de bulunmaktadır.

**Resim 2.18:** Mobil küresel robotlar

**Hibrit Robotlar:** Bu tanım hem birden fazla hareket mekaniğine sahip robotlar için hem de sibernetik robotlar için kullanılmaktadır. Sibernetik robotlar hem elektronik hem de biyolojik (canlı) elemanları içermektedir. Biyolojik elemanlar olarak deney hayvanlarının nöronları (genellikle fare) kullanılmaktadır. Bu nöronlara bağlı çipler robotik sistemin temelini oluşturmaktadır. Bu robotların birer sibernetik organizma olduğu rahatlıkla söylenebilir. Üniversite ve araştırma kuruluşlarında geliştirilen araştırma amaçlı deneysel robotlardır.



**Resim 2.19:** Hibrit robotlar

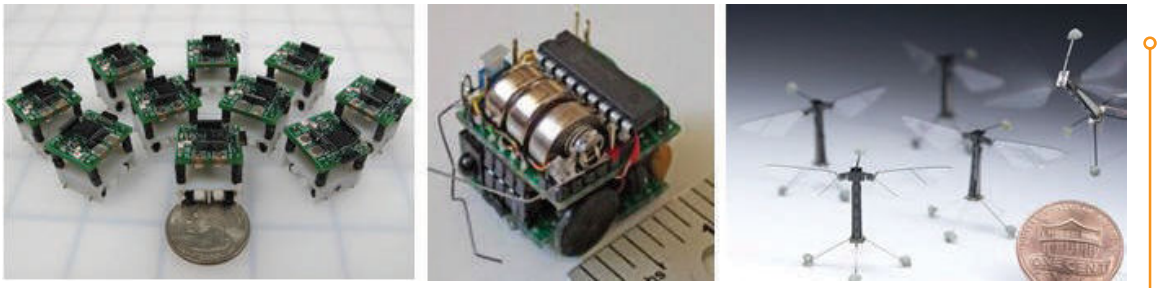
**Sürü Robotları:** Sürü robotları, yapı olarak birleşik ve tek olmak yerine çok sayıda benzer ve basit fonksiyonellikte robotun ortak çalışmaları ile işleyen robotlardır. Modüler robotlarla benzerlikler gösterseler de sürü robotlarının elemanları çok daha fazla sayıda ve fonksiyonel açıdan çok daha basittir.

**Resim 2.20:** Sürü robotları

**Modüler Robotlar:** Modüler robotlar da sürü robotlar gibi robotik sistemi değişik robotik parçalara dağıtılmış robot sistemleridir. Bu tür robotlar yeni koşullara uyum veya yeni görevleri gerçekleştirmek amacıyla kendilerini yeniden yapılandırabilmektedir. Bu amaçla kendi parçalarının bağlantılarını yeniden düzenleyerek kendi şeklini değiştirebilmektedir. Bu robotların sürü robotlarından farkı ise, parçaların daha gelişmiş ve nispeten daha az sayıda olmasıdır. Modüler robotların bir diğer özelliği ise parçalar arası birleşimlerle oluşturdukları konfigürasyonların değişik fonksiyonları bulunan farklı robotlar oluşturabilmesidir. Modüler yapı blokları genellikle tutucular, ayaklar, tekerlekler, kameralar, yük ve enerji depolama gibi birimlerden oluşmaktadır.

**Resim 2.21:** Modüler robotlar

**Mikro Robotlar:** Mikro robotlar, hem mikro hassasiyette işlem yapabilen farklı boyutlardaki robotları hem de mikrometre boyutlarında olup mikro hassasiyette işlem yapabilen robotları ifade etmektedir. Bu tür robotlar uzay çalışmalarında, tıpta, askerî uygulamalarda ve daha pek çok yerde kullanılmaktadır. Mikro robotların kullanıldığı en önemli alan, tıbbi mikro robot uygulamalarıdır. Bu alan insan vücudundaki çeşitli hastalıkları insana rahatsızlık vermeden tanıyıp, doğrudan hasta olan noktaya ilaç verebilecek, biyopsi ve cerrahi müdahale yapabilecek küçük kablosuz robotların geliştirilmesini amaçlamaktadır.



**Resim 2.22:** Mikro robotlar

**Nano Robotlar:** Nano robotlar nanometre düzeyinde hassasiyetle işlem yapabilen çok hassas robotlardır. Bu tür robotlar boyut olarak nanometre düzeyinde ifade edilen çok küçük ölçülerde (atom ve molekül boyutlarında) yapılmış olabildiği gibi nano ölçekte doğrulukla hareket edebilen makro veya mikro ölçekli robotlar da olabilmektedir. Bu robotlar nanoteknoloji, biyoteknoloji ve biyomedikal alanlarının gelişimine katkıda bulunmak için yine bu alanlardaki gelişmelerden yararlanarak üretilmektedir. Mikron ve nanometre boyutlarında cisimleri, parçaları ve biyolojik maddeleri çok hassas olarak manipüle edebilecek nano robotların geliştirilmesi çalışmaları sürdürülmektedir.

**Resim 2.23:** Nano robotlar

**Beam Robotlar:** Beam (Biology, Electronics, Aestetics, Mechanics) robotlar yapılarında temel elektronik bileşenlerin kullanıldığı robotlardır. Bu nedenle beam robotların yapımında genellikle programlanabilir mikroişlemci veya mikrodenetleyici kullanılmaz. Bu tür robotlar temel elektronik elemanlarıyla (foto-diyotlar, kapasitörler, tersleyiciler ve transistörler gibi) yapılan basit lojik devrelerle tıpkı bir sinir ağı gibi oluşturulur. Genellikle oluşturulan mantık devreleri ile algılayıcılardan algıladıkları sinyalleri yorumlayarak kendinden içgüdümle hareket ederler. Genellikle güneş enerjisinden güçlerini alırlar. Bu tür robotlar doğadan esinlenerek yapılmaktadır.

**Resim 2.24:** Beam robotlar

## 2.4. Eğitsel Amaçlı Robotlar

Yüzyılımızda robotların eğitsel amaçlarla kullanımı giderek artmaktadır. Eğitsel amaçlarla geliştirilen ve kullanılan çok fazla sayı ve türde eğitsel robot, robot kiti ve seti ortaya çıkmıştır. Robotlar eğitimde daha çok FTMM (Fen, Teknoloji, Mühendislik ve Matematik) eğitimini desteklemek amaçlı kullanılmaktadır. Fakat günümüzde 21. yüzyıl becerileri olarak adlandırdığımız problem çözme, birlikte çalışma, karar verme, bilgi-işlemsel düşünme gibi çeşitli becerilerin kazanılmasında da etkili olduklarının belirlenmesiyle diğer eğitim alanlarında da (sosyal bilimlerde) kullanımı yaygınlaşmaya başlamıştır. Özellikle öğrencilerin keşfetme, eleştirel düşünebilme ve sosyal becerilerini geliştirmedeki etkileri dikkati çekmektedir. Bu konuda yapılan çalışmaların büyük bir bölümü robotların eğitime olan pozitif etkisini ortaya koymasıyla sonuçlanmıştır. Örneğin programlama dilleri öğretiminde robot kullanımıyla birlikte öğrencilerin problem çözme becerilerinin geliştiğini, iş birliği içerisinde takım çalışmalarıyla bilgiyi paylaşarak öğrendikleri belirlenmiştir. Bunların sağlanmasında kullanılabilecek eğitsel robotlar farklı şekillerde sınıflandırılmaktadır. Bazı sınıflamalar robotun kullanılacağı eğitim türüne göre, bazı sınıflamalar robotun yapısına (Elektronik Robot Kitleri, Mekanik Robot Kitleri, İnsansı –Hümanoid– Robotlar gibi), bazı sınıflamalar maliyetine göre, bazı sınıflamalar da kullanılabilecek yaş guruplarına göre yapılmaktadır.

**Blok (LEGO Benzeri) Tabanlı Robot Montaj Setleri:** Öğrencilerin kendi robotlarını tasarlamaları, inşa etmeleri ve onları programlayarak harekete geçirmeleri için birbirine kolayca bağlanabilen parçalardan oluşan robot setleridir. Bu tür robotik setler oldukça fazla sayıda yapı ve hareket bileşenlerinden oluşmaktadır. Örneğin VEX IQ Süper Kit içerisinde 850 adet yapısal ve hareket bileşeni, 4 adet akıllı motor, 7 çeşit algılayıcı, robot kontrol kumandası, robot kontrol kartı ve piller bir saklama kutusu içerisinde yer almaktadır. LEGO® MINDSTORMS® EV3 Education Ana Set toplam 541 parçadan oluşmakta, içerisinde yapısal bileşenler, EV3 programlanabilir kontrolör, renk algılayıcı, ultrasonik algılayıcı, buton algılayıcı ve jiroskop algılayıcı bulunmaktadır. Yine aynı şekilde Fischertechnik ROBOTICS TXT Discovery Set 310 parçadan oluşmaktadır.

**Resim 2.25:** Blok (LEGO benzeri) tabanlı tobot montaj setleri

**Düşük Maliyetli Programlanabilir Robotik Kol Setleri:** Robotik kollar insan kollarından esinlenerek tasarlanmış ve benzer fonksiyonlara sahip robotik sistemlerdir. Robotik kol, programlanabilir yapıda, mekanik parçaların bütünü ya da karmaşık bir robotun bir parçası olarak nitelendirilebilir. Bir kol sistemi farklı eklemlerin birbirlerine bağlanması ile oluşmaktadır. Eklemlerin bağlantı noktalarında bulunan motorların hareketleri robot kolun yapabileceği hareketlerin göstergesini oluşturmaktadır. Robot kolların uçlarında gerçekleştirilmesi istenen işlemlere uygun bir araç bulunur. Bu araç kavrama, kaldırma, boyama, resim çizme veya yazma gibi değişik işlemler için kullanılabilir. Bu sayede temel robotik ilkelerin ve programlamanın öğretilmesinde kullanmak mümkündür. Bu amaçla gerçekleştirilmiş montajlı veya montajsız olarak bulunabilen pek çok kit satılmaktadır.

**Resim 2.26:** Düşük maliyetli programlanabilir robotik kol setleri

**Düşük Maliyetli Minimum Özelliklerde Mobil Robot Tasarım Kitleri:** Pek çok firmanın ürettiği bu tür eğitsel robotlar kullanıma hazır ama tamamen montajlanmamış şekilde satışa sunulmaktadır. Temel düzeyde özelliklere ve algılayıcılara sahip, ancak genişleme özellikleri ile sonradan herhangi bir bileşenin eklenmesine olanak veren kitlerdir. Parallax Robotics Kitleri (Robotics Arduino Shield Kit, Boe-Bot Robot Kit, ActivityBot), Pololu Robot Kitleri (Zumo Robots, 3pi Robot) ve Makeblock (mBot - STEM Educational Robot Kit, mBot Ranger, Starter Robot Kit) bunlara örnek olarak verilebilir.


**Resim 2.27:** Düşük maliyetli minimum özelliklerde mobil robot tasarım kitleri

**Açık Kaynaklı Düşük Maliyetli Mobil Robot Platformları:** Eğitim amaçlı bu robotlar tamamen açık kaynak kodlu (mekanik ve elektronik yapı) ve açık kaynak yazılım araçları (OpenScad, FreeCAD ve Kicad) ile özel olarak tasarlanmış ve paylaşıma sunulmuş robotlardır. Bu robot platformları öğrenci-

lerin robot programlamayı öğrenmelerine, aynı zamanda kolayca kasayı, yapıyı değiştirebilmelerine ve yeni özel parçaları oluşturmalarına izin vermektedir. Açık kaynak, donanım ve yazılım robotun serbestçe değiştirilebilmesine, kopyalanabilmesine ve İnternet üzerinden paylaşılabilmesine olanak vermektedir. Genel olarak son derece ekonomik bileşenlerden oluşturulmaktadır. Teknoloji ve robot marketlerde satılan onlarca model dışında, Mini Skybot Robot V1, Miniskybot 2, MIT SEG: An Origami-Inspired Segway Robot gibi tanınmış modeller bu tür robotlara örnek olarak verilebilir.

**Resim 2.28:** Açık kaynaklı düşük maliyetli mobil robot platformları

**Düşük Maliyetli, Tam Monte Edilmiş Mobil Robotlar:** Bu robotlar tamamen montajı yapılmış, kullanıma hazır olarak satışa sunulan eğitsel robotlardır. Bazılarında kendine özgü görsel veya metin tabanlı programlama araçları kullanılırken bazılarında açık kaynak programlama araçları kullanılabilmektedir. Genişleme özellikleri daha sınırlı olabilmektedir.

**Resim 2.29:** Düşük maliyetli, tam monte edilmiş mobil robotlar

**Modüler Eğitsel Robot Kitleri:** Modüler eğitsel robotların robotik sistemi değişik robotik parçalara ayrılmıştır. Bu tür robotlar uygun modüllerin eklenmesi veya çıkarılmasıyla farklı iş ve işlemleri için yeniden yapılandırılabilmektedir. Öğrenciler farklı parçaları bir araya getirerek farklı yapıda robotlar ortaya çıkarabilmektedir. Kinematics Modular Robotic Construction Kit, MOSS Modular Robot Construction Kit, Modular Robotics tarafından geliştirilen Cubelets bu tür robotlara örnek olarak verilebilir.


**Resim 2.30:** Modüler eğitsel robot kitleri

**Açık Kaynaklı Minyatür Sürü Robotlar:** Sürü robotları, yapı olarak birleşik ve tek olmak yerine çok sayıda benzer ve basit fonksiyonellikte robotun ortak çalışmaları ile işleyen robotlardır. Daha çok üniversite düzeyinde robotik araştırmacılar için çok sayıda robotun, merkezi bir kontrole ihtiyaç duymadan birlikte çalışarak, sürü seviyesinde hareket etme davranışlarının incelenmesi, algoritmaların denenmesi ve testlerinin yapılması için tasarlanmış robotlardır. Kilobot, Robomote ve Alice bu tür robotlara örnek olarak verilebilir.

**Resim 2.24:** Açık kaynaklı minyatür sürü robotlar

## 2.5. Düşünelim / Araştıralım

Robot programlama dersinde kullanmak üzere eğitsel amaçlı olarak sunulmuş montaj setlerinden, kol setlerinden, tasarım kitlerinden, robot platformlarından, robot kitlerinden birini seçin veya doğrudan bir eğitsel robot seçimi yapınız. Bu seçim için İnternet'te araştırma yapınız. Niçin bu seçimi yaptığınızı, eğitsel robotun hangi özelliklerinin bu seçiminizde etkili olduğunu açıklayınız.

## 2.6. Değerlendirme Soruları

1. **Hangi robot türünün en önemli özelliği kollara sahip olmasıdır?**
   a) Eğitsel robotlar
   b) Servis robotlar
   c) Endüstriyel robotlar
   d) Savaş robotlar
   e) Hibrit robotlar

2. **Aşağıda verilen robot türlerinden hangisi kullanılan uygulama alanlarına göre yapılan sınıflamaya girmez?**
   a) Endüstriyel robotlar
   b) Nano robotlar
   c) Ev robotları
   d) Tıbbi robotlar
   e) Servis robotları

3. **Aşağıda verilen robot türlerinden hangisi hareket mekaniğine göre yapılan sınıflamaya girmez?**
   a) Sabit robotlar
   b) Tekerlekli robotlar
   c) Mobil küresel robotlar
   d) Uzay robotları
   e) Hibrit robotlar

4. **İnsanların fiilî olarak bulunmaması gereken nükleer, kimyasal felaketler gibi senaryolarda, sağlık alanında, askerî casusluk gibi birçok görevde kullanılması öngörülmüş insan kontrolünde çalışan robot türü aşağıdakilerden hangisidir?**
   a) Telepresence robotlar
   b) Endüstriyel robotlar
   c) Tıbbi robotlar
   d) Hibrit robotlar
   e) Modüler robotlar

5. **Düzensiz, yumuşak, kaygan, karlı ya da çamurlu olabilen zor zeminlerde hangi robot türü diğerlerine göre daha fazla avantaj sağlamaktadır?**
   a) Tekerlekli robotlar
   b) Paletli robotlar
   c) Uçan robotlar
   d) Mobil küresel robotlar
   e) Çok ayaklı robotlar

**6. Sabit robotlar, sürekli tekrarlayan görevlerini pozisyonlarını değiştirmeden yapan robotlar için aşağıdaki ifadelerden hangisi yanlıştır?**

a) Sabit robotların temeli bulundukları yüzeye sabitlenmiştir.
b) Sabit robotların kolları hareket halindedir.
c) Sabit robotların robotik sistemi değişik robotik parçalara dağıtılmış robot sistemleridir.
d) Silindirik robotlar, küresel robotlar, SCARA robotlar, belden robotlar (robotik kollar) ve paralel sabit robotlar gurubuna girmektedir.
e) Çoğu sabit robotlar sanayi ortamlarında imalat ve montaj sektöründe çalışmaktadır.

**7. Hangi tür eğitsel robotlar uygun modüllerin eklenmesi veya çıkarılmasıyla farklı iş ve işlemler için yeniden yapılandırabilmektedir?**

a) Açık kaynaklı minyatür sürü robotlar
b) Modüler eğitsel robot kitleri
c) Düşük maliyetli programlanabilir robotik kol setleri
d) Düşük maliyetli minimum özelliklerde mobil robot tasarım kitleri
e) Açık kaynaklı düşük maliyetli mobil robot platformları

**8. Aşağıda verilen eğitsel robot türlerinden hangisi robotun serbestçe değiştirilebilmesine, kopyalanabilmesine ve İnternet üzerinden paylaşılabilmesine olanak vermektedir?**

a) Düşük maliyetli, tam monte edilmiş mobil robotlar
b) Açık kaynaklı minyatür sürü robotlar
c) Modüler eğitsel robot kitleri
d) Açık kaynaklı düşük maliyetli mobil robot platformları
e) Blok (LEGO Benzeri) tabanlı robot montaj setleri

**9. Kavrama, kaldırma, boyama, resim çizme veya yazma gibi değişik işlemler için kullanılabilir eğitsel robot türü aşağıdakilerden hangisidir?**

a) Düşük maliyetli programlanabilir robotik kol setleri
b) Düşük maliyetli minimum özelliklerde mobil robot tasarım kitleri
c) Açık kaynaklı düşük maliyetli mobil robot platformları
d) Blok (LEGO Benzeri) tabanlı robot montaj setleri
e) Modüler eğitsel robot kitleri

**10. Öğrencilerin farklı parçaları bir araya getirerek farklı yapıda robotlar ortaya çıkarabilmeleri için hangi tür eğitsel robota ihtiyacı bulunmaktadır?**

a) Açık kaynaklı minyatür sürü robotlara
b) Düşük maliyetli minimum özelliklerde mobil robot tasarım kitlerine
c) Açık kaynaklı düşük maliyetli mobil robot platformlarına
d) Blok (LEGO Benzeri) tabanlı robot montaj setlerine
e) Modüler eğitsel robot kitlerine

# 3. EĞİTSEL ROBOTTA MEKANİK BİLEŞENLER

Bu bölümün sonunda,

- ✓ Eğitsel robotta kullanılan yapısal bileşenleri listeleyebilecek,
- ✓ Yapısal bileşenlerin görevlerini açıklayabilecek,
- ✓ Eğitsel robotta kullanılan montaj bileşenlerini tanımlayabilecek,
- ✓ Montaj bileşenlerinin görevlerini örneklendirebilecek,
- ✓ Eğitsel robotta kullanılan hareket/eylem bileşenlerini sıralayabilecek,
- ✓ Hareket/eylem bileşenlerinin görevlerini açıklayabileceksiniz.

## 3.1. Eğitsel Robotta Mekanik Bileşenler

Eğitsel robotta kullanılan mekanik bileşenler; gövde veya iskeleti oluşturan şasi, mekanik kollar, aktüatörler ve robot mekanik parçaları gibi yapısal bileşenler, vida, somun, rondela gibi bağlantı parçalarından oluşan bağlantı bileşenleri ile tekerlek, palet ve ayak gibi parçalardan oluşan mekanik hareket/eylem bileşenleridir.

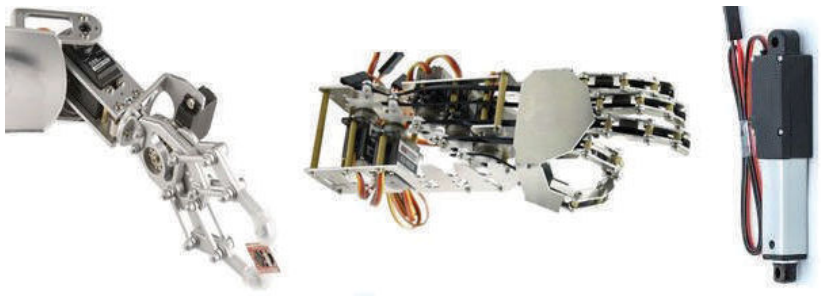
## 3.2. Yapısal Bileşenler (Gövde, İskelet)

Yapısal bileşenler; robotun gövdesini, ana yapıyı oluşturan, diğer bileşenleri üstünde taşıyan gövde, iskelet gibi yapılardır. Plastikten, metalden veya her ikisinden de yapılabilir. Üreticiler tarafından satılan hazır gövdeler, gövde elemanları ve kitleri bulunmaktadır. Standart olarak üretilen pek çok yapısal bileşen bulunmaktadır:

**1. Şaseler:** Robot gövdesini oluşturmak üzere kullanılan çeşitli türde plastik veya metal delikli plakalar veya biçimlendirilerek gerekli bağlantı delikleri açılmış montaja hazır gövdelerdir. Kare, dikdörtgen veya yuvarlak çeşitleri vardır. Ayrıca kullanıma hazır fakat üzerinde herhangi bir elektronik bileşenin bulunmadığı veya gövdeyle birlikte sadece motorların yer aldığı kit şeklinde olanları da bulunmaktadır.

**Resim 3.1:** Şaseler

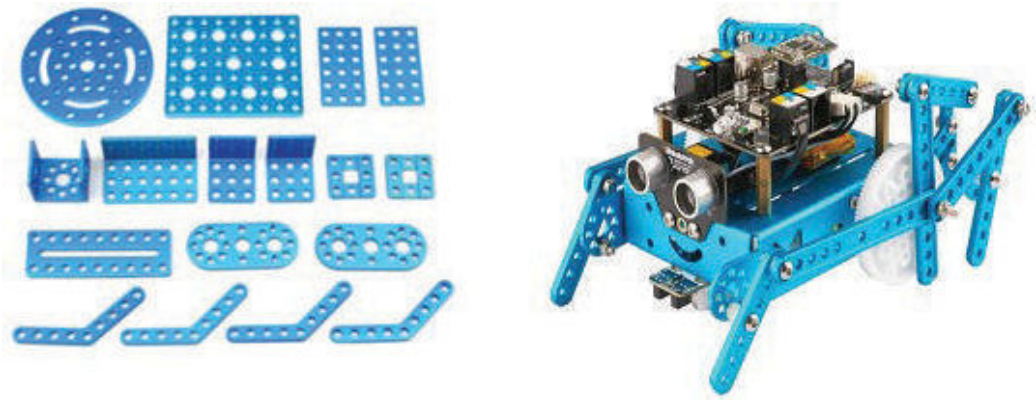
**2. Mekanik Kollar, Aktüatörler:** Robotun bir nesneyi tutması, kaldırması, sürüklemesi sağ sol, yukarı aşağı (pan/tilt) hareketi yapması için kullanılan mekanik bileşenlerdir. Genellikle iki kıskaçtan oluşan kol şeklindedir. Fakat daha fazla uzvu bulunan el şeklinde kollar da bulunmaktadır. Elektronik bileşenleri olmayan veya sadece adım veya servo motorlara sahip çeşitleri birçok üretici tarafından hazır olarak sunulmaktadır.



**Resim 3.2:** Mekanik kollar, aktüatörler

**3. Robot Mekanik Parçaları:** Robota ve robot gövdesine (şase) ekleme yaparak robotik platformu istenilen şekilde oluşturmayı ve geliştirmeyi amaçlayan yapısal bileşenlerdir. Bağlantı elemanları kul-

lanılarak robota mekanik eklemeler yapılabilmektedir. Aşağıdaki fotoğrafta yer alan tekerlekli robota mekanik parçalar eklenerek, ayaklı robot haline getirilmiştir.



**Resim 3.3:** Mekanik robot parçaları

### 3.2.1.Yapısal Bileşenlerin Görevleri

Yapısal bileşenlerin temel görevi robot için ana taşıyıcı yapıyı oluşturmaktır. Gerektiği zaman eklemeler yapılmasına olanak sağlayarak robotun geliştirilmesini, yeni eklemeler yapılabilmesini sağlamaktır. Şasi ve mekanik parçalarının üzerinde bulunan çeşitli deliklerin yardımıyla kullanılacak bileşenlerin montajını oldukça kolaylaştırırlar. Robot bileşenlerinin kolay ve hızlıca montajına izin veren bir yapıya sahiptirler.

## 3.3. Montaj Bileşenleri (Bağlantı Parçaları)

Robotu meydana getiren bileşenleri gövdeye veya birbirine bağlamak için kullanılan vida, somun, rondela, yükselteç, küçük delikli levha gibi elemanlardır. Metal veya plastik çeşitleri bulunmaktadır.


**Resim 3.4:** Montaj Bileşenleri (Bağlantı Parçaları)

### 3.3.1. Montaj Bileşenlerinin (Bağlantı Parçaları) Görevleri

Montaj bileşenlerinin görevleri robotu meydana getiren bileşenleri gövdeye veya birbirine bağlayarak bir bütün oluşturmalarını sağlamaktır. Bu sayede hem bileşenler bir arada olurken hem de hareket esnasında robotun zarar görmesi önlenmektedir. Ayrıca bileşenlerin istenilen şekilde bağlanmasını sağladıkları için daha esnek kullanım sunarlar. Örneğin yükselteçler kullanarak bileşenleri gövde üzerinde daha yüksek veya daha yakın bağlamak mümkün olur.

## 3.4. Mekanik Hareket/Eylem Bileşenleri (Tekerler, Paletler, Ayaklar)

Robotun tercih edilen hareketine uygun olarak kullanılan mekanik bileşenlerdir. Tekerleğe dayalı hareket için çok çeşitli ölçü ve türlerde tekerlekler veya paletler kullanılırken, yürümeye dayalı hareket için servo veya step (adım) motor içeren çeşitli türlerde ayaklar kullanılmaktadır. Tekerlekli, paletli, iki veya daha çok bacaklı mekanik kitler bulunmaktadır.

**Resim 3.5:** Mekanik Hareket/Eylem Bileşenleri (Tekerler, Paletler, Ayaklar)

### 3.4.1. Mekanik Hareket/Eylem Bileşenlerinin (Tekerler, Paletler, Ayaklar) Görevleri

Hareket/eylem bileşenlerinin temel görevi robotun hareketini (yürümesini) sağlamak için gerekli mekanik yapıyı sağlamaktır. Robotun sınıf ortamında kullanımında tekerlekli çözümler tercih edilirken, dış mekân kullanımında paletli çözümlerin tercih edilmesi yüzey şartları nedeniyle daha uygun olabilir. İnsansı robotlarda ise hareket için ayaklı çözümler tercih edilebilir. Bu tür robotların hareketlerinin programlanması diğerlerine göre daha zor olabilir.

## 3.5. Düşünelim / Araştıralım

Robot programlama dersinde kullanmak üzere bir eğitsel robot yapacağınızı düşünerek gerekli olabilecek mekanik bileşenlerin seçimi için İnternet'te araştırma yapınız. Niçin bu bileşenleri seçtiğinizi, bileşenlerin hangi özelliklerinin seçiminizde etkili olduğunu açıklayınız.

## 3.6. Değerlendirme Soruları

1. **Robotun gövdesini, ana yapıyı oluşturan diğer bileşenleri üstünde taşıyan gövde, iskelet gibi yapıların genel adı aşağıdakilerden hangisidir?**
   a) Elektromekanik bileşenler
   b) Yapısal bileşenler
   c) Montaj bileşenleri
   d) Mekanik hareket/eylem bileşenleri
   e) Elektronik bileşenler

2. **Robotun gövdesini oluşturmak üzere kullanılan çeşitli türde plastik veya metal delikli plakalar veya biçimlendirilerek gerekli bağlantı delikleri açılmış montaja hazır bileşenlere ne ad verilir?**
   a) İskelet
   b) Gövde
   c) Şase
   d) Montaj bileşeni
   e) Aktüatör

3. **Robotun bir nesneyi tutması, kaldırması, sürüklemesi sağ-sol, yukarı-aşağı (pan/tilt) hareketi yapması için kullanılan mekanik bileşenlere ne ad verilir?**
   a) İskelet
   b) Gövde
   c) Şase
   d) Montaj bileşeni
   e) Aktüatör

4. **Robota gövdesine çeşitli mekanik eklemeler yaparak, robotik platformu istenilen şekilde oluşturmayı veya geliştirmeyi amaçlayan bileşenlere ne ad verilir?**
   a) Robot mekanik parçaları
   b) İskelet
   c) Gövde
   d) Şase
   e) Montaj bileşeni

5. **Aşağıdakilerden hangisi yapısal bileşenlerin görevlerinden biri değildir?**
   a) Robot için ana taşıyıcı yapıyı oluşturmaktır.
   b) Gerektiği zaman eklemeler yapılmasına olanak sağlamaktır.
   c) Robotun yeteneklerinin geliştirilmesini, yeni özellikler kazanmasını sağlamaktır.
   d) Kullanılacak bileşenlerin montajını kolaylaştırmaktır.
   e) Robot bileşenlerinin kolay ve hızlıca adaptasyonunu sağlamaktır.

**6. Robotu meydana getiren bileşenleri gövdeye veya birbirine bağlamak için kullanılan vida, somun, rondela, yükselteç, küçük delikli levha gibi elemanlara ne ad verilir?**

a) Elektromekanik bileşenler
b) Mekanik hareket/eylem bileşenleri
c) Yapısal bileşenler
d) Montaj bileşenleri
e) Elektronik bileşenler

**7. Aşağıdakilerden hangisi montaj bileşenlerin görevlerinden biri değildir?**

a) Robotu meydana getiren bileşenleri gövdeye veya birbirine bağlamaktır.
b) Robotun mekanik tasarımını kolaylaştırmaktır.
c) Robotu meydana getiren bileşenlerin bir bütün oluşturmalarını sağlamaktır.
d) Hareket esnasında robotun zarar görmesini önlemektir.
e) Bileşenlerin istenilen şekilde bağlanmasını sağlayarak daha esnek kullanım olanağı sunmaktır.

**8. Düzgün olmayan yüzeylerde hızlıca hareket etmesi için geliştirilen bir robot için uygun hareket/eylem bileşeni aşağıdakilerden hangisidir?**

a) Tekerlek
b) İki ayak
c) İkiden fazla ayak
d) Palet
e) Kanat

**9. Eğitsel robotta kullanılan mekanik bileşenler için aşağıdaki ifadelerden hangisi yanlıştır?**

a) Mekanik bileşenleri olmayan robot yapmak imkânsızdır.
b) Mekanik bileşenler robotun bir bütün olmasını sağlar.
c) Mekanik bileşenler sağlam robotlar yapmak için gereklidir.
d) Mekanik bileşenler sayesinde modüler robotlar geliştirilebilmektedir.
e) Mekanik bileşenler metal, plastik veya ağaç gibi materyallerden meydana gelebilir.

**10. Aşağıdakilerden hangisi hareket/eylem bileşenlerinden biri değildir?**

a) Tekerlekler
b) Ayaklar
c) Paletler
d) Kanatlar
e) Kollar

# 4. EĞİTSEL ROBOTTA ELEKTROMEKANİK BİLEŞENLER

Bu bölümün sonunda,

- ✓ Buton, anahtarlar ve konektör bileşenlerinin görevlerini açıklayabilecek,
- ✓ Güç bileşenlerini listeleyebilecek,
- ✓ Güç bileşenlerinin görevlerini örneklendirebilecek,
- ✓ DC motorların görevlerini tanımlayabilecek,
- ✓ Servo motorların görevlerini sıralayabilecek,
- ✓ Adım (Step) motorların görevlerini açıklayabileceksiniz.

## 4.1. Eğitsel Robotta Elektromekanik Bileşenler

Eğitsel robotlarda kullanılan mekanik bileşenler; butonlar, anahtarlar ve konektörler gibi bağlantı bileşenleri; pil, akü, batarya gibi güç bileşenleri; hareket sağlamak için kullanılan doğru akım, servo ve adım motor gibi bileşenlerdir.

## 4.2. Bağlantı Bileşenleri (Butonlar, Anahtarlar, Konektörler ve Klemensler)

**1. Butonlar:** Üzerine basıldığında, robottaki veya yazılımdaki önceden belirlenmiş mekanik veya elektronik bir sürecin başlamasını, sonlanmasını veya kontrolünü sağlayan basit kontak mekanizmalarıdır. Butonların pek çok çeşitleri bulunmaktadır. Fakat hepsi itme veya üzerine uygulanan kuvvet karşısında tepki veren yay sisteminden oluşurlar. Genellikle butonların tek konumu ve tek kontağı vardır. Stop (durdurma) butonları buna örnek olarak verilebilir.

**Resim 4.1:** Butonlar

**2. Anahtarlar:** Elektrikle çalışan bütün sistem ve devrelerde, devreyi açıp kapatmaya yarayan elemanlardır. Basmalı veya çevirmeli tiplerde pek çok çeşidi bulunmaktadır. Örneğin robotun veya kontrol devrelerinin açılması ve kapatılması için kullanılan iki yollu anahtarların iki konumu ve normalde kapalı ve açık olmak üzere iki kontağı vardır. Kalıcı tip anahtarlar ise bir yollu ve iki yollu yapılabilmektedir. Bir yollu çeşitlerinde; anahtara basılınca veya çevrilince kontak açık ise kapanır, kapalı ise açılır. İki yollu çeşitlerinde; anahtara basılınca veya çevrilince, kontaklardan biri açılır, diğeri kapanır.


**Resim 4.2:** Anahtarlar

**3. Konektörler ve Klemensler:** Robotun yapısında kullanılan dc, servo veya adım motor gibi elektromekanik ve robotik kontrol kartları, algılayıcılar, güç kaynakları ve motor sürücüleri gibi elektronik bileşenlerin birbirine bağlantısı için kullanılan kablo bağlantı elemanlarıdır. Her türlü bileşenin kablolarla birbirine bağlanması için geliştirilmiş çok fazla tür ve sayıda konektör çeşidi bulunmaktadır. Klemensler ise kabloların birbirine bağlanması için kullanılmaktadır.

**Resim 4.3:** Konektörler

### 4.2.1. Bağlantı Bileşenlerinin (Butonlar, Anahtarlar ve Konektörler) Görevleri

Butonların görevi, üzerine basıldığında robottaki veya yazılımdaki önceden belirlenmiş mekanik veya elektronik bir sürecin başlamasını, sonlanmasını veya kontrol edilmesini sağlamaktır. Anahtarların görevi ise elektrikle çalışan bütün sistem ve devrelerde, devreyi açıp kapatmaktır. Konektörlerin görevi ise her türlü donanımın kablolarla birbirine bağlanmasını sağlamaktır.

## 4.3. Güç Bileşenleri (Pil, Akümülatör, Batarya)

**1. Piller:** Kimyasal enerjinin depolanabilmesi ve elektriksel forma dönüştürülebilmesi için kullanılan küçük hacimli temel güç kaynaklarıdır. Piller, bir veya daha fazla elektrokimyasal hücre, yakıt hücreleri veya akış hücreleri gibi, farklı elektrokimyasal yapılardan meydana gelir. Genel olarak kullanıldıktan sonra atılan (Non-rechargeable) ve tekrar şarj edilebilen (Rechargeable) piller olarak ikiye ayrılır. Eğitsel robotların enerji kaynağı olarak bu pillerin her iki türü de oldukça yaygın olarak kullanılmaktadır.


**Resim 4.4:** Piller

**2. Akümülatörler:** Elektrik enerjisini kimyasal enerji olarak depolayıp, istenildiğinde bunu tekrar elektrik enerjisi olarak geri veren pillerden daha güçlü enerji kaynaklarıdır. Yüksek güç tüketimi olan robotların enerji ihtiyaçlarını karşılamak için kullanılmaktadır. Piller gibi elektrokimyasal yapılardan meydana gelirler.


**Resim 4.5:** Akümülatörler

**3. Bataryalar:** Paralel ya da seri bağlanan birden çok pil veya akümülatör gibi kimyasal enerjiyi elektrik enerjisine dönüştüren üreteçlerden oluşturulan güç kaynaklarıdır. Robotlarda, genel olarak tablet ve taşınabilir bilgisayarda yaygın olarak bataryalar kullanılmaktadır.


**Resim 4.6:** Bataryalar

### 4.3.1.Güç Bileşenlerinin (Pil, Akümülatör, Batarya) Görevleri

Güç bileşenlerinin görevi robotun çalışması için ihtiyaç duyduğu elektrik enerjisini karşılamaktır. Bu amaçla gerekli voltaj ve akım değerlerinin karşılanması güç bileşenlerinin görevidir. Kesintisiz ve/veya yedek enerji ihtiyaçları için elektrik enerjisinin depolanması ve gerektiğinde geri alınması (kullanılması) yine güç bileşenlerinin görevidir. Hareketsiz ve sabit robotların elektrik ihtiyacı için yukarıda açıklanan güç bileşenleri yerine şehir şebekesinden adaptörle elektrik alınması daha uygun seçenek olacaktır.

## 4.4. Hareket Bileşenleri (Doğru Akım -DC-, Servo ve Adım Motorlar)

**1. Doğru Akım (DC) Motorlar:** Doğru akım elektrik enerjisini dairesel mekanik enerjiye dönüştüren makinelerdir. Robotun hareketi için kullanılan temel bileşenlerden biridir. Düşük maliyetli robotlar

üretmek için uygundur. ırçalı, fırçasız, reduktörlü, enkoderli, enkoderli ve reduktörlü çeşitleri bulunmaktadır. Fırçalı motor, motorun hareketli olan bölümüne elektrik akımını aktarılabilmek için fırça ve kolektör kullanılan motor türüdür. Fırçasız motor ise motorun hareketli olan bölümüne elektrik akımı aktarılabilmek için fırça ve kolektör yerine elektronik aksam kullanılan motor türüdür. Reduktörlü motor, şanzıman, dişli kutusu veya dişli sistemi kullanılan motor türüdür. Enkoderli motor ise dönme hareketini ardışık sayısal sinyallere çevirerek dönme hızı ve dönme sayısı hakkında bilgi veren motor türüdür. Standart robot uygulamaları için fırçalı motorlar kullanılırken, yüksek performans isteyen uygumalar için fırçasız motorlar kullanılmaktadır. Motorun devir hızını azaltarak daha yüksek tork (motordan tekerleğe iletilen itme -dönme momenti- kuvveti) elde etmeyi gerektiren uygulamalar için ise redüktörlü bulunan motorlar tercih edilmektedir. Dönme hızı ve dönme sayısını kontrol etmeyi gerektiren uygulamalar için enkoderli motorlar kullanılmaktadır.

**Resim 4.7:** Doğru akım (DC) motorlar

**2. Servo Motorlar:** Hareket kontrolü yapılabilen (dönüş yönü, mekaniksel konum, hız veya ivme gibi parametrelerin kontrol edilebildiği) motor çeşitleridir. Bu amaçla gerekli olan sürücü ve kontrol devresi motor içerisinde bulunmaktadır. Bu motorlar, DC motorlardan farklı olmak üzere istenilen pozisyonda sabit kalacak şekilde tasarlanmıştır. Çoğunlukla 0 ile 180 derece arası açılarda çalışırlar. Robotun bileşenlerinin hareketi (kol, ayak, dönen gövde, baş gibi) ve bunların hassas pozisyon kontrolü için kullanılan temel bileşenlerden biri olduğu için robot teknolojisinde en çok kullanılan motor çeşididir. Yürüyen robotlar için yine bu tip motorlar kullanılmaktadır.

**Resim 4.8:** Servo motorlar

**3. Adım (Step) Motorlar:** Çok hassas konum kontrol olanağı ve düşük devirde yüksek tork sağlayan motorlardır. Bu motorlarda dönme hareketi istenildiği kadar açıya bölünerek, açısal konumu adımlar halinde değiştirilebilmekte, hassas konum ve pozisyon düzenlemeleri yapılabilmektedir. Adım açısı motorun yapısına bağlı olarak 90°, 45°, 18°, 7.5°, 1.8° veya daha değişik açılarda olabilmektedir. Örneğin robotun kolunun 17° dönmesini istiyorsak adım motor kullanılmalıdır. Adım motor kullanarak tekerlekli robotların daha hassas ve ölçülebilir manevralar yapabilmesi de sağlanmaktadır.

**Resim 4.9:** Adım (Step) motorlar

### 4.4.1. Hareket Bileşenlerinin (Doğru Akım -DC-, Servo ve Adım Motorlar) Görevleri

Hareket bileşenlerinin görevi robotun hareketi için gerekli motor gücünü sağlamaktır. Bu amaçla mekanik hareket/eylem bileşenlerinin ihtiyaç duyduğu türde dairesel mekanik enerji, hareket bileşenleri tarafından karşılanır. Bu dairesel enerji robotun hareket biçimine göre değiştirilebilmektedir. İstenildiğinde doğrusal şekle de dönüştürülebilmektedir. Örneğin robotun hareketi için tekerlek kullanılıyorsa tekerleği döndürmek, ayakla yürüyorsa ayakları yürütmek bu bileşenlerin görevidir.

## 4.5. Düşünelim / Araştıralım

Robot programlama dersinde kullanmak üzere bir eğitsel robot yapacağınızı düşünerek gerekli olabilecek elektromekanik bileşenlerin seçimi için İnternet'te araştırma yapınız. Niçin bu bileşenleri seçtiğinizi, bileşenlerin hangi özelliklerinin seçiminizde etkili olduğunu açıklayınız.

## 4.6. Değerlendirme Soruları

1. **Her türlü elektrik ve elektronik bileşenin kablolarla birbirine bağlanması için geliştirilmiş kablo bağlantı yapılarına ne ad verilir?**
   a) Buton
   b) Anahtar
   c) Konektörler
   d) Klemens
   e) Duy

2. **Aşağıdakilerden hangisi bağlantı bileşenlerinin görevi değildir?**
   a) Önceden belirlenmiş bir sürecin başlamasını, sonlanmasını veya kontrol edilmesini sağlamak
   b) Bütün elektrik ve elektronik sistem ve devrelerde, devreyi açıp kapatmak
   c) Her türlü donanımın kablolarla birbirine bağlanmasını sağlamak
   d) Her türlü kablonun birbirine bağlanmasını sağlamak
   e) Robotun bileşenlerini birbirine bağlamak

3. **Kimyasal enerjinin depolanabilmesi ve elektriksel forma dönüştürülebilmesi için kullanılan küçük hacimli temel güç kaynakları aşağıdakilerden hangisidir?**
   a) Fotovoltaik panel
   b) Akümülatör
   c) Batarya
   d) Pil
   e) Yakıt hücresi

4. **Elektrik enerjisini kimyasal enerji olarak depolayıp, istenildiğinde bunu tekrar elektrik enerjisi olarak geri veren güçlü enerji kaynaklarına ne ad verilir?**
   a) Fotovoltaik panel
   b) Akümülatör
   c) Batarya
   d) Pil
   e) Yakıt hücresi

5. **Pillerin bir araya gelerek oluşturdukları pil gruplarına ne ad verilmektedir?**
   a) Batarya
   b) Fotovoltaik panel
   c) Akümülatör
   d) Pil
   e) Yakıt hücresi

6. **Yüksek güç tüketimi olan robotların enerji ihtiyaçlarını karşılamak için aşağıdaki seçeneklerden hangisinin kullanılması daha uygundur?**
   a) Batarya
   b) Akümülatör
   c) Fotovoltaik panel
   d) Pil
   e) Yakıt hücresi

7. **Motorun devir hızını azaltarak daha yüksek tork elde etmeyi gerektiren uygulamalar için hangi motor türü tercih edilmelidir?**
   a) Fırçalı motor
   b) Fırçasız motor
   c) Servo motor
   d) Enkoderli motor
   e) Redüktörlü motor

**8. Aşağıdakilerden hangisi hareket kontrolü yapılabilen (dönüş yönü, mekaniksel konum, hız veya ivme gibi parametrelerin kontrol edilebildiği) motor çeşididir?**

a) Fırçasız motor
b) Step motor
c) Enkoderli motor
d) Servo motor
e) Redüktörlü motor

**9. Dönme hareketini istenildiği kadar açıya bölerek, açısal konumu adımlar hâlinde değiştirebilen, hassas konum ve pozisyon düzenlemeleri yapabilen motor çeşidi aşağıdakilerden hangisidir?**

a) Fırçasız motor
b) Step motor
c) Enkoderli motor
d) Servo motor
e) Redüktörlü motor

**10. Dönme hızı ve dönme sayısını kontrol etmeyi gerektiren uygulamalar için hangi tür motorlar kullanılmalıdır?**

a) Redüktörlü motor
b) Enkoderli motor
c) Fırçasız motor
d) Fırçalı motor
e) Adım motor

# 5. EĞİTSEL ROBOTTA ELEKTRONİK BİLEŞENLER

Bu bölümün sonunda,

- ✓ Motor sürücü katlarının görevlerini listeleyebilecek,
- ✓ USB-UART çeviricilerin görevlerini tanımlayabilecek,
- ✓ Kablosuz iletişim bileşenlerinin görevlerini özetleyebilecek,
- ✓ Algılayıcı çeşitlerini listeleyebilecek,
- ✓ Algılayıcı çeşitlerinin görevlerini açıklayabilecek,
- ✓ Robotik programlamada kullanılan işlemcileri tanımlayabilecek,
- ✓ Robotik programlamada kullanılan işlemcilerinin görevlerini yorumlayabilecek,
- ✓ Robot kontrol kartlarını listeleyebilecek,
- ✓ Robot kontrol kartlarının görevlerini açıklayabileceksiniz.

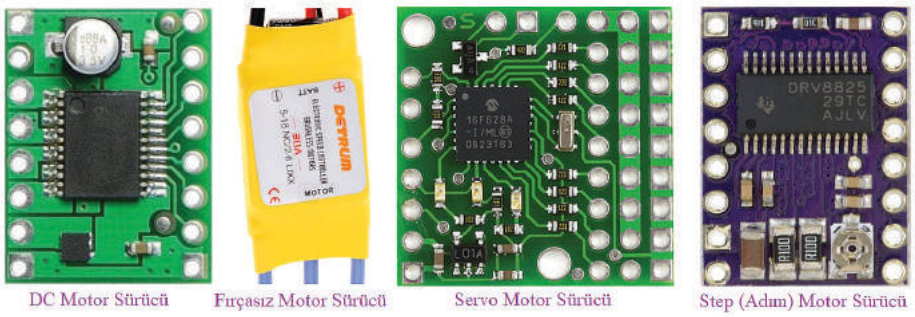
## 5.1 Eğitsel Robotta Elektronik Bileşenler

Bu bölümde eğitsel robotta kullanılan elektronik bileşenler ve bu bileşenlerin görevleri açıklanmıştır. Bu kapsamda motor sürücü kartları, usb-uart çeviriciler, kablosuz iletişim bileşenleri, robotik uygulamalarda kullanılan algılayıcılar (sensörler), algılayıcıların mikrodenetleyici kartlarla haberleşmesi/bağlanması, robotik programlamada kullanılan işlemciler, mikrodenetleyici kartlar (geliştirme kartları), mikrodenetleyici kartlar için kalkanlar (shields) konuları ele alınmıştır.

## 5.2. Motor Sürücü Kartları ve Görevleri

Robotlarda kullanılan motorların kontrol edilebilmesi (çalışma, durma, ileri geri hareket etme, hızlanma, yavaşlama vb.) için kullanılan bileşenlerdir. Ayrı bir kart olarak alınabileceği gibi, robot kontrol kartlarının ya da mikro kontrolör kartlarının dâhilî bir bileşeni olarak da bulunabilmektedir. Tek bir motorun kontrolünden, çok sayıda ve türde motorun kontrolüne kadar çok çeşitli yapıda motor kontrol kartları bulunmaktadır. Birden fazla sayıda ve türde motorun hız ve yönlerini birbirinden bağımsız olarak kontrol edebilmektedir.

Tercih edilecek motorun türüne göre farklı motor sürücü kartlarının kullanılması gerekmektedir. Fırçalı doğru akım motorları için DC Motor Sürücüler, fırçasız doğru akım motorları için Fırçasız Motor Sürücüler (Bunlara Electronic Speed Controller, ESC adı verilmektedir.) kullanılmaktadır. Aynı şekilde Servo motorlar için Servo Motor Sürücüler ve Adım (Step) motorlar için Adım Motor Sürücülerin kullanılması gerekmektedir. Fırçasız doğru akım motorları hariç diğer türlerin kontrolü için ortak kullanımlı (her üç tür motoru bir arada kontrol edebilen) kartlar bulunmaktadır.



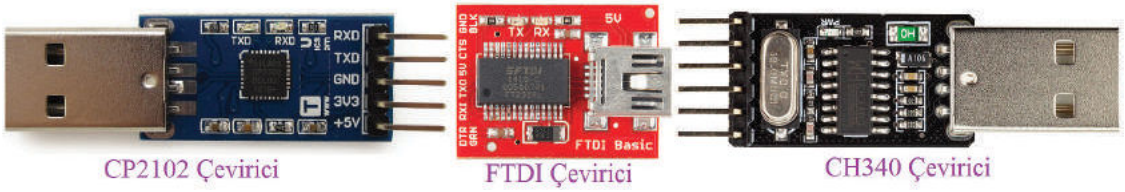

**Resim 5.1:** Motor sürücü kartları

## 5.3. USB-UART Çeviriciler ve Görevleri

Bilgisayar ve ona bağlanabilen her türlü çevresel aygıt seri haberleşme tekniğini (seri iletişim) kullanmaktadır. Bu amaçla bilgisayar ve çevresel aygıtların üzerinde seri iletişim bağlantı noktaları bulunmaktadır. Günümüzde kullanılan seri iletişim bağlantı noktası temelde USB'dir (Universal Serial Bus-Evrensel Seri Veriyolu).

Robotik programlamada kullanılan işlemcilerin, bunların üzerinde bulunduğu mikrodenetleyici kartların ve robotik kontrol kartların bilgisayara bağlanıp programlanabilmesi için de USB bağlantı noktası kullanılmaktadır. USB'nin görevi, bilgisayar ile kontrol kartı (örneğin Arduino) üzerinde yer alan mikrodenetleyici arasında iletişimi sağlamaktır. Bu sayede kartların programlanması ve kontrolü gerçekleşebilmektedir. Fakat bazı mikrodenetleyici kartlarda ve robotik kontrol kartlarında (aynı şekil-

de mikroişlemcilerde) USB bağlantı seçeneği bulunmamaktadır. Yalnızca UART (Universal Asynchronous Receiver/Transmitter - Evrensel Asenkron Alıcı / Verici) bulunmaktadır. Bu durumda bu tür birimlerle iletişim kurulabilmesi için USB-UART çeviricilere ihtiyaç duyulmaktadır. Ancak iletişimin sağlanabilmesi için bilgisayarın kullanılan çevirici ile nasıl haberleşeceğini biliyor olması, başka bir deyişle çeviricinin aygıt sürücülerinin bilgisayarda yüklü olması gerekmektedir. Farklı türlerde USB-UART çeviriciler bulunmaktadır.



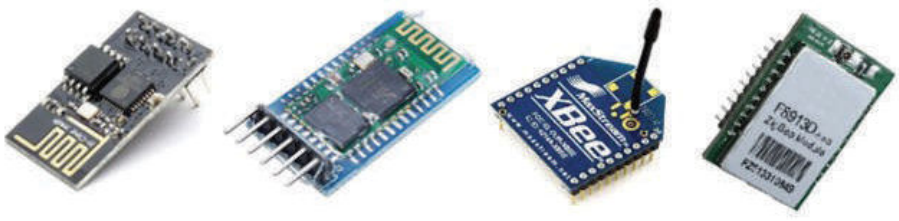

**Resim 5.2:** USB-UART çeviriciler

## 5.4. Kablosuz İletişim Bileşenleri ve Görevleri

Robotun kontrol edileceği, programlanacağı aygıtlara (Bilgisayar, tablet veya akıllı telefon olabilir.) kablosuz olarak bağlanabilmesi için kullanılan haberleşme bileşenlerdir. Genellikle Wi-Fi, Bluetooth, XBee ve ZigBee parçaları bu amaçla tercih edilmektedir. Bu parçalar kullandıkları protokole, haberleşme frekansına, anten tiplerine ve güçlerine göre sınıflandırılmaktadır. Mikrodenetleyici kartların ve robotik kontrol kartların bilgisayara bağlanıp kontrol edilebilmesi için bu teknolojilerin hepsi de kullanılabilmektedir.

Wi-Fi (Wireless Fidelity-Kablosuz Bağlantı Alanı) kişisel bilgisayar, tablet, video oyunu konsolları, dijital ses ve video oynatıcıları ve akıllı telefonlar gibi cihazların kablosuz olarak İnternet'e ve birbirlerine bağlanması için kullanılmaktadır. Wifi teknolojisi ile 4900 Mbps'ye kadar ses ve veri iletimi yapabilmektedir. Wi-Fi destekli cihazların ve parçaların etkin olduğu mesafe, kapalı alanlarda en fazla 300 metre civarındadır.

Bluetooth kişisel bilgisayar, çevre birimleri ve diğer cihazların birbirleri ile kablo bağlantısı olmadan haberleşmelerine olanak sağlayan kısa mesafe radyo frekans (RF) teknolojisidir. Bluetooth teknolojisi ile 24 Mbps'ye kadar ses ve veri iletimi yapabilmektedir. Bluetooth destekli cihazların etkin olduğu mesafe, yaklaşık 10 ile 100 metre arasındadır. Robotik uygulamalarda yaygın olarak kullanılmaktadır.




**Resim 5.3:** Kablosuz iletişim bileşenleri

XBee ve ZigBee düşük maliyetli, düşük güçlü kablosuz kısa mesafe radyo frekans (RF) teknolojisidir. Düşük maliyetli teknoloji olduğu için, kablosuz aygıtların kontrol ve izleme uygulamalarında kulla-

nılmaktadır. Düşük güç kullanımı daha küçük pil ile daha uzun ömür sunmaktadır. XBee ve ZigBee destekli 2. Nesil cihazların etkin olduğu mesafe düşük veri iletişim hızlarında (10-20 kbit/sn) ve yüksek kazançlı antenler kullanılarak 45 km'ye kadar ulaşabilmektedir. Genellikle veri iletim hızı çeşitlerine göre 20 ile 1000 kilobit/saniye arasında değişmektedir. Oldukça küçük yapıda üretilebilmektedir.

## 5.5. Robotik Uygulamalarda Kullanılan Algılayıcılar (Sensörler)

Robot teknolojisinin veya genel anlamda otomasyon sistemlerinin en önemli kısımlarından birisi algılamadır. Algılamayı sağlayan aygıtlara sensör ya da algılayıcı adı verilmektedir. Algılayıcıları bu sistemlerin duyu organları olarak değerlendirebiliriz. Çünkü insanların çevrelerinde olup bitenleri duyu organlarıyla algılamasına benzer biçimde, robotlar ve otomasyon sistemleri de çevresindeki sıcaklık, basınç, hız, yön, eğim ve benzeri değişkenleri algılayıcıları vasıtasıyla algılarlar. Algılama algılananları ölçme ve ölçümleri kontrol aygıtına (mikroişlemci) iletme şeklinde gerçekleşir. Mikroişlemci algılananları yorumlamak ve ona göre karar döngülerini yürütmek zorundadır. Algılanması gereken farklı değişkenler farklı tiplerde algılayıcılar gerektirir. Neyin ya da nelerin algılanacağı kullanılan algılayıcının seçimine bağlıdır. Algılayıcı seçimi robotun görevine uygun olarak yapılır. Örneğin robotun herhangi bir engele çarpmadan dolaşabilmesini istiyorsak bir mesafe ölçüm algılayıcısının kullanılması gerekmektedir. Algılayıcılar ile algılanan çok farklı türde değişken bulunmaktadır. Bu değişkenler şu şekilde özetlenebilir:

- Mekanik Değişkenler: Uzunluk, alan, miktar, kütlesel akış, kuvvet, tork (moment), basınç, hız, ivme, pozisyon, ses dalga boyu ve yoğunluğu gibi değişkenlerin ölçülmesidir.
- Termal Değişkenler: Sıcaklık, ısı akışı gibi değişkenlerin ölçülmesidir.
- Elektriksel Değişkenler: Voltaj, akım, direnç, endüktans, kapasitans, dielektrik katsayısı, polarizasyon, elektrik alanı ve frekans gibi değişkenlerin ölçülmesidir.
- Manyetik Değişkenler: Alan yoğunluğu, akı yoğunluğu, manyetik moment, geçirgenlik gibi değişkenlerin ölçülmesidir.
- Işıma Değişkenleri: Yoğunluk, dalga boyu, polarizasyon, faz, yansıtma, gönderme gibi değişkenlerin ölçülmesidir.
- Kimyasal Değişkenler: Yoğunlaşma, içerik, oksidasyon/redaksiyon, reaksiyon hızı, pH miktarı gibi değişkenlerin ölçülmesidir.

### 5.5.1. Robotik Algılayıcı Türleri

Günümüzde çok çeşitli algılayıcı bulunmaktadır. Bunları birbirinden farklı birçok sınıfa ayırmak mümkündür. Genelde ölçülen büyüklüğe göre, çıkış büyüklüğüne göre, besleme ihtiyacına göre yapılan sınıflandırmalar kullanılmaktadır. Örneğin besleme ihtiyaçlarına göre algılayıcılar, pasif ve aktif algılayıcılar; çalıştıkları sinyallere göre ise dijital ve analog algılayıcılar olarak iki grupta sınıflandırılır.

Özellikle mobil robotlar için kullanılan algılayıcılar işlevlerine göre sınıflandırılmaktadır. Bazı algılayıcılar, bir robotun elektroniğinin iç sıcaklığı veya motorların dönme hızı gibi basit değerleri ölçmek için kullanılır. Bazıları ise robotun çevresi hakkında bilgi edinmek veya robotun küresel konumunu doğrudan ölçmek gibi daha karmaşık değerleri ölçmek için kullanılabilir. Robotik algılayıcılar bu iki önemli fonksiyonel eksende propriyoseptif ve exteroseptif algılayıcılar olarak ayrılmaktadır.

**Propriyoseptif Algılayıcılar:** Robotik sistemin içindeki motor hızı, tekerlek yükü, robot kolu eklem açısı ve akü gerilimi gibi değerleri ölçmek için kullanılan algılayıcılardır.

**Eksteroseptif Algılayıcılar:** Robotun bulunduğu ortamdan bilgi alan algılayıcılardır. Örneğin mesafe ölçümleri, ışık yoğunluğu ve ses dalga genliği ölçümü gibi işlemleri yaparlar. Bu nedenle, eksteroseptif algılayıcı ölçümleri, anlamlı çevre özelliklerini çıkarmak için robot tarafından yorumlanırlar.

**Pasif Algılayıcılar:** Dışarıdan harici hiçbir güç kaynağına ihtiyaç duymadan çevrelerinden aldıkları fiziksel ya da kimyasal sinyalleri ölçen algılayıcılardır. Başka bir deyişle, algılayıcıya giren çevre ortam enerjisini ölçerler. Pasif algılayıcı çeşitlerine en basit örnek ise buton ve anahtardır. Bunlardan farklı olarak potansiyometre, limit anahtarları, ısı, ışık, basınç algılayıcıları, dokunma algılayıcılar, mikrofonlar, CCD veya CMOS kameralar örnek olarak verilebilir. Bu algılayıcıların çalışması için harici hiçbir enerjiye ihtiyaç yoktur. Bu algılayıcılar sadece giriş değişkenlerini ölçerek tepki verirler.

**Aktif Algılayıcılar:** Sinyallerini kendileri üretip çevrelerine yayar ve bu sinyallerin çevreleriyle olan etkileşimlerini ölçen algılayıcılardır. Aktif algılayıcılar sinyallerini kendileri yaydıklarından daha fazla enerjiye ihtiyaç duyarlar. Bu nedenle fiziksel ya da kimyasal değerleri ölçmek için dışarıdan haricî bir güç kaynağı kullanılmaktadır. Bu algılayıcıların en önemli özelliklerinden biri, zayıf sinyalleri oldukça hassas biçimde ölçmek için kullanılabilmeleridir. Kızılötesi algılayıcılar, mesafe algılayıcılar, enkoderler, lazer mesafe bulucular ve ultrasonik uzaklık algılayıcıları aktif algılayıcılara örnek olarak verebiliriz. Aktif algılayıcılar ürettiği sinyal türüne göre analog veya dijital sinyal çıkışı vermektedir.

**Dijital Sinyal Veren Algılayıcılar:** Dijital algılayıcılar ayrık sinyaller üretir. Bu, değerlerin sınırlı sayıda ve kesikli olduğu anlamına gelir. Dijital algılayıcılardan alınan ham bilgiler belli adımlarla yükselen değerlere sahiptirler. Örneğin bir dijital pusula 360 farklı değer üretirken, dijital algılayıcı olan anahtarlar açık ya da kapalı olarak iki değer üretirler.

**Analog Sinyal Veren Algılayıcılar:** Analog algılayıcılar, devre 0 V - 5 V arasında ya da 4 mA - 20 mA arasındaki değerleri algılayacak şekilde çalışırlar ve bu durumda bu iki değer arasındaki tüm değerleri okuyabilirler. Analog sinyal belli iki değer arasında herhangi bir değerdir. Sürekli sinyal ürettikleri için sinyaller arası aralık yoktur. Analog algılayıcılar kullanıldığında bunları mikroişlemcilere yönlendirmeden önce analog/dijital (A/D) çeviriciler kullanılarak analog sinyallerin dijital sinyallere çevrilmeleri gerekir. Çünkü mikroişlemciler dijital sinyallerle çalışırlar.

## 5.5.2. Yaygın Kullanılan Robotik Algılayıcılar ve Görevleri

Burada genel olarak robotik uygulamalarda en fazla kullanılan algılayıcı çeşitleri ve görevleri kısaca açıklanmıştır. Sınıflandırma, algılayıcıların aktif veya pasif olmasına göre yapılmış analog veya dijital sinyal üretmesi gibi özelliklerine değinilmemiştir. Çünkü aynı amaç için kullanılan fakat farklı özellikler taşıyan çok fazla sayıda ve türde algılayıcı bulunduğu gibi hem analog hem de dijital sinyali birden verenleri de bulunmaktadır.

## 5.5.3. Aktif Algılayıcılar

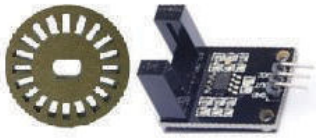
**Çizgi Takip Algılayıcıları (Line Sensors):** Robot uygulamalarında, robotun kalınca çizgilerle çizilen belirli bir alan içerisinde kalması veya çizilen çizgileri izlemesi için kullanılan algılayıcılardır.


**Resim 5.4:** Çizgi takip algılayıcı

**Engel Kaçınma Algılayıcıları (Obstacle Avoidance Sensors):** Robotun bir engele çarpmadan önce onu algılayıp kaçınması için kullanılan algılayıcılardır.

**Resim 5.5:** Engel kaçınma algılayıcı

**Enkoder Algılayıcılar (Encoder Sensors):** Robotik uygulamalarda motorların dönüş yönünü, hızlarını ve tur sayılarını belirlemek için kullanılan, motor kontrol sistemleri için geri bildirim sağlayan algılayıcılardır. Optik ve manyetik yöntemle çalışan çeşitleri bulunmaktadır. Doğrusal ve döner olmak üzere ikiye ayrılırlar.



**Resim 5.6:** Enkoder algılayıcı

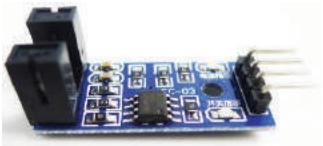
**Hareket Algılayıcılar (PIR Motion Sensors):** İnsan ve hayvanların robot tarafından algılanması için kullanılan algılayıcılardır. PIR (Passive Infrared Sensor) algılayıcılar insanlar veya sıcakkanlı hayvanlar tarafından üretilen kızılötesi ışığı algılarlar. Algılayıcının ön yüzünde ısı ışınlarını IR algılayıcı üzerinde çeşitli noktalara odaklayan çok sayıda fresnel mercekler bulunmaktadır.

**Resim 5.7:** Hareket algılayıcı

**Hareket Kontrol Algılayıcılar (Gesture Sensors):** Robotun elle yapılan hareketlerle kontrol edilebilmesi için kullanılan algılayıcılardır. Bu algılayıcılar, kullanıcıdan yansıyan kızılötesi ışınları tespit ederek basit el hareketlerini robotun tanımasını sağlar.

**Resim 5.8:** Hareket kontrol algılayıcı

**Işık Kesici Algılayıcılar (Photo Interrupter Sensors):** Algılayıcının kolları arasında bulunan kızılötesi ışık demeti arasından bir nesne geçtiğinde ışının kırılması sonucu robotun o nesneyi algılamasını sağlayan algılayıcılardır.



**Resim 5.9:** Işık kesici algılayıcı

**Kızılötesi Termometre Algılayıcılar (Infrared Thermometer Sensors):** Robotun temassız olarak (uzaktan) ortam sıcaklığını algılaması, vücut ısısı ölçümü veya hareket algılaması gibi uygulamaları için kullanılan algılayıcılardır.

**Resim 5.10:** Kızılötesi termometre algılayıcı

**Kızılötesi Yakınlık Algılayıcılar (Infrared Proximity Sensors):** Robotun belirli bir nesneye veya duvara olan mesafesini ölçmek için kullanılan algılayıcılardır. Genellikle 3 ile 150 cm aralığındaki uzunluğu ölçebilmektedir.

**Resim 5.11:** Kızılötesi yakınlık algılayıcı

**Lazer Tarama Algılayıcılar (Laser Scanner Senors):** Robotun engellerden kaçınması, bulunduğu ortamı haritalaması, lokalizasyon, rota planlaması gibi işlemleri yapabilmesi için kullanılan algılayıcılardır. Robot 360° tarama yaparak bulunduğu ortamın 2 veya 3 boyutlu gerçek görüntülerini oluşturmaktadır.

**Resim 5.12:** Lazer tarama algılayıcı

**Mikrodalga Hareket Dedektörü Algılayıcılar (Microwave Motion Detector Sensors):** Robotun mikrodalgalar kullanılarak cansız hareketli nesneleri algılaması, hız ölçmesi için kullanılan algılayıcılardır. Sistemin çalışma mantığı Doppler Efektine dayanır.

**Resim 5.13:** Mikrodalga hareket detektörü algılayıcı

**Optik Algılayıcılar (Optical Detectors):** Bu algılayıcılar robotun yansıyan kızılötesi sinyalleri algılaması için kullanılır. Siyah beyaz renk geçişlerini algılama veya yakındaki cisimleri (0,5-1 cm) tespit etmek için de kullanılmaktadır.

**Resim 5.14:** Optik algılayıcı

**Sonar Mesafe Bulucular (Sonar Range Finders):** Robotun belirli bir nesneye veya duvara olan mesafesini ölçmek için kullanıldıkları gibi algılama bölgesindeki nesneleri tespit etme ve bir nesne (bir kişi gibi) algılama bölgesine girdiğinde rapor vermek için de kullanılan algılayıcılardır. 0 ile 765 cm aralığındaki uzunluğa kadar 2,5 mm hassasiyete ölçme yapabilen, bu mesafeler içerisindeki engelleri algılayabilen çeşitli modelleri bulunmaktadır.

**Resim 5.15:** Sonar mesafe algılayıcı

**Ultrasonik Uzaklık Algılayıcılar (Ultrasonic Distance Sensors):** Robotun belirli bir nesneye veya duvara olan mesafesini ölçmek için kullanılan algılayıcılardır. Genellikle 2 ile 400 cm aralığındaki uzunluğu 3 mm hassasiyete ölçebilmekte, bu mesafeler içerisindeki engelleri algılayabilmektedir.


**Resim 5.16:** Ultrasonik uzaklık algılayıcı

**Yansıtıcı Optik Algılayıcılar (Reflective Optical Sensors):** Robotun siyah beyaz renk değişimini algılaması için kullanılan algılayıcılardır. Genelde çizgi izleyen robotlar için kullanılmaktadır.

**Resim 5.17:** Yansıtıcı optik algılayıcı

**Tampon Algılayıcılar (Bumper Sensors):** Robotun herhangi bir nesneye veya yapıya çarpmadan önce onu algılaması için kullanılan algılayıcılardır. Algılama çarpmadan önce gerçekleşmektedir.

**Resim 5.18:** Tampon algılayıcı

### 5.5.4. Pasif Algılayıcılar

**Açısal Algılayıcılar (Angular Sensors):** Robotun bir bağlantı mekanizmasının açısal değerini veya robota ait bir eklemin açı değerini tespit için tasarlanmış algılayıcılardır.

**Resim 5.19:** Açısal algılayıcı

**Ağırlık Algılayıcılar (Load Sensors):** Robotun ağırlıkları algılayabilmesi, ölçebilmesi için kullanılan algılayıcılardır. Çok çeşitli tür ve ağırlık kapasitelerinde üretilmektedir.


**Resim 5.20:** Ağırlık algılayıcı

**Akım Algılayıcılar (Current Sensors):** Robotun kendi genel güç tüketimlerini ölçmek ve değerlendirmek için kullandığı algılayıcılardır.


**Resim 5.21:** Akım algılayıcı

**Alev Algılayıcılar (Flame Sensors):** Robotun alevi, ateşi uzaktan algılaması için kullanılan algılayıcılardır.

**Resim 5.22:** Alev algılayıcı

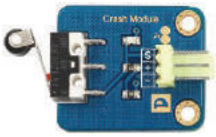
**Basınç / Yükseklik Algılayıcılar (Barometric Pressure /Altitude Sensors):** Robotun barometrik basınç ölçmesi için kullanılan algılayıcılardır. Basınç yükseklik ile değiştiği için aynı zamanda bir altimetre (yüksekliköçer) olarak da kullanılabilmektedir.

**Resim 5.23:** Basınç algılayıcı

**Buhar Algılayıcılar (Steam Sensors):** Robotun ortamdaki nem ve buhar varlığını algılaması için kullanılan algılayıcılardır. Nem ve buhar miktarının ölçümü için kullanılabilmektedir.

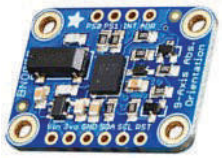
**Resim 5.24:** Buhar algılayıcı

**Çarpma Algılayıcılar (Crash Sensors):** Robotun herhangi bir nesneye veya yapıya çarptığını algılaması için kullanılan algılayıcılardır. Algılama çarptıktan sonra gerçekleşmektedir.

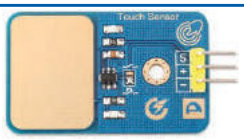


**Resim 5.25:** Çarpma algılayıcı

**Çoklu Algılayıcılar (IMU-Inertial Measurement Unit- Atalet Ölçüm Birimi):** Robotun gerçek dünyadaki konumu, hızı, yüzeyle olan açısı ve yüksekliği gibi bilgileri algılamasını sağlayan entegre algılayıcılardır. 3 eksen jiroskop, 3 eksen ivmeölçer, 3 eksen pusula ve dijital barometre algılayıcılarının birleştirildiği bir mini kart şeklindedir.


**Resim 5.26:** Çoklu algılayıcı

**Dokunma Algılayıcılar (Touch Sensors):** Robotun kendisine dokunulduğunu anlamasını sağlayan algılayıcılar-



**Resim 5.27:** Dokunma algılayıcı

dır. İnsan derisine duyarlıdırlar. Açma/kapama düğmesi kullanmadan bir açma/kapama işlemi yapmak veya robotun insan eliyle dokunmaya duyarlı bir eylem veya hareketi yapması için kullanılmaktadır.

**Eğim Algılayıcılar (Tilt Sensors):** Robotun bulunduğu yerdeki eğimi, eğimin yönünü veya sarsıntıyı tespit edebilmesi için kullanılan algılayıcılardır.

**Resim 5.28:** Eğim algılayıcı

**Esnek Kuvvet, Güç, Basınç Algılayıcılar (Flexiforce Pressure Sensors):** Robotun kuvvet, güç ya da üzerine uygulanan basıncı algılayabilmesi için kullanılan algılayıcılardır. Robot üzerindeki belirli bir alana (kare veya dairesel olabilir) uygulanan, güç ya da basıncın algılanması söz konusudur.

**Resim 5.29:** Esnek algılayıcı

**Gaz Algılayıcılar (Gas Sensors):** Havadaki Karbon Monoksit (CO), Azot dioksit (NO2), Doğalgaz (CNG), Hidrojen (H2), sıvılaştırılmış petrol gazı (LPG), Bütan, Propan, Metan (CH4), Alkol, Amonyak (NH3) ve duman gibi gazlarla, toksik gazları algılamak için kullanılan algılayıcılardır. Hava kalitesini ölçmek için kullanılan çeşitleri de bulunmaktadır.


**Resim 5.30:** Gaz algılayıcı

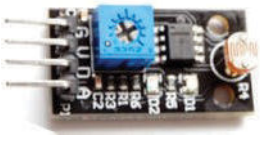
**Görüntü Algılayıcılar (Image Sensors):** Robotun nesneleri tanıması, öğrenmesi ve istenildiğinde bulması için kullanılan görme sistemleridir. Öğretilen nesneleri gördüğünde algılamaktadır. Gerçek zamanlı görüntü işleme görevleri için kullanılmaktadır.

**Resim 5.31:** Görüntü algılayıcı

**GPS Algılayıcılar (GPS Sensors):** Robotun bulunduğu noktayı enlem ve boylam olarak tespit edebilmesi, kendine verilen rota doğrultusunda hareket edebilmesi, gerçek hızı ve yüksekliğini belirleyebilmesi için kullanılan küresel konumlandırma (Global Positioning System -GPS) algılayıcılarıdır.

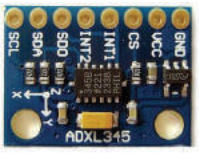

**Resim 5.32:** GPS algılayıcı

**Işık Algılayıcılar (Light Sensors):** Robotun ortamdaki ışık miktarını, yoğunluğunu ölçmesi, buna göre herhangi bir eylem veya hareket yapması için kullanılan algılayıcılardır. Kızılötesi ve normal ışık için kullanılan çeşitleri bulunmaktadır.



**Resim 5.33:** Işık algılayıcı

**İvme Algılayıcılar (Accelerometer Sensors):** İvme ölçmek için kullanılan algılayıcılardır. Robotun eklem hareketlerini, eğilme derecesini ve titreşimleri algılayabilmesini



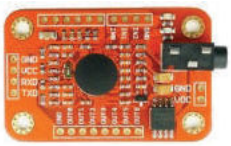

**Resim 5.34:** İvme algılayıcı

sağlayan algılayıcılardır. X, Y ve Z eksenlerinde yapılan temel hareketleri algılamak için bu algılayıcılar kullanılmaktadır. Tek, iki veya üç eksende oluşan ivmeyi ölçebilen çeşitleri bulunmaktadır.

**Jiroskop Algılayıcılar (Gyroscope Sensors):** Robotun yön ölçümü ve ayarlanmasında konumsal ve hareketsel yönünü hesaplamayı sağlayan algılayıcılardır. Bu amaçla X, Y ve Z eksenleri arasındaki açısal oranların ölçümü yapılmaktadır. Jiroskop dış etkenlerden, yer çekiminden ve merkezkaç kuvvetinden etkilenmeyen bir referans etkeni sağlamaktadır.

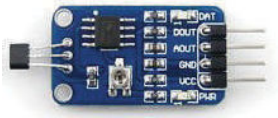
**Resim 5.35:** Jiroskop algılayıcı

**Konuşma, Ses Tanıma Algılayıcıları (Speech, Voice Recognition Sensors):** Robotun sesle verilen emirleri anlayıp uygulayabilmesi için sesi ve konuşmayı tanımasını sağlayan algılayıcılardır. Bu sayede robotla konuşarak iletişim kurmak ve istenileni yaptırmak mümkün hâle gelmektedir.



**Resim 5.36:** Konuşma, ses tanıma algılayıcı

**Manyetik Alan Algılayıcılar (Hall Effect Sensors):** Robotun manyetik malzeme ve ortamları algılamasını sağlayan algılayıcılardır. Robotun manyetik alana duyarlı bir eylem veya hareketi yapması için kullanılmaktadır.



**Resim 5.37:** Manyetik alan algılayıcı

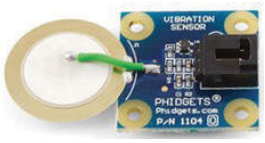
**Nem Algılayıcılar (Humidity Sensors):** Robotun ortamdaki nem miktarını ölçmesi için kullanılan algılayıcılardır.


**Resim 5.38:** Nem algılayıcı

**Parlaklık Algılayıcılar (Luminosity Sensors):** Robotun ışığın parlaklık düzeyini algılaması ve ölçmesi için kullanılan algılayıcılardır.

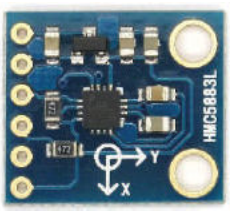

**Resim 5.39:** Parlaklık algılayıcı

**Piezo Titreşim Algılayıcılar (Piezo Vibration Sensors):** Robotun esneme, dokunma, titreşim ve şok ölçümleri yapabilmesi, çarpışmaları algılayabilmesi veya esnek anahtar uygulamaları için kullanılan algılayıcılardır.




**Resim 5.40:** Piezo titreşim algılayıcı

**Pusula, Manyetometre Algılayıcılar (Compass, Magnetometer Sensors):** Dijital yön algılayıcılardır. Dünyanın manyetik alanına ilişkin ölçmeye dayalı yönlendirme ile robotun her zaman otomatik veya programlı olarak istenilen gerçek fiziksel yönde hareket etmesi için kullanılır. Tek, iki veya üç eksen ölçen çeşitleri bulunmaktadır.




**Resim 5.41:** Pusula algılayıcı

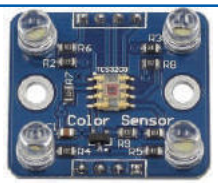
**Ses Algılayıcılar (Sound Sensors):** Robotun sesi algılaması, sese duyarlı bir eylem veya hareketi yapması için kullanılan algılayıcılardır. Bu algılayıcılar sesi tanımlayamaz, anlayamaz, sadece sesi fark eder.

**Resim 5.42:** Ses algılayıcı

**Sıcaklık Algılayıcılar (Temperature Sensors):** Robotun ortam ve çalışma sıcaklığını ölçmesi için kullanılan algılayıcılardır.

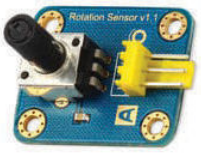
**Resim 5.43:** Sıcaklık algılayıcı

**Renk Algılayıcılar (Color Sensors):** Robotun renkleri algılaması, tanımlaması ve renk ölçümlerini doğru yapabilmesi için kullanılan algılayıcılardır.




**Resim 5.44:** Renk algılayıcı

**Rotasyon Algılayıcılar (Rotation Sensors):** Robotun herhangi bir bileşeninin (kol, ayak, baş, gövde vb.) kaç derece hareket ettiğini mekanik bağlantıyla algılaması için kullanılan algılayıcılardır.



**Resim 5.45:** Rotasyon algılayıcı

**Titreşim Algılayıcılar (Vibration Sensors):** Robotun meydana gelen titreşimleri ve hızlanmayı algılaması için kullanılan algılayıcılardır. Titreşim miktarının veya hızlanmanın ölçümü için kullanılmaz.

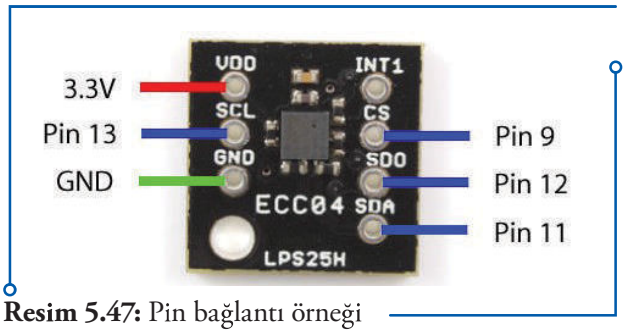
**Resim 5.46:** Titreşim algılayıcı

## 5.5.5. Algılayıcıların Mikrodenetleyici Kartlarla Haberleşmesi/Kartlara Bağlanması

Algılayıcıların mikrodenetleyici kartlarla haberleşmesi ile ilgili birçok standart protokol bulunmaktadırdır. SPI (Seri Çevresel Arayüz - Serial Peripheral Interface) ve I²C (Inter-Integrated Circuit) protokolleri en fazla kullanılan haberleşme protokolleridir. Arduino kartları, diğer Arduino kartlarıyla veya algılayıcılarla (sensörlerle) haberleşmek için bu haberleşme protokollerini kullanmaktadır. Bu protokollerin kullandıkları uç sayıları, ulaşabilecekleri maksimum hızları ve kullanım şekilleri birbirinden farklılık göstermektedir. Pasif algılayıcılar genellikle (+ volt), [- volt (toprak hattı)] ve (S -Sinyal) olmak üzere 3 pin üzerinden Arduino kartlarla haberleşirler. Algılayıcının analog veya dijital çıkış vermesine bağlı olarak Arduino kartının analog veya dijital çıkışlarına bağlanmaları gerekmektedir. Bağlantı algılayıcının voltaj girişlerinin Arduinonun Vcc (+Volt) ve Gnd (-Volt) pinlerine, sinyal çıkışlarının analogsa Arduinonun analog pinlerine (A0, A1, A2, A3 gibi), dijitalse Ardıninonun dijital pinlerine (D2, D3, D4 gibi) yapılması gerekmektedir. Bazı algılayıcılarda hem analog hem de dijital sinyal çıkışı bulunabilmektedir. Bu durumdaki algılayıcıların hangi çıkış tipi kullanılacaksa ona uygun pine bağlantılarının yapılması gerekmektedir.

Aktif algılayıcıların mikrodenetleyici kartlara haberleşmesi ise daha fazla pin kullanımını gerektirebilir. Eğer algılayıcıda I²C protokolu kullanılıyorsa Arduino kartlarla haberleşme için + Volt (Vcc) ve – Volt (toprak hattı) dışında SDA (SerialDAta) ve SCL (SerialCLock) olmak üzere iki bağlantı hattının daha kullanılması gerekmektedir. Bu algılayıcıların haberleşmesi için kullanılan Arduino kart türüne

göre hangi pinlerin SDA ve SCL pini olduğunun bilinmesi ve bağlantıların buna göre yapılması gerekmektedir. Eğer algılayıcılar Arduino kartla haberleşmek için SPI protokolü kullanıyorlarsa, kullanılan Arduino kartın türüne göre hangi pinlerin MISO (Master In Slave Out), MOSI (Master Out Slave In) ve SCLK (Serial Clock) ve SS (Slave Select) pinleri olduğunun bilinmesi ve bağlantılarının buna göre yapılması gerekmektedir. Genellikle algılayıcı ile ilgili dokümanlarda bağlantılar ve kullanılacak pinler aşağıdali örnekte olduğu gibi gösterilmekte veya açıklanmaktadır. Konu ile ilgili ayrıntılar Arduino IDE konusunda verilmektedir.




**Resim 5.47:** Pin bağlantı örneği

## 5.6. Robotik Programlamada Kullanılan İşlemciler

Robotik programlamada kullanılan işlemcilere mikrodenetleyici (Microcontroller) adı verilmektedir. Mikrodenetleyiciler endüstriye ve elektronik sanayisine yönelik olarak kontrol ve otomasyon işlemlerini gerçekleştirmek için tasarlanmış özel mikroişlemciler olarak tanımlanabilmektedir. Bir mikroişlemci sadece işlem ve hafıza birimlerinden oluşurken bu özel mikroişlemciler birçok bileşenden oluşmaktadır.

Mikrodenetleyicilerin içerisinde; aritmetik ve mantıksal işlemler yapan bir mikroişlemci CPU (Central Process Unit), sistemin komutlarının kalıcı olarak tutulduğu ROM (Read Only Memory) bellek, geçici verilerin ve sonuçların tutulduğu RAM (Random Access Memory) bellek bulunmaktadır. Ayrıca seri giriş ve çıkış birimleri I/O (input – output), SPI (Serial Peripheral Interface) ile bilgi alışverişi için kullanılan programlanabilen seri iletişim (UART-USART/Ethernet) ara birimleri, Analogdan Dijitale (A/D) ya da Dijitalden Analoğa (D/A) çeviriciler (konvektör), sayıcılar (counter), PWM sinyal üretici gibi çevre birimlerini de türüne göre yer almaktadır. Bu özellikleri ile değerlendirildiğinde mikrodenetleyici programlanabilme, bir programı içerisinde depolayıp daha sonra çalıştırabilme özelliklerine sahip tek bir işlemciden (tümleşik devre) oluşan bir mikro bilgisayardır. Bu özellikleriyle mikrodenetleyiciler mikroişlemcilerden ayrılmaktadır.

Günümüzde birçok üretici (Intel, Atmel, Michrochip, National Semiconductror, Texas Instruments, vb.) çeşitli tür ve modellerde 8, 16 veya 32 bit mikrodenetleyiciler üretmektedir. Bunlardan en yaygın olanları, Microchip firmasının PIC (Peripheral Interface Controller) ailesini oluşturan PIC10, 12, 16, 17, 18, 24 ve PIC32M model mikrodenetleyiciler, Atmel firmasının AVR ailesini oluşturan tinyAVR, Mega AVR, XMEGA, AVR32 serisi mikrodenetleyiciler, Texas Instruments firmasının MSP430 ailesini oluşturan mikrodenetleyiciler ile ARM tabanlı TI, ST ve ATMEL mikrodenetleyicileridir.


**Resim 5.48:** Robotik programlamada kullanılan işlemciler

### 5.6.1. Robotik Programlamada Kullanılan İşlemcilerin Görevleri

Öncelikli olarak tek başlarına çalışmaları, bütün iş ve işlemleri tek başına yapmaları gerekmektedir. Ayrıca donanımı oluşturan diğer elektronik devrelerle (düşük hızlı çevre birimleri, örneğin algılayıcı) iletişim kurmak mikrodenetleyicinin görevidir. Bunun için gerekli iletişim yapılarını sağlaması, algılayıcılardan gelen analog verileri dijital verilere dönüştürmesi gerekmektedir. Ayrıca algılayıcılardan gelen her türlü verinin toplanması ve işlenmesini yapmak, uygulamanın gerektirdiği bütün fonksiyonları gerçekleştirmek, üzerinde çalışan programda bir sorun olduğu takdirde programın sıfırlanmasını sağlamak yine temel görevlerini oluşturmaktadır.

## 5.7. Mikrodenetleyici Kartlar (Geliştirme Kartları) ve Görevleri

Mekanik, elektromekanik ve elektronik sistemlerin veya bunların bileşeni olan robotların kontrolü için kullanılabilen, üzerinde 8, 16 veya 32 bit mikrodenetleyicilerin bulunduğu, çeşitli fiziksel boyutları olan genelde mini bir kart şeklindeki elektronik platformdur. Her amaca uygun farklı büyüklük ve özelliklerde farklı tür ve modeli olan single-board (bütün üyeleri tek bir kart üzerinde olan sistem) mini bilgisayardır. Kartlara göre farklılık göstermekle beraber kart ile bilgisayar arasındaki bağlantı için genellikle USB iletişim birimi kullanılmaktadır. Dâhilî Wi-Fi veya bluetooth parçası olan çeşitleri de bulunmaktadır.

Geliştirme kartları için farklı programlama ortamları (bilgisayar üzerinde, web üzerinde) ve programlama dilleri bulunmaktadır. Kendine özgü kolay blok veya metin tabanlı programlama dilleri yanında C/C++, Pyhton gibi yüksek seviyeli dillerle de programlanabilmektedir. Hemen hemen bütün işletim sistemleri ile kullanılabilmektedir. Bu geliştirme ortamları kodları derleyip kolayca mikrodenetleyiciye yüklemeyi sağlamaktadır. Geliştirme kartları için oluşturulan kütüphaneler, birçok işlemi donanım seviyesine inmeden mikrodenetleyicinin kaydedicileri üzerinde işlemler yapmaya gerek duymadan yapmayı sağlayacak şekilde oluşturulmuştur. Birçok işlem bu kütüphane fonksiyonları ile yapıldığından kullanıcı daha az kodla ve kolayca programlama yapabilmektedir. Bu tür kartların en büyük özelliğinin kullanım kolaylığı olduğunu belirtebiliriz. Ardunio UNO, Raspberry PI, Beagle Bone robotik uygulamalar için yaygın olarak kullanılan kartlardan bazılarıdır.


**Resim 5.49:** Mikrodenetleyici kartlar (Geliştirme kartları)

## 5.8. Mikrodenetleyici Kartlar (Geliştirme Kartları) İçin Kalkanlar (Shields) ve Görevleri

Mikrodenetleyici kartların özelliklerini geliştirmek, yeni fonksiyon ve özellikler kazandırmak veya kolayca diğer kart yapısındaki bileşenleri eklemek için kullanılan, doğrudan mikrodenetleyici kart üzerine takılabilen (eklenebilen katmanlar) farklı tür ve çeşitlerde kartlardır. Örneğin bluetooth shield

adından da anlaşılabileceği gibi mikrodenetleyici kartla bluetooth kullanarak haberleşmeyi, veri alışverişi yapmayı sağlarken aynı şekilde ethernet shield de mikrodenetleyici kartla ethernet üzerinden haberleşmeyi sağlamaktadır. Bazı kalkanlar katmanlar şeklinde üst üste takılabilmektedir. Kalkanların mikrodenetleyici kart üzerine takılmasında herhangi bir kablolama ihtiyacı bulunmamaktadır. Soket yapıdaki ürünler birbirine geçirilerek kullanılmaktadır.

**Resim 5.50:** Mikrodenetleyici kartlar (Geliştirme kartları) için kalkanlar (Shields)

## 5.9. Robot Kontrol Kartları

Özellikle robotik uygulamalar için geliştirilmiş olup üzerinde bir mikrodenetleyici, motor sürücü, Wi-Fi veya Bluetooth gibi kablosuz iletişim parçası bulunan kartlardır. Bazılarında her üç bileşen bulunabildiği gibi, daha az veya daha çok bileşen bir arada bulunabilir. Bazı çeşitlerde bir robotu programlayarak kontrol etmek için gerekli tüm elektronik donanımlar kart üzerinde yer alabilmektedir. Genellikle giriş çıkış bağlantıları (I/O portları) soketli olarak yapılmıştır. Bu sayede soketli bileşenler soketli birimlere kolayca bağlanabilir. Bunun yanında pin içeren bağlantılar da kullanılabilir. Robot kontrol kartları üzerindeki motor sürücüler ile kullanılacak motorlar doğrudan bağlanabilmektedir. Bu tür kartlar robot yapımı ve kontrolünü oldukça kolaylaştırır.

**Resim 5.51:** Çeşitli robot kontrol kartları

### 5.9.1. Robot Kontrol Kartlarının Görevleri

Robot kontrol kartlarının görevi, robot için gerekli elektronik bileşenlerin tamamını veya önemli bir kısmını karşılayarak robot yapımını kolaylaştırmaktır. Ayrıca kart üzerinde bulunmayan bileşenler için

bağlantı ortamı oluşturmak, çok bileşenli yapıdan daha az bileşenli yapıya geçişi sağlayarak karmaşıklığı azaltmak, kullanımı kolaylaştırmak, fiziksel büyüklükten tasarruf sağlamak gibi görev ve yaraları bulunmaktadır.

## 5.10. Düşünelim / Araştıralım

Robot programlama dersinde kullanmak üzere bir eğitsel robot yapacağınızı düşünerek gerekli olabilecek elektronik bileşenlerin seçimi için İnternet'te araştırma yapınız. Niçin bu bileşenleri seçtiğinizi, bileşenlerin hangi özelliklerinin seçiminizde etkili olduğunu açıklayınız.

## 5.11. Değerlendirme Soruları

1. **Robotlarda kullanılan motorların kontrol edilebilmesi (çalışma, durma, ileri geri hareket etme, hızlanma, yavaşlama vb.) için kullanılan bileşenlere ne ad verilir?**
   a) Robot kontrol kartı
   b) Mikrokontrolör
   c) Motor sürücü
   d) Motor denetleyici
   e) Mikroişlemci

2. **Bilgisayara bağlamak istediğimiz bir çevresel aygıtın (örneğin mikrodenetleyici kartın) üzerinde seri iletişim bağlantı noktası bulunmuyorsa aşağıdaki seçeneklerden hangisinin kullanılması uygundur?**
   a) Paralel bağlantı noktası
   b) Wi-Fi
   c) USB-Seri çevirici
   d) USB-UART çevirici
   e) Bluetooth

3. **Robotun kontrol edileceği, programlanacağı aygıtlara (bilgisayar, tablet veya akıllı telefon) kablosuz olarak bağlanabilmesi için yaygın olarak kullanılan haberleşme bileşeni aşağıdakilerden hangisidir?**
   a) Bluetooth
   b) Wi-Fi
   c) XBee
   d) ZigBee
   e) WiMAX

**4. Robotun bulunduğu ortamdan bilgi alan algılayıcılara ne ad verilir?**

a) Propriyoseptif algılayıcılar
b) Eksteroseptif algılayıcılar
c) Pasif algılayıcılar
d) Aktif algılayıcılar
e) Dijital sinyal veren algılayıcılar

**5. Aşağıdakilerden hangisi dışarıdan haricî hiçbir güç kaynağına ihtiyaç duymadan çevrelerinden aldıkları fiziksel ya da kimyasal sinyalleri ölçen algılayıcılardır?**

a) Propriyoseptif algılayıcılar
b) Eksteroseptif algılayıcılar
c) Pasif algılayıcılar
d) Aktif algılayıcılar
e) Analog sinyal veren algılayıcılar

**6. Aşağıdaki algılayıcıların hangisi mikrodenetleyici kartların haberleşmesi için kullanılan standart protokollerden biridir?**

a) USB
b) UART
c) Seri
d) Paralel
e) $I^2C$

**7. Mikrodenetleyicilerde aşağıdaki bileşenlerden hangisi yer almaz?**

a) I/O (İnput – Output-Seri Giriş ve Çıkış Birimleri)
b) GUI (Grafiksel Kulanıcı Arayüzü)
c) ROM (Read Only Memory-Sadece Okunabilir Bellek)
d) RAM (Random Access Memory-Rastgele Erişimli Bellek)
e) SPI (Serial Peripheral Interface-Seri Çevresel Arayüz)

**8. Mekanik, elektromekanik ve elektronik sistemlerin veya bunların bileşeni olan robotların kontrolü için kullanılabilen, üzerinde 8, 16 veya 32 bit mikrodenetleyicilerin bulunduğu çeşitli fiziksel boyutlardaki temelde mini bir kart şeklinde elektronik platformlara ne ad verilir?**

a) Mikrobilgisayar kartı
b) Mikroişlemci kartı
c) Mikroprogramlayıcı kart
d) Mikrodenetleyici kart
e) Geliştirme kiti

**9. Robotikte kullanılan kartların özelliklerini geliştirmek, yeni fonksiyon ve özellikler kazandırmak veya kolayca diğer kart yapıdaki bileşenleri eklemek için kullanılan ve doğrudan mevcut kartın üzerine takılabilen kartlara ne ad verilir?**

a) Mikrobilgisayar kartı
b) Mikroişlemci kartı
c) Mikroprogramlayıcı kart
d) Mikrodenetleyici kart
e) Kalkan (Shields) kart

**10. Robot için gerekli elektronik bileşenlerin tamamını veya önemli bir kısmını karşılayarak robot yapımını kolaylaştıran kart türü aşağıdakilerden hangisidir?**

a) Robot kontrol kartı
b) Mikrobilgisayar kartı
c) Mikroprogramlayıcı kart
d) Mikrodenetleyici kart
e) Kalkan (Shields) kart

# 6. BLOK TABANLI ROBOT PROGRAMLAMA YAZILIMLARI VE ORTAMLARI

Bu bölümün sonunda,

- ✓ Blok tabanlı ortamların ve yazılımların temel yapısını tanımlayabilecek,
- ✓ mBlock robot programlama ortam ve yazılımının kullanım özelliklerini açıklayabilecek,
- ✓ mBlock'ta programlama yapılan bilgisayarla robot ve geliştirme kartları arasında bağlantı oluşturulabilecek,
- ✓ mBlock'ta programlama yapılan bilgisayarla robot ve geliştirme kartları arasında bağlantı ayarlarını yapabilecek,
- ✓ Genel programlama yapılarının çalışma mantığını özetleyebilecek,
- ✓ Harekete yönelik yapıların çalışma mantığını tanımlayabilecek,
- ✓ Harekete yönelik yapıları programlayabilecek,
- ✓ Görünüme yönelik yapıların çalışma mantığını açıklayabilecek,
- ✓ Görünüme yönelik yapıları programlayabilecek,
- ✓ Sese yönelik yapıların çalışma mantığını tanımlayabilecek,
- ✓ Sese yönelik yapıları programlayabilecek,
- ✓ Veriye yönelik yapıları, geliştirdiği program için uygun şekilde kullanabilecek,
- ✓ Olaylara yönelik yapıları, geliştirdiği program için uygun şekilde kullanabilecek,
- ✓ Kontrol yapılarını, geliştirdiği programa uygun şekilde kullanabilecek,
- ✓ Algılama yapılarının çalışma mantığını açıklayabilecek,
- ✓ Algılama yapılarını programlayabilecek,
- ✓ İşlem yapılarını, geliştirdiği program için uygun şekilde kullanabilecek,
- ✓ Robota özgü yapıları listeleyebilecek,
- ✓ Robota özgü yapıların çalışma mantığını açıklayabilecek,
- ✓ mBlock robot programlama ortam ve yazılımında geliştirilen programları yorumlayabilecek,
- ✓ mBlock robot programlama ortam ve yazılımında geliştirilen programları yeniden düzenleyebilecek,
- ✓ mBlock robot programlama ortam ve yazılımında program geliştirilebileceksiniz.

## 6.1. Blok Tabanlı Robot Programlama Yazılımları ve Ortamları

Robot programlama için oluşturulmuş metin tabanlı programlama yazılımlarına göre daha kullanıcı dostu, öğrenici ve hobi kullanıcılarına hitap eden diller ve ortamlar da bulunmaktadır. Birçoğu ücretsiz olan bu araçlarla hiçbir kod kullanmadan, sürükle bırak veya yapboz oynar gibi programlar oluşturmak olanaklı hale gelmiştir. Bu tür ortamlara blok programlama ortamları adı verilmektedir. Lego NXT-G, EV3, Enchanting, Robo Pro, Modkit, miniBlog, Ardublock, Snap4Arduino, Srach for Arduino (S4A) ve mBlock bu ortamlara örnek olarak verilebilir. Burada blok tabanlı programlama yazılımı ve ortamı olarak mBlock tercih edilmiştir.

## 6.2. mBlock Grafik Programlama Yazılımı ve Ortamı

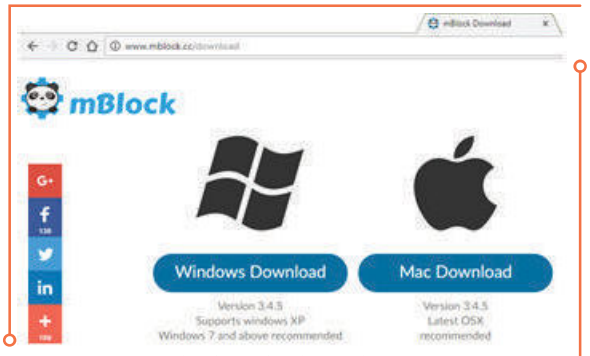
mBlock Makeblock tarafından geliştirilen Arduino projelerinde programlamayı ve etkileşimli uygulamalar oluşturmayı kolaylaştıran grafik ara yüzlü görsel programlama yazılımı ve ortamıdır. Fiziksel dünya ile etkileşim içinde interaktif uygulamalar (oyun, hikâye, animasyon) ve kablosuz olarak programlanabilen robotlar oluşturmak için modüler ve geliştirilebilir şekilde tasarlanmıştır. Gerçek zamanlı kod üretme desteğine sahiptir. Scratch tarzındaki (sürükle-bırak) açık kaynak kodlu bu kod yazma programı Makeblock tarafından geliştirilen mBot eğitim robotlarının programlanmasında kullanıldığı gibi Arduino temelli robotların programlanmasında da kullanılabilmektedir. Bu amaçla Arduino Board Standartlarını desteklemektedir. Arduino UNO, Leonardo, Nano, Mega128, Mega 2560, PicoBoard, Makeblock mCore ve Arduino uyumlu diğer kartlarla kullanılabilmektedir.

Bu görsel programlama ortamı, açık haberleşme protokolleri ve kaynak kodları kullanmaktadır. Windows ve MAC uyumlu güncel sürümü 20'den fazla dili desteklemektedir. Ücretsizdir ve kaynak kodları açıktır. Herhangi bir yardımcı ek uygulama olmaksızın kullanılabilmektedir. Kablosuz haberleşme protokolleri de desteklendiği için daha esnek kullanım olanağı sunmaktadır.

mBlock ortamında hazırlanan örnek uygulamalar dersin sitesinde resim numarası ile yer almaktadır. Bilgisayarınıza kopyalayarak farklı parametrelerle test ediniz.

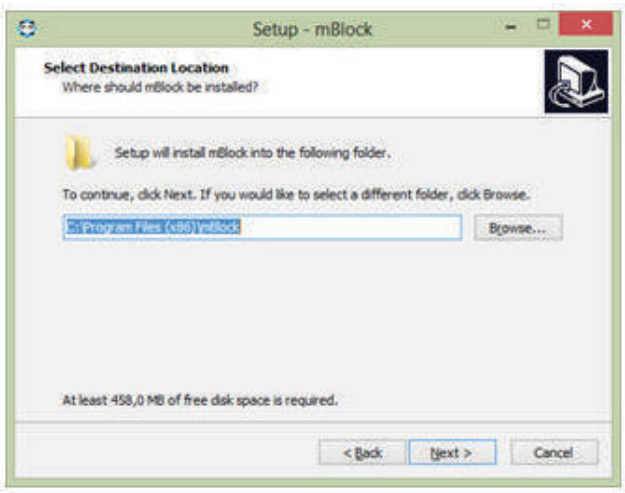
## 6.3. mBlock Grafik Programlama Yazılımının Yüklenmesi

mBlock yazılımının son sürümü http://www.mblock.cc/download sitesinden, kullanılacak işletim sistemi (Windows veya Mac) seçilerek ücretsiz olarak indirilebilmektedir. Hem Windows hem de Mac işletim sistemlerinin eski ve güncel sürümlerinde çalışabilmektedir. Seçimden sonra doğrudan exe dosyası olarak indirilmektedir. İndirildikten sonra mBlock_win_V3.4.5.exe'ye tıklayarak programın kurulması yeterlidir. Kurulum işlemi tamamlandıktan sonra program kullanılmaya hazırdır. Önemli kurulum aşamaları aşağıda verilmiştir.



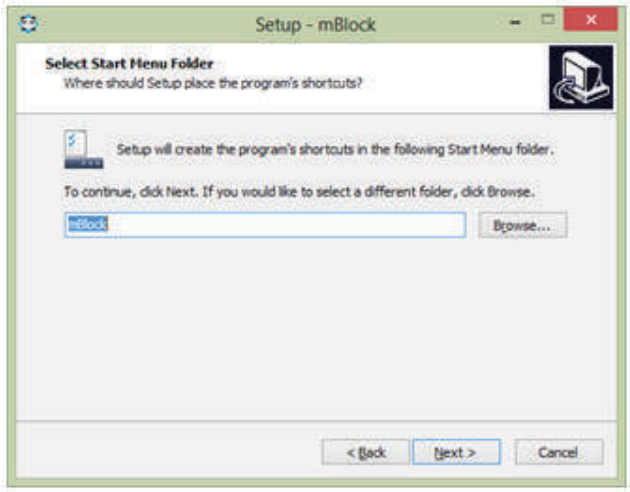

**Resim 6.1:** mBlock indirilme ekranı

Dil seçimi yapıldıktan sonra ilk aşamada lisans sözleşmesi onaylanmalıdır. Daha sonra üçüncü aşamada kurulum klasörü seçilmelidir. İstenirse olduğu gibi bırakılabilir.



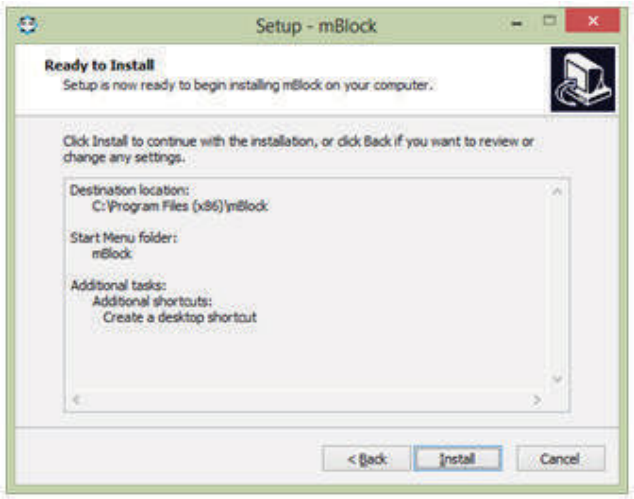

**Resim 6.2:** mBlock kurulum klasörü seçimi

Kurulumun bu aşamasında programın başlangıç menüsü kısayolu için klasör seçimi yapılabilir. İstenirse olduğu gibi bırakılabilir.




**Resim 6.3:** mBlock başlangıç klasörü seçimi

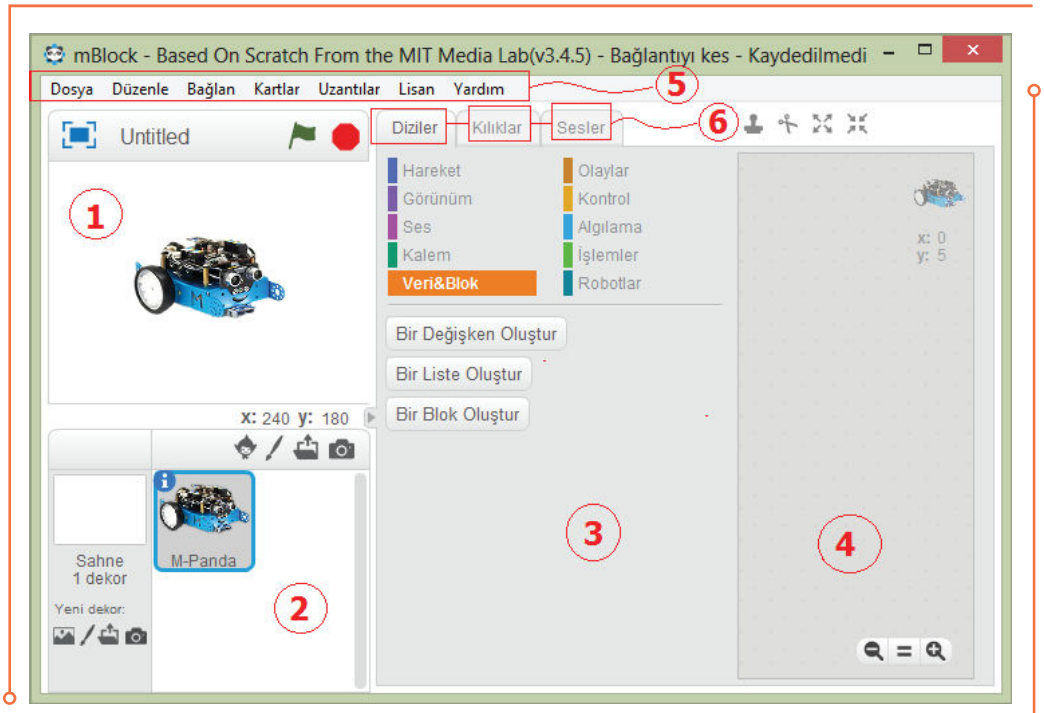
Kurulum yapılacak başlangıç menüsü klasörü seçildikten sonra programın kurulumu install seçeneğinin tıklanmasıyla başlar. Bittiği zaman programın masaüstündeki kısa yoluna tıklanarak program çalıştırılır.




**Resim 6.4:** mBlock kurulum ekranı

## 6.4. mBlock Programının Arayüzü, Yapısı ve Temel Özellikleri

Program çalıştırıldığı zaman karşımıza aşağıdaki arayüz çıkmaktadır. Arayüz üzerindeki yapılar ve görevleri aşağıda açıklanmıştır.




**Resim 6.5:** mBlock programının arayüzü ve yapısı

Programlama ortamı menü yapısında (5 numaralı alan) sunulmakta, 4 temel bölümden oluşmaktadır (Görsel 6.65). 1 numaralı bölümde seçilen dekor içerisinde seçilen figür (kukla) yer almaktadır. 2

numaralı bölümde sahne (dekor) ve figür (kılık, kukla) seçimi yapılmaktadır. 3 numaralı bölüm programlama ortamını oluşturan blok yapılara ayrılmıştır. Programın yazıldığı alan ise 4 numara ile belirtilmiştir. mBlock programlama ortamı (6 numaralı alan) diziler (blok kategorileri), sahneler (dekor), figürler (kılık, kukla) ve seslerden oluşan bir yapı içerisinde sunulmaktadır. Bu ortamı kullanarak etkileşimli hikâyelerle, oyunlarla, animasyonlarla ve robotlarla eğlenceli ve kolay şekilde programlama yapmak veya yapmayı öğretmek mümkündür. mBlock'ta robot programlamayı öğrenmek veya öğretmek için kullanabilecek yapılar "Diziler" başlığı altında; "Hareket", "Görünüm", "Ses", "Kalem", "Veri&Blok", "Olaylar", "Kontrol", "Algılama", "İşlemler" ve "Robotlar" olmak üzere toplam 10 kategoride toplanmıştır. Bunların kullanımıyla programlar oluşturulmaktadır. Seksenden fazla sahne, yüzden fazla kılık ve ses; çeşitli kategori, tema veya türlere göre ortamın kütüphanesinden seçilebilmekte, istenildiğinde bilgisayarda bulunan veya oluşturulan dekor, kukla ve sesler kullanılabilmektedir. İstenirse programın sunduğu dekor, kukla ve ses aracıyla da oluşturulabilmektedir.

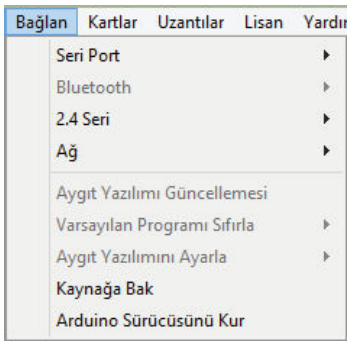
Dosya menüsü yeni proje açmak, proje yüklemek, proje kaydetmek, projeyi farklı kaydetmek gibi temel dosya işlemlerine ayrılmıştır. Bu menü kullanılarak oluşturulan resimler bilgisayara veya bilgisayardan programa aktarılabilmektedir.


**Resim 6.6:** mBlock dosya menüsü

Düzenle menüsü ile silinen sahnelerin geri alınması, sahnenin gizlenmesi, küçük sahne seçimi, jet hızında ve Arduino kipinde çalışma gibi seçenekler yer almaktadır.

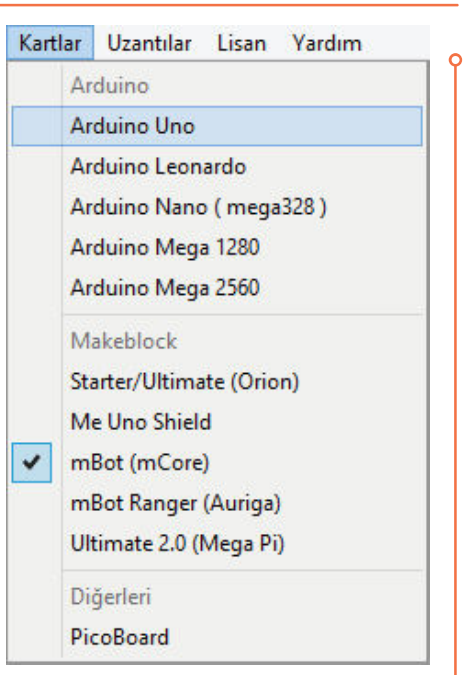

**Resim 6.7:** mBlock düzenle menüsü

Bağlan menüsü robotun mBlock programlama ortamına bağlanması için kullanılabilecek seçenekleri barındırmaktadır. Seri port (USB portu üzerinden), bluetooth, 2.4 seri (USB Dongle üzerinden) ve ağ üzerinden bağlantı seçenekleri sunulmaktadır. Makeblock tarafından sunulan robotlar için aygıt yazılım güncellemeleri, varsayılan programın sıfırlanması, aygıt yazılımının ayarlanması, aygıt yazılımının kaynağının görüntülenmesi ile Arduino sürücülerin kurulması yine bu menüden yapılmaktadır. Kurulum sırasında mevcut Arduino modellerinin sürücüleri standart olarak kurulmaktadır.



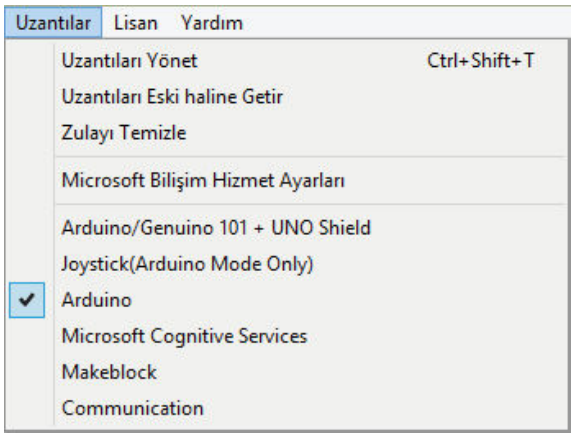

**Resim 6.8:** mBlock bağlan menüsü

Kartlar menüsü ile kullanılan Arduino kart modelleri, Arduino uyumlu kartlar ve Makeblock tarafından üretilen robot kontrol kartları içerisinden seçim yapılmaktadır. Robotik kontrol kartı hangi model Arduino içeriyorsa o kart modeli seçilmelidir. Arduino uyumlu kartlar için Arduino, diğer karlar için diğerleri seçilebilir. Burada Scratch ile programlanabilen sensör kartı PicoBoard desteği de yer almaktadır.



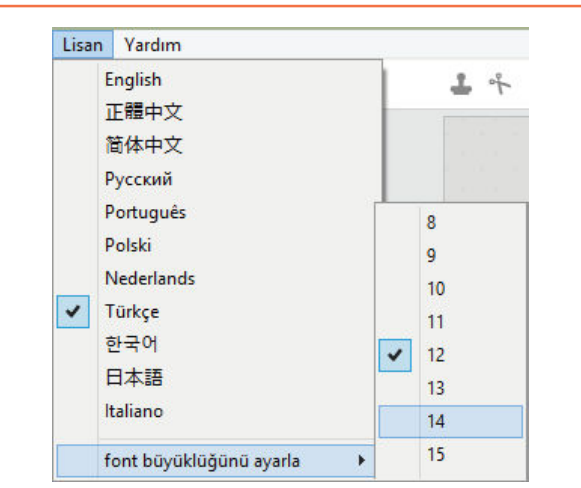

**Resim 6.9:** mBlock kartlar menüsü

Uzantılar menüsü ile kullanılacak kartın türü seçilmektedir. Seçilen türe özgü kod blokları dizeler sekmesinde kullanılabilir hâle gelmektedir. Örneğin Arduino seçildiğinde bunun için oluşturulmuş kod blokları dizeler sekmesine eklenecektir. Birden fazla kart türü seçilebilmektedir.




**Resim 6.10:** mBlock uzantılar menüsü

Lisan menüsü ile istenilen dil ve font büyüklüğü seçilebilmektedir. mBlock şu anda yirmi dilde kullanılabilmektedir.




**Resim 6.11:** mBlock lisan menüsü

Yardım menüsünde yardım konularının yanında güncelleştirmelerin kullanımı ve üretici firmanın robotik ürünlerinin tanıtıldığı site ile örnek uygulamaların bulunduğu siteye yönlendirme yapılmaktadır.

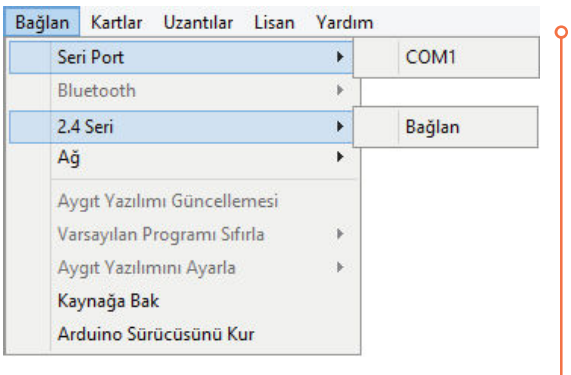

**Resim 6.12:** mBlock yardım menüsü

## 6.5. mBlock Yüklü Bilgisayarla Robot Arasında Bağlantının Oluşturulması

mBlock ile robotun programlanması ve kontrol edilmesi için kablolu veya kablosuz olarak gerçekleştirilebilecek dört farklı seçenek sunulmaktadır. Bunlar; USB port ve Ağ üzerinden kablolu bağlantı, Bluetooth modülü ile ve 2.4 Ghz Dongle modülü üzerinden kablosuz bağlantı seçenekleridir. Kullanılacak robot hangi bağlantı türünü destekliyorsa o seçilmelidir. Kablolu bağlantılar için USB, kablosuz bağlantılar için Bluetooth en çok tercih edilen bağlantı şekilleridir.

USB bağlantı Arduino temelli robotlara program yüklemek için gereklidir. Ayrıca Makeblock tarafından üretilen robotların programlanması, güncellemesi için yine USB bağlantı türü kullanılmaktadır. Bağlantı gerçekleştirildiğinde blokların üzerinde yer alan kırmızı noktanın rengi yeşile dönmektedir.

Bağlan menüsü (Görsel 6.73) robotun mBlock programlama ortamına bağlanması için kullanılabilecek seçenekleri barındırmaktadır. Seri port seçeneği robotun USB kablosu ile bilgisayara bağlanması için kullanılmaktadır. Uygun olan COM (iletişim kapısı) listeden seçilmelidir. Bluetooth seçeneği ile Bluetooth üzerinden, 2.4 seçeneği ile USB Dongle (Mbot robot ile sunulmaktadır) üzerinden kablosuz bağlantı, ağ seçeneği ile de ağ üzerinden bağlantı seçenekleri sunulmaktadır. Makeblock tarafından sunulan robotlar için aygıt yazılım güncellemeleri, varsayılan programın sıfırlanması, aygıt yazılımının ayarlanması, aygıt yazılım kaynağının görüntülenmesi ile Arduino sürücülerin kurulumu yapılmaktadır. Kurulum sırasında mevcut Arduino modellerinin sürücüleri standart olarak kurulmaktadır.




**Resim 6.13:** mBlock ile robot arasındaki bağlantının oluşturulması

## 6.6. mBlock Programlama Dilinin Temel Özellikleri ve Programlama Yapısı

Blok tabanlı programlama dilleri ve ortamlarında komutu oluşturan bloklar, bulunduğu alandan programlama alanına sürükle bırak tekniği ile aktarılır. Bütün bloklar kullanıma açık olmalarına karşın robotik uygulamalarda kullanılmayan bloklar vardır. Bunlar kullanılmak istendiğinde herhangi bir işlev göstermeyecektir.

mBlock'ta robot programlama yapmak ve robotları kontrol etmek için kullanılabilecek komut blokları "Diziler" başlığı altında; "Hareket", "Görünüm", "Ses", "Kalem", "Veri&Blok", "Olaylar", "Kontrol", "Algılama", "İşlemler" ve "Robotlar" olmak üzere toplam on kategoride toplanmıştır. Burada yer alan blok yapıları fiziksel bir robotun programlanması ve kullanımının sağlanması dışında sanal bir robotun (kukla, figür kullanılarak) programlanması ve kullanımını da içermektedir.

### 6.6.1. Hareket Alt Başlığı Altında Verilen Komut Blokları

"Hareket" alt başlığı altında verilen komut blokları sanal bir robotun (kütüphaneden kukla, figür kullanılarak veya oluşturularak) hareket kullanımı ve diğer uygulamalar için kullanılmaktadır. Buradaki blokların kullanılmasıyla sanal robotun istenilen yöne döndürülmesi, belirli bir koordinata gitmesi, belirli bir zaman aralığında belirli bir koordinata yavaşça ilerlemesi (süzülebilmesi) sağlanabilir. Ayrıca sanal robotun kenara geldiği zaman sekmesi de buradan gerçekleştirilebilir. Sanal robotun hangi yönlere dönmesine izin verilip verilmeyeceği de belirlenebilir. "Hareket" alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

| | |
|---|---|
| 10 adım git | Belirtilen adım kadar hareket etmesi için kullanılır. |
| 15 derece dön | Belirtilen açıda soldan sağa dönmesi için kullanılır. |
| 15 derece dön | Belirtilen açıda sağdan sola dönmesi için kullanılır. |
| 90 yönüne dön | 89 Derce sağa, -90 derece sola, 0 derce yukarı ve 180 derce aşağı dönmesi için kullanılır. |
| 'ye doğru dön | Kuklanın fare okuna, Kuklaya, rasgele yatay, dikey ve sahne noktasına dönmesi için kullanılır. |
| x: 0 y: 5 noktasına git | Belirtilen x ve y noktasına gitmesi için kullanılır. |
| fare oku 'na git | Kuklanın fare okuna, Kuklaya, rasgele yatay, dikey ve sahne noktasına gitmesi için kullanılır. |
| 1 sn.de x: 0 y: 5 a süzül | Belirtilen süre içerisinde, belirtilen x ve y noktalarına süzülmesi için kullanılır. |
| x'i 10 arttır | x'i belirtilen değer kadar artırmak için kullanılır. |
| x, 0 olsun | x'i belirtilen nokta yapmak için kullanılır. |
| y 'yi 10 arttır | y'i belirtilen değer kadar artırmak için kullanılır. |
| y, 0 olsun | y'i belirtilen nokta yapmak için kullanılır. |
| kenara geldiysen sek | Kenara geldiyse sekmesi için kullanılır. |
| kuklanın şekli sağa-sola dönebilsin | Sağa sola dönebilmesi, hiç dönememesi veya her yöne dönebilmesi için kullanılır. |
| x konumu | x konumunun ekranda gösterimi için kullanılır. |
| y konumu | y konumunun ekranda gösterimi için kullanılır. |
| yönü | Yönünün ekranda gösterimi için kullanılır. |

**Tablo 6.1:** Hareket alt başlığı altında verilen komut blokları

**Hareket Örneği:** Bu örnekte sanal robot, ortamdaki ses seviyesi 10 birimden yüksek olunca 250 adım gitmekte; 1 sn bekledikten sonra ekrana "Merhaba!" yazısı 2 sn boyunca çıkmaktadır. 1 sn daha bekledikten sonra geldiği noktaya geri dönmektedir.


**Resim 6.14:** Hareket örneği

Yandaki diğer örnekte ise basit bir animasyon çalışması görülmektedir. Kullanılan mBot kuklası 1 ile 20 arasında adımla hareket etmekte, 15 derece dönmekte ve eğer kenara geldiyse sekmektedir. Her işlemden sonra bir sonraki kılığa geçerek animasyon oluşturulmaktadır. Her hareketin ardışık olarak gerçekleşmesi için birbirini izleyen hareket görüntülerinden oluşan kukla resimleri kullanılmalıdır. Animasyon sırasında "computer beeps2" sesini çıkarmaktadır.


**Resim 6.15:** mBlock hareket örneği

## 6.6.2. Görünüm Alt Başlığı Altında Verilen Komut Blokları

"Görünüm" alt başlığı altında verilen bloklar sanal bir robotun (kütüphaneden kukla, figür kullanılarak veya oluşturarak) görünüm kullanımı ve diğer uygulamalar için kullanılmaktadır. Buradaki blokların kullanılmasıyla sanal robotların veya sahnelerin görünümünde değişiklik yapılabilir. Belirli bir süre ya

da sürekli olarak bir mesaj verilebilir, bir düşünme cümlesi belirtilebilir. İhtiyaca göre istenilen sanal robotun sahnede görünüp gizlenmesi sağlanabilir. Sahnede birden fazla dekor (arka plan, hareketsiz nesneler) ve birden fazla sanal robot bulunabilir. Bunlar arasında geçişler yapılabilir. Sanal robotun veya dekorun renk etkisi (0-200 arası) değiştirilebilir. Sanal robotun büyüklüğünde değişiklik yapılabilir ya da yapılacak uygulamalara göre belirli bir büyüklük seçilebilir. Sahnede bulunacak nesnelerin istenilen oranla önde ya da arkada görünmesi sağlanabilir. Görünüm alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

| | |
|---|---|
| Merhaba ! de 2 saniye | Belirtilen süre kadar, belirtilen ifadeyi konuşma balonu içerisinde yazdırmak için kullanılır. |
| Merhaba ! de | Belirtilen ifadeyi konuşma balonu içerisinde yazdırmak için kullanılır. |
| Eee... diye düşün 2 saniye | Belirtilen süre kadar, belirtilen ifadeyi düşünme balonu içerisinde yazdırmak için kullanılır. |
| Eee... diye düşün | Belirtilen ifadeyi düşünme balonu içerisinde yazdırmak için kullanılır. |
| görün | Gizlenmiş (ekranda görünmeyen) kuklanın görünmesi için kullanılır. |
| gizlen | Kuklanın gizlenmesi (ekranda görünmmemesi) için kullanılır. |
| Panda-b kılığına geç | Kuklanın seçilen kılığa geçmesi için kullanılır. |
| sonraki kılık | Kuklanın sonraki kılığa geçmesi için kullanılır. |
| dekor1 dekoruna geç | Dekorun program içerisinde değiştirilmesi için kullanılır. |
| renk etkisini 25 arttır | Belirtilen efekt etkisini belirtilen oranda artırmak için kullanılır. |
| renk etkisi 0 olsun | Belirtilen efekt etkisini belirtilen değere getirmek için kullanılır. |
| görsel etkileri temizle | Görsel etkileri temizlemek için kullanılır. |
| 10 birim büyüt | Kuklayı belirtilen birim kadar büyütmek için kullanılır. |
| büyüklüğü % 100 yap | Kuklanın büvüklüğünü belirtilen yüzdelik değer kadar ayarlamak için kullanılır. |
| üste çık | Kuklanın üst katmana çıkması için kullanılır. |
| 1 katman alta in | Kuklanın belirtilen katman alta inmesi için kullanılır. |

**Tablo 6.2:** Görünüm alt başlığı altında verilen komut blokları

**Görünüm Örneği:** Bu küçük örnekte sanal robot ortamdaki ses şiddeti 10 birimden büyük olunca, 250 adım gidip 1 sn sonra merhaba demekte ve görüntüsü 25 birim büyümektedir. 5 sn bekledikten sonra büyüklüğü %50 oranında değiştirilmektedir.


**Resim 6.16:** Görünüm örneği

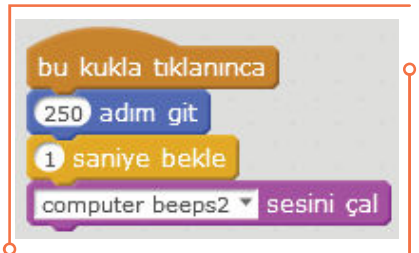
### 6.6.3. Ses Alt Başlığı Altında Verilen Komut Blokları

"Ses" alt başlığı altında verilen bloklar sanal bir robotun (kütüphaneden kukla, figür kullanılarak veya oluşturarak) sesle kullanımı, ses şiddetine dayalı ve sesle ilgili diğer robotik uygulamalar için kullanılmaktadır. Bilgisayarda bulunan mikrofondan elde edilen sesler de kullanılabilir. Buradaki blokların kullanılmasıyla sahnelere veya sanal robotlara ses eklenebilir. Sanal robot bir hedefe ulaştığında ses çıkarması ya da uygulama devam ettiği sürece arka fon müziği bulunması sağlanabilir. Uygulamaya istenirse ses kaydı ya da var olan bir ses dosyası eklenebilir. Ayrıca mBlock içinde bulunan çalgı aletleri de uygulamalara eklenebilir. Sesin temposuyla ilgili düzenlemeler yapılabilir. "Ses" alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

| | |
|---|---|
| patlat sesini çal | Belirtilen sesi çalmak için kullanılır. |
| patlat sesini bitene kadar çal | Belirtilen ses bitene kadar çalmak için kullanılır. |
| tüm sesleri durdur | Tüm sesleri durdurmak için kullanılır. |
| 8 davulunu 0.25 vuruş çal | Belirtilen ses kaynağını (18 adet) belirtilen vuruş oranı kadar çalmak için kullanılır. |
| 0.25 vuruş sus | Belirtilen vuruş oranı kadar susması için kullanılır. |
| 60 notasını 0.5 süresince çal | Belirtilen notayı belirtilen süre boyunca çalmak için kullanılır. |
| çalgıyı 1 yap | Müzik enstrumanını (21 adet) değiştirmek için kullanılır. |
| sesi -10 birim yükselt | Sesi düzeyini belirtilen birim kadar değiştirmek için kullanılır. |
| ses şiddeti % 100 olsun | Ses şiddetini belirtilen oran kadar değiştirmek için kullanılır. |
| ☐ ses şiddeti | Ses şiddetin ekranda göstermek için kullanılır. |
| tempoyu 20 arttır | Tempoyu belirtilen oranda değiştirmek için kullanılır. |
| tempo 60 vuruş/dk olsun | Tempoyu belirtilen vuruş/dakika oranına ayarlamak için kullanılır. |
| ☐ tempo | Tempoyu ekranda göstermek için kullanılır. |

**Tablo 6.3:** Ses alt başlığı altında verilen komut blokları

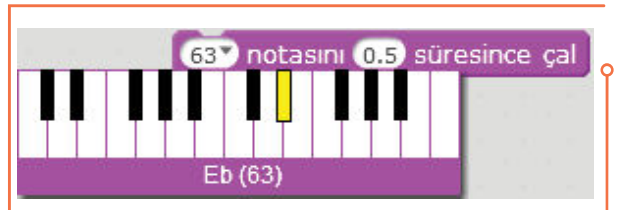
**Ses Örneği:** Bu basit örnekte sanal robot görüntüsüne tıklanınca 250 adım ilerlemekte ve 1 sn bekledikten sonra "computer beeps2" sesini çalmaktadır. Ses kaynağı ses kütüphanesinden örnek olarak seçilmiştir.




**Resim 6.17:** Ses örneği

Aşağıdaki diğer örnekte ise belirlenen üç ayrı tuşa basılmasıyla piyanodan seçilen üç ayrı nota çalınmaktadır. Diğer notalar da aynı şekilde eklenerek bir piyano yapılabilir. Hangi tuş için hangi sesin seçileceği aşağıdaki blokun nota numarasına tıklanarak belirlenebilir.


**Resim 6.18:** 2. Ses örneği




**Resim 6.19:** 2. Ses örneği için nota

Örnekte piyanonun farklı müzik aletleri ile kullanılabilmesi için "müzik aletini seç" adında bir değişken oluşturulmuştur. Değişkenin nasıl oluşturulacağı "Veri&Blok" alt başlığı altında verilen komut blokları konusunda açıklanmıştır.


**Resim 6.20:** 2. Ses örneği için değişken oluşturulması

"müzik aletini seç" değişkeni ile farklı müzik aletlerinin seçimi sürgünün kaydırılması ile yapılmaktadır. Seçim için değişken sağ tıklanarak "sürgü" görünümü tercih edilmiştir.


**Resim 6.21:** 2. Ses örneği için müzik aleti seçimi

### 6.6.4. Kalem Alt Başlığı Altında Verilen Komut Blokları

"Kalem" alt başlığı altında verilen bloklar, sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robotun kalem kullanımı, çizim ve grafik uygulamaları ile diğer uygulamaları için kullanılmaktadır. Buradaki blokların kullanılmasıyla geometrik çizimler yapılabilir. Kalemin rengi, tonu ve kalınlığı değiştirilebilir. Ayrıca sanal robotun olduğu yerde izini bırakması sağlanabilir. "Kalem" alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

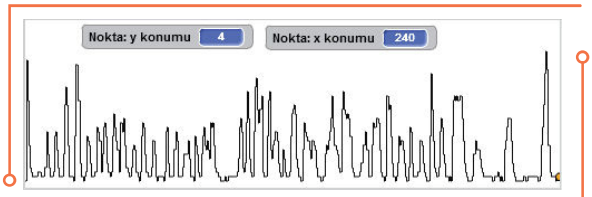
| | |
|---|---|
| temizle | Kalemin bıraktığı izleri temizlemek için kullanılır. |
| iz bırak | Kalemin çizmeye başlaması için kullanılır. |
| kalemi bastır | Kalemın kullanılacak yüzeyde çizmeye başlaması için kullanılır. |
| kalemi kaldır | Kalemi kaldırarak çizimi durdurmak için kullanılır. |
| kalem rengini ☐ yap | Kalemin rengini değiştirmek için kullanılır. |
| kalem rengini 10 değiştir | Kalemın rengını belırtılen değer kadar değıştırmek için kullanılır. |
| kalem rengini 0 yap | Kalemin rengini belirtilen renk yapmak için kullanılır. |
| kalem tonunu 10 arttır | Kalemin tonunu belitirlen oranda artırmak için kullanılır. |
| kalem tonunu 50 yap | kalemin tonunu belitirlen değer yapmak için kullanılır. |
| kalem kalınlığını 1 arttır | Kalemin kalınlığını belirtilen değerde artırmak için kullanılır. |
| kalem kalınlığını 1 yap | Kalemin kalınlığını belirtilen değere getirmek için kullanılır. |

**Tablo 6.4:** "Kalem" alt başlığı altında verilen komut blokları

**Kalem Örneği:** Bu örnekte ses şiddeti ile ekranda ses grafiği çizilmektedir. X koordinatında -240 noktasından başlayıp, +239 olunca ekranı temizlemektedir. Y koordinatı sıfır olarak alınmıştır (Koordinat düzlemi en solda -240, en sağda +240, en yukarıda +180, en aşağıda -180 arasında değişmektedir). Kalem kalınlığı 1, kalem rengi olarak da siyah seçilmiştir. Tanımlana "Ses" değişkeni ile her ekran

temizlemesinden sonra eğer ses varsa program çalışması sağlanmaktadır. Elde edilen grafik boyutunun büyütülmesi için ses şiddeti 4 kat oranında artırılmıştır. Ses kaynağı olarak bilgisayara bağlı bulunan mikrofon veya web kamera mikrofonu kullanılabilmektedir. Çizilen ses grafik örneği aşağıda verilmiştir. Çizilen noktanın x ve y konumlarını göstermek için hareket blok gurubunda bulunan x ve y konumu blokları işaretlenmiştir.


**Resim 6.22:** Kalem uygulaması




**Resim 6.23:** Kalem örneğinin ekran görüntüsü

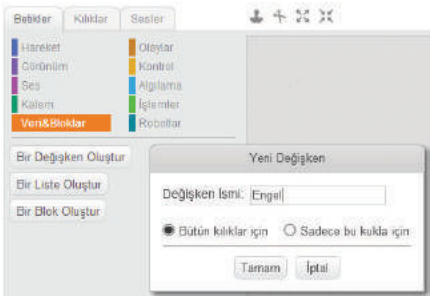
## 6.6.5. Veri&Blok Alt Başlığı Altında Verilen Komut Blokları

"Veri&Blok" alt başlığı altında verilen bloklar sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robotun programlanması ile diğer uygulama programlarında gerekli olabilecek "değişken", değişken listesi "liste" ve "blok" oluşturulması ve bunların düzenlenmesi için kullanılmaktadır.

| | |
|---|---|
| Bir Değişken Oluştur | Bir değişken oluşturmak için kullanılmaktadır. |
| Bir Liste Oluştur | Bir liste oluşturmak için kullanılmaktadır. |
| Bir Blok Oluştur | Bir blok oluşturmak için kullanılmaktadır. |

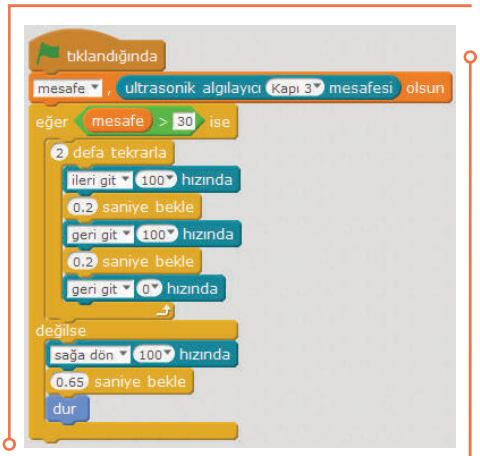
**Tablo 6.5:** Veri&Blok alt başlığı altında verilen komut blokları

**Değişkenler ve Oluşturulması:** Girilen değerleri alan veya programın çalışmasıyla bazı değerlerin atandığı veri tutucularından oluşan temel yapılarından biridir. Değişkenlerin taşıdığı değerler programın akışı içinde farklılaşabilir. Değişkenler, değişken adı ve değeri olmak üzere iki kısımdan oluşurlar. Basit değişken tipleri; sayısal, metin ve boolean tipindedir. Aşağıda "Engel" adında bir değişkenin oluşturulması işlemi gösterilmiştir.

| | |
|---|---|
| mBlock'ta değişkenler "Veri&Bloklar" kategorisinde "Bir Değişken Oluştur" seçeneğinde bulunmaktadır. "Bir Değişken Oluştur" butonuna tıklandıktan sonra, değişken için bir ad girilmelidir. |  |
| Değişkenler programın sol tarafında listelenmektedir. Eklenen her bir değişken için değer atama blokları otomatik olarak eklenmektedir. |  |
| Değer atama bloklarından hangi değişkene değer atanacağı aşağı okuyla seçilmektedir. |  |

**Tablo 6.6:** Değişken oluşturulması aşamaları

**Değişken Oluşturma Örneği:** Bu örnekte robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 30 cm'den büyük ise robot her tıklamada 100 rpm hıza göre 0.2 saniye ileri doğru ve 0.2 saniye geriye doğru 2 defa harekat ettikten sonra durmaktadır. Engele olan uzaklık 30 cm'den küçük ise 0.65 saniye sağa dönüp durmaktadır. Örnekte engele olan uzaklık için "mesafe" adında bir değişken oluşturulmuştur. "Bir Değişken Oluştur" butonuna tıklandıktan sonra, değişken için "mesafe" ad olarak girilmiştir. "Mesafe" değişkeni için değer atama blokları bu aşamada otomatik olarak eklenmiştir. Bu bloklar program içinde kullanılarak,




**Resim 6.24:** Değişken oluşturma örneği

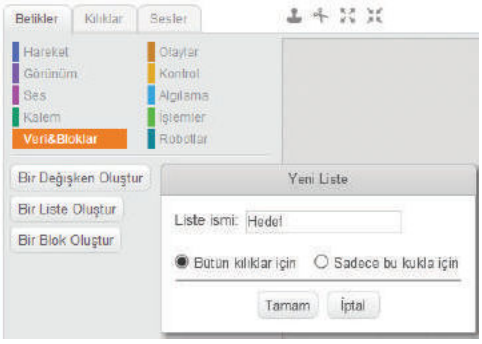
ultrasonik algılayıcı mesafesi "mesafe" değişkeni ile ifade edilmiştir. Uzaklık şartı olan 30 cm "işlemler" kategorisinde bulanan  işlem bloğu kullanılarak yazılmış ve "mesafe" değişkenine eşlenmiştir.

**Listeler, Diziler ve Oluşturulması:** Çok sayıda değişkenle çalışmak için oluşturulmuş temel yapılarından biridir. Listeler değişkenlerden farklı olarak birden fazla değer taşırlar. Dizilerse köşeli parantez içinde virgülle ayrılmış değerler taşır. Aşağıda "Hedef" adında bir değişkenin oluşturulması işlemi gösterilmiştir.

mBlock'ta listeler "Veri&Bloklar" kategorisinde "Bir Liste Oluştur" seçeneğinde bulunmaktadır. "Bir Liste Oluştur" butonuna tıklandıktan sonra, liste için bir ad girilmelidir.

Liste değişkenleri ekranın sol tarafında listelenmektedir. Listeyle yapılabilecek işlemlere ait bloklar otomatik olarak eklenmektedir.

Liste sol üst köşede yer almaktadır. Diziye değerler bu tablonun sol alt köşesindeki + işaretine tıklayarak oluşan kutucuğa yazıyla girilebilir. İstenirse program akışı içerisinde de listeye değer eklenip çıkarılabilmektedir.

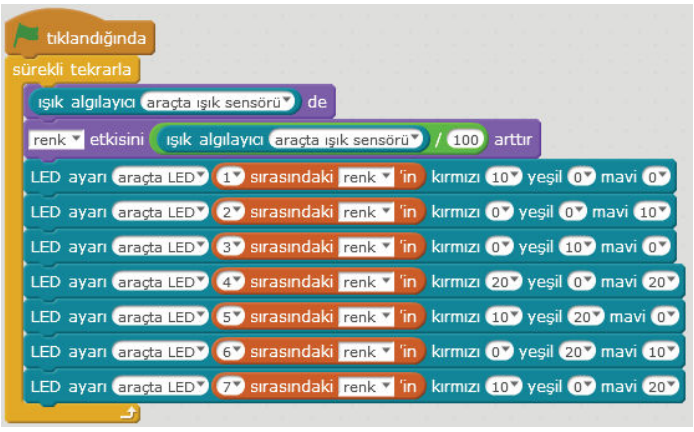
| | |
|---|---|
| mBlock'ta listeler "Veri&Bloklar" kategorisinde "Bir Liste Oluştur" seçeneğinde bulunmaktadır. "Bir Liste Oluştur" butonuna tıklandıktan sonra, liste için bir ad girilmelidir.t |  |
| Liste değişkenleri ekranın sol tarafında listelenmektedir. Listeyle yapılabilecek işlemlere ait bloklar otomatik olarak eklenmektedir. |  |

| | |
|---|---|
| Liste sol üst köşede yer almaktadır. Diziye değerler bu tablonun sol alt köşesindeki + işaretine tıklayarak oluşan kutucuğa yazıyla girilebilir. İstenirse program akışı içerisinde de listeye değer eklenip çıkarılabilmektedir. |  |

**Tablo 6.7:** Liste ve dizi oluşturulması aşamaları

**Liste Oluşturma Örneği:** Bu örnekte robotun üzerinde bulunan ışık algılayıcısı kullanılarak, ortamdaki ışık miktarına göre robotun renk değiştirmesi, aynı zamanda oluşturulan renk listesindeki renklere göre mBot üzerinde bulunan RGB LED'lerin sıra ile yanması ve mBot üzerinde ışık seviyesinin konuşma balonu şeklinde görülmesi sağlanmıştır. Bu amaçla; "Görünüm" kategorisinde bulunan Merhaba ! de bloğu seçilerek buraya "Merhaba" yerine "Robotlar" kategorisinde bulunan ışık algılayıcı araçta ışık sensörü bloğu yerleştirilmiştir. Böylece ışık algılayıcının kullanılması sağlanmıştır. Yine "Görünüm" kategorisinde bulunan renk etkisini 25 arttır bloğu seçilerek üzerine "İşlemler" kategorisinde bulanan işlem bloğu 10/100 oranıyla yerleştirilmiş, bunun üzerine de ışık algılayıcı araçta ışık sensörü bloğu yerleştirilerek renk etkisi artırılarak araç gövdesinin ışık miktarına göre renk değiştirmesi sağlanmıştır. "Veri&Bloklar" kategorisinde bulunan "Bir Liste Oluştur" seçeneği


**Resim 6.25:** Liste oluşturma örneği ekran görüntüsü



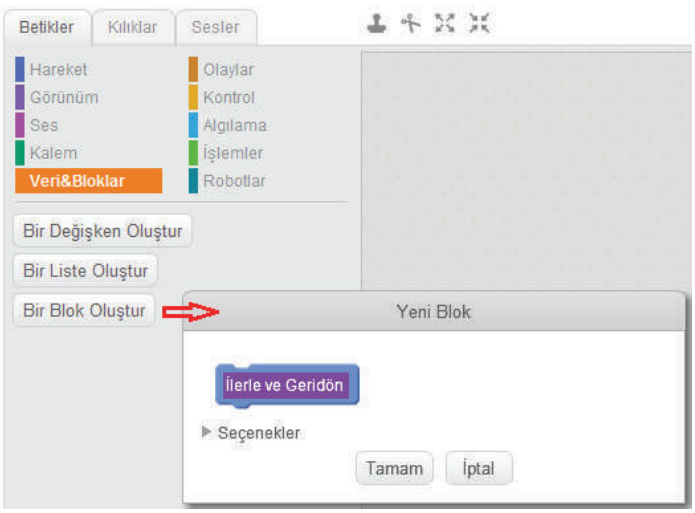
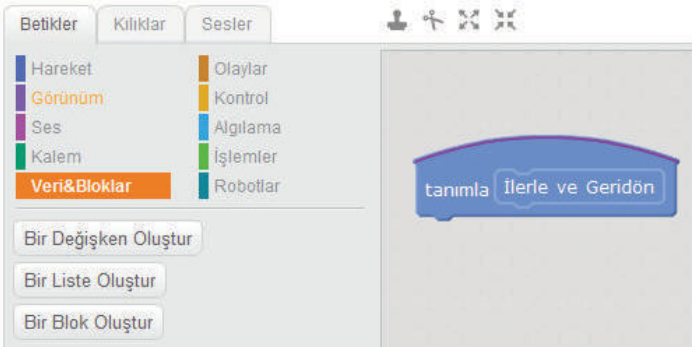

**Resim 6.26:** Liste oluşturma örneği

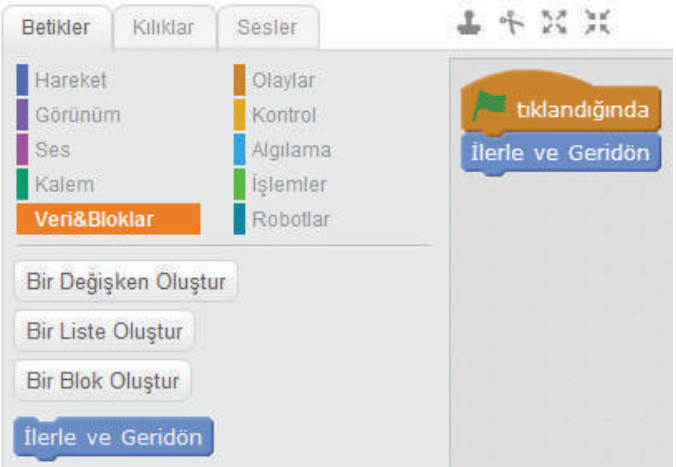
kullanılarak "renk" adını taşıyan bir liste yapılmıştır. "renk" listesini oluşturan değişkenler yazılım tarafından programın sol tarafında otomatik olarak listelenmiştir. "renk" adlı listeyle yapılabilecek işlemlere ait bloklar yine yazılım tarafından otomatik olarak eklenmiştir. Liste sol üst köşede tablo şeklinde yer almıştır. Buraya değerler (renk adları) tablonun sol alt köşesindeki + işaretine tıklayarak oluşan kutucuğa yazılarak girilmiştir. "Robotlar" kategorisinde bulunan 

bloğu seçilerek "hepsi" yerine, oluşturulan listeden 

bloğu seçilerek üzerine yerleştirilmiştir. Bu işlem listedeki 7 renk için tekrarlanarak programa eklenmiştir. Renkler için RGB kodları girilerek LED'lerin listede belirtilen renkte yanması sağlanmıştır.

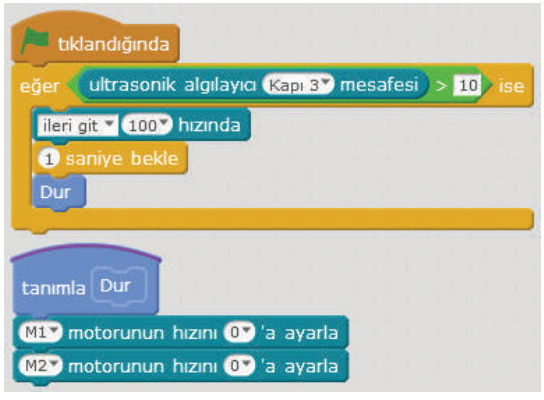
**Blok (Prosedür) Oluşturma:** Kodu bir kez yazıp defalarca kullanmak için ortaya konmuş temel yapılardır. Program akışı içinde tekrarlayan ifadelerin her seferinde tekrar tekrar yazılması yerine, bir kere ayrı bir yerde yazılıp tekrarlanan her yerde kullanmak için uygundurlar. Aşağıda "İlerle ve Geridön" adında bir blok oluşturulması işlemi gösterilmiştir. Burada bu blokun oluşturulmasıyla yapılmak istenilen, bir işlem gurubunu tek bir blok olarak belirleyip her seferinde aynı blokların ayrı ayrı kullanılmasını ortadan kaldırmaktır. Örneğimizde robotun önce ileri gidip sonra geri gelme işlemi 7 adımdan oluşmaktadır ancak İlerle ve Geri Dön bloğu oluşturulduğunda bunu tek bir blokla gerçekleştirmek mümkün olmaktadır.

| | |
|---|---|
| mBlock'ta blok oluşturulması "Veri&Bloklar" kategorisinin altında "Bir Blok Oluştur" seçeneğinde bulunmaktadır. "Bir Blok Oluştur" butonuna tıklandıktan sonra, oluşan "Yeni Blok" penceresindeki isimsiz alana blok için bir ad girilmeli ve onaylanmalıdır. |  |
| Programda kullanılacak ve çağrılabilecek blok otomatik olarak eklenmektedir. |  |

| | |
|---|---|
| Bu şekilde blok oluşturulması yapıldıktan sonra blok tarafından yapılacak işlemlerin blok altında tanımlanması gerekmektedir. Örnekte robotu 1 sn ileri ve 1 sn geri getirdikten sonra durmasını sağlayan bir blok oluşturulmuştur. |  |
| Blok kullanılmak istendiğinde ana programdan çağrılmaktadır. |  |

**Tablo 6.8:** Blok oluşturma aşamaları

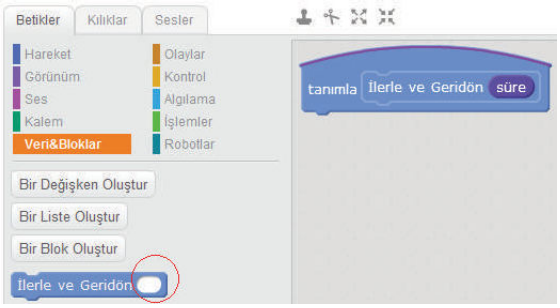
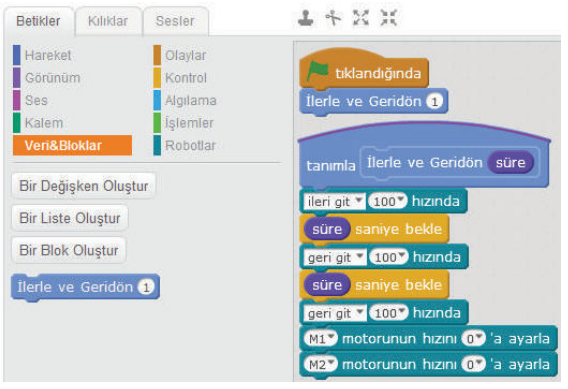
**Blok (Prosedür) Oluşturma Örneği:** Bu örnekte robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 10 cm'den büyük ise program çalışmaktadır. Engele olan uzaklık 10 cm'den büyük ise her tıklamada 100 rpm hıza göre 1 saniye ileri doğru 10 cm kalıncaya kadar gitmektedir. Koşul sağlanınca "Dur" prosedürü ile robot durmaktadır. Bu prosedür "Veri&Bloklar" kategorisinde bulunan "Bir Blok Oluştur" seçeneği ile oluşturulmuş ve prosedür tanımlanması "Dur" şeklinde yapıldıktan sonra kullanacağımız işlemler prosedürün al-




**Resim 6.27:** Blok oluşturma örneği

tında M1 ve M2 motorlarının hızlarını 0'a ayarla şeklinde tanımlanmıştır. Bu işlemlerin tanımlanması için "Robotlar" kategorisinde bulunan yandaki blok kullanılmıştır. Bloğun bir kopyası oluşturularak M1 ve M2 seçenekleri seçilmiştir. Uzaklık şartı olan 10 cm "İşlemler" kategorisinde bulanan  işlem bloğu kullanılarak yazılmıştır.

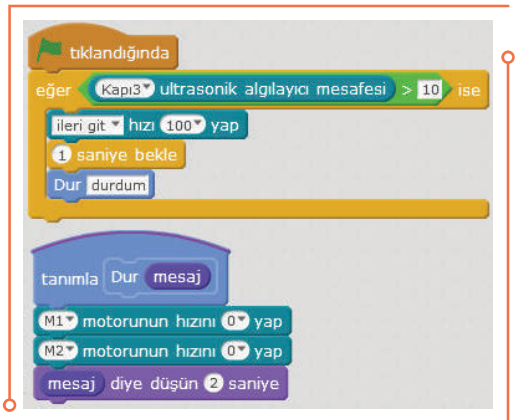
**Oluşturulan Blok (Prosedür) İçin Parametre Tanımlanması:** Prosedürlere değer taşıyan değişkenlere parametre adı verilir. Bir prosedür çağrıldığı zaman aynı zamanda parametre değerlerini de vermek gerekmektedir. Oluşturulan "İlerle ve Geridön" bloğu bu anlamda bir prosedürdür. Bu prosedüre parametre ekleme işlemi aşağıdaki örnekte açıklanmıştır.

| | |
|---|---|
| mBlock'ta oluşturulan bloka parametre tanımlanması için "Bir Blok Oluştur" seçeneğiyle açılan "Yeni Blok" penceresinde blok oluşturulup ad verilmesinden sonra yine bu pencerede bulunan "Seçenekler" düğmesine tıklanmalıdır. Uygun olan sayısal veya sözel parametre tipi seçildikten sonra parametre adı girilmelidir. |  |
| Yeni tanımlanan bu prosedürde yapılacak işler parametre değeri ile birlikte tanımlanır. |  |
| Hazırlanan prosedüre parametre değeri vererek kullanılır. Bunun için prosedür bloğu içinde tanımlanan kutucuğun içerisine doğrudan bir değer girerek ya da bir değişken atayarak kullanılır. Girilen bu değer prosedüre aktarılacak ve bu değerle işlem yapılacaktır. |  |

**Tablo 6.9:** Prosedüre parametre ekleme aşamaları

**Parametre Örneği:** Bu örnekte robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 10 cm'den büyük ise program çalışmaktadır. Engele olan uzaklık 10 cm'den büyük ise her tıklamada 100 rpm hıza göre 1 saniye ileri doğru, engele 10 cm kalıncaya kadar gitmektedir. Koşul sağlanınca oluşturulan "Dur" değişkeni ile robot durmaktadır. Bu aşamaya kadar

olan işlemler için yandaki uygulama örneği kullanılmıştır. Uzaklık şartı olan 10 cm "İşlemler" kategorisinde bulanan  işlem bloğu kullanılarak yazılmıştır.




**Resim 6.28:** Parametre örneği


**Resim 6.29:** Parametre örneği ekran görüntüsü

Bu prosedüre parametre tanımlamak için "Dur" değişkenine sağ tıklanmış ve açılan "Düzenle" düğmesinde bulunan "Seçenekler" içerisinden "Sözel girdi ekle" kullanılarak parametre tipi seçilmiştir. Prosedür bloğu içinde tanımlanan kutucuğun içerisine "durdum" yazılarak parametre tamamlanmıştır. Program çalıştırılıp koşul sağlanınca programdaki kukla "durdum" ifadesini 2 saniye boyunca ekrana yazmaktadır.

### 6.6.6. Olaylar Alt Başlığı Altında Verilen Komut Blokları

Olaylar alt başlığı altında verilen bloklar sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robot için oluşturulan programların ve diğer uygulamaların çalıştırılmasında kullanılmaktadır. Buradaki blokların kullanılmasıyla bir tuşun basılı olup olmaması veya seçilen nesnenin aktif olup olmaması durumuna göre uygulamanın çalışması sağlanabilir. Belirlenen bir haberin gelmesi durumunda yapılacak işlemler kontrol edilebilir. Olaylar alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

| | |
|---|---|
| tıklandığında | Herhangi bir uygulamanın tıklandığında çalışması için kullanılır. |
| boşluk tuşu basılınca | Herhangi bir uygulamanın belirlenen tuşa basılmasıyla çalışması için kullanılır. |
| boşluk tuşu bırakılınca | Herhangi bir uygulamanın belirlenen tuşun bırakılmasıyla çalışması için kullanılır. |
| bu kukla tıklanınca | Herhangi bir uygulamanın kuklaya tıklanmasıyla çalışması için kullanılır. |
| dekor dekor1 olunca | Herhangi bir uygulamanın belirlenen dekorun kullanılmasıyla çalışması için kullanılır. |

| | |
|---|---|
| ses şiddeti > 10 olunca | Herhangi bir uygulamanın ses şiddeti, süre ölçer veya video hareketi belirlenen değerden büyük olunca çalışması için kullanılır. |
| message1 haberi gelince | Herhangi bir uygulamanın belirtilen ileti veya belirtilen yeni ileti gelince çalışması için kullanılır. |
| message1 haberini sal | Herhangi bir uygulamada belirtilen ileti veya belirtilen yeni ileti gelince açıklanması için kullanılır. |
| message1 haberini sal ve bekle | Herhangi bir uygulamada belirtilen ileti veya belirtilen yeni ileti gelince açıklanması ve beklenmesi için kullanılır. |

**Tablo 6.10:** Olaylar alt başlığı altında verilen komut blokları

**Olay Örneği:** Robot kontrol kartı üzerindeki RGB LED'lerin rengini kontrol edebilen bu uygulama tıklandığında, a tuşu basılınca, a tuşundan el çekince, bunun için oluşturulmuş kuklaya tıklanınca veya ses şiddeti 10'dan büyük olunca çalıştırılabilir. Bunun için yapılması gereken tek şey istenilen bloku tıklandığında blokuyla yer değiştirmektir.


**Resim 6.30:** Olay örneği

## 6.6.7. Kontrol Alt Başlığı Altında Verilen Komut Blokları

Kontrol alt başlığı altında verilen bloklar sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robot için hazırlanan programların ve diğer uygulamaların oluşturulmasında kullanılan temel programlama komutlarından oluşmakta ve tüm programlama uygulamalarında kullanılmaktadır. Buradaki blokların kullanılmasıyla birden fazla yapılması gereken döngüsel işlemler gerçekleştirilebilir. Koşullu durumunda gerçekleştirilmesi gereken işlemler yapılabilir. Ayrıca çalışan tüm kodların durdurulması ve nesnenin ikizi ile ilgili işlemlerin yapılması sağlanabilir. Robotik uygulamalarda, robot hareketinin durdurulması için "durdur" bloğu kullanılamamaktadır. Bu durumlara motor hızlarının sıfırlanarak durdurulması pratik bir çözüm olabilir. Aşağıda verilen örnekleri inceleyip uygulayınız. Kontrol alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

| | |
|---|---|
| 1 saniye bekle | Programın belirtilen saniye kadar beklemesi için kulanılır. |
| 10 defa tekrarla | Döngüler tekrarlanan işleri yapmak için kullanılan temel yapılardır. Bu döngü bloğu belirtilen sayı kadar işlemi tekrarlamak için kullanılır. |
| sürekli tekrarla | Verilen işlemi sürekli tekrarlamak için kullanılan döngü bloğudur. |
| eğer ise | Olumlu koşul ifadesi "eğer" "ise" koşulu gerçekleşene kadar işlemi tekrarlamak için kullanılır. Döngü bloğunda bir koşul tanımlanır ve o koşul gerçekleşene kadar döngü devam eder. |

| | |
|---|---|
| eğer ise<br>değilse | Olumlu koşul ifadesi "eğer" "ise" ve olumsuz koşul ifadesi "değilse" koşulu gerçekleşene kadar işlemi tekrarlamak için kullanılır. |
| olana kadar bekle | Belirtilen koşul gerçekleşene kadar işlemi bekletmek için kullanılır. |
| olana kadar tekrarla | Belirtilen koşul gerçekleşene kadar işlemi tekrarlamak için kullanılır. Koşul komutları program akışını farklı durumlara göre değiştirmek, yönlendirmek için kullanılan temel karar yapılardır. |
| hepsi durdur | Çalışan betik, kuklanın diğer betikleri veya hepsinde işlemi durdurmak icin kullanılır. |
| ikiz olarak başladığımda | Program ikiz olarak başlatıldığında kullanılır. |
| kendim in ikizini oluştur | Programcının kendisi veya kullanılan kukla tarafından ikizinin oluşturulması için kullanılır. |
| bu ikizi sil | Oluşturulan ikizin silinmesi için kullanılır. |

**Tablo 6.11:** Kontrol alt başlığı altında verilen komut blokları

### 6.6.7.1. Kontrol Örnekleri-Döngüler

**Verdiğimiz Sayı Kadar İşlemi Tekrarlayan Döngü Örneği:** Bu örnekte robot 100 rpm hıza göre 1 saniye ileri, 1 saniye de geri hareket etmekte ve toplamda bunu 2 defa tekrarlamaktadır. Tekrarın sonunda geri gelme hızını sıfırlayarak durmaktadır.


**Resim 6.31:** Verdiğimiz sayı kadar işlemi tekrarlayan döngü örneği

**İşlemi Sürekli Tekrarlayan Döngü Örneği:** Bu örnekte robot 100 rpm hıza göre 1 saniye ileri, 1 saniye geri hareket etmekte ve bu işlemi sürekli olarak tekrarlamaktadır.


**Resim 6.32:** İşlemi sürekli tekrarlayan döngü örneği

**Belirli Bir Şart Gerçekleşene Kadar Bekleme Örneği:** Bu örnekte robot ses şiddeti 10'dan büyük olunca çalışmaya başlamaktadır. Robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 20 cm'den büyük ise robot beklemeye geçmektedir. Eğer engele olan uzaklık 20 cm'den küçük olursa (örneğin elinizi ultasonik algılayıcıya yaklaştırırsanız) koşul gerçekleşince 100 rpm hıza göre 1 saniye ilerleyip M1 ve M2 motorlarının hızlarını sıfıra düşürmektedir. Uzaklık şartı olan 20 cm "İşlemler" kategorisinde bulanan  işlem bloğu kullanılarak yazılmıştır.


**Resim 6.33:** Belirli bir şart gerçekleşene kadar bekleme örneği

**Belirli Bir Şart Gerçekleşene Kadar Döngü Örneği:** Bu örnekte robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 20 cm'den büyük ise program çalışmaktadır. Engele olan uzaklık 20 cm'den küçük olana kadar 100 rpm hıza göre 1 saniye 1 saniye ilerleyip koşul gerçekleşince M1 ve M2 motorlarının hızlarını sıfıra düşürmektedir. Uzaklık şartı olan 20 cm "İşlemler" kategorisinde bulanan  işlem bloğu kullanılarak yazılmıştır.


**Resim 6.34:** Belirli bir şart gerçekleşene kadar döngü örneği

### 6.6.7.2. Kontrol Örnekleri -Koşullar

**Olumlu Koşul İfadesi "eğer" "ise" Örneği:** Bu örnekte robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 20 cm'den küçük ise program çalışmaktadır. Engele olan uzaklık 20 cm'lik alan içerisinde kalıncaya kadar her tıklamada 100 rpm hıza göre 1 saniye

gerileyip, M1 ve M2 motorlarının hızlarını sıfıra düşürmektedir. Uzaklık şartı olan 20 cm "İşlemler" kategorisinde bulanan  işlem bloğu kullanılarak yazılmıştır.


**Resim 6.35:** Olumlu koşul ifadesi "eğer" "ise" örneği

**Olumlu Koşul İfadesi "eğer" "ise" ve Olumsuz Koşul İfadesi "değilse" Örneği:** Bu örnekte de robotun ultrasonik algılayıcısı kullanılarak engele olan uzaklık ölçülmektedir. Engele olan uzaklık 10 cm'den büyük ise her tıklamada 100 rpm hıza göre 1 saniye ileri doğru 10 cm kalıncaya kadar gitmektedir. Eğer engele olan uzaklık 10 cm'den küçük ise robot geriye doğru 10 cm oluncaya kadar 100 rpm hıza 1 saniye boyunca çalışmaktadır. Koşul sağlanınca M1 ve M2 motorlarının hızlarını sıfıra düşürmektedir. Uzaklık şartı olan 10 cm "İşlemler" kategorisinde bulanan  işlem bloğu kullanılarak yazılmıştır.


**Resim 6.36:** Olumlu koşul ifadesi "eğer" "ise" ve olumsuz koşul ifadesi "değilse" örneği

### 6.6.8. Algılama Alt Başlığı Altında Verilen Komut Blokları

Algılama alt başlığı altında verilen bloklar sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robot için hazırlanan programların ve diğer uygulamaların oluşturulmasında ve tüm uygulama programlarında kullanılmaktadır. Buradaki blokların kullanılmasıyla bir nesneye, bir renge değme durumu ya da fareye olan mesafe algılanabilir. Mesaj vermek için kullanılabilir. Klavye ve fare kullanımına dayalı işlemler yapılabilir. Bilgisayara bağlı kameranın aktif olması ve sanal robotun video hareketlerine göre değişiklik göstermesi sağlanabilir. Ayrıca nesnelerin konumunu algılama işlemleri de yapılabilmektedir. Algılama alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

| | |
|---|---|
| a değdi (mi?) | Kuklanın fare okuna, sağ, sol, üst veya alt kenara değdiğini belirlemek için kullanılır. |
| rengine değdi (mi?) | Kuklanın belirlenen herhangi bir renge değdiğini belirlemek için kullanılır. |
| rengi rengine değdi (mi?) | Kuklanın belirlenen herhangi bir renginin belirlenen alanda herhangi bir renge değdiğini belirlemek için kullanılır. |
| 'a mesafe | Fare okuna, kuklaya, rasgele yatay, dikey ve sahne noktasına olan mesafeyi belirlemek için kullanılır. |
| İsmin ne? diye sor ve bekle | Belirtilen ifadeyi (ad, isim, yer vs.) sormak ve beklemek için kullanılır. |
| yanıt | Yanıtını ekranda almak için kullanılır. |
| boşluk tuşu basılı (mı?) | Belirtilen tuşun basılı olup olmadığını belirlemek için kullanılır. |
| fareye basılı mı? | Farenin tuşunun (aktif tuş) basılı olup olmadığını belirlemek için kullanılır. |
| farenin x'i | Farenin x koordinatını belirlemek için kullanılır. |
| farenin y'si | Farenin y koordinatını belirlemek için kullanılır. |
| ses şiddeti | Ses şiddetini ekranda görmek için kullanılır. |
| video hareket on bu kukla | Kullanılan kukla veya sahne üzerinde video hareketi ve yönü belirtmek için kullanılır. |
| videoyu aç | Videoyu açmak, kapatmak ve açıp solu sağ yapmak için kullanılır. |
| video saydamlığı % 50 olsun | Video saydamlığını belirtilen orana ayarlamak için kullanılır. |
| süre ölçer | Süre ölçeri ekran üzerinde açmak için kullanılır. |
| süre ölçeri sıfırla | Süre ölçeri sıfırlamak için kullanılır. |
| x konumu değeri mBot 4_1_1 'in | x, y konumu, yönü, kılık no, kılığın ismi, büyüklük ve ses şiddeti değerleri mevcut kukla veya sahne için kullanılır. |
| şimdiki dakika | Ekranda saniye, dakika, saat, haftanın günü, tarih, ay ve yıl bilgilerini göstermek için kullanılır. |
| 2000'den beri geçen gün | 2000 yılından beri geçen gün sayısını belirlemek için kullanılır. |

**Tablo 6.12:** Algılama alt başlığı altında verilen komut blokları

**Algılama Örneği:** Bu örnekte yukarı ve aşağı ok tuşları kullanılarak sanal robotun dikey yönde aşağı veya yukarı hareket etmesi sağlanmıştır. Program çalıştırıldığında robot x=-180, y=0 konumuna gitmektedir. Yukarı ve aşağı hareket y değerinin artırılıp azaltılmasıyla sağlanmaktadır.


**Resim 6.35:** Algılama örneği

Bu örnekten yararlanarak her yöne hareket edebilen bir sanal robot tasarlayınız.

### 6.6.9. İşlemler Alt Başlığı Altında Verilen Komut Blokları

İşlemler alt başlığı altında verilen bloklar sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robot için hazırlanan programların ve diğer uygulamaların oluşturulmasında ve tüm uygulama programlarında kullanılmaktadır. İşlemler bloğu matematiksel işlemlerin bulunduğu bloktur. Dört işlem gerçekleştirme, iki değer arasında rastgele değer üretme, karşılaştırma yapma, birden fazla durumu veya bir durumu seçmek veya seçmemek için kullanılmaktadır. Ayrıca iki farklı ifadeyi birleştirme, karakter uzunluğu belirtme işlemleri ve basit matematiksel hesaplamalar (mod alma, yuvarlama, karekök alma gibi) yapılabilmektedir. İşlemler alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

| | |
|---|---|
| + | Toplama işlemleri için kullanılır. |
| - | Çıkarma işlemleri için kullanılır. |
| * | Çarpma işlemleri için kullanılır. |
| / | Bölme işlemleri için kullanılır. |
| 1 ile 10 arasında bir sayı (tut) | Belirtilen değerler arasında rastgele sayı üretmek için kullanılır. |
| < | "<" karşılaştırma işlemleri için kullanılır. |
| = | "=" karşılaştırma işlemleri için kullanılır. |
| > | ">" karşılaştırma işlemleri için kullanılır. |
| ve | Mantıksal işlemler için kullanılır. Her iki koşul da doğru olduğunda "ve" yapı taşı doğru olacaktır. Aksi takdirde yanlış olur. |

| | |
|---|---|
| veya | Mantıksal işlemler için kullanılır. Her iki durumdan biri doğru olduğunda, "veya" yapı taşı doğrudur. Aksi takdirde yanlış olur. |
| değil | Mantıksal işlemler için kullanılır. Matıksal işlem "değil" ise yapı taşı doğrudur. |
| hello ile world i birleştir | Belirtlen iki kelimeyi birleştirmek için kullanılır. |
| 1 . harfi world in | Belirtilen kelimenin belirtilen harfiyla işlem yapmak için kullanılır. |
| world in uzunluğu | Belirtilen kelimenin uzunluğuyla işlem yapmak için kullanılır. |
| index of world in hello world | Belirtilen kelime içinden alanan kelimenin indeksi ile işlem yapmak için kullanılır. |
| oyuncu 1 'i dizgeye | Belirtilen oyuncuyu dizgeye taşımak için kullanılır. |
| modüler | Mod işlemleri yapmak için kullanılır. |
| i yuvarla | Belirtilen değeri yuvarlamak için kullanılır. |
| karekök 9 'nin | Belirtilen sayının karakökünü, mutlak değerini almak, aşağı veya yukarı yuvarlamak ve açısal değerleri için kullanılır. |

**Tablo 6.13:** İşlemler alt başlığı altında verilen komut blokları

**İşlem Örneği:** Bu örnekte 1 ile 10 arasında rastgele oluşturulan sayıya göre kuklanın hareketi sağlanmıştır. Bu amaçla "Sayı" adında bir değişken oluşturulmuştur. Tutulan sayı 5'ten küçükse kukla 20 adım ilerlemektedir. Eğer tutulan sayı 5'ten büyükse kukla 20 adım geri gitmektedir. Eğer sayı 5'e eşitse 15 derece dönmekte, eğer sayı 0'a eşitse -15 derece dönmekte, eğer sayı 1'e eşitse 90 (sağa) yönüne dönmektedir. Eğer sayı 2 veya 3 ise -90 (sola) yönüne dönmektedir. Eğer sayı 4 değilse ekrana 2 sn boyunca Merhaba! şeklinde yazmaktadır.


**Resim 6.38:** İşlem örneği

## 6.6.10. Robotlar Alt Başlığı Altında Verilen Komut Blokları

Robotlar alt başlığı altında verilen bloklar Arduino uyumlu kartlar için uygulama programları hazırlanmasında, fiziksel bir robot için programların hazırlanmasında, Makeblock tarafından üretilen robot ve robot kontrol kartları ve kalkanların (shield) programlanmasında ve diğer tüm donanım tabanlı programlama uygulamalarında kullanılmaktadır. Kullanılan kart türüne göre desteklenen bloklar değişmekte olup aşağıda

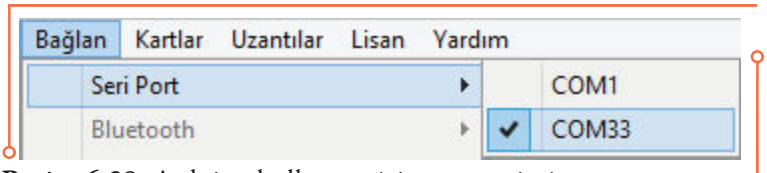
Arduino uyumlu kartlar için kullanılabilecek bloklar verilmiştir. Robotlar alt başlığı altında verilen komut blokları ve ne amaçla kullanıldıkları aşağıdaki tabloda açıklanmıştır.

| | |
|---|---|
| Arduino Programı | Arduino programları için kullanılır. |
| 9 sayısal pini oku | Belirtilen sayısal pini okumak için kullanılır. |
| (A) 0 analog pini oku | Belirtilen analog pini okumak için kullanılır. |
| 13 darbe pini oku, zaman aşımı 20000 olsun | Belirtilen PWM pinini, verilen zaman aşımı içinde okuması için kullanılır. |
| 9 sayısal pini YÜKSEK yap | Belirtilen sayısal pini LOW veya HIGH yapmak için kullanılır. |
| 5 pwm pini 0 yap | Belirtilen PWM pinini, 0,50,100,150 veya 255 değerlerine ayarlamak için kullanılır. |
| 9 ses tonu pini C4 notasında Yarım vuruş çal | Belirtilen ses tonu pini, istenilen notada, istenilen vuruş kadar çalması için kullanılır. |
| 9 servo pini açısını 90 yap | Belirtilen servo pini açısını 0, 45, 90, 135 veya 180 derece yapmak için kullanılır. |
| seri porta merhaba yaz | Verilen ifadeyi seri porta yazdırmak için kullanılır. |
| seri portta byte var | Seri porttan gelen byte'ı tespit için kullanılır. |
| seri porttan byte oku | Seri porttan gelen byte'ı okumak için kullanılır. |
| ultrasonik 13 tetik pini 12 okuma pini | Ultrasonik sensörün tetikleme ve okuma pinini belirtmek için kullanılır. |
| süre ölçer | Süre ölçümü için kullanılır. |
| süre ölçeri sıfırla | Süre ölçeri sıfırlamak için kullanılır. |

**Tablo 6.14:** Robotlar alt başlığı altında verilen komut blokları

## 6.7. mBlock ile Arduino Kullanımı

Arduino kullanım örnekleri UNO R3 üzerinden verilmiştir. mBlok üzerinden Arduino kullanımı için "Bağlan" sekmesinden Arduino'nun bağlandığı Seri Port, ayrıca "Kartlar" sekmesinden de "Arduino Uno" seçilmelidir.

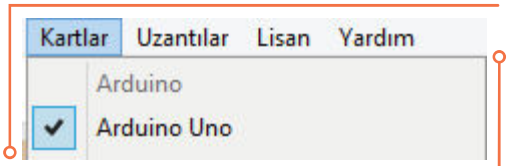



**Resim 6.39:** Arduino kullanımı için port seçimi

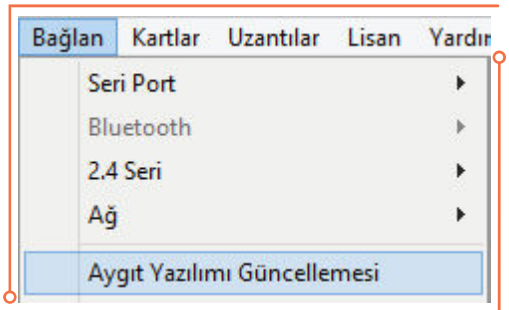
Arduino ile kullanılacak elektronik bileşenlerin önceden hazırlanması, bağlantılarının yapılması sonra da uygulamanın hazırlanarak Arduino'ya yüklenmesi gerekmektedir. Yüklemek için "Arduino Prog-

ramı” 

blokunun üzerine tıklanarak açılan ekrandan “Arduinoya Yükle” seçeneğinin tıklanması gerekmektedir. Bu durumda uygulama ekranın sağında Arduino kodları şeklinde görülecek ve yükleme bitince “Yükleme Bitti” şeklinde uyarı verecektir. Bu şekildeki kullanımda sadece Arduino için oluşturulmuş bloklar kullanılabilir. Yüklenen uygulama mBlock olmadan Arduino üzerinde kullanılabilir. Fakat istenirse “Bağlan” sekmesinden “Aygıt Yazılımı Güncellemesi” seçeneği ile Arduino kullanım kodları yüklenerek Arduino blokları ve diğer bloklar birlikte kullanılabilir. Bu durumda hazırlanan uygulama Arduino hafızasında yer almayacak yalnızca mBlock üzerinden kullanılabilecektir.

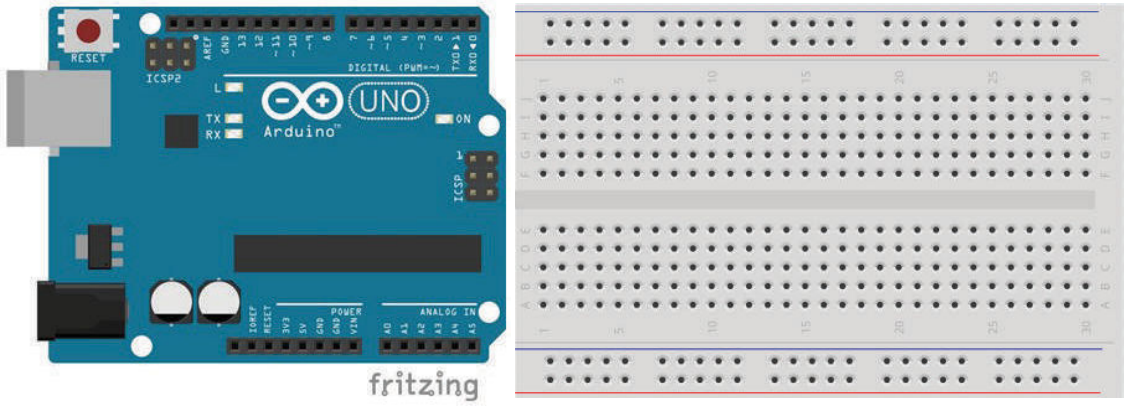


Resim 6.40: Arduino Uno seçimi



**Resim 6.41:** Aygıt yazılımının güncellenmesi

Arduino uygulamalarında kullanılacak elektronik bileşenler için Breadboard 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 Arduino UNO R3 ve bir breadboard örneği yer almaktadır.

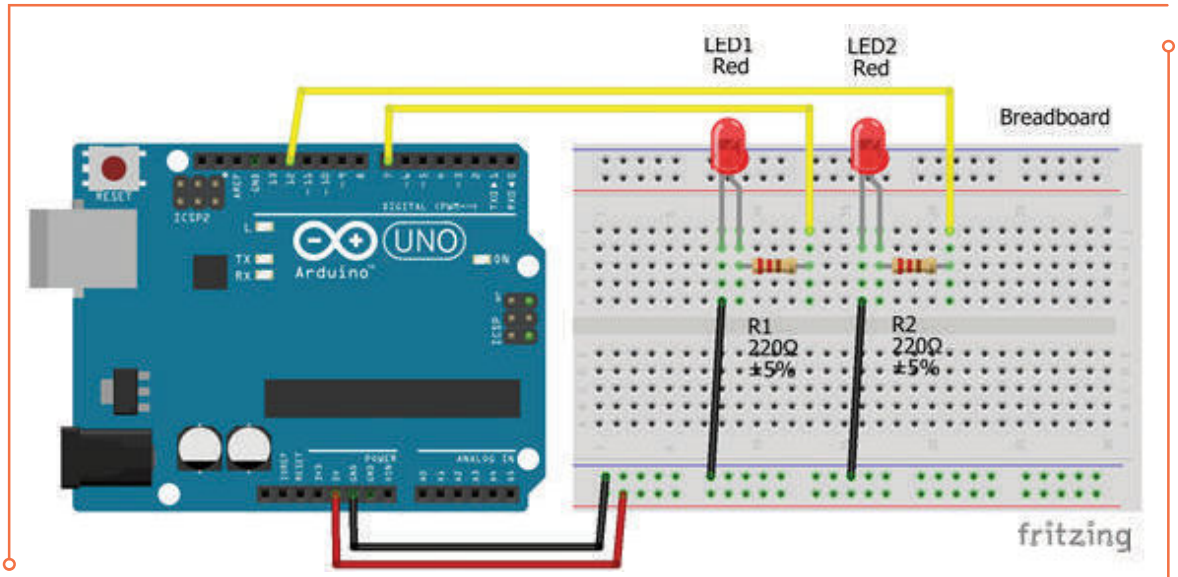


**Resim 6.42:** Arduino UNO R3 ve Breadboard örneği

**Dijital Kontrol Pimlerin Ayarlanması ve Okunması:** Arduino UNO kartında toplamda 14 dijital kontrol pimi (6 adet PWM dâhil) bulunmaktadır. Dijital pinlerin çıkış değerini 0 (DÜŞÜK) ya da 1 (YÜKSEK) olarak ayarlamak ve aynı zamanda, dijital pinlerin giriş değerlerini ekranda okumak için yandaki örneği inceleyiniz.

**Resim 6.43:** Dijital kontrol pimlerin ayarlanması ve okunması

Aşağıda verilen iki aynı örnekte Arduino’nun 7 ve 12 numaralı dijital pinlerine bağlı 2 LED’in 1 sn aralıklarla yanıp sönmesi sağlanmıştır. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir. LED’ler düşük güçle çalıştıkları ve zarar görmemeleri için birer 220 ohm dirençle bağlanmıştır.

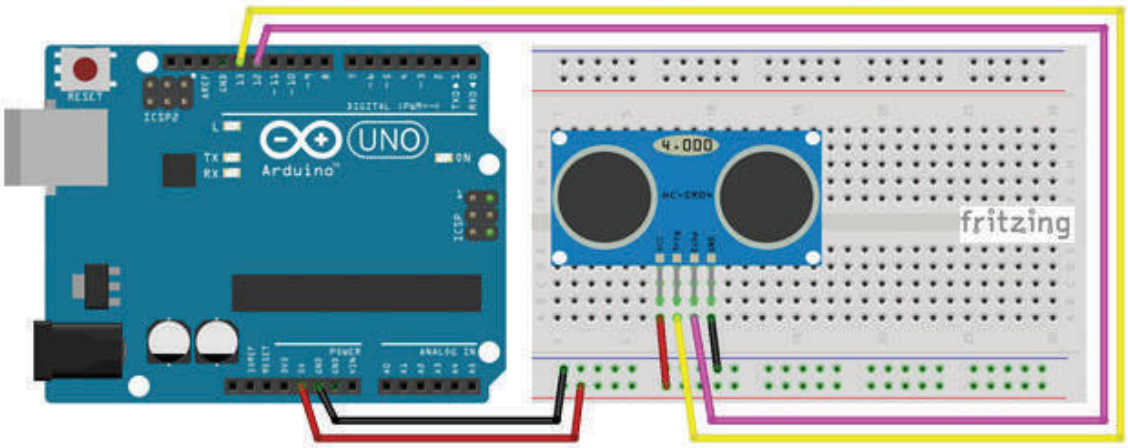



**Resim 6.44:** Arduino UNO ile LED uygulaması Breadboard çizimi

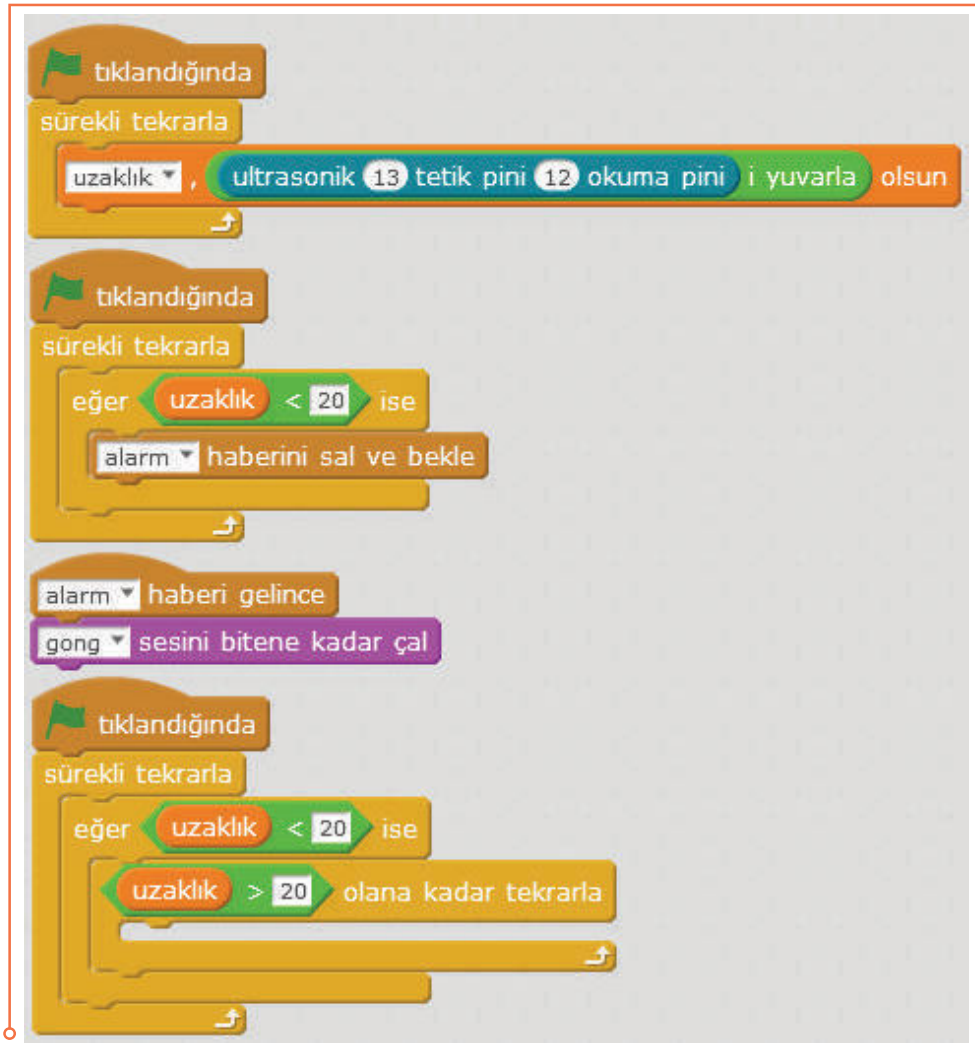
Aşağıdaki LED uygulamasının çalışması için "Arduino Programı" blokunun üzerine tıklanarak açılan ekranda "Arduinoya Yükle" seçeneğinin tıklanması gerekmektedir. Örnekteki diğer programın tıklandığında çalışabilmesi için önceden "Bağlan" sekmesinden "Aygıt Yazılımı Güncellemesi" seçeneği ile güncelleme yapılması gerekmektedir.


**Resim 6.45:** Arduino UNO ile LED uygulaması örneği

Aşağıda verilen örnekte Arduino'nun dijital pinlerine bağlı HY-SRF 05 ultrasonik sensör kullanılarak bir alarm uygulaması hazırlanmıştır. Ultrasonik sensörün tetik (Trig) pini 13 numaralı, okuma (Echo) pini 12 numaralı Arduino UNO pinine bağlanmıştır. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir. Eğer ultrasonik sensöre olan uzaklık 20 cm'den küçük ise alarm devreye girmekte ve uzaklık 20 cm'den büyük oluncaya kadar devam etmektedir.




**Resim 6.46:** Arduino UNO ile Ultrasonik Sensör uygulaması Breadboard çizimi



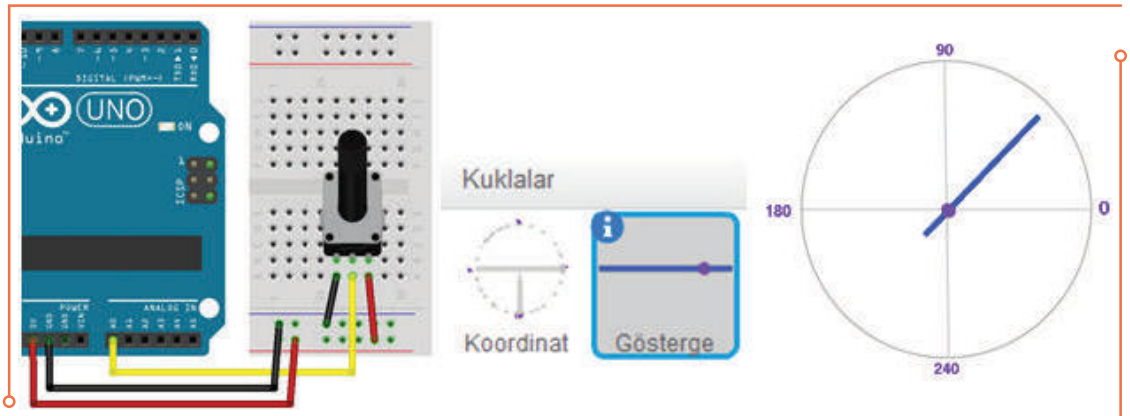

**Resim 6.47:** Arduino UNO ile Ultrasonik Sensör uygulaması örneği

**Analog Kontrol Pimlerin Ayarlanması ve Okunması:** Arduino UNO kartında toplamda 6 analog kontrol pimi bulunmaktadır (A0, A1, A2, A3, A4, A5, A6 numaralı pinler). Analog pinlerin çıkış değerini 0 ile 1023 arasında ayarlamak ve aynı zamanda, analog pinlerin giriş değerlerini ekranda okumak için yandaki örneği inceleyiniz.


**Resim 6.48:** Analog pimlerin ayarlanması ve okunması

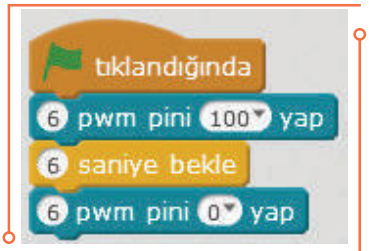
Aşağıda verilen örnekte A1 pinine bağlı bir potansiyometre kullanılarak 3600'lik bir açıölçer uygulaması yer almaktadır. Potansiyometrenin hareketi ile çizilen koordinat sistemi üzerinde gösterge istenen açıda ayarlanabilmektedir. Arduino ile okunan analog değerler 0 ile 1023 arasında olduğu, koordinat sisteminde ise 3600 istendiği için dönüştürme işlemi uygulanmıştır. Bunun için potansiyometre değeri 1024'e bölünerek 360 ile çarpılmıştır. Böylece 0 ile 1023 arasındaki bir değer 0 ile 360 arasına taşınmıştır. Potansiyometrenin Arduino ile bağlantısı ve koordinat sistemi aşağıda gösterilmektedir. Kullanılan göstergenin kaç derecelik açıyla başlayacağını belirlemek için "yönüne don" blokunda -90° alınmıştır. Kullanılacak koordinat sistemi ve gösterge şekline göre istenilen açıdan başlanabilir.


**Resim 6.49:** Arduino UNO ile Potansiyometre uygulaması örneği




**Resim 6.50:** Arduino UNO ile Potansiyometre uygulaması Breadboard çizimi ve ekran görüntüleri

**PWM Dijital Kontrol Pimlerin Ayarlanması ve Okunması:** Arduino UNO kartında toplamda 6 adet PWM dijital kontrol pimi bulunmaktadır (D3, D5, D6, D9, D10, D11 numaralı pinler). PWM pinlerin çıkış değerini 0 ile 1023 arasında ayarlamak ve aynı zamanda, PWM pinlerin giriş değerlerini ekranda okumak için hazırlanan yandaki örneği inceleyiniz.

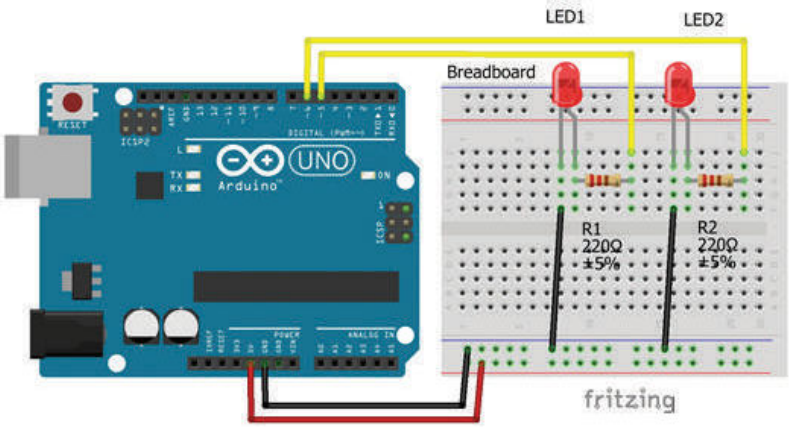



**Resim 6.51:** PWM Dijital kontrol pimlerin ayarlanması ve okunması işlemi

Aşağıda verilen örnekte Arduino'nun 5 ve 6 numaralı pwm pinlerine bağlı 2 LED'in boşluk tuşuna basıldığı sürece ışık seviyesinin 0.5 birim artarak yanması sağlanmıştır. Bunun için ışık seviyesi adını taşıyan bir değişken oluşturulmuştur. Örnek çalıştırıldığı zaman boşluk tuşunun basılı olup olmadığı kontrol etmekte, eğer basılı değilse ışık seviyesini sıfırlamakta, basılı ise ışık düzeyini artırmaktadır. Aynı zamanda boşluk tuşuna basıldığı süre, süre ölçer kullanılarak ekrana yazılmaktadır. Süre ölçer değeri 100'e bölünerek ve tam sayıya yuvarlanarak sürenin 1'den başlaması sağlanmıştır. Eğer ışık seviyesi 30 birim olursa ışık düzeyi sıfırlanmaktadır.


**Resim 6.52:** Arduino UNO ile PWM uygulaması

Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için birer 220 ohm dirençle 5 ve 6 numaralı PWM pinine bağlanmıştır.




**Resim 6.53:** Arduino UNO ile PWM uygulaması Breadboard çizimi

## 6.8. Robotlar Alt Başlığı Altında Verilen mBot Komut Blokları

Makeblock tarafından üretilen mBot robot, robot kontrol kartları ve kalkanların (shield) programlanmasında kullanılabilecek bloklar aşağıda iki bölüm halinde açıklanmıştır.

| | |
|---|---|
| mBot Programı | mBot programını çalıştırmak için kullanılır. |
| ileri git hızı 0 yap | Robotun belirtilen hızda (0 ile 255 arası), ileri veya geri gitmesi, sağ veya sola dönmesi için kullanılır. |
| M1 motorunun hızını 0 yap | Belirtilen moturun (M1 veya M2) hızını (0 ile 255 arası) ayarlamak için kullanılır. |
| Kapı1 Kanal1 servo açısını 90 yap | Belirtilen kapıya bağlı, seçilen kanaldaki servo motorun açısını (0, 45, 90, 135 veya 180) derece yapmak için kullanılır. |
| kart ledler hepsi kır 0 yeş 0 mav 0 | Robot kontrol kartı üzerinde bulunan RGB ledlerin renklerini ayarlamak için kullanılır. |
| Kapı1 led hepsi kır 0 yeş 0 mav 0 | Genişleme kapılarına bağlı bulunan RGB ledlerin (4 adet) renklerini ayarlamak için kullanılır. |
| Kapı1 şerit led Kanal2 -- hepsi kır 0 yeş 0 mav 0 | Genişleme kapılarına bağlı bulunan serit RGB ledlerin (4 adet), seçilen kanaldaki renklerini ayarlamak için kullanılır. |
| ses tonunu C4 notasında Yarım vuruş çal | Ses tonunu belirtilen notada, istenen vuruş kadar çalmak için kullanılır. |
| Kapı1 de 0 numaralı yüzü göster | Belirtilen kapıda, belirtilen numaralı yüzü göstermek için kullanılır. |
| Kapı1 de x: 0 y: 0 konumunda Hi harf(ler)ini göster | Belirtilen kapıda, belirtilen x ve y konumunda belirtilen kelimeyi göstermek için kullanılır. |
| saati göster Kapı1 saat: 10 : dakika: 20 | Belirtilen kapıda, belirtilen saati göstermek için kullanılır. |
| Kapı1 de x: 0 y: 0 konumunda çizimi göster | Belirtilen kapıda, belirtilen x ve y konumunda belirtilen çizimi göstermek için kullanılır. |
| Kapı1 7 parçalı displeyde 100 yaz | Belirtilen kapıya bağlı 7 parçalı displeye, belirtilen sayıyı yazdırmak için kullanılır. |

| | |
|---|---|
| Kapı3 ışık algılayıcıyı Aç | Kapı 3 veya 4'te bulunan ışık algılayıcıyı açmak veya kapatmak için kullanılır. |
| Kapı1 kamera perdesini Basıldı yap | Belirtilen kapıya bağlı kamera perdesini basıldı, bırak, odaklan veya kaydır yapmak için kullanılır. |
| Kapı1 mini fan saat yönünde döndür | Belirtilen kapıya bağlı mini fanı saat yönünde, saat yönünün tersinde döndürmek vaya durdurmak için kullanılır. |
| ışık algılayıcıyı kartta ışık sensörü değeri | Robot kontrol kartı üzerinde bulunan veya kapılara bağlı olan ışık sensorunun değeri ile ilgili işlem yapmak için kullanılır. |
| basıldı düğmesi basıldığında | Basıldı düğmesine basıldığında veya serbest bırakıldığında programı çalıştırmak için kullanılır. |
| basıldı düğmesi | Basıldı düğmesine basıldığında veya serbest bırakıldığında işlem yapmak için kullanılır. |

**Tablo 6.15:** Robotlar alt başlığı altında verilen mBot komut blokları 1

| | |
|---|---|
| Kapı3 ultrasonik algılayıcı mesafesi | Belirtilen kapıya bağlı ultrasonik algılayıcı mesafesi ile ilgili işlemler yapmak için kullanılır. |
| Kapı2 çizgi izleyen | Belirtilen kapıya bağlı çizgi izleme algılayıcılarıya ilgili işlemler yapmak için kullanılır. |
| Kapı2 çizgi izleyen solTaraf siyah ise | Belirtilen kapıya bağlı sağ veya sol çizgi izleme algılayıcılarıya ilgili işlemler yapmak için kullanılır. |
| Kapı3 joystick X-ekseni | Belirtilen kapıya bağlı joystick'in x ve y ekseni ile ilgili işlemler yapmak için kullanılır. |
| Kapı3 potansiyometre | Belirtilen kapıya bağlı potansiyometre ile ilgili işlemler yapmak için kullanılır. |
| Kapı3 ses algılayıcı | Belirtilen kapıya bağlı ses algılayıcı ile ilgili işlemler yapmak için kullanılır. |
| Kapı1 limit anahtarı Kanal1 | Belirtilen kapıya bağlı limit anahtarı ile ilgili işlemler yapmak için kullanılır. |
| Kapı3 sıcaklık Kanal1 °C | Belirtilen kapıya bağlı sıcaklık algılayıcı ile ilgili işlemler yapmak için kullanılır. |
| Kapı2 pır hareket algılayıcı | Belirtilen kapıya bağlı pır hareket algılayıcı ile ilgili işlemler yapmak için kullanılır. |
| 3-eksenli jirosjop X-ekseni açısı | Belirtilen kapıya bağlı 3 eksenli jiroskopun x, y ve z ekseniyle ilgili işlemler için kullanılır. |
| Kapı1 nem algılayıcı nem | Belirtilen kapıya bağlı nem ve sıcaklık algılayıcıyla ilgili işlemler için kullanılır. |
| Kapı3 alev algılayıcı | Belirtilen kapıya bağlı alev algılayıcıyla ilgili işlemler için kullanılır. |
| Kapı3 gaz algılayıcı | Belirtilen kapıya bağlı gaz algılayıcıyla ilgili işlemler için kullanılır. |
| Kapı1 pusula | Belirtilen kapıya bağlı pusula algılayıcıyla ilgili işlemler için kullanılır. |
| Kapı1 dokunma algılayıcı | Belirtilen kapıya bağlı dokunma algılayıcıyla ilgili işlemler için kullanılır. |
| Kapı3 buton key1 basıldı | Belirtilen kapıya bağlı butonlarla ilgili işlemlerler için kullanılır. |

| | |
|---|---|
| kızıl ötesi kumandanın A düğmesi basıldı | Kızıl ötesi kumandayla ilgili işlemler yapmak için kullanılır. |
| merhaba mesajını mBot'a gönder | Belirtilen mesajı mBot robota göndermek için kullanılır. |
| mBot iletisi alındı | Alınan mBot iletisi ile ilgili işlemler için kullanılır. |
| süre ölçer | Süre ölçümü için kullanılır. |
| süre ölçeri sıfırla | Süre ölçümünü sıfırlamak için kullanılır. |

**Tablo 6.16:** Robotlar alt başlığı altında verilen mBot blokları 2

### 6.8.1. mBlock ile Arduino ve mBot Uyumlu Robot Kullanımı

Makeblock tarafından üretilen mBot robot, mBot Ranger robot, robot kontrol kartları ve kalkanların (Me Uno Shield) programlanmasında kullanılabilecek bloklar gruplar halinde verilmektedir. Hangi robot ya da kalkan kullanılıyorsa onun mBlock Kartlar sekmesinden seçilmesi gereklidir. Aşağıda Arduino UNO uyumlu robot ve mBot robot için programlama örnekleri verilmiştir.

**Robot Hareketi için DC Motor Kullanımı:** mBot robot Arduino uyumlu olduğu için buradaki bloklar Arduino uyumlu robotların kullanımı için de kullanılabilmektedir. DC motorlar doğrudan kontrol kartına bağlanmazlar. Bu nedenle, uygun DC motor sürücüsü ile istenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) ve uygun kanala (Kanal 1 veya 2) bağlayarak kullanılması gerekmektedir. Yandaki örnekte uyumlu robot 1 sn boyunca ileri, 1 sn boyunca da geri hareketini iki defa tekrar ettikten sonra hızı sıfırlanarak durmaktadır.


**Resim 6.54:** Robot hareketi için DC motor kullanımı örneği

**mBot Robotun Hareketi için DC Motor Kullanımı:** mBot robot üzerinde bulunan M1 ve M2 motorları kullanılarak robotun hareket kullanımı sağlanmaktadır. Yandaki örnekte mBot robot 1 sn boyunca ileri, 1 sn boyunca da geri hareketini iki defa tekrar ettikten sonra motor hızları sıfırlanarak durmaktadır.


**Resim 6.55:** mbot Robotun hareketi için DC motor kullanım örneği

**mBot Robot Üzerindeki RGB LED Kullanımı:** mBot robot üzerinde bulunan RGB LED1 ve RGB LED2 istenilen renkleri üretmek için programlanabilmektedir. Ayarlanması gereken üç parametre vardır: Led seçimi (sağ, sol veya hepsi), kırmızı değer (0 ile 255 arası), mavi değer (0 ile 255 arası), yeşil değer (0 ile 255 arası). Aşağıdaki örnekte RGB LED'lere kırmızı, yeşil ve mavi renk değerlerinin verilmesi gösterilmiştir.


**Resim 6.56:** mbot Robot üzerindeki RGB LED kullanım örneği

**8*16 LED Matris Kullanımı:** Bu parça sekiz âdeti dikeyde, 16 âdeti yatayda olmak üzere toplam 128 adet mavi LED'den oluşmaktadır. Bu parça mBot robot üzerinde kayan yazılar yazmak, algılayıcı değerlerini veya işlem sonuçlarını göstermek için kullanılabilir. Bu amaçla parçanın istenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) bağlayarak kullanılması gerekmektedir. Yandaki örnekte istenilen metnin bu parça üzerinde yazdırılması gösterilmiştir. Önce metin girişi yapılmakta, girilen metinin (yanıt)

uzunluğu tespit edilmekte ve bu değer 6 ile çarpılarak buna "yer" bilgisi eklenmektedir. Her bir karakter yatayda 6 sıra LED kullandığı için 6 ile çarpılmaktadır. Yer bilgisi için "yer" adında bir değişken oluşturulmuş olup, değişkeninin sağdan başlanarak ("yer" 16 olsun bloğu) her harf için -1 azaltılarak yeni konumuna yerleştirilmesi sağlanmıştır. Yanıtın kayarak gösterilmesi için kapı 4'de x:yer y:0 konumunda yanıt harflerini göster bloğu kullanılmıştır.


**Resim 6.57:** 8*16 LED Matris kullanım örneği

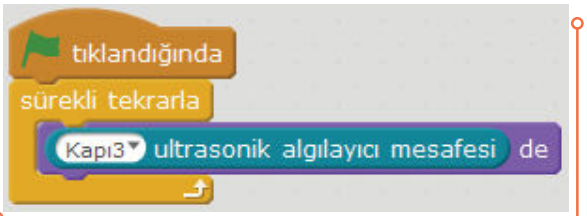
**Servo Motor Kullanımı:** Servo motorlar doğrudan kontrol kartına bağlanmazlar. Bu nedenle, uygun servo motor sürücüsü ile istenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) ve uygun kanala (Kanal 1 veya 2) bağlanarak kullanılması gerekmektedir. Yandaki örnekte servo motor açısının 90 derece yapılması gösterilmiştir.


**Resim 6.58:** Servo motor kullanım örneği

**7 Segment Ekran Kullanımı:** 7 segmentli ekranlar, genellikle hızı, zamanı, algılayıcıların değerini veya puanı göstermek için robot projelerinde kullanılmaktadır. Doğrudan istenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) bağlanabilirler. Yandaki örnekte 7 segmentli ekrana 100 yazılması gösterilmiştir.

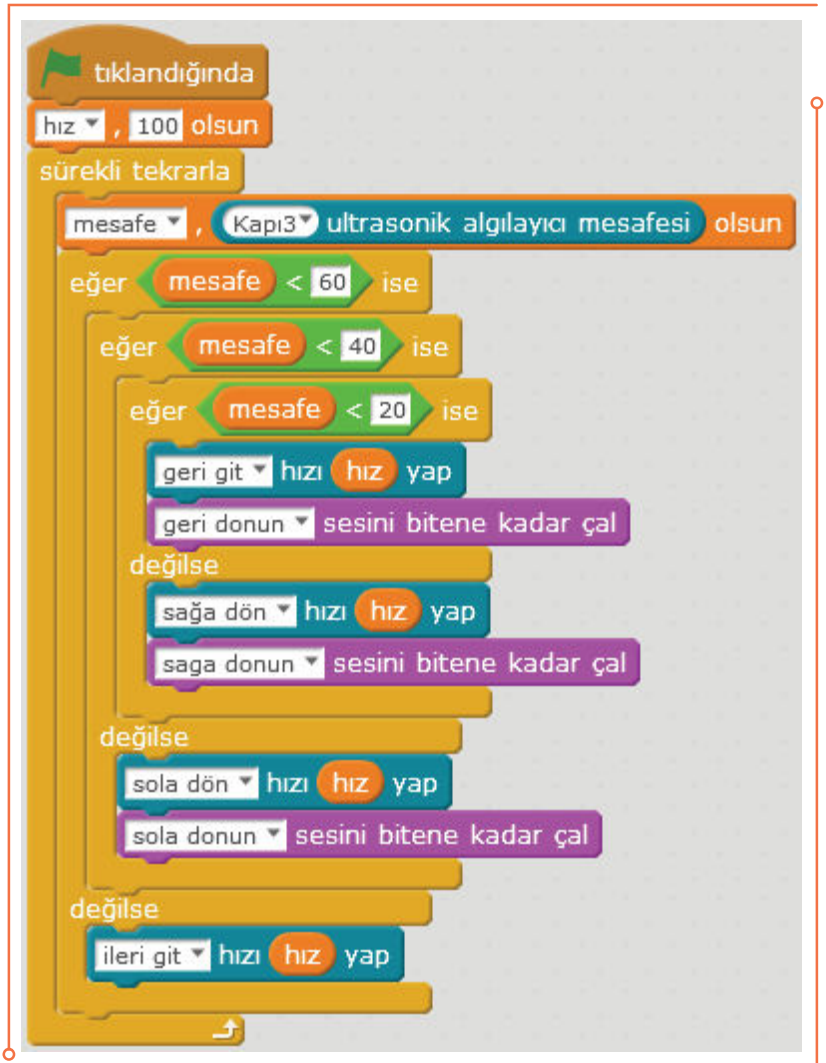

**Resim 6.59:** Servo motor kullanım örneği

**Ultrasonik Algılayıcı Kullanımı:** Ultrasonik algılayıcılar genellikle algılayıcı ile engeller arasındaki mesafeyi ölçmek için kullanılırlar. Direkt olarak (Kapı 1, 2, 3 veya 4) bağlanabilirler. Yandaki örnekte engele olan uzaklığın ölçülmesi gösterilmiştir.




**Resim 6.60:** Ultrasonik algılayıcı ile mesafe ölçme örneği

Aşağıdaki diğer örnekte ultrasonik algılayıcıdan alınan mesafe bilgisi kullanılarak robotun herhangi bir nesneye çarpmadan dolaşması sağlanmıştır. Bu amaçla oluşturulan "hız" değişkeni ile istenilen hız belirlenmekte "mesafe" değişkeni ile ölçülen uzunluğa göre robotun yönlendirme işlemi sağlanmaktadır. Buna göre nesneye 20 cm'den az kalmışsa robot geri dönmekte, 20 ile 40 cm arasındaki mesafeler için sağa dönmekte, 40 ile 60 cm mesafeler için sola dönmekte ve 60 cm'den büyük mesafe varsa ileri gitmektedir. Her hareket eylemini de sesle belirtmektedir. Ses dosyaları önceden kaydedilerek mBlock ortamına aktarılmıştır.




**Resim 6.61:** Ultrasonik algılayıcı kullanarak engele çarpmadan dolaşma uygulaması

**Işık Algılayıcı Kullanımı:** Işık algılayıcı genellikle ortamdaki ışığın yoğunluğunu ölçmek için kullanılır. Direkt olarak karttaki ışık algılayıcı kullanılabildiği gibi (Kapı 3 veya 4) de bağlanabilirler. Yandaki örnekte ortamdaki ışık seviyesinin ölçülmesi gösterilmiştir.


**Resim 6.62:** Işık algılayıcı kullanım örneği

Aşağıdaki diğer örnekte ışık algılayıcı kullanılarak robot kuklanın ışık değerine göre renk değiştirmesi sağlanmıştır. Bu amaçla "Görünüm" blok gurubunda bulunan "… etkisini artır" bloğu alınmış, burada renk seçeneği kullanılmıştır. Işık değeri / 100 oranında artırılmıştır.


**Resim 6.63:** Işık algılayıcı kullanarak renk değiştirme örneği

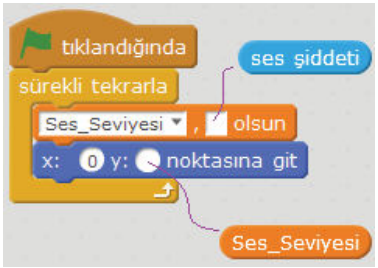
**Ses Algılayıcı Kullanımı:** Ses algılayıcı, ortamdaki ses şiddetini ölçmek için tasarlanmıştır. Direkt olarak (Kapı 3 veya 4) de bağlanabilirler. Yandaki örnekte ortamdaki ışık seviyesinin harici bir ses algılayıcı bağlanarak ölçülmesi gösterilmiştir. İstenirse ses seviyesine bağlı uygulama yapmak için bilgisayarda bulunan mikrofon veya web kamera mikrofonu da kullanılabilir.


**Resim 6.64:** Ses algılayıcı kullanım örneği

Yandaki diğer örnekte bilgisayara bağlı mikrofon kullanılarak ses seviyesine göre ekrandaki kuklayı zıplatan bir uygulama gösterilmiştir.

Kuklanın x ekseninde sıfır noktasında, y ekseninde "Ses_Seviyesi" düzeyinde olması için "Hareket" blok gurubunda bulunan "x:0 y: noktasına git" bloğu kullanılmıştır. "Ses_Seviyesi" değişkeninin işaretlenmesi ile ölçüm sonucunun bilgisayar ekranında gösterilmesi sağlanmıştır.


**Resim 6.65:** Ses seviyesine göre ekrandaki kuklayı zıplatan uygulama




**Resim 6.66:** Ses seviyesine göre ekrandaki kuklayı zıplatan uygulama

Bu amaçla "Ses_Seviyesi" adını taşıyan bir değişken oluşturulmuştur. Bu değişkene "Algılama" blok gurubunda bulunan "ses şiddeti" bloğu eklenerek ses şiddetinin tespiti sağlanmıştır.


**Resim 6.67:** Uygulama için değişken oluşturulması

**Çizgi Algılayıcı Kullanımı:** Çizgi algılayıcılar, robota kalınca çizilmiş çizgileri veya yakında bulunan nesneleri algılama yeteneği kazandırmak, belirli bir yolu takip etmesini sağlamak için kullanılırlar. Direkt olarak (Kapı 1, 2, 3 veya 4) bağlanabilirler. Yandaki örnekle algılayıcıların çizgi üzerinde olup olmadıkları tespit edilmektedir.

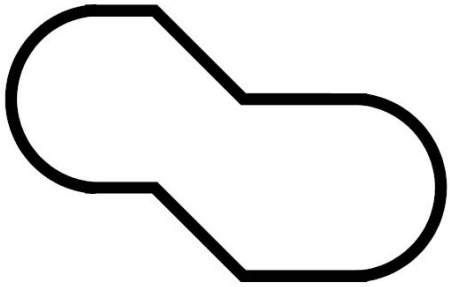

**Resim 6.68:** Çizgi algılayıcı kullanım örneği

Aşağıda verilen örnekte ise robotun çizgi algılayıcıları çizgi üzerinde veya çizgi dışında olup olmadıkları her bir algılayıcı için belirlenip ekrana yazılmaktadır. Kullanılan blokları göstermek için ilk eğer işlemini oluşturan bloklar yerlerine konmadan gösterilmiştir.


**Resim 6.69:** Robotun çizgi algılayıcıları ile çizginin neresinde olduğunu belirleyen uygulama örneği

Aşağıda verilen örnekte ise bir çizgi takip uygulaması bulunmaktadır. Robotun çizgileri takip edebilmesi için önce algılayıcıların her birinin konumu belirlenmekte, buna göre de robotun yönlendirilmesi sağlanmaktadır. Bu amaçla "hiz" ve "alan" adında iki değişken oluşturulmuştur. "hiz" değişkeni ile hız belirlenmekte, "alan" değişkeni ise algılayıcıların konumuna göre hareketin yönünün belirlenmesinde kullanılmaktadır. Algılayıcılar tarafından; eğer her iki algılayıcı çizgi üzerinde ise "0", sol algılayıcı çizgi

üzerinde ise “1”, sağ algılayıcı çizgi üzerinde ise “2” ve her iki algılayıcı çizgi dışında ise “3” değeri üretildiği için yönlendirmeler buna göre yapılmaktadır. Aşağıda örnek bir çizgi takip haritası yer almaktadır.



**Resim 6.70:** Çizgi takip haritası


**Resim 6.71:** Çizgi takip uygulaması

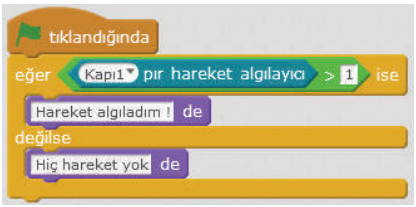
**Potansiyometre Kullanımı:** Potansiyometre potun dönüşüne bağlı olarak sürekli değişen bir değer üreten elektronik parçadır. Döndürülmek suretiyle 0 ile 1023 arasında çıkış değeri üretir. Direkt olarak (Kapı 3 veya 4) bağlanabilirler. Yandaki örnekle portun döndürülmesi ile oluşan potansiyometre değeri ekranda okunabilmektedir.


**Resim 6.72:** Potansiyometre kullanım örneği

**Sıcaklık Algılayıcı Kullanımı:** Sıcaklık algılayıcı uç noktalardaki ve ortamdaki sıcaklığı ya da uzaktaki sıcak bir şeyi algılamak için kullanılmaktadır. İstenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) ve uygun kanala (Kanal 1 veya 2) bağlayarak kullanılması gerekmektedir. Yandaki örnekle ısı değeri ekranda okunabilmektedir.


**Resim 6.73:** Sıcaklık algılayıcı örneği

**PIR Hareket Algılayıcı Kullanımı:** PIR hareket algılayıcı, hayvanların ve insanların hareketini yaklaşık 6-7 metre uzaklıktan algılamak için kullanılmaktadır. Yandaki örnekle hareket algılandığında ekrana "Hareket algıladım !" ifadesi yazılmaktadır.




**Resim 6.74:** PIR Hareket algılayıcı kullanım örneği

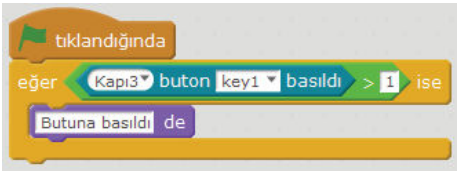
**Dokunma Algılayıcı Kullanımı:** Dokunma algılayıcı, insanların dokunma hareketini algılamak için kullanılmaktadır. Yandaki örnekle dokunma algılandığında ekrana "Dokunmayın bana !" ifadesi yazılmaktadır.


**Resim 6.75:** Dokunma algılayıcı kullanım örneği

**Limit Anahtarı Kullanımı:** Limit anahtarları, belirlenen seviyeye ulaşılıp ulaşılmadığının kullanımı için kullanılmaktadır. Yandaki örnekle limite ulaşıldığında ekrana "Limite ulaşıldı !" ifadesi yazılmaktadır.


**Resim 6.76:** Limit anahtarı kullanım örneği

**Buton Kullanımı:** Buton parçası üzerinde bulunan butonların kullanımı için kullanılmaktadır. Butonlar robot kontrol etmek için, işlev düğmesi veya yön düğmesi olarak çalışabilirler. Yandaki örnekle butona basıldığında ekrana "Butona basıldı!" ifadesi yazılmaktadır.




**Resim 6.77:** Buton kullanım örneği

**Kızılötesi Kumanda Kullanımı:** Kızılötesi kumandanın kullanımı için kullanılmaktadır. Kızılötesi kumanda robot kontrol etmek için, işlev düğmesi veya yön düğmesi olarak da çalışabilirler. Yandaki örnekle kızılötesi kumandanın A düğmesine basıldığında ekrana "Butona basıldı!" ifadesi yazılmaktadır.


**Resim 6.78:** Kızılötesi kumanda kullanım örneği

**Nem ve Sıcaklık Algılayıcı Kullanımı:** Nem ve sıcaklık algılayıcı ortamın sıcaklığı ya da nem durumunu algılamak için kullanılmaktadır. İstenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) bağlayarak kullanılması gerekmektedir. Yandaki örnekle nem veya sıcaklık değeri ekranda okunabilmektedir.


**Resim 6.79:** Nem ve sıcaklık algılayıcı kullanım örneği

**Alev Algılayıcı Kullanımı:** Alev algılayıcı yangın ve ateş durumunu algılamak için kullanılmaktadır. İstenilen bağlantı noktalarına (Kapı 3 veya 4) bağlayarak kullanılması gerekmektedir. Yandaki örnekle alev, ateş algılandığında ekranda okunabilmektedir.


**Resim 6.80:** Alev algılayıcı kullanım örneği

**Gaz Algılayıcı Kullanımı:** Gaz algılayıcı ortamdaki çeşitli gazların (kullanılan türe göre) varlığını algılamak için kullanılmaktadır. İstenilen bağlantı noktalarına (Kapı 3 veya 4) bağlayarak kullanılması gerekmektedir. Yandaki örnekle gaz algılandığında ekranda okunabilmektedir.


**Resim 6.81:** Gaz algılayıcı kullanım örneği

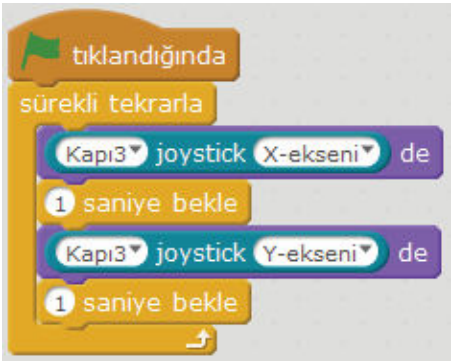
**Pusula Algılayıcı Kullanımı:** Pusula algılayıcı yön belirlemek için robot uygulamalarında kullanılmaktadır. İstenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) bağlayarak kullanılması gerekmektedir. Yandaki örnekle pusula algılandığında ekranda okunabilmektedir.


**Resim 6.82:** Pusula algılayıcı kullanım örneği

**Jiroskop Algılayıcı Kullanımı:** Jiroskop algılayıcı robotun hareket ve duruş algılaması, dengelenmesi için kullanılmaktadır. Genellikle 3 eksenli ivmeölçer ile 3 eksenli açısal hız algılayıcı ve hareket işlemcisi birlikte bulunmaktadır. İstenilen bağlantı noktalarına (Kapı 1, 2, 3 veya 4) bağlayarak kullanılması gerekmektedir. Yandaki örnekle x, y ve z eksen değerleri algılandığında ekranda okunabilmektedir.


**Resim 6.83:** Jiroskop algılayıcı kullanım örneği

**Joystick Kullanımı:** Joystick, yönsel hareketleri analog değerlere (0 ile 1023 arasında) dönüştürebilen elektronik bir parçadır. Robotu X ve Y ekseni yönünde kontrol etmek için kullanılmaktadır. İstenilen bağlantı noktalarına (Kapı 3 veya 4) bağlayarak kullanılması gerekmektedir. Yandaki örnekle x ve y eksenindeki joystick hareketleri ekranda okunabilmektedir.




**Resim 6.84:** Joystick kullanım örneği

## 6.9. Düşünelim / Araştıralım / Uygulayalım

mBlock blok tabanlı robot programlama yazılımını öğrenmek ve bu ortamda programlama yapabilmek için her bölümde verilen örnekleri ve uygulamaları aynı şekilde hazırlayınız. Eğer elinizde eğitsel robot bulunmuyorsa birçok uygulama için sanal robot kullanabileceğinizi unutmayınız. Tercih edilen fiziksel bir robotun olmasıdır. Örnekleri ve uygulamaları parametreleri değiştirerek tekrar tekrar deneyiniz. mBlock sitesinde verilen örnekleri ve İnternette yer alan uygulamaları araştırınız. Bu ortamda hazırlayabileceğiniz özgün uygulamalar düşünerek gerçekleştirmeye çalışınız.

## 6.10. Değerlendirme Soruları

1. **mBlock yüklü bilgisayarla Arduino destekli robot arasında bağlantının oluşturulması için aşağıdaki seçeneklerden hangisi kullanılabilir?**
   a) USB port
   b) Ağ üzerinden bağlantı
   c) Bluetooth modülü ile
   d) 2.4 Ghz Dongle modülü
   e) Hepsi

2. **mBlock yüklü bilgisayarla Arduino destekli robotun güncellenmesi ve programlanması için aşağıdaki seçeneklerden hangisi kullanılabilir?**
   a) USB port
   b) Ağ üzerinden bağlantı
   c) Bluetooth modülü
   d) 2.4 Ghz Dongle modülü
   e) Hepsi

3. **Girilen değerleri alan veya programın çalışmasıyla bazı değerlerin atandığı veri tutucularından oluşan temel programlama yapılarına ne ad verilir?**
   a) Fonksiyon
   b) Liste
   c) Değişken
   d) Dizi
   e) Prosedür

4. **Çok sayıda değişkenle çalışmak için oluşturulmuş temel programlama yapılarına ne ad verilir?**
   a) Fonksiyon
   b) Liste
   c) Değişken
   d) Dizi
   e) Prosedür

5. **Program akışı içinde tekrarlayan ifadelerin her seferinde tekrar tekrar yazılması yerine, bir kere ayrı bir yerde yazılıp tekrarlanan her yerde kullanmak için aşağıdakilerden hangisinin yapılması uygundur?**
   a) Prosedür oluşturmak
   b) Dizi oluşturmak
   c) Liste oluşturmak
   d) Blok oluşturmak
   e) Fonksiyon oluşturmak

**6. Prosedürlere değer taşıyan değişkenlere ne adı verilir?**

a) Blok
b) Fonksiyon
c) Dizi
d) Liste
e) Parametre

**7. Sanal (kütüphaneden kukla, figür kullanılarak veya oluşturarak) veya fiziksel bir robot için oluşturulan programların ve diğer uygulamaların çalıştırılması için hangi alt başlık altında verilen blok gurupları kullanılmaktadır?**

a) Hareket alt başlığı altında verilen blok gurupları
b) Görünüm alt başlığı altında verilen blok gurupları
c) Olaylar alt başlığı altında verilen blok gurupları
d) Kontrol alt başlığı altında verilen blok gurupları
e) İşlemler alt başlığı altında verilen blok gurupları

**8. Arduino blokları ile diğer blokların Arduino uygulamalarında birlikte kullanılabilmesi için mBlock ortamında aşağıdakilerden hangisinin yapılması gereklidir?**

a) Aygıt Yazılımı Güncellemesi yapılmalıdır
b) Arduino Programı blokunun üzerine tıklanmalıdır
c) Arduino'ya Yükle seçeneğini üzerine tıklanmalıdır
d) Varsayılan Program sıfırlanmalıdır
e) Arduino sürücüsü yüklenmelidir

**9. Arduino uygulamalarında elektronik bileşenlerin lehim yapmadan kullanılabilmesi için aşağıdakilerden hangisi kullanılmalıdır?**

a) Soket
b) Konnektör
c) Delikli kart
d) Breadboard
e) Pertinaks

**10. Aşağıdakilerden hangisi blok tabanlı robot programlama yazılımlarından ve ortamlarından biri değildir?**

a) miniBlog
b) ModKit
c) Ardublock
d) Snap4Arduino
e) Srach for Arduino (S4A)

# 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

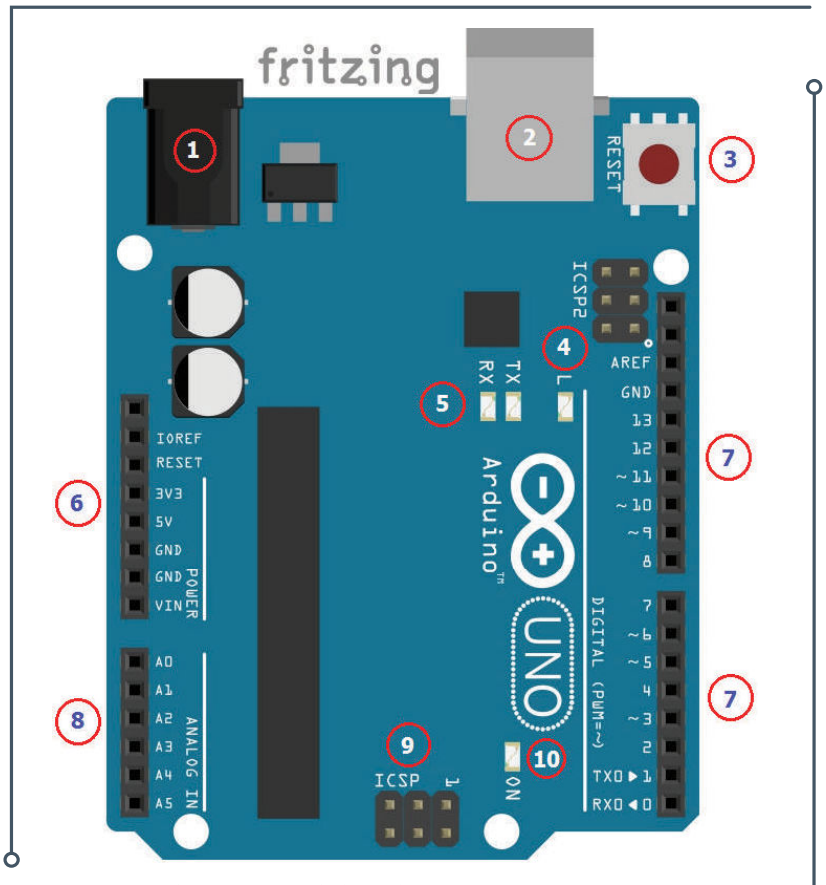
Arduino, ileri derecede elektronik ve mikrodenetleyici bilgisi gerektirmeden çok çeşitli projelerin uygulanabileceği açık kaynaklı, donanımında Atmel firması tarafından üretilen AVR mikrodenetleyici içeren bir elektronik geliştirme platformudur. Kolaylıkla analog ve dijital sinyaller alınarak işlenebilmektedir. 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.



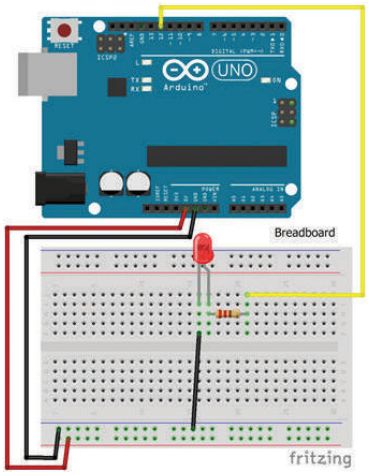

**Resim 7.2:** Arduino UNO bileşenleri

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.

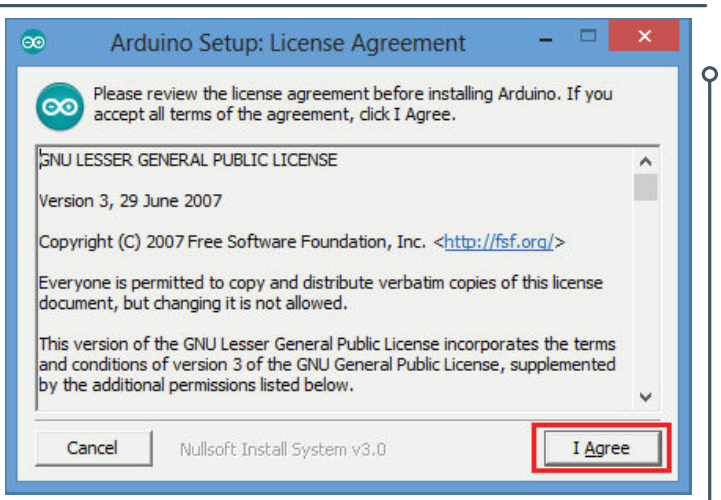
Arduino yazılımı bir tümleşik geliştirme ortamı (IDE) ve mikrodenetleyici ve elektronik konusunda detaylı bilgi sahibi olmayı gerektirmeden herkesin programlama yapabilmesini sağlayan kütüphanelerden oluşmaktadır. Arduino kütüphaneleri, geliştirme ortamı ile birlikte gelmekte ve "libraries" klasörünün 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.

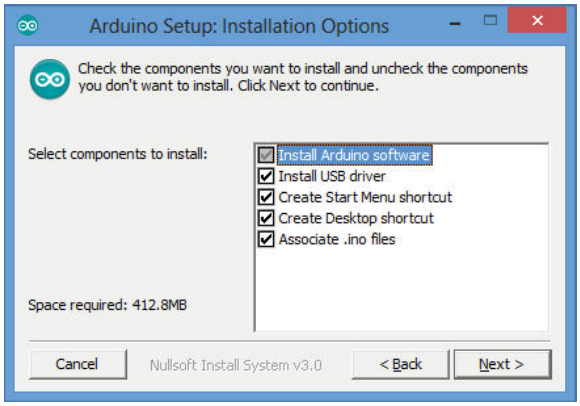
İlk aşamada lisans sözleşmesi onaylanmalıdır.


**Resim 7.5:** Arduino UNO indirme ekranı



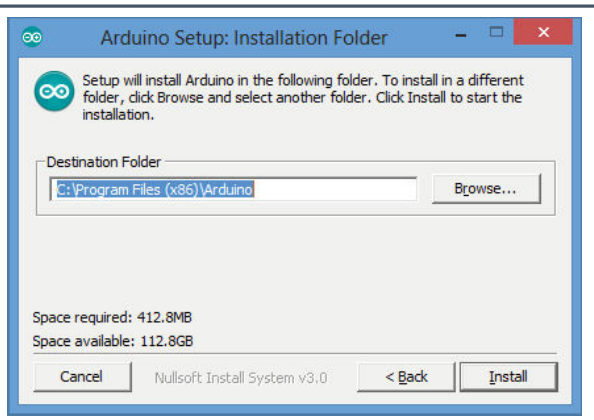

**Resim 7.6:** Arduino IDE kurulum aşaması

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.

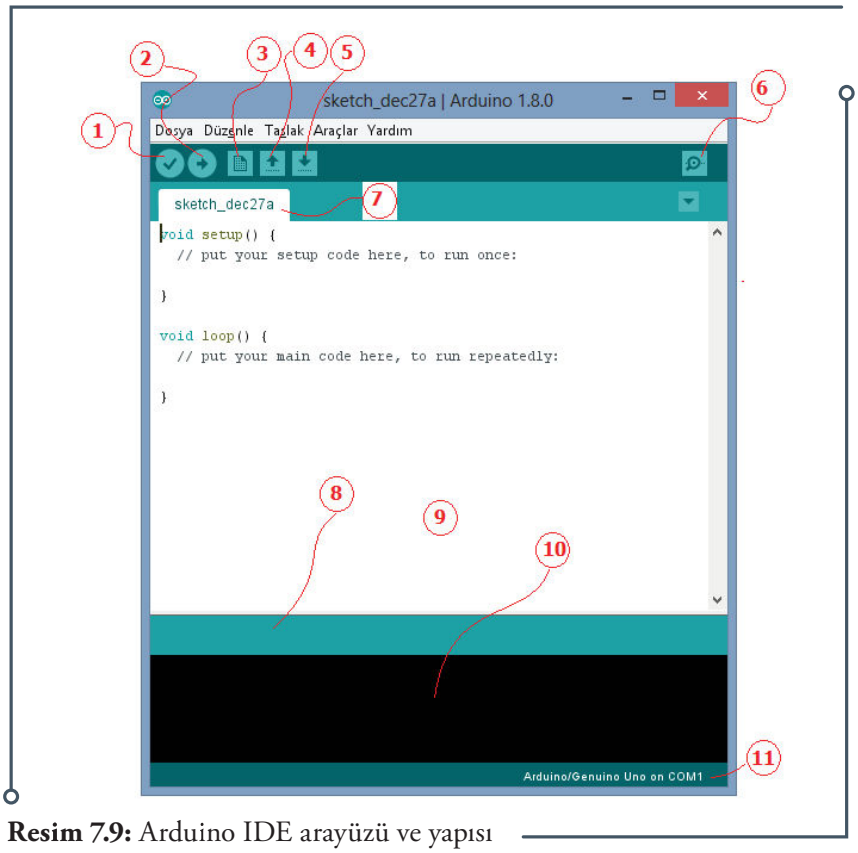



**Resim 7.8:** Arduino IDE kurulum klasörü

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

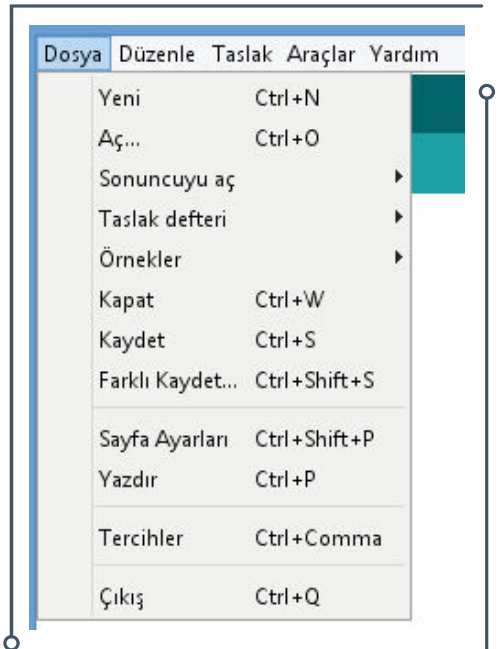
1. **Kontrol Et:** Yazılan kodları derler ve hataları bulur.
2. **Yükle:** Yazılan programı Arduino kartına yükler.
3. **Yeni:** Yeni çalışma sayfası açar.
4. **Aç:** Kayıtlı bir programı açar.
5. **Kaydet:** Yazılan programı kaydeder.
6. **Seri Port Ekranı:** Arduino ile seri iletişim yaparak ekran 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.



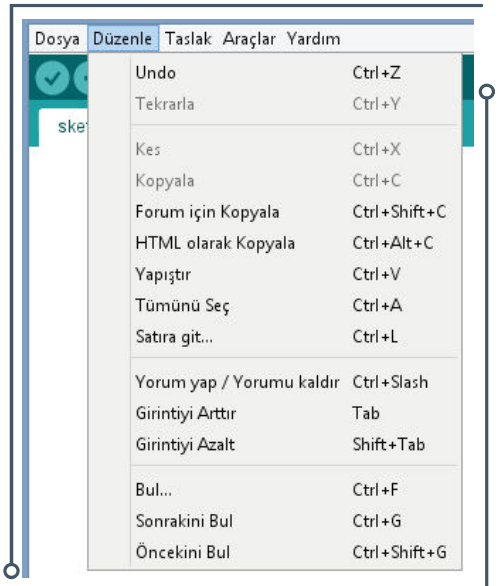

**Resim 7.9:** Arduino IDE arayüzü ve yapısı

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.



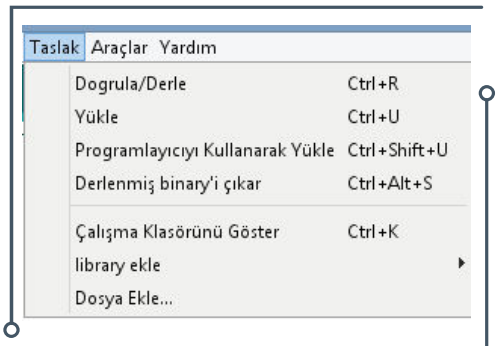

**Resim 7.10:** Arduino IDE dosya menüsü

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



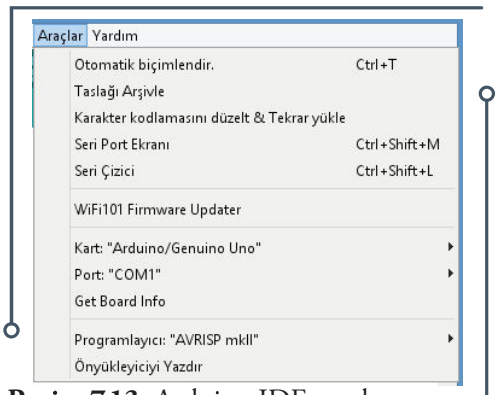

**Resim 7.11:** Arduino IDE düzenle menüsü

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

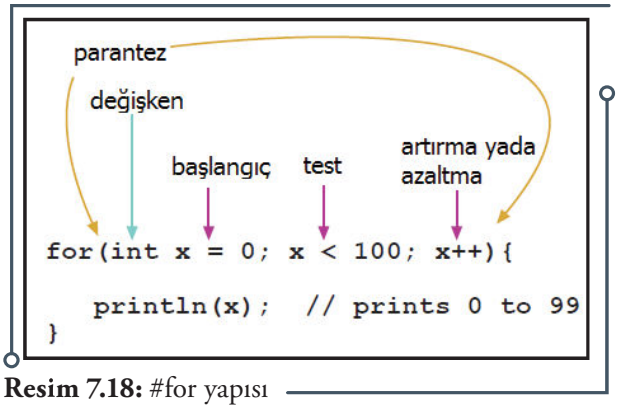
**Resim 7.17:** #if /else örneği

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 oku1= digitalRead(1);
int oku2= digitalRead(2);
if (oku1 == 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 oku1= digitalRead(1);
int oku2= digitalRead(2);
if (oku1 == 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.

### noTone()

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

```
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

### 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.

```
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

## 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

**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<br>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

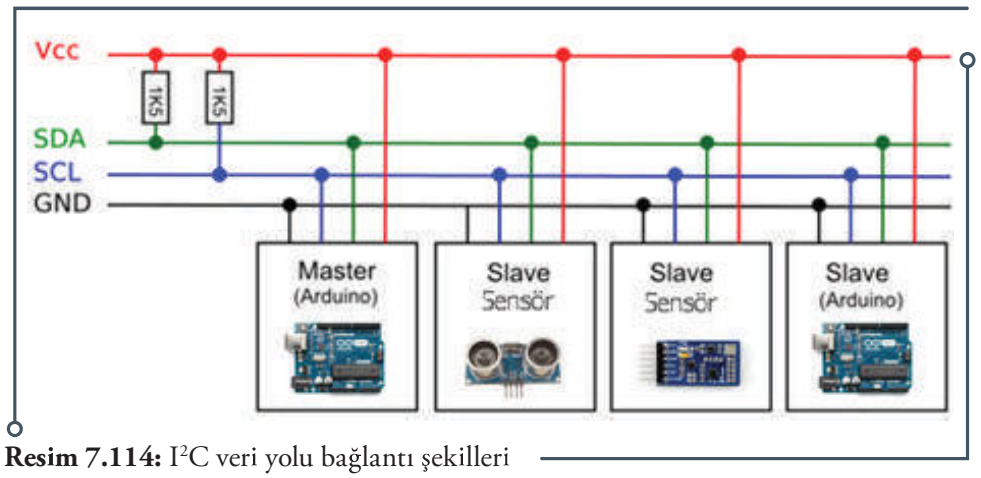
I²C (Inter-Integrated Circuit), oldukça hızlı veri aktarımına olanak tanıyan seri haberleşme türlerindendir. Bir arada çalışan, belirli aralıklarla birbiriyle haberleşen, çeşitli çevresel cihazların çok az harici donanım gereksinimiyle haberleşmelerini sağlar. Uzun mesafeli haberleşmelerde tercih edilmez. 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.

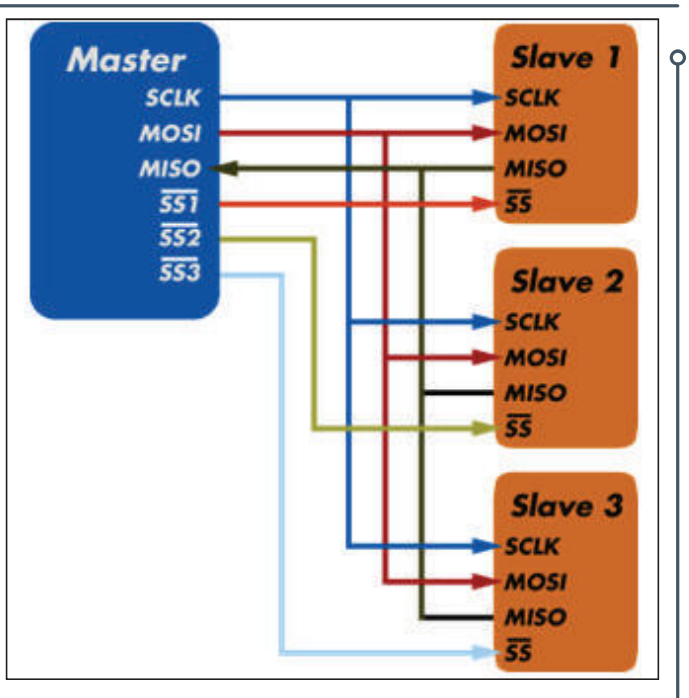
```
#include "I2Cdev.h"  // I2C kütüphanesi ekleniyor
#include "MPU6050.h" // Mpu6050 kütüphanesi ekleniyor
#include "Wire.h"    // Seri bağlantı kütüphanesi ekleniyor
MPU6050 accelgyro;   // Mpu6050 sensör tanımlanıyor
int16_t ax, ay, az;  // İvmeölçer tanımlanıyor
int16_t gx, gy, gz;  // Gyro sensör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

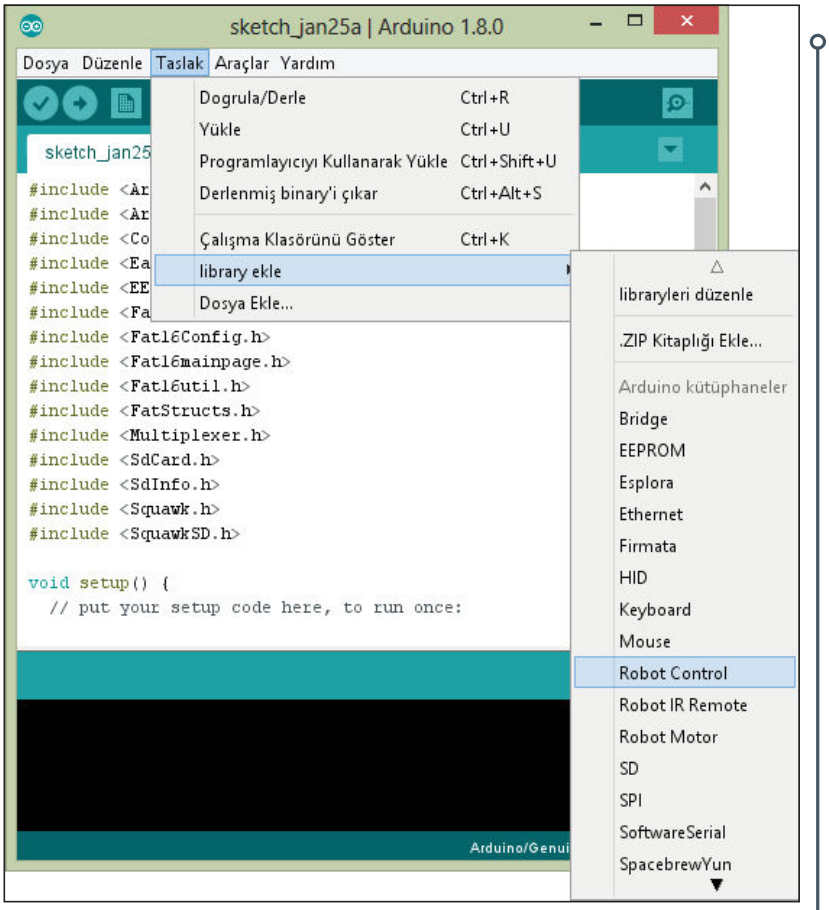
### Slave Kodu

```
#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.

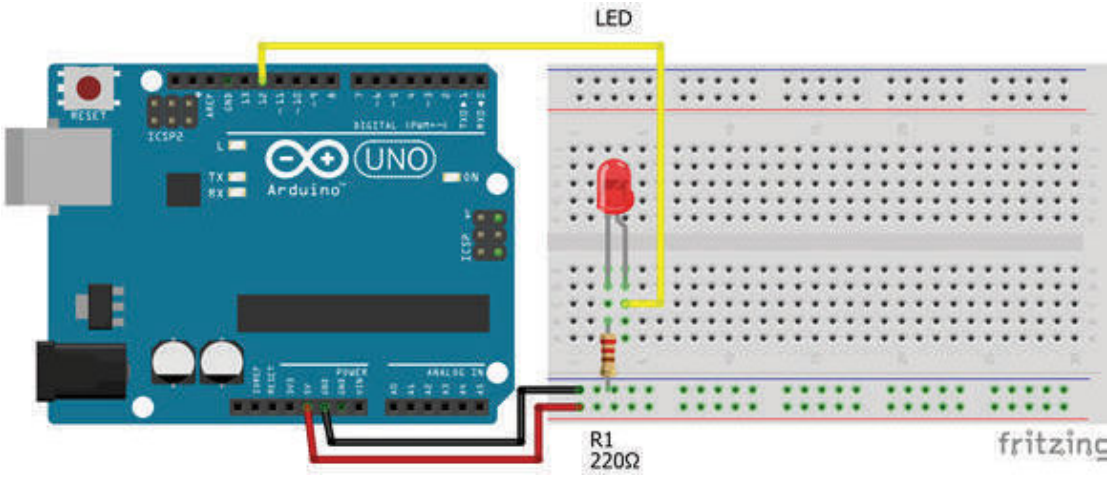
Eğer uygulamalarda kullanılan algılayıcıların arduino ile haberleşmesinde kütüphane kullanılacaksa veya bu zorunluysa uygun kütüphanenin çalışmaya eklenmesi gerektiği unutulmamalıdır. Zira eklenen kütüphane dosyaları algılayıcıdan gelen farklı yapıdaki bilginin okunması görevini üstlenerek kullanımı 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

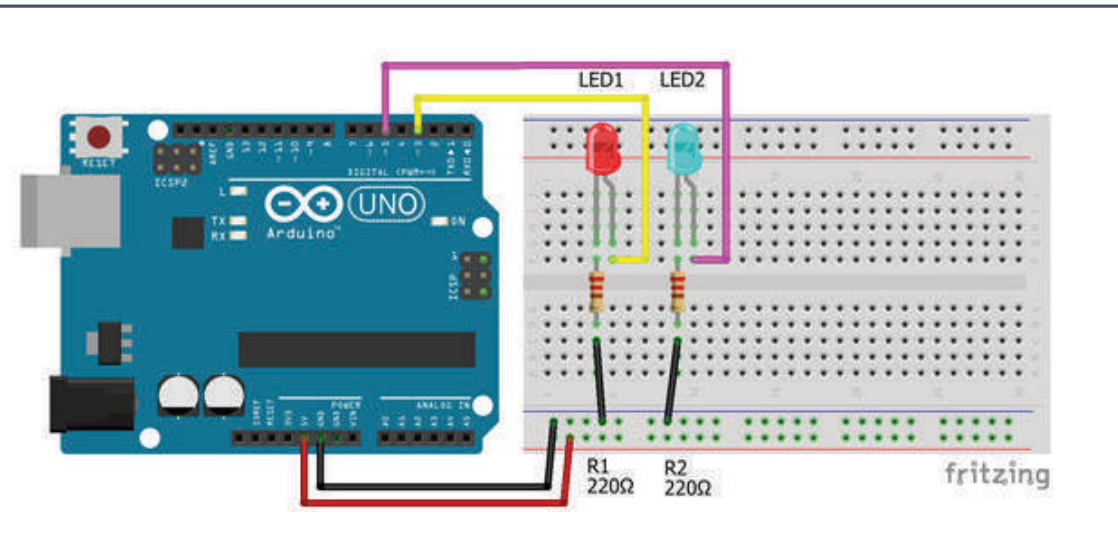
Arduino'nun 12 numaralı dijital pinlerine bağlı bir LED'in 1 sn aralıklarla yanıp sönmesini sağlayacak bir uygulama hazırlayınız. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir. LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada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

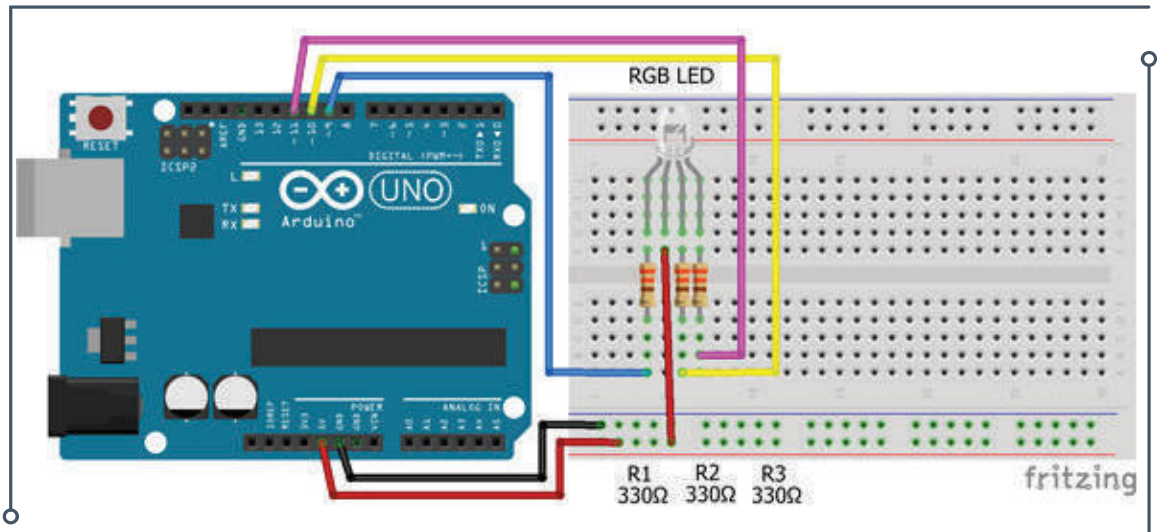



**Resim 7.121:** PWM uygulaması breadboard çizimi

### 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

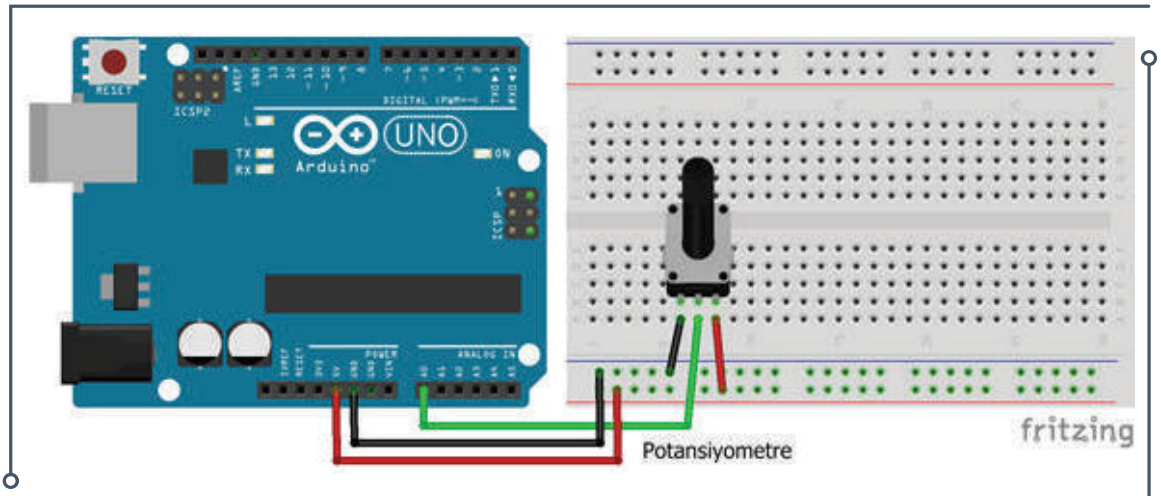
sırayla yanması sağlayacak bir uygulama hazırlayınız. Bağlantılar aşağıdaki breadboard çizimi üzerinde gösterilmiştir RGB LED'ler düşük güçle çalıştıkları ve zarar görmemeleri için uygun bir dirençle bağlanmalıdır. Örnek uygulamada 3 adet 330 Ω direnç kullanılmıştır.




**Resim 7.122:** PWM uygulaması breadboard çizimi

### 7.14.4. Potansiyometre Uygulaması

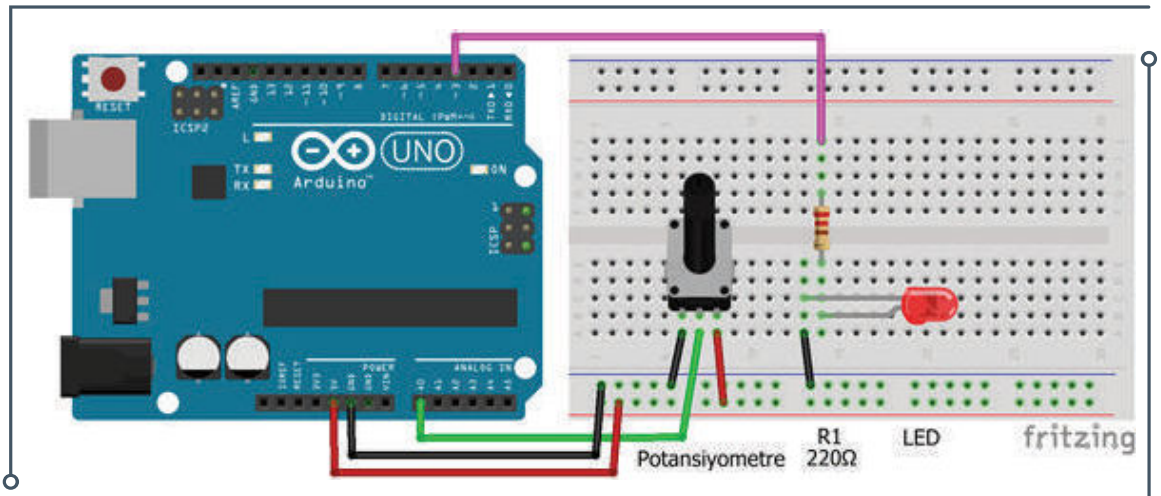
Arduino'nun A1 numaralı analog pinine bağlı bir potansiyometre kullanılarak değer okuma uygulaması hazırlayınız. Potansiyometrenin hareketi ile değişen direnç değerleri Arduino IDE seri port ekranında okunmalıdır. Potansiyometrenin Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir.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




**Resim 7.123:** Potansiyometre uygulaması breadboard çizimi

### 7.14.5. Potansiyometre ile PWM LED Uygulaması

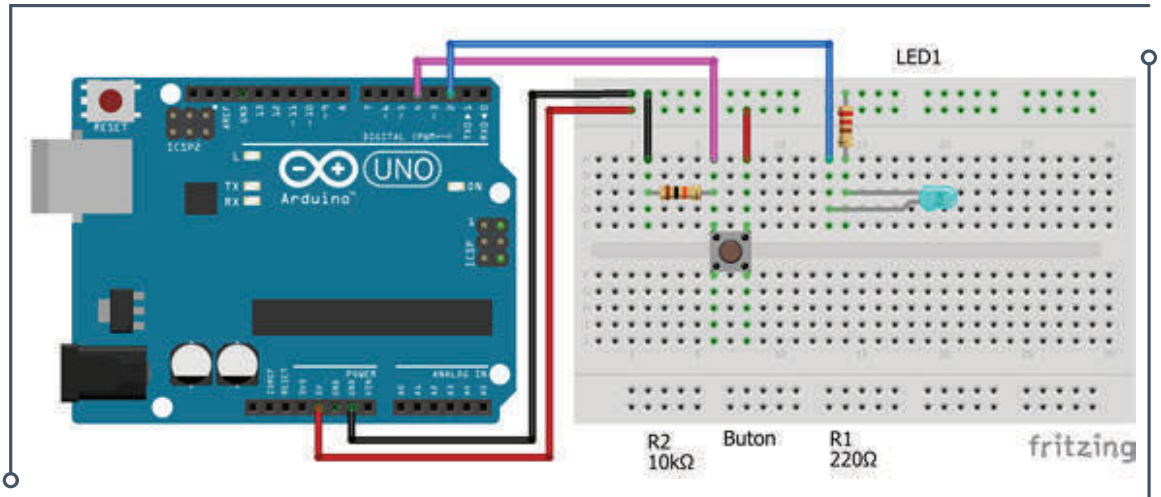
Arduino'nun A0 numaralı analog pinine bağlı bir potansiyometre kullanılarak, Arduino'nun 3 numaralı PWM pinine bağlı bir LED'in ışık seviyesinin kontrolülü sağlayacak bir uygulama hazırlayınız. Potansiyometrenin hareketi ile ışığın parlaklığı ayarlanabilmelidir. Potansiyometre ve LED'in Arduino ile bağlantısı 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.

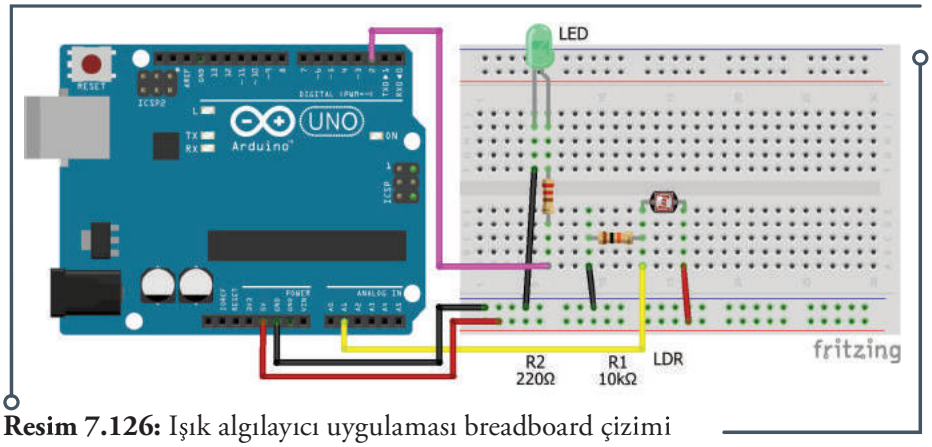



**Resim 7.125:** Buton uygulaması breadboard çizimi

### 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.

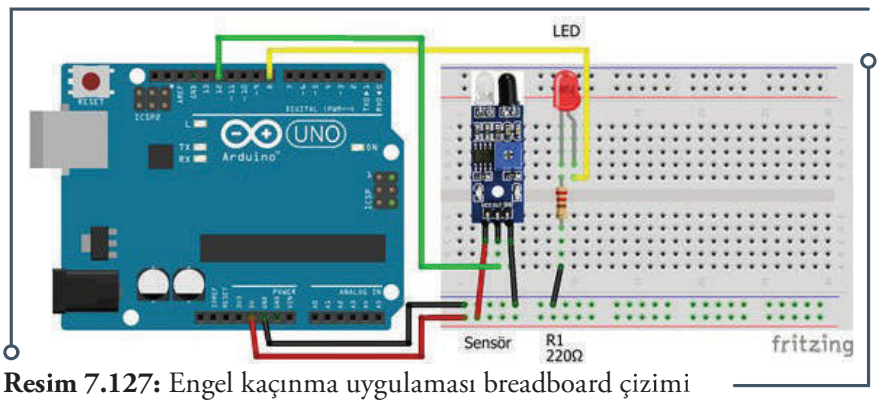
Aynı devre üzerinde yapacağınız ikinci uygulamada ise LED'in karanlıkta yanmasını aydınlıkta sönmesini sağlayınız. LDR algılayıcının algıladığı ışık şiddetini belirleyerek, okunan değer sizin belirleyeceğiniz değerden küçük olunca LED'in yanmasını değilse (büyükse) sönmesini sağlayınız. Işık algılayıcının 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

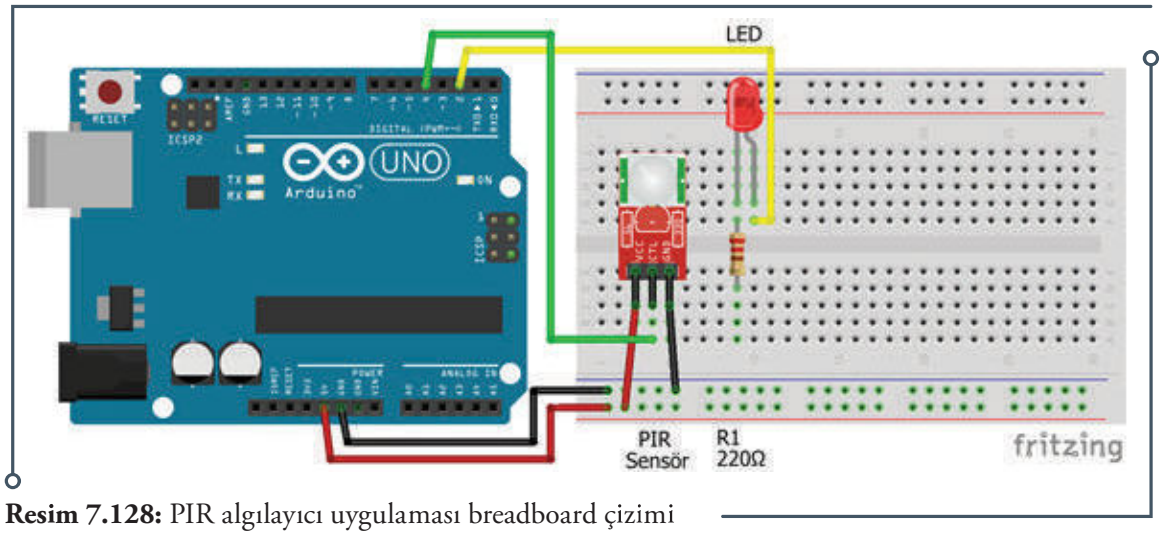
Sinyal çıkışı Arduino'nun 12 numaralı pinine bağlı bir engel kaçınma algılayıcısı kullanılarak engeli algıladığı zaman 4 numaralı dijital pine bağlı LED'i yakan bir uygulama hazırlayınız. Engel kaçınma algılayıcısının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler 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

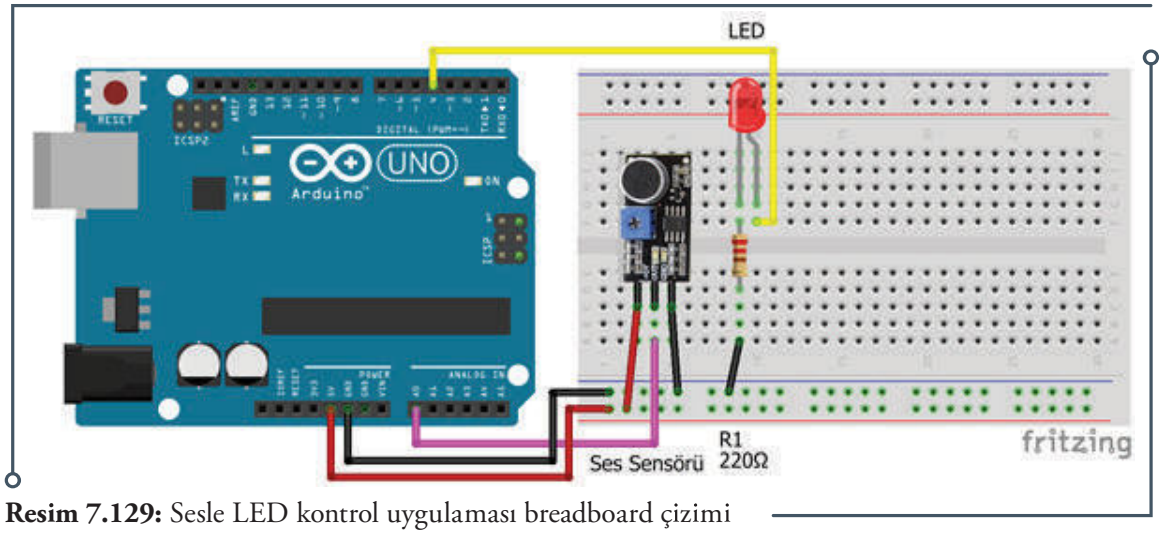
Sinyal çıkış pini Arduino'nun 2 numaralı dijital pinine bağlı bir PIR algılayıcısı kullanarak canlı algıladığı zaman 4 numaralı dijital pine bağlı LED'i yakan bir uygulama hazırlayınız. PIR algılayıcının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. LED'ler düşük güçle 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

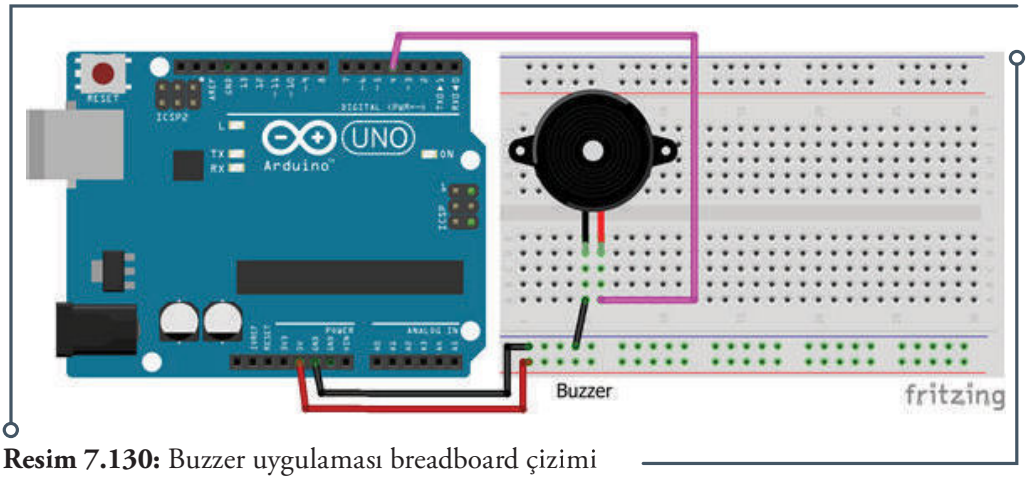
Sınyal çıkışı Arduino'nun A0 numaralı analog pinine bağlı bir ses algılayıcısı kullanarak ses seviyesine göre 4 numaralı dijital pine bağlı LED'i yakıp söndüren ses ile LED kontrol uygulaması hazırlayınız. Ses algılayıcısının Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmektedir. 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

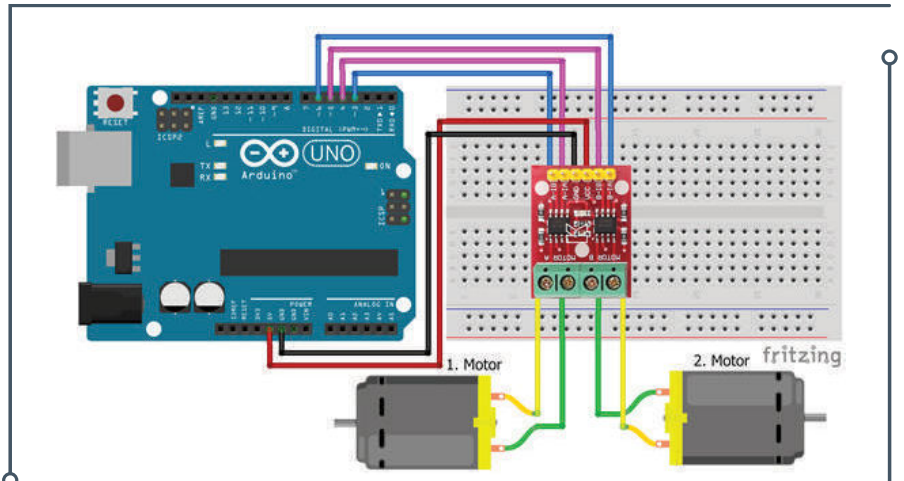
Arduino'nun 4 numaralı dijital pinine bağlı bir buzzer kullanılarak 2 farklı tonla alarm sesi üretmeyi sağlayacak bir uygulama hazırlayınız. Buzzer'in Arduino ile bağlantısı aşağıdaki breadboard çizimi üzerinde gösterilmiştir. Örnek uygulamada standart bir buzzer kullanılmıştır.




**Resim 7.130:** Buzzer uygulaması breadboard çizimi

### 7.14.12. DC Motor Kontrol Uygulaması

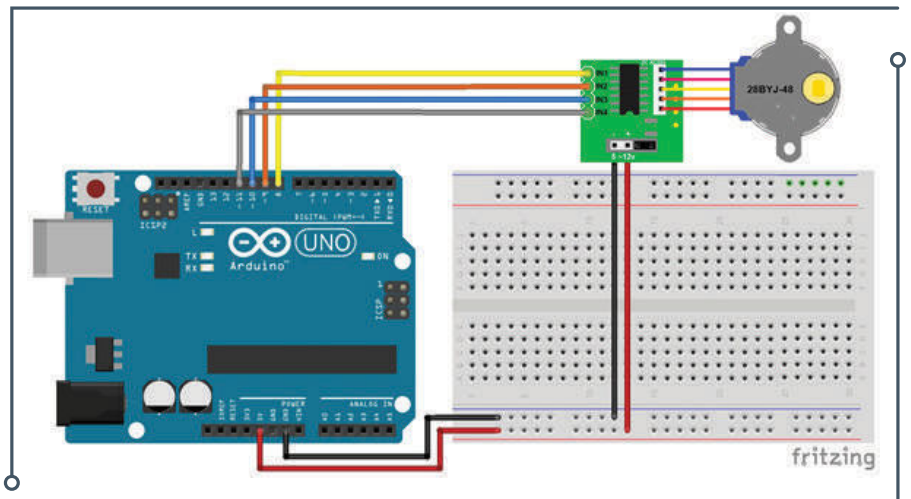
Motor sürücüye bağlı bir çift motoru ileri, geri, sağa ve sola hareket ettirip durduran bir motor kontrol uygulaması hazırlayınız. Bu amaçla farklı motor sürücü kartları kullanılabilir. Uygun voltajda (en fazla 6 volt) herhangi bir dc motor kullanılabilir. Daha güçlü motor kullanılacaksa Arduino UNO'nun 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

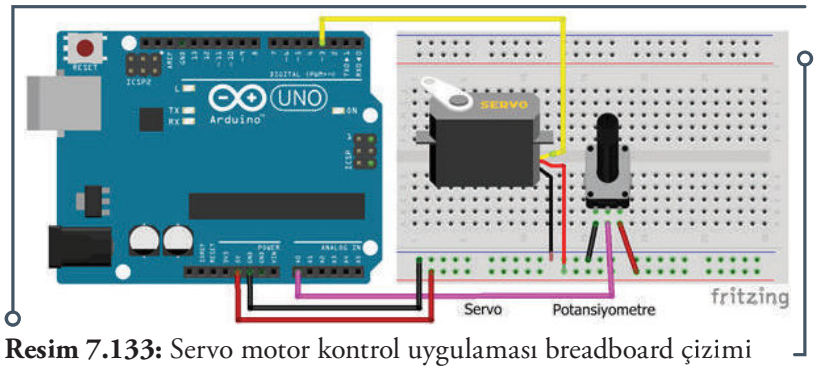
Adım motor sürücüye bağlı bir adım motorun; saat yönünde ve saat yönünün tersinde, verilecek olan adım sayısı kadar adım atmasını sağlayacak adım motor kontrol uygulaması hazırlayınız. Bu amaç için hazırlanan örnekte 28 BYJ-48 redüktörlü adım motor ve ULN2003A adım motor sürücü kartı kullanılmıştır. 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

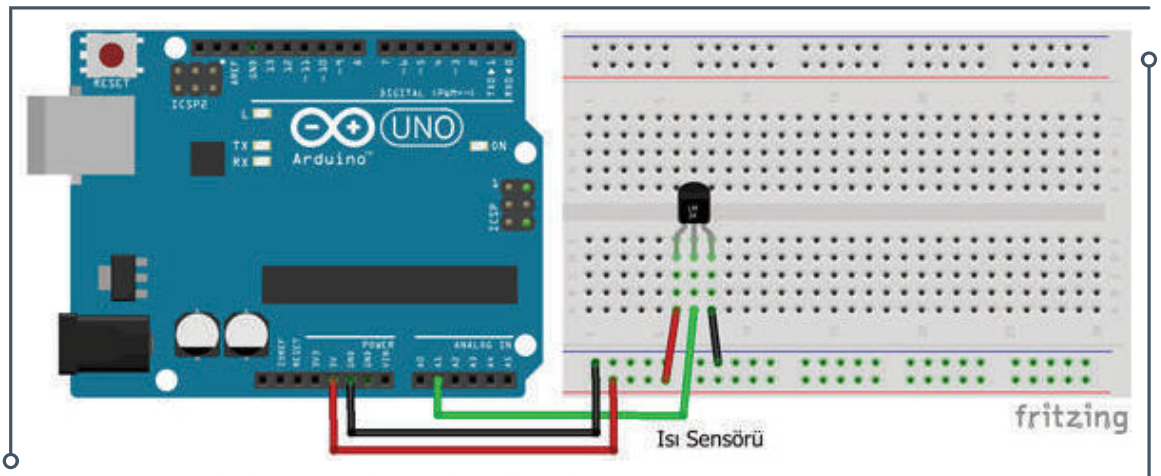
Arduino'nun A0 numaralı analog pinine bağlı bir potansiyometre kullanılarak 0 ile 180 derece arasında hareket eden bir servo motor kontrol uygulaması hazırlayınız. Potansiyometrenin değerini değiştirerek servo motorun 0 ile 180 derece arasında pozisyon alması gerekmektedir. Potansiyometreden okunan analog 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

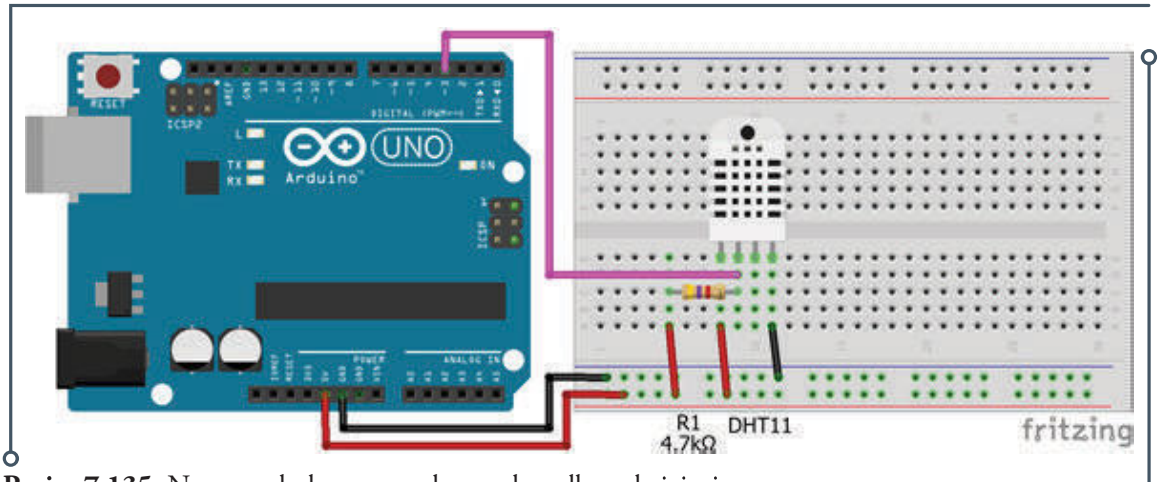
Sensör ucu Arduino'nun A1 numaralı analog pinine bağlı bir ısı algılayıcısı kullanılarak ortam ısısını ölçmeyi sağlayacak bir uygulama hazırlayınız. Ortam ısısı bilgisi Arduino IDE seri port ekranından okunmalıdır. Bu amaç için hazırlanan örnekte LM34 ısı algılayıcısı kullanılmıştır. Isı algılayıcısının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

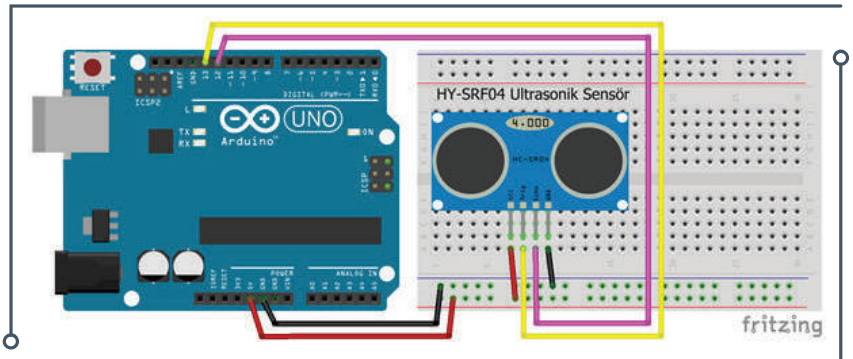



**Resim 7.135:** Nem-ısı algılayıcı uygulaması breadboard çizimi

### 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.

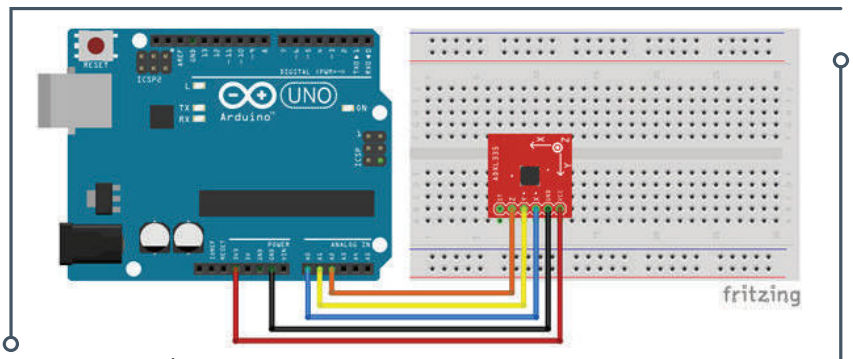
Aynı devre üzerinde yapacağınız ikinci mesafe ölçme uygulamasında ise aynı ultrasonik algılayıcının tetik (trig) ve okuma (echo) pini birleştirilerek, Arduino'nun 12 numaralı dijital pinine bağlanmalıdır. Bunun dışındaki bağlantılar aynı şekilde kalmalıdır. Uygulamada mesafe bilgisi Arduino IDE seri 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

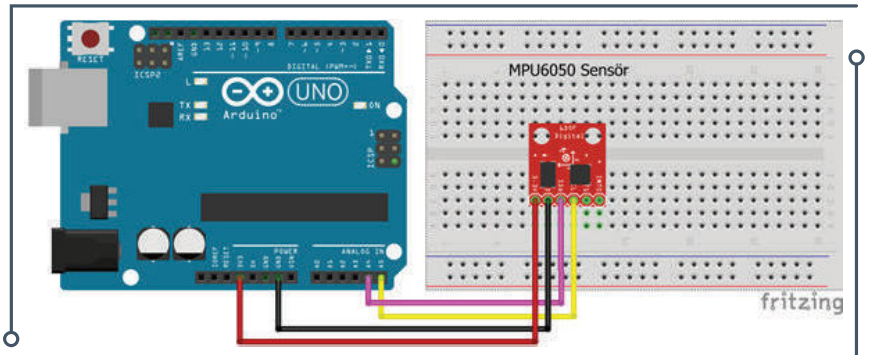
İvme ölçerin X ekseni sınyal çıkışı Arduino'nun A0, Y ekseni sınyal çıkışı Arduino'nun A1 ve Z ekseni sınyal çıkışı Arduino'nun A2 numaralı analog pinlerine bağlı ivmeölçer algılayıcısı kullanarak 3 eksenli ivme ölçme uygulaması hazırlayınız. Bu amaçla farklı ivmeölçer agılayıcıları kullanılabilir. Bu 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

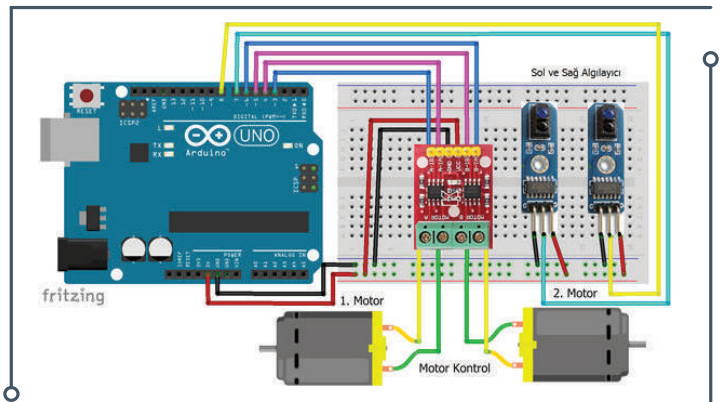
Sinyal çıkışları Arduino'nun A4 (SDA) ve A5 (SCL) numaralı analog pinlerine bağlı gyro ve ivmeölçer algılayıcısı kullanarak bir açıölçer uygulaması hazırlayınız. Bu amaçla farklı gyro algılayıcılar kullanılabilir. Bu amaç için hazırlanan örnekte MPU-6050 algılayıcı kartı kullanılmıştır. Bu üzerinde 3 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].