Bu kitaba sığmayan daha neler var!

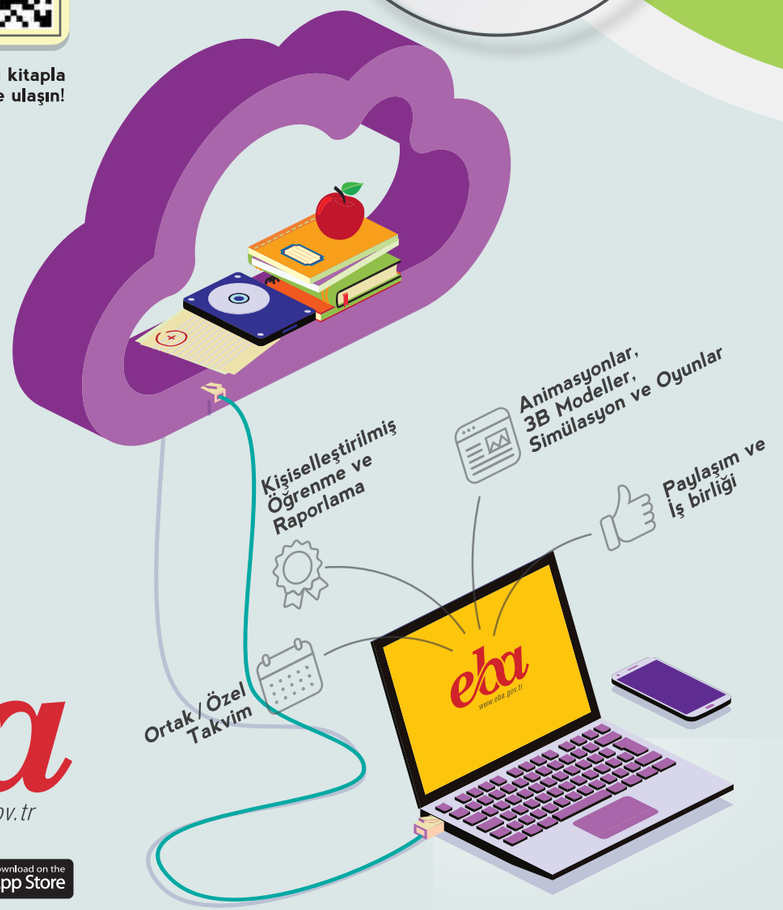
Karekodu okutun, bu kitapla ilgili EBA içeriklerine ulaşın!

ÖDS

ÖĞRENCİ/ÖĞRETMEN DESTEK SİSTEMİ

https://ods.eba.gov.tr

- Konu Anlatımlı Ders Videoları
- Soru Çözüm Videoları
- Ders Anlatım Videoları
- Çoktan Seçmeli Sorular



eba
www.eba.gov.tr

Google play | Download on the App Store

BU DERS KİTABI MİLLÎ EĞİTİM BAKANLIĞINCA ÜCRETSİZ OLARAK VERİLMİŞTİR.
PARA İLE SATILAMAZ.

ISBN: 978-975-11-6958-7

Bandrol Uygulamasına İlişkin Usul ve Esaslar Hakkında Yönetmelik'in 5'inci Maddesinin İkinci Fıkrası Çerçevesinde Bandrol Taşıması Zorunlu Değildir.

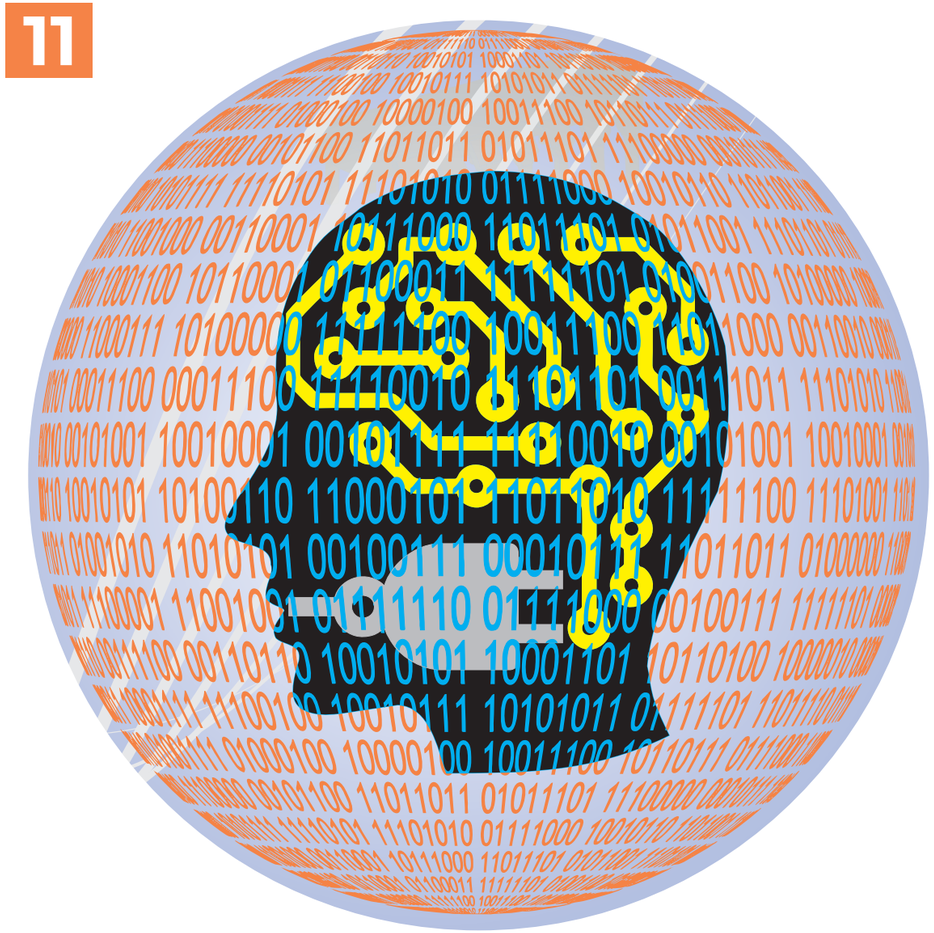
ELEKTRİK-ELEKTRONİK TEKNOLOJİSİ ALANI MİKRODENETLEYİCİ İLE PROGRAMLAMA 11 DERS MATERYALİ

ELEKTRİK-ELEKTRONİK TEKNOLOJİSİ ALANI

MİKRODENETLEYİCİ İLE PROGRAMLAMA

DERS MATERYALİ

11



MESLEKİ VE TEKNİK ANADOLU LİSESİ

MESLEKİ VE TEKNİK ANADOLU LİSESİ

ELEKTRİK-ELEKTRONİK TEKNOLOJİSİ ALANI

MİKRODENETLEYİCİ İLE PROGRAMLAMA

11

DERS MATERYALİ

Yazarlar

Ahmet Zeki AKKAYA
Bekir İŞENGER
Erhan UYTUN
Fatih AKÇAÖZOĞLU
Fatih ÇANKAYA
İsmail SÖNMEZ
Murat ŞAŞAL
Nazım YILDIZ

MİLLÎ EĞİTİM BAKANLIĞI YAYINLARI..... 8362
DERS KİTAPLARI DİZİSİ 2254

HAZIRLAYANLAR

Dil Uzmanı
Jülide ALTAY

Rehberlik Uzmanı
Feyza SÜNBÜL

Ölçme Değerlendirme Uzmanı
Neslihan KOSER

Görsel Tasarım Uzmanı
Firdevs ŞIK

ISBN: 978-975-11-6958-7

Millî Eğitim Bakanlığının 24.12.2020 gün ve 18433886 sayılı oluru ile Mesleki ve Teknik Eğitim Genel Müdürlüğünce ders materyali olarak hazırlanmıştır.

İSTİKLÂL MARŞI

Korkma, sönmez bu şafaklarda yüzen al sancak;
Sönmeden yurdumun üstünde tüten en son ocak.
O benim milletimin yıldızıdır, parlayacak;
O benimdir, o benim milletimindir ancak.

Çatma, kurban olayım, çehreni ey nazlı hilâl!
Kahraman ırkıma bir gül! Ne bu şiddet, bu celâl?
Sana olmaz dökülen kanlarımız sonra helâl.
Hakkıdır Hakk'a tapan milletimin istiklâl.

Ben ezelden beridir hür yaşadım, hür yaşarım.
Hangi çılgın bana zincir vuracakmış? Şaşarım!
Kükremiş sel gibiyim, bendimi çiğner, aşarım.
Yırtarım dağları, enginlere sığmam, taşarım.

Garbın âfâkını sarmışsa çelik zırhlı duvar,
Benim iman dolu göğsüm gibi serhaddim var.
Ulusun, korkma! Nasıl böyle bir imanı boğar,
Medeniyyet dediğin tek dişi kalmış canavar?

Arkadaş, yurduma alçakları uğratma sakın;
Siper et gövdeni, dursun bu hayâsızca akın.
Doğacaktır sana va'dettiği günler Hakk'ın;
Kim bilir, belki yarın, belki yarından da yakın.

Bastığın yerleri toprak diyerek geçme, tanı:
Düşün altındaki binlerce kefensiz yatanı.
Sen şehit oğlusun, incitme, yazıktır, atanı:
Verme, dünyaları alsan da bu cennet vatanı.

Kim bu cennet vatanın uğruna olmaz ki feda?
Şüheda fışkıracak toprağı sıksan, şüheda!
Cânı, cânânı, bütün varımı alsın da Huda,
Etmesin tek vatanımdan beni dünyada cüda.

Ruhumun senden İlâhî, şudur ancak emeli:
Değmesin mabedimin göğsüne nâmahrem eli.
Bu ezanlar -ki şehadetleri dinin temeli-
Ebedî yurdumun üstünde benim inlemeli.

O zaman vecd ile bin secde eder -varsa- taşım,
Her cerîhamdan İlâhî, boşanıp kanlı yaşım,
Fışkırır ruh-ı mücerret gibi yerden na'şım;
O zaman yükselerek arşa değer belki başım.

Dalgalan sen de şafaklar gibi ey şanlı hilâl!
Olsun artık dökülen kanlarımın hepsi helâl.
Ebediyyen sana yok, ırkıma yok izmihlâl;
Hakkıdır hür yaşamış bayrağımın hürriyyet;
Hakkıdır Hakk'a tapan milletimin istiklâl!

Mehmet Âkif Ersoy

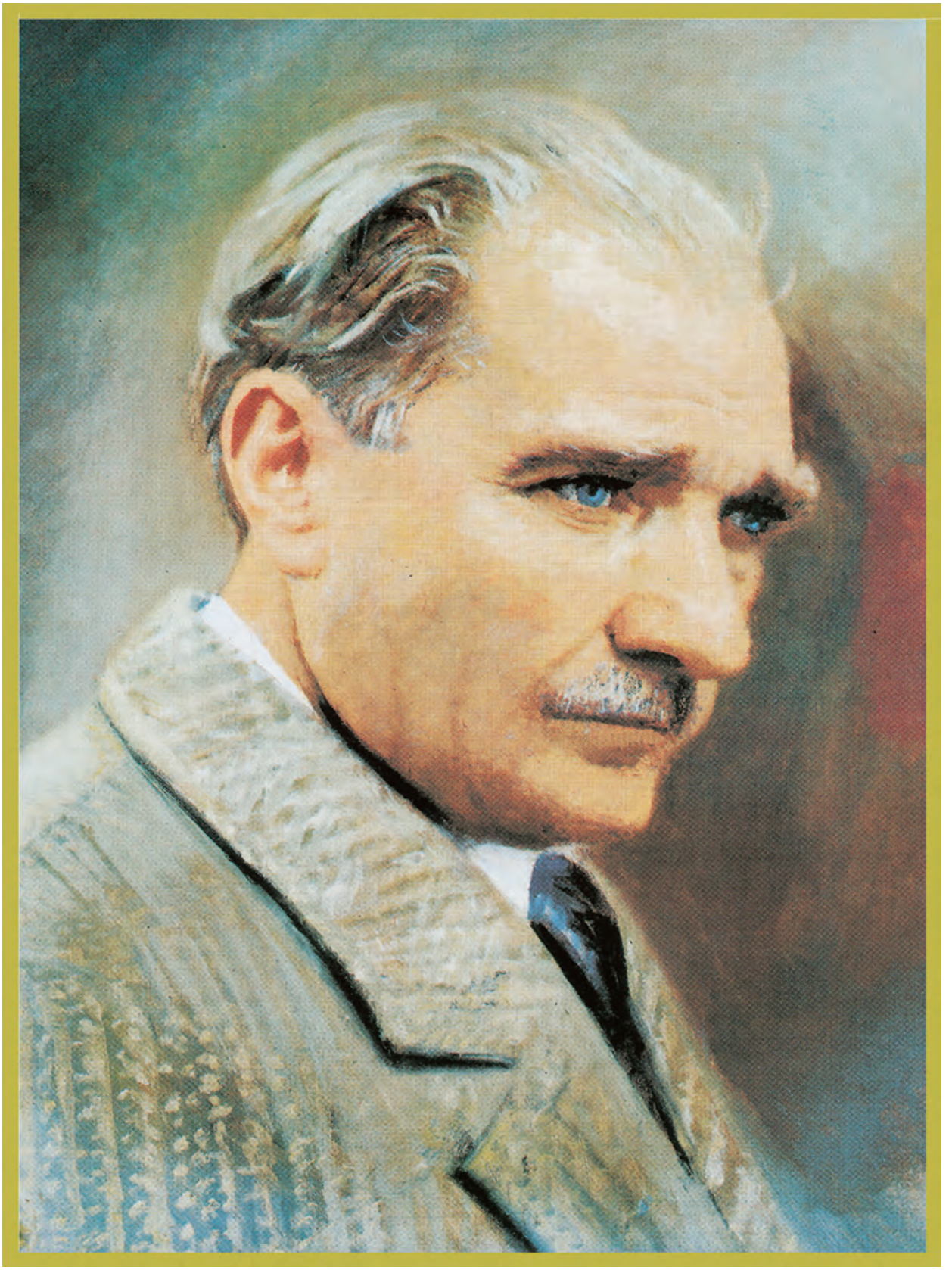
GENÇLİĞE HİTABE

Ey Türk gençliği! Birinci vazifen, Türk istiklâlini, Türk Cumhuriyetini, ilelebet muhafaza ve müdafaa etmektir.

Mevcudiyetinin ve istikbalinin yegâne temeli budur. Bu temel, senin en kıymetli hazinendir. İstikbalde dahi, seni bu hazineden mahrum etmek isteyecek dâhilî ve hâricî bedhahların olacaktır. Bir gün, istiklâl ve cumhuriyeti müdafaa mecburiyetine düşersen, vazifeye atılmak için, içinde bulunacağın vaziyetin imkân ve şeraitini düşünmeyeceksin! Bu imkân ve şerait, çok namüsait bir mahiyette tezahür edebilir. İstiklâl ve cumhuriyetine kastedecek düşmanlar, bütün dünyada emsali görülmemiş bir galibiyetin mümessili olabilirler. Cebren ve hile ile aziz vatanın bütün kaleleri zapt edilmiş, bütün tersanelerine girilmiş, bütün orduları dağıtılmış ve memleketin her köşesi bilfiil işgal edilmiş olabilir. Bütün bu şeraitten daha elîm ve daha vahim olmak üzere, memleketin dâhilinde iktidara sahip olanlar gaflet ve dalâlet ve hattâ hıyanet içinde bulunabilirler. Hattâ bu iktidar sahipleri şahsî menfaatlerini, müstevlîlerin siyasî emelleriyle tevhit edebilirler. Millet, fakr u zaruret içinde harap ve bîtap düşmüş olabilir.

Ey Türk istikbalinin evlâdı! İşte, bu ahval ve şerait içinde dahi vazifen, Türk istiklâl ve cumhuriyetini kurtarmaktır. Muhtaç olduğun kudret, damarlarındaki asil kanda mevcuttur.

Mustafa Kemal Atatürk



MUSTAFA KEMAL ATATÜRK

İÇİNDEKİLER

Öğrenme biriminin kapağını gösterir.

Öğrenme biriminin görselini gösterir.

Öğrenme biriminin numarasını gösterir.

Öğrenme biriminin ismini gösterir.

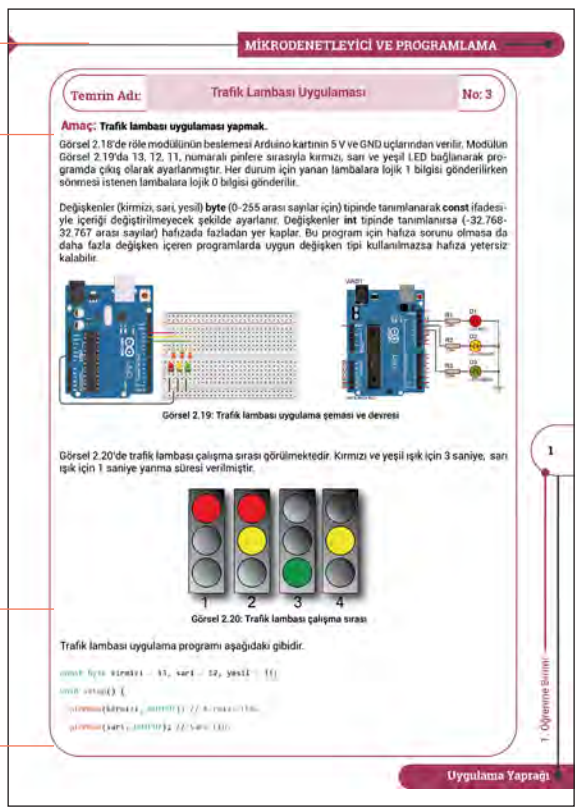
Öğrenme biriminindeki konu başlıklarını gösterir.

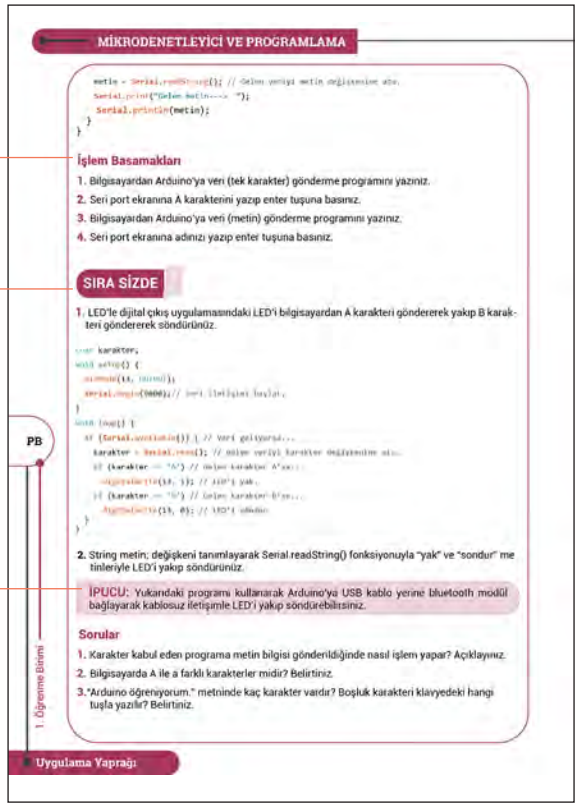
Öğrenme biriminin ismini gösterir.

Uygulamanın amacını gösterir.

Görsel numarasını gösterir.

Uygulama yaprağında olduğunuzu gösterir.

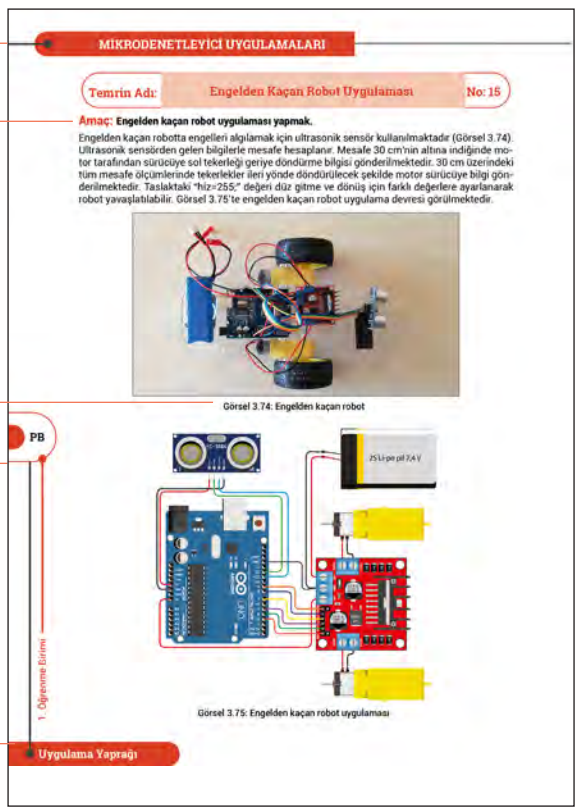




Öğrenme biriminin sonundaki işlem basamaklarını gösterir.

Uygulama yaprağındaki sıra sizde bölümünü gösterir.

Uygulama yaprağında ipucu ve sorularda olduğunuzu gösterir.



Öğrenme biriminin ismini gösterir.

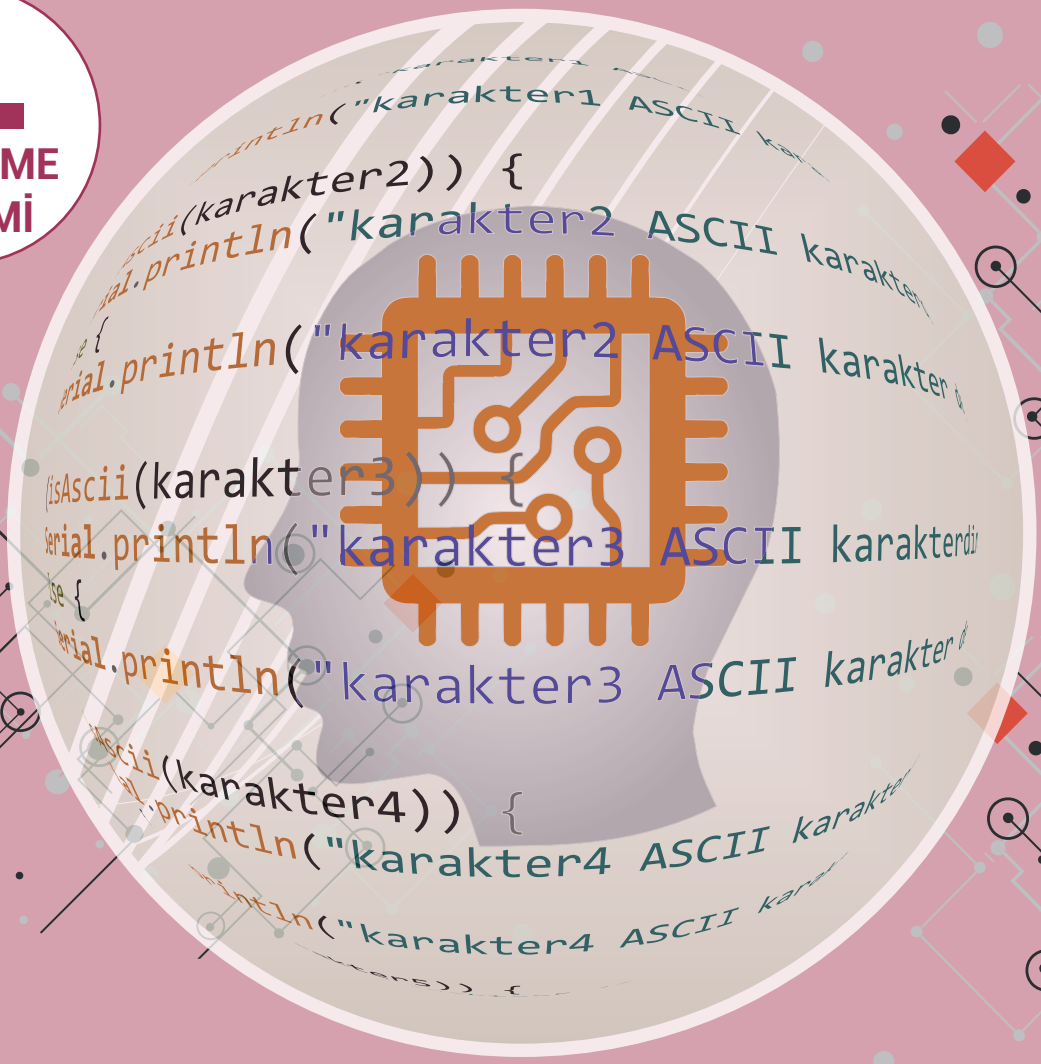
Uygulamanın amacını gösterir.

Görsel numarasını gösterir.

Sayfa numarasını gösterir.

Uygulama yaprağında olduğunuzu gösterir.

1. ÖĞRENME BİRİMİ



MİKRODENETLEYİCİ VE PROGRAMLAMA

NELER ÖĞRENECEKSİNİZ ?

- Mikroişlemciler ve Mikrodenetleyiciler
- Mikrodenetleyici Editör Programı
- Mikrodenetleyiciye Program Yükleme
- Algoritma Hazırlama
- Dijital Giriş Çıkış İşlemleri
- Seri Port Analog Giriş Çıkış İşlemleri

TEMEL KAVRAMLAR

mikrodenetleyici
algoritma
fonksiyon
koşul
döngü
değişken
EEPROM
işlem
mantık
dijital
analog
PWM
kesme

1. 1. MİKROİŞLEMCİLER VE MİKRODENETLEYİCİLER

Mikroişlemci, işlemci ana işlem biriminin [CPU Central Process Unit (Sentrıl Proses Yunıt)] fonksiyonlarını tek bir yarı iletken tüm devrede [IC Integrated Circuit (İntıgreytıd Sırkıt)] birleştiren, programlanabilir bir sayısal elektronik bileşendir. Kullanıcı ya da programcı tarafından yazılan programları meydana getiren komutları veya bilgileri yorumlamak ve yerine getirmek için gerekli olan tüm mantıksal devreleri kapsar. Bu devreler genelde transistörlerden meydana gelmektedir.

1.1.2. Mikrodenetleyici

Bir mikroişlemcili sistemi meydana getiren temel bileşenlerden mikroişlemci, bellek ve G/Ç birimlerinin, bazı özelliklerinin kırpılarak (azaltılarak) tek bir entegre içerisinde üretilmiş biçimine **mikrodenetleyici (microcontroller)** denir. Denetim teknolojisi gerektiren uygulamalarda kullanılmak üzere tasarlanmış olan mikrodenetleyiciler, mikroişlemcilere göre çok daha basit ve ucuzdur. Endüstrinin her kolunda kullanılan mikrodenetleyiciler; otomobil, kamera, cep telefonu, fotokopi ve çamaşır makineleri, televizyon, oyuncak vb. cihazlarda sıklıkla kullanılmaktadır.

1.1.3. Mikroişlemci ve Mikrodenetleyici Arasındaki Farklar

Bir mikroişlemcinin çalışabilmesi için en az üç ayrı ek devre gerekir. Bunlar yazılan programın tutulacağı kalıcı bellek birimi, değişken verilerin saklanacağı geçici bellek birimi ve dış dünyadan veri alışverişinin yapılmasını sağlayan giriş/çıkış birimleridir. Bunlar ve daha fazlası bir mikroişlemcili sistemde ayrı ayrı devreler şeklinde yerini alır. Bu sebeple mikroişlemcili sistemlere çok entegreli sistemler de denir. Mikroişlemci ve diğer gerekli olabilecek çeşitli devreler, bir entegre devrede birleştirilmiş ve mikrodenetleyiciler üretilmiştir.

Bilgisayar gibi mikroişlemcili sistemlere verilen bir örnekte, bir bilgisayarın bir çamaşır makinesinde veya cep telefonunda kullanılması elbette mümkün olmayacaktır. Bilgisayar aynı anda milyonlarca işi yapabildiğinden ve çok yer kapladığından böyle yerlerde kullanılması mantıklı değildir ve maliyeti çok olur. Bu sebeple sistemi meydana getiren elemanların birçok özelliğinden feragat edilerek ve elemanlar bir entegrede birleştirilerek mikroişlemcilerin yeni türevleri oluşturulmuştur.

1.1.4. Mikrodenetleyici Çeşitleri

Mikrodenetleyicilerin üretildikleri firmalara göre pek çok çeşidi vardır. Piyasada Atmel, Texas Instruments, ST Microelectronics, Microchip firmalarının mikrodenetleyicilerini kullanan pek çok geliştirme kartı mevcuttur. Atmel mikrodenetleyicileri kullanılan Arduino geliştirme kartları, Arduino topluluğu tarafından donanım tasarımları ve geliştirme ortamı kodları açık kaynak kodlu olarak herkesin kullanımına sunulmuştur. Bu donanım tasarımlarını kullanarak herkes kendine özgü geliştirme kartları tasarlayıp kullanabilir. Bu kitapta Atmega328p mikrodenetleyicisini kullanan Arduino Uno ve Nano modelleri üzerinden mikrodenetleyici programlama teknikleri gösterilmiştir.

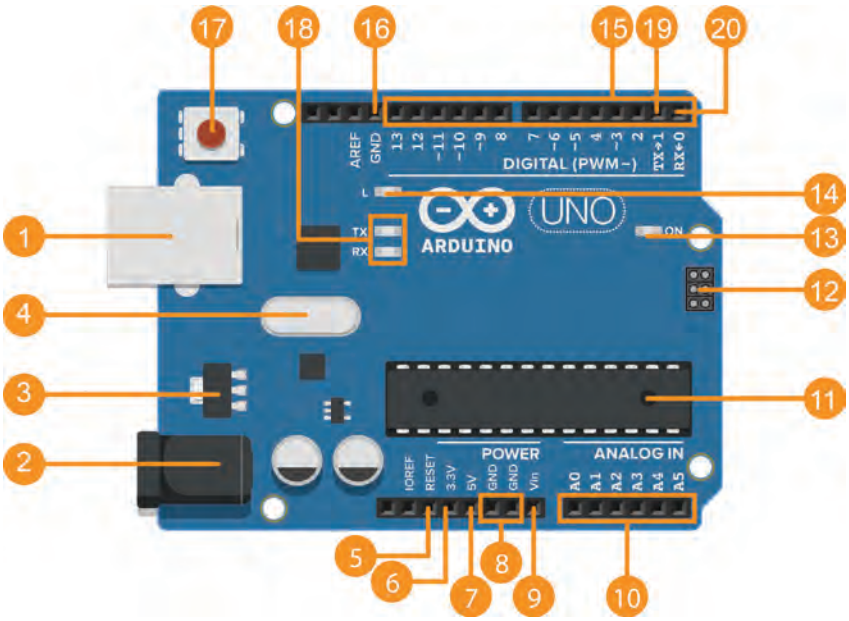
1.2. MİKRODENETLEYİCİ KARTININ DONANIM YAPISI VE ÖZELLİKLERİ

Arduino mikrodenetleyicisinde kullanılan işlemci Microchip firması bünyesindeki Atmele ait ATmega328P'dir. Geliştirme kartı Arduino geliştirici topluluğuna aittir. Mikrodenetleyicinin temel özellikleri aşağıda maddeler hâlinde verilmiştir:

1. 7 V-12 V besleme voltajı
2. 5 V çalışma voltajı

3. Pin başına 20 mA akım (en fazla 40 mA)
4. 32 kB program belleği (0,5 kB'ı bootloader (önyükleyici) tarafından kullanılan Arduino kodunun yüklendiği kalıcı hafıza)
5. 2 kB RAM bellek (geçici hafıza)
6. 1 kB EEPROM bellek (kalıcı hafıza)
7. 16 MHz çalışma frekansı
8. 2 x 8 bit 1 x 16 bit toplam 3 adet timer
9. 1 x I2C, 2 x SPI, 1 UART

Görsel 1.1'de Arduino Uno kartı ve donanımsal yapısı görülmektedir.



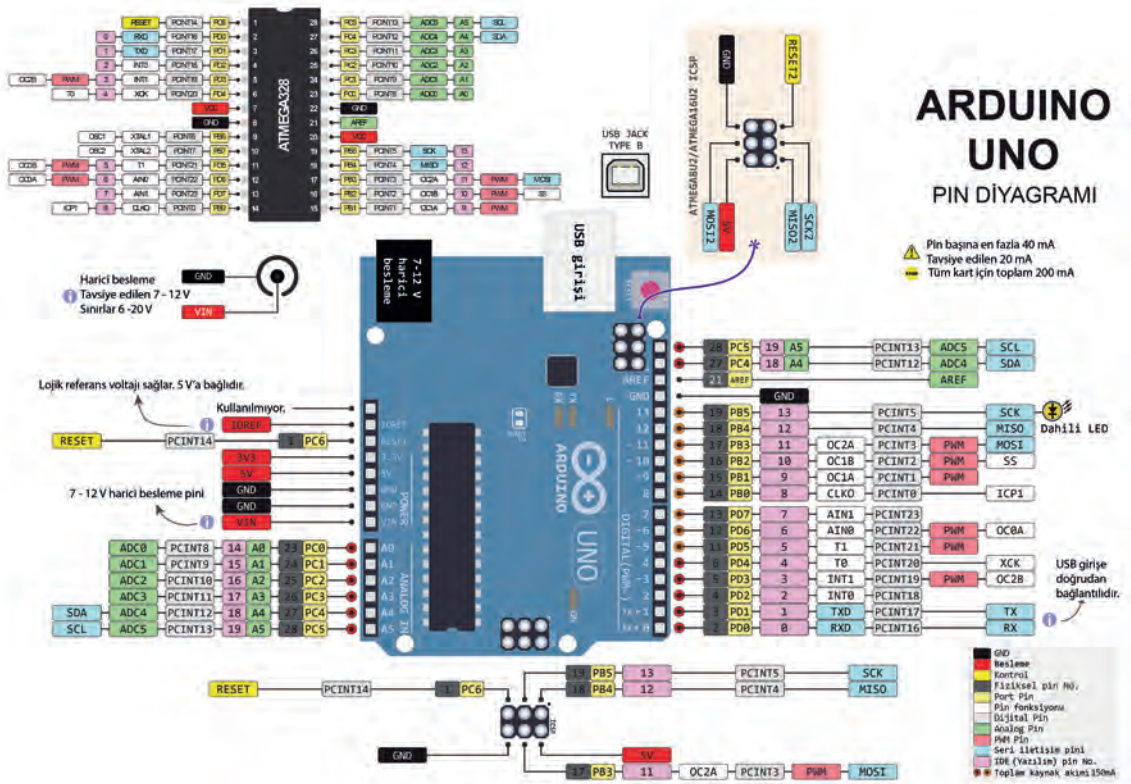
Görsel 1.1: Arduino Uno kartı ve donanımsal yapısı

1. USB girişi: Arduino ile bilgisayar arasındaki iletişimi sağlar. Arduino 5 V besleme gerilimini USB üzerinden alır.
2. Haricî besleme: Arduino'nun çalışması için gereken güç, DC adaptör veya pil ile bu soket üzerinden sağlanır (7 V-12 V).
3. Voltaj regülatörü: Haricî beslemeden gelen 7 V-12 V aralığındaki gerilimi 5 V'a dönüştürür.
4. Kristal osilatör: Arduino işlemcisinin çalışma frekansını üretir.
5. Reset: Çalışan kodu baştan başlatır.
6. 3,3 V çıkış sağlar (50 mA).
7. 5 V çıkış sağlar.
8. GND (ground-toprak) 0 V besleme pinidir.
9. Vin: Pin şeklindeki 7 V-12 V haricî besleme girişidir.
10. Analog In: Altı adet analog pin bulunmaktadır. A0...A5 olarak isimlendirilir. Dijital giriş çıkış olarak da kullanılır. ADC 10 bittir.
11. ATmega328P mikrodenetleyicisidir.

12. ICSP: Bu pinler SPI kütüphanesiyle SPI haberleşmeyi sağlar.
13. Power LED: Arduino'ya besleme verildiğinde yanar.
14. Dâhilî LED: Arduino'nun 13 No.lu pinine bağlı LED'dir.
15. On dört adet dijital giriş / çıkış pinleridir. Bazı pinlerin özel fonksiyonları vardır.
16. 0 (RX) ve 1 (TX) pinleri seri data almak [receive (risiv) RX] ve yaymak [transmit (tranzmit) TX] için kullanılır. 2 ve 3 numaralı pinler haricî kesme [interrupt (intırapt)] pinleridir. 3, 5, 6, 9, 10 ve 11 numaralı PWM pinleri analogWrite() fonksiyonuyla 8-bit PWM sinyali sağlar.
17. AREF: Analog girişler için referans voltajıdır. analogReference() fonksiyonuyla kullanılır.

Arduino Uno Pin Diyagramı

Arduino'da pinlerin birçok işlevi vardır. Görsel 1.2'de uygulamalarda kullanılan özellikler için ayrıntılı pin diyagramı verilmiştir.



Görsel 1.2: Arduino Uno pin diyagramı

Bellekler

Avr tabanlı Arduino kartlarında kullanılan mikrodenetleyicide üç tip bellek vardır:

- Flash bellek (program alanı), Arduino programının yüklendiği yerdir.
- SRAM (statik rastgele erişimli bellek), taslak çalıştığında geçici değişkenleri oluşturduğu ve yönettiği yerdir.
- EEPROM, bilgileri depolamak için (sabit disk gibi) kullanılan bellek alanıdır.

Flash bellek ve EEPROM bellek kalıcıdır (Bilgi, güç kapatıldıktan sonra da saklanır devam eder.).

SRAM geçicidir. Güç kapatıldığında bilgi kaybolur.

Ardunio Uno'da bulunan ATmega328P yongası aşağıdaki bellek miktarlarına sahiptir:

- Flash 32 kB (0,5 kB'ı bootloader tarafından kullanılır.)
- SRAM 2 kB
- EEPROM 1 kB

Ardunio Mega2560'taki ATmega2560, daha büyük bellek alanına sahiptir:

- Flash 256 kB (8 kB'ı bootloader tarafından kullanılır.)
- SRAM 8 kB
- EEPROM 4 kB

Uno'da çok fazla SRAM bulunmadığı için programda çok sayıda dizi bulunduğu zaman hafıza kolayca tükenir. SRAM biterse program başarıyla yükleniyor gibi görünse de program çalışmayabilir. Bu durumda program çalışmayacak veya garip bir şekilde çalışacaktır. SRAM bitmesinden kaynaklı bir problem olup olmadığını anlayabilmek için diziler programdan kaldırılır. Daha sonra başarılı bir şekilde çalışırsa muhtemelen SRAM bitiyor demektir. Bu sorunu çözmek için aşağıdaki işlemler yapılabilir:

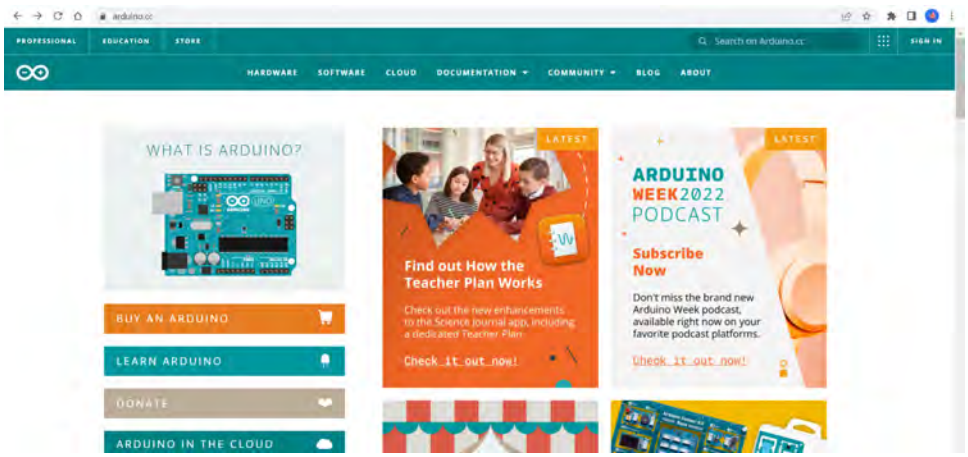
Yazılan program, bilgisayarda çalışan bir programla beraber çalışıyorsa veriler veya hesaplamalar bilgisayardaki programa kaydırılarak Arduino üzerindeki yük azaltılabilir.

Değerleri depolamak için gereken en küçük veri türü kullanılmalıdır. Örneğin int veri tipi iki byte yer kaplarken byte veri tipi için yalnızca bir byte'lık yer kullanılır (Değeri ne kadar küçük olsa da en küçük bilgi bile hafızada bir byte yer kaplamaktadır.).

Taslakta çalışırken dizileri veya verileri değiştirmek gerekmiyorsa bunlar SRAM yerine flash (program) bellekte saklanabilir. Bunu yapmak için PROGMEM anahtar sözcüğü kullanılır. Serial. print() fonksiyonuyla kullanılan F() makrosu da benzer işlev görmektedir.

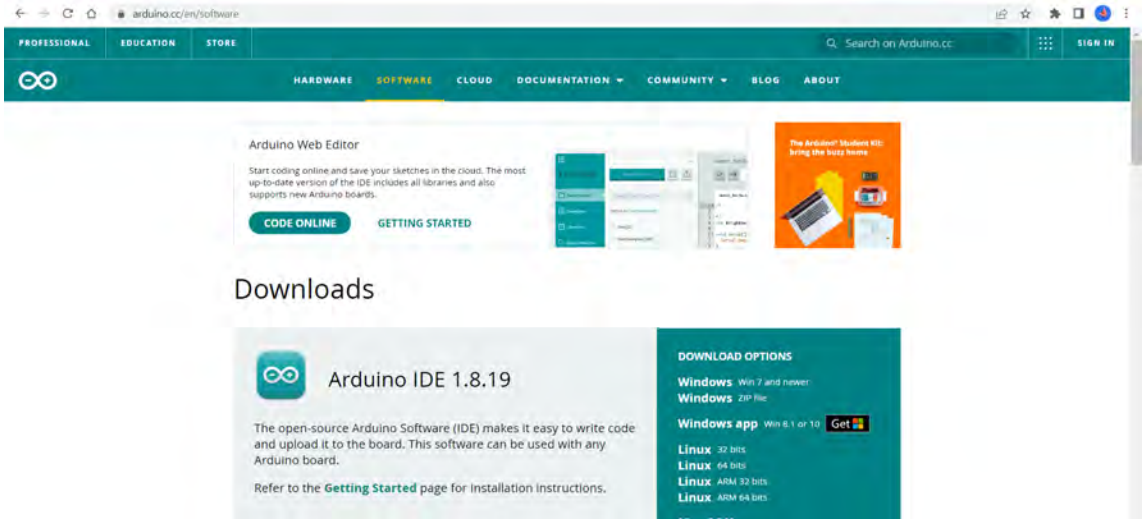
1.3. MİKRODENETLEYİCİ EDİTÖR PROGRAMI

Arduino yazılımını indirmek için www.arduino.cc adresinden software sekmesi tıklanır (Görsel 1.3).



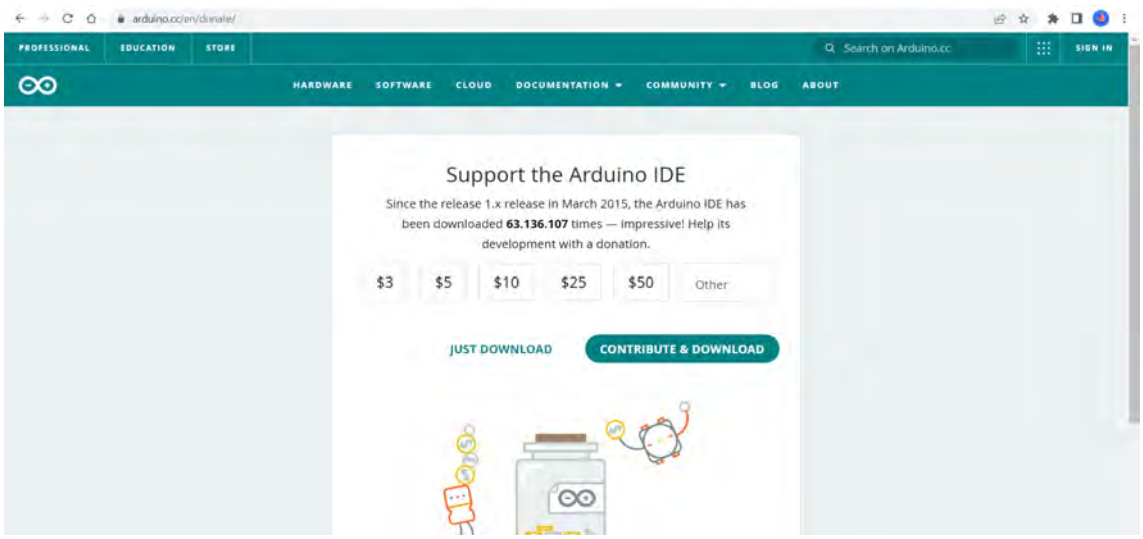
Görsel 1.3: Arduino resmî sitesi

İndirme sayfasında Arduino IDE'nin yayımlanmış son sürümü bulunmaktadır. Sağ taraftaki indirme seçeneklerinden işletim sistemine uygun seçenek tıklanır (Görsel 1.4).



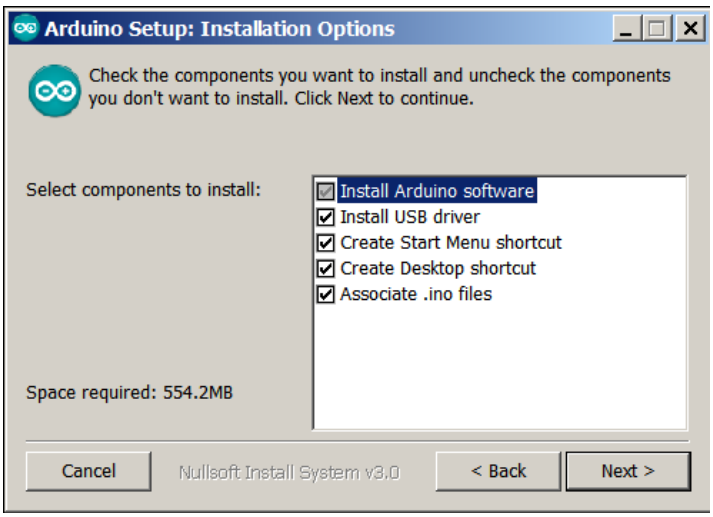
Görsel 1.4: Arduino IDE indirme sayfası

Just download düğmesine tıklanarak Arduino yazılımı bilgisayara indirilir (Görsel 1.5).



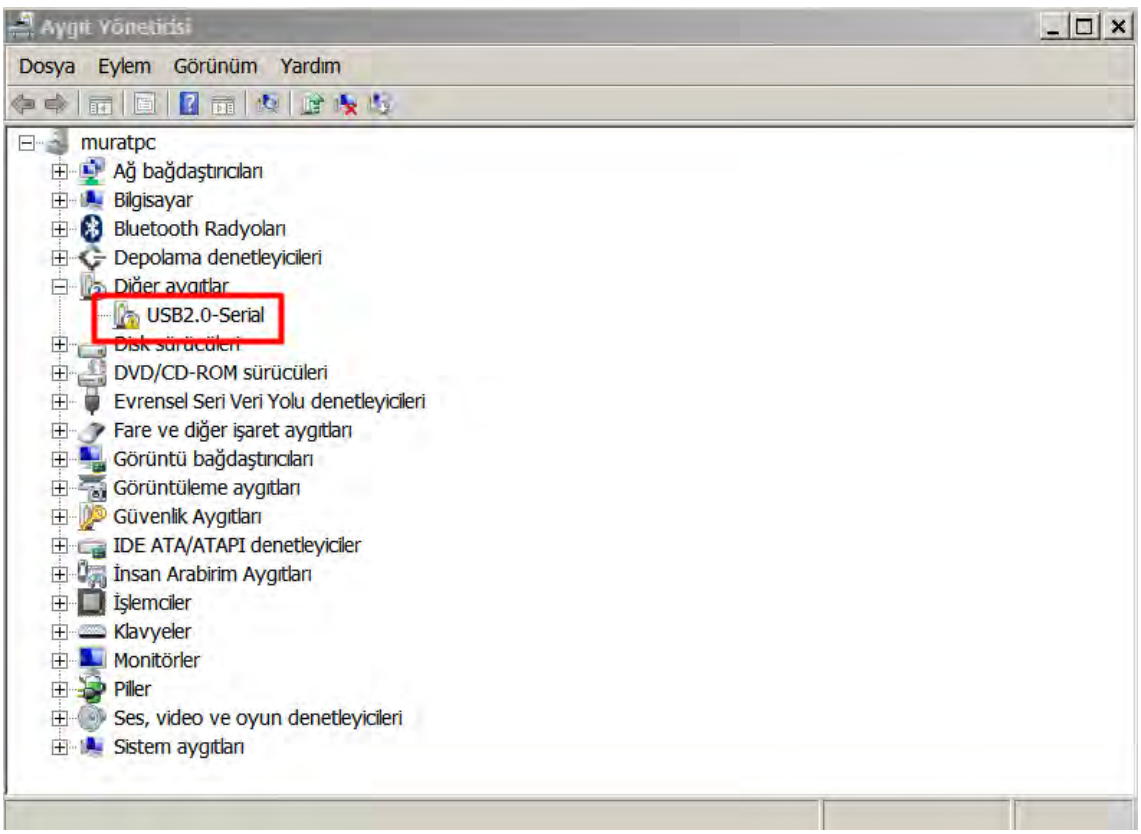
Görsel 1.5: Arduino IDE indirme sayfası

İndirilen kurulum dosyasıyla Arduino yazılımı bilgisayara kurulur (Görsel 1.6).

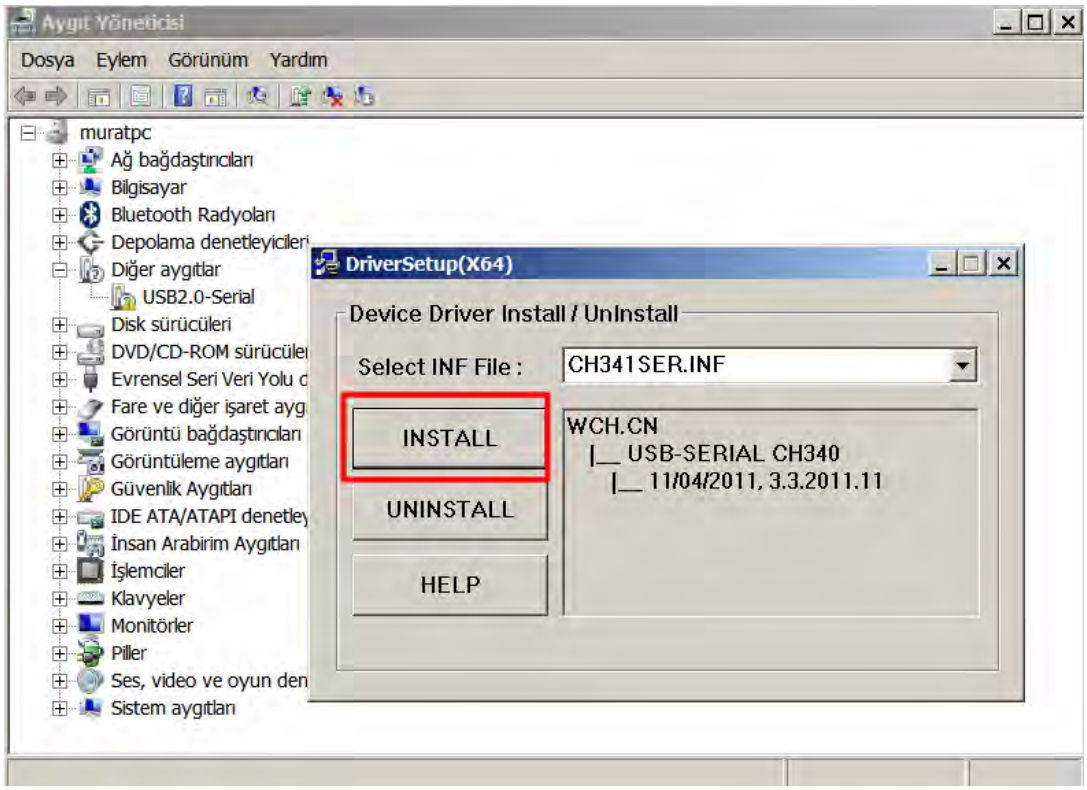


Görsel 1.6: Arduino IDE kurulumu

Arduino üzerinde USB bağlantıyı seri UART bağlantıya dönüştüren ATmega16u2 isimli çip bulunur. Ancak bazı kartlarda CH340 isimli bir çip mevcuttur. Arduino kartı bilgisayara bağlıyken aygıt yöneticisinde Görsel 1.7'deki sürücü yükleme hatası varsa Görsel 1.8'deki CH340.exe isimli sürücü dosyası yüklenir.

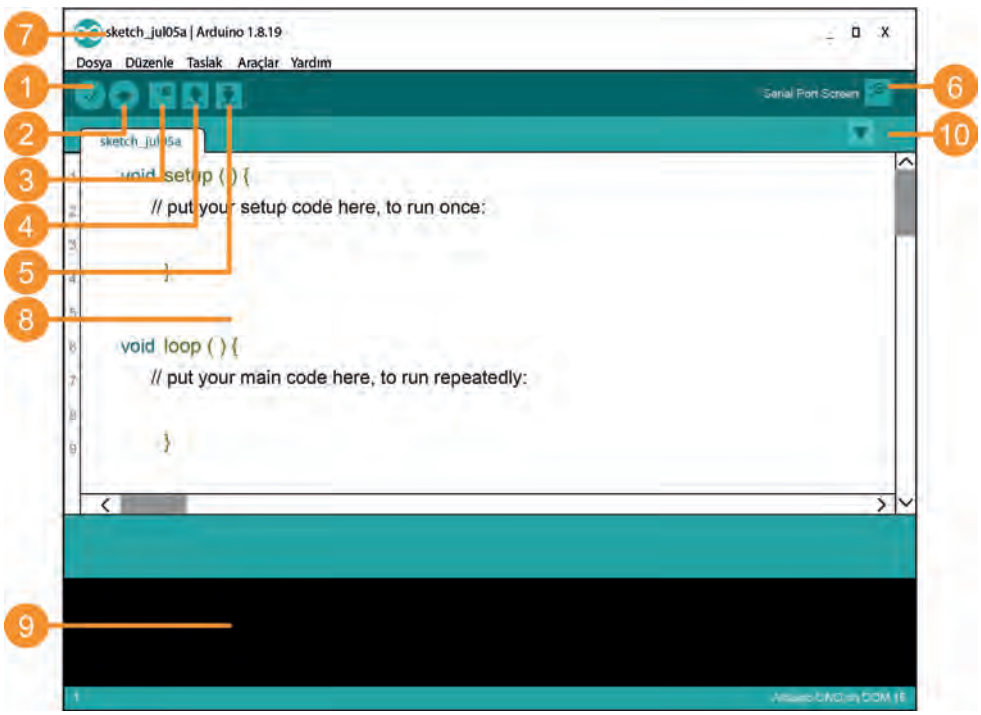


Görsel 1.7: Aygıt yöneticisi sürücü yükleme hatası



Görsel 1.8: CH340 sürücü dosyası yükleme

Sade bir arayüze sahip olan Arduino yazılımındaki butonlar ve görevleri Görsel 1.9'da verilmiştir.



Görsel 1.9: Arduino IDE üzerindeki butonlar

Sade bir arayüze sahip olan Arduino yazılımındaki butonların görevleri aşağıda verilmiştir:

1. Derleme düğmesi yazılan kodu derler ve söz dizimi hatalarını bulur. Mantıksal hataları bulmaz
2. Yükleme düğmesi programı Arduino kartına yükler. Bu sırada RX ve TX pinleri meşguldür.
3. Yeni düğmesi yeni kod penceresi açar.
4. Aç düğmesi bilgisayarda bulunan .ino uzantılı Arduino dosyasını açar.
5. Kaydet düğmesi yazılan programı kaydeder.
6. Seri Monitör düğmesi seri iletişimi görüntüler.
7. Çalışılan programın dosya adı yer alır.
8. Programın yazıldığı alandır.
9. Mesaj alanı derleme ve yükleme butonlarına basıldığında bir hata varsa hatayı açıklamasıyla birlikte kırmızı olarak gösterir.
10. Sekme kullanım seçenekleri yer alır.

Arduino IDE Menüler

Dosya Menüsü: Dosya işlemlerinin yapıldığı komutlar bulunur.

Yeni: Yeni sketch (taslak) oluşturur.

Aç: Bilgisayarda bulunan .ino uzantılı Arduino dosyasını açar.

Sonuncuyu Aç: Son kullanılan dosyaları açmak için kullanılır.

Taslak Defteri: Programların kaydedildiği klasördeki program listesini görüntüler.

Örnekler: Arduino IDE'le gelen örnek kodlar bulunur.

Kapat: Editörü kapatır.

Kaydet: Üzerinde çalışılan program dosyasını kaydeder.

Farklı Kaydet: Üzerinde çalışılan program dosyasını farklı isimle veya farklı adrese kaydeder.

Sayfa Ayarları: Yazıcıdan çıktı almak için sayfa ayarlarını düzenler.

Yazdır: Yazıcıdan programın çıktısını alır.

Tercihler: Arduino programıyla ilgili ayarlar yapılır. Program dosyalarının kaydedileceği klasör, font büyüklüğü, satır numaraları gösterme, kod katlama vb. seçenekler bulunur.

Çıkış: Programı kapatır.

Düzenle Menüsü: İşlemi geri alma, ileri alma, kesme, kopyalama, yapıştırma, seçme, düzenleme işlemleri yapılır.

Geri Al (Undo): Kod yazarken yapılan son işlem geri alınır.

Tekrarla: Geri alınan işlem tekrar gerçekleştirilir (İleri alınır.) .

Kes: Seçilen metni keserek panoya kopyalar.

Kopyala: Seçilen metni panoya kopyalar.

Forum İçin Kopyalama: Yazılan kodları forumlarda paylaşmak üzere panoya kopyalar.

HTML Olarak Kopyalama: Yazılan kodları web sayfalarına entegre etmek için HTML olarak panoya kopyalar (Renkli çıktı almak için kullanılabilir.).

Yapıştır: Panodaki metni imlecin bulunduğu yere yapıştırır.

Tümünü Seç: Tüm kodu seçer.

Satıra Git: İmleci, yazılan satır numarasında gösterir.

Yorum Yap/Yorumu Kaldır: Bir satırı otomatik olarak yorum hâline getirir (Seçilen alanın başına // ekler.). Tekrarında eski hâline getirir.

Girintiyi Artır: Satır başına boşluk ekler.

Girintiyi Azalt: Satır başındaki boşluğu azaltır.

Yazı Tipi Boyutunu Küçült/Büyüt: Yazı tipi boyutunu küçültmek büyütmek için kullanılır. Ctrl + mouse tekerleği kullanılarak da aynı işlem yapılır.

Bul: Kodların içinde aradığımız metni bulur.

Sonrakini Bul: Aranan metnin bir sonraki örneğini bulur.

Öncekini Bul: Aranan metnin bir önceki örneğini bulur.

Taslak Menüsü: Kod derleme, yükleme, kütüphane ekleme gibi işlemlerin yapıldığı menüdür.

Doğrula/Derle: Kodu derler.

Yükle: Kodları derler ve karta yükler.

Programlayıcıyı Kullanarak Yükle: Farklı programlayıcı kullanarak kod yüklemesi yapmak için kullanılır.

Derlenmiş Binaryi Çıkar: Hex dosyasını .ino dosyasıyla aynı klasöre çıkarır.

Çalışma Klasörünü Göster: Derlenen kodların bulunduğu klasörü ve içeriğini gösterir.

Library Ekle: Program yazarken kullanacağımız kütüphane dosyasını kodların başına ekler.

Dosya Ekle: Başka bir konumdaki program dosyasını sketch içine ekler. Eklenen dosya yeni bir sekmede görünür. Eklenen dosya mevcut .ino dosyasıyla aynı klasöre kopyalanır.

Araçlar Menüsü: Kullanılan kartın ve portun seçimi, seri monitörü görüntüleme, oluşturulan programı sıkıştırılmış dosya şeklinde kaydetme gibi işlemlerin yapıldığı menüdür.

Otomatik Biçimlendir: Koddaki girinti ve boşlukları standart yazım şeklinde ayarlar.

Taslağı Arşivle: Yazılan programı, sıkıştırılmış dosya (.zip formatı) olarak kaydeder.

Karakter Kodlamasını Düzelt & Tekrar Yükle: Programda karakter kodlamalarıyla ilgili bir problem varsa bu hataları düzeltir. Web siteleri ve forumlardan alınan kodlarda hata veren durumlarda kullanılır.

Kütüphane Yöneticisi: Kurulu olan veya internetten indirilebilen kütüphaneleri yönetir.

Seri Port Ekranı: Seri iletişim monitörünü açar ve buradan gelen verileri görüntüler.

Seri Çizici: Arduino Serial Monitor'den gelen seri verileri grafik olarak gösterir.

Kart: Kart seçimini yapar.

Port: Bilgisayarla kart arasında haberleşme amacıyla oluşturulan COM portunun seçimi yapılır.

Kart Bilgisi Al: Arduino kartının seri numarasını gösterir.

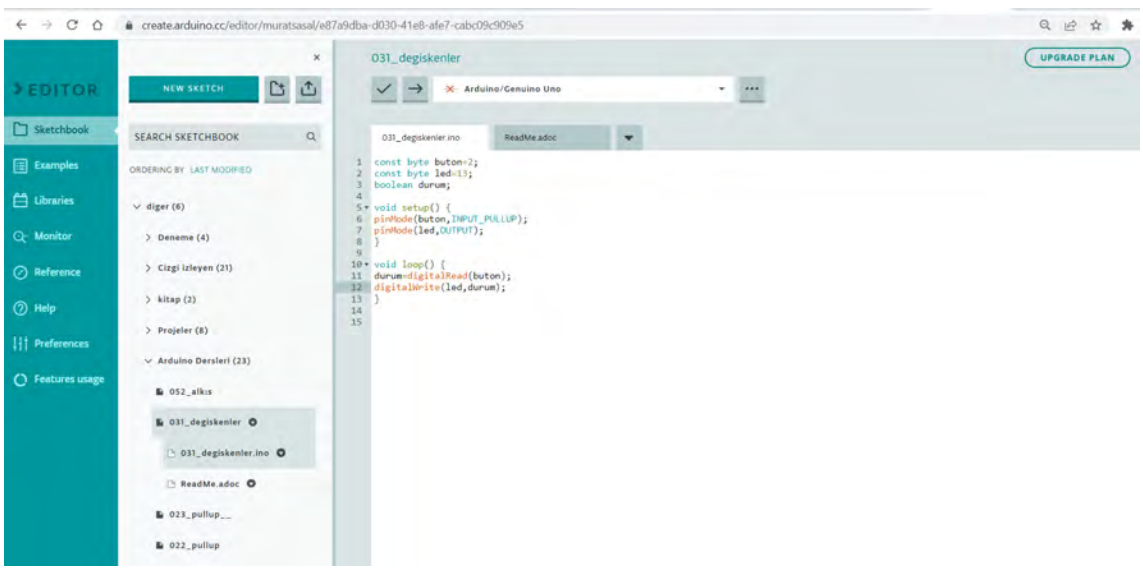
Programlayıcı: Kullanılan programlayıcı tipini gösterir.

Önyükleyici Yazdır: Önyükleyicisi (bootloader) olmayan kartlara bootloader yüklemek için kullanılır.

Yardım Menüsü: www.arduino.cc adresinden ulaşabileceğiniz kaynaklara yönlendirir.

Arduino Web Editör

Arduino IDE yazılımını bilgisayara kurmadan https:// create.arduino.cc/editor sayfasından çevrimiçi olarak da Arduino kartları programlanır. Yazılan programlar web sitesinin bulut saklama alanına kaydolmaktadır (Görsel 1.10).

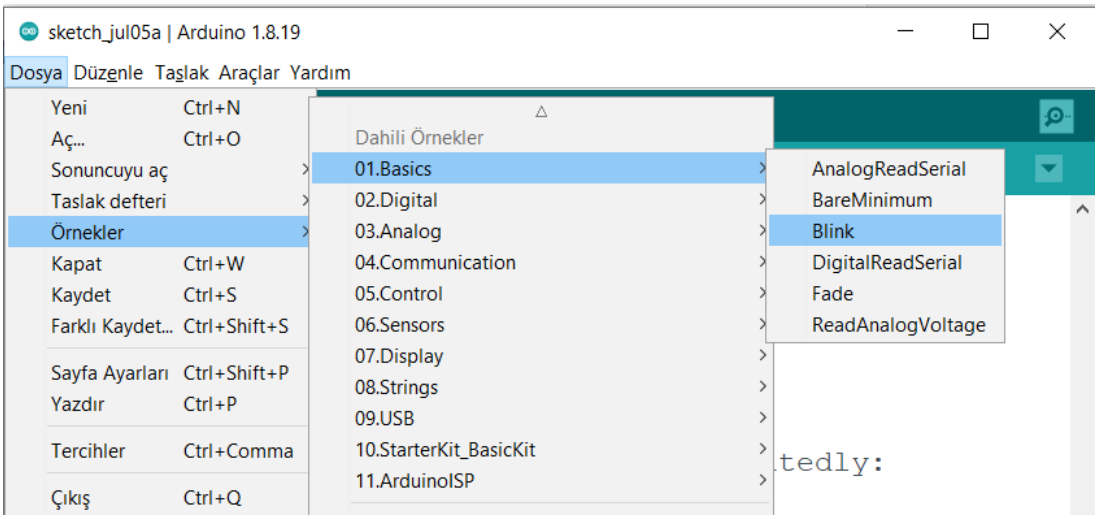


Görsel 1.10: Arduino web editörü

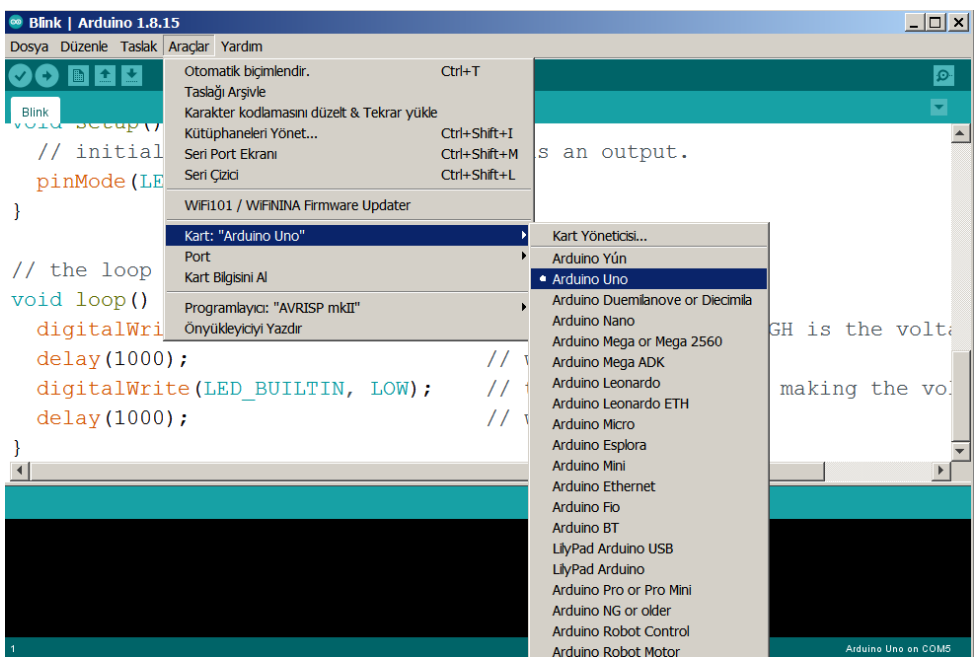
1.4. MİKRODENETLEYİCİYE PROGRAM YÜKLEME

Arduino "Dosya-Örnekler" menüsünde pek çok hazır uygulama mevcuttur. Temel uygulamalardan Blink (yanıp sönen LED) uygulaması Arduino üzerindeki dâhilî LED'i (Bu LED 13 numaralı pine bağlıdır.) 1 saniye aralıkla yakıp söndürmektedir.

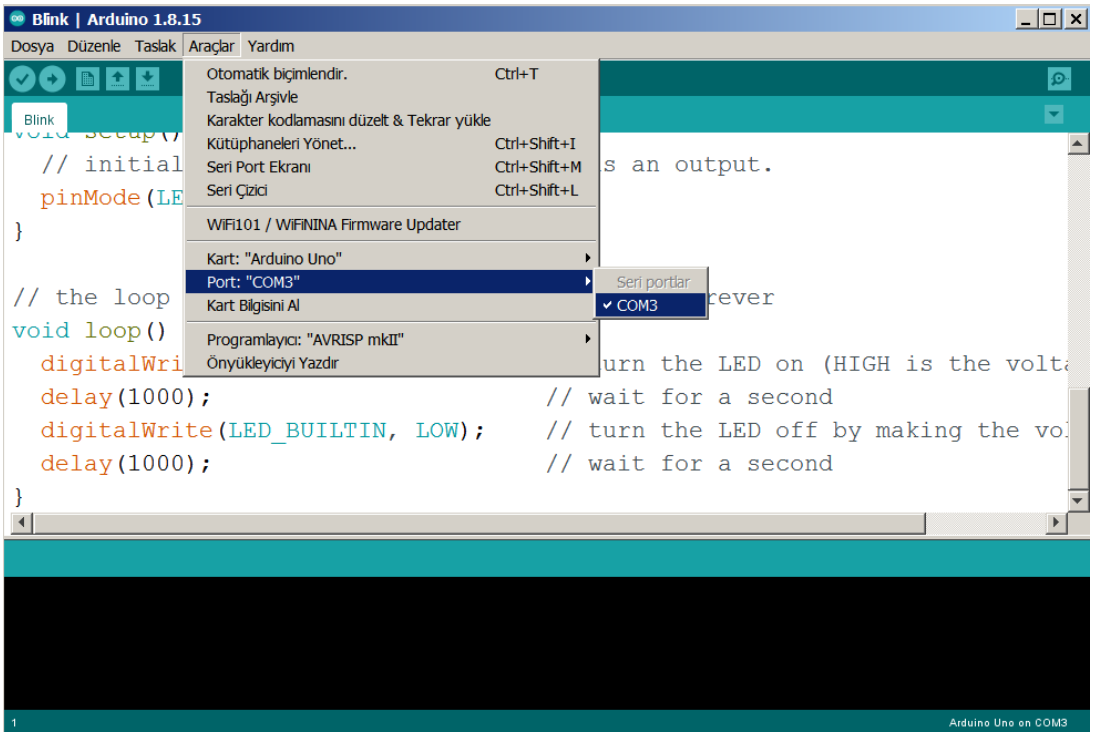
Görsel 1.11'de Dosya → Örnekler → 01.Basics → Blink yolunu izleyerek Blink uygulaması seçilir. Görsel 1.12'de "Araçlar" menüsündeki kart seçeneğinden Arduino Uno (kullanılacak kart çeşidi) seçilir. Görsel 1.13'te "Araçlar" menüsündeki port seçeneğinden Arduino kartımızın kartının bilgisayara bağlı olduğu USB portu seçilir. Görsel 1.14'te "Yükle" düğmesine basılarak Blink uygulaması Arduino kartına yüklenir. Yükleme tamamlandıktan sonra 13 numaralı pinin hizasındaki dâhilî LED birer saniye aralıklarla yanıp söner.



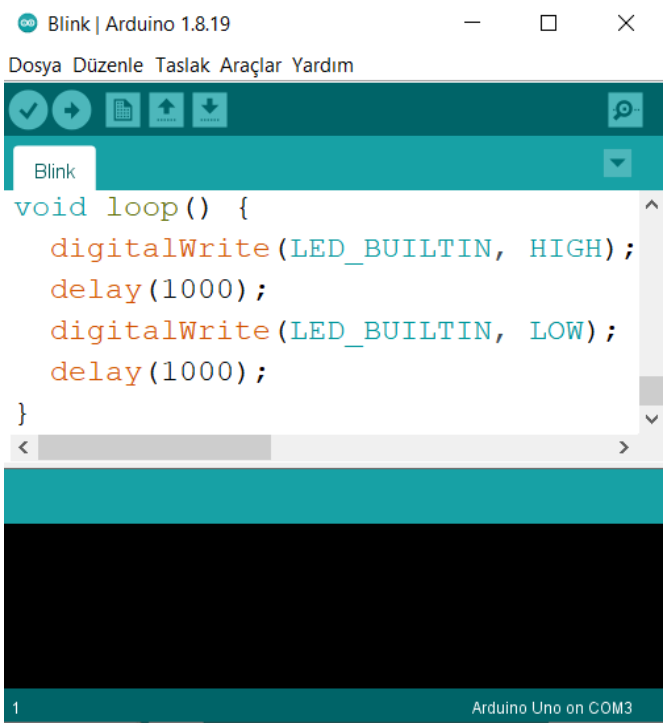
Görsel 1.11: Örnek programlara ulaşma



Görsel 1.12: Arduino kart seçimi



Görsel 1.13: Arduino kartının bağlı olduğu port seçimi



Görsel 1.14: Blink uygulaması

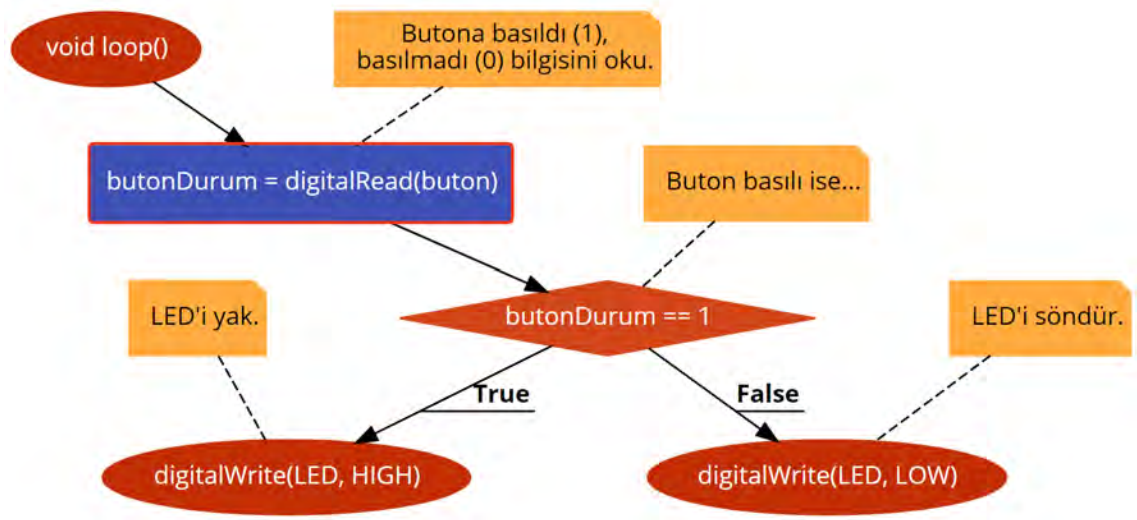
1.5. ALGORİTMA HAZIRLAMA

Algoritma, bir problemi çözmek için takip edilecek sonlu sayıda adımdan oluşan bir çözüm yoludur. Diğer bir ifadeyle algoritma, bir problemin mantıksal çözümünün adım adım nasıl gerçekleştirileceğinin sözlü ifadesidir.

Algoritmayla oluşturulan çözümler sözel olarak ifade edildiğinden algoritmayı herkesin aynı sonucu çıkarabileceği hâle getirmek için akış diyagramları kullanılır. Akış diyagramları sembollerden oluşmaktadır. Her sembolün belli bir işlevi vardır.

Algoritması oluşturulmuş bir problemin bilgisayar ortamına aktarılmış hâline **program** denir. Program, problemin çözümünde yapılması gereken işlemler bütününün kod karşılığıdır. Algoritmaların program hâline getirilmesi için programlama dilleri kullanılır. Programlama dilleri kullanılarak yazılımlar geliştirilir. Arduino programlama dili C/C++ programlama dilinden üretilmiş bir dildir.

Görsel 1.15'te butona basıldığında LED'in yandığı bir olayın algoritması, akış diyagramı ve programı görülmektedir.



Görsel 1.15: Karar yapısı algoritması

1.6. TEMEL PROGRAMLAMA İŞLEMLERİ

Arduino programlama dili, C programlama diline benzemektedir. Her programlama dilinde yapılan temel işlemler Arduino programlama dilinde de yapılmaktadır.

Arduino IDE'de yeni bir taslak oluşturulduğunda taslak (sketch) Görsel 1.16'da görüldüğü gibi iki hazır fonksiyonla gelmektedir. Bunlar setup() (ayar) ve loop() (döngü) fonksiyonlarıdır. Bu fonksiyonlar taslakta bulunmazsa derleyici hata verir.

setup() Fonksiyonu

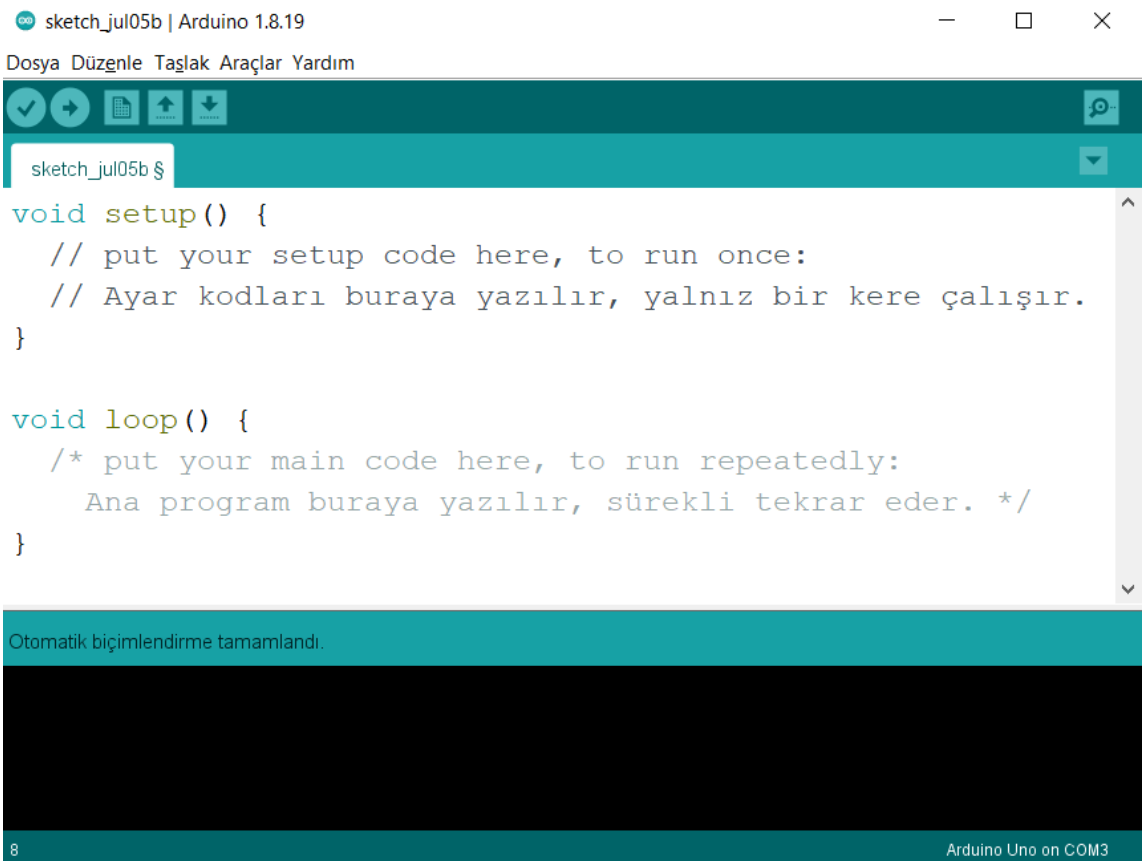
setup() fonksiyonu donanımsal ayarların yapıldığı kart başlatıldığında veya resetlendiğinde bir kez çalışan kısımdır. Void ifadesi setup() fonksiyonun geri herhangi bir değer döndürmediğini ifade etmektedir.

loop() Fonksiyonu

loop() fonksiyonu ana programın olduğu ve kartın enerjisi kesilene kadar sürekli çalışan döngüdür. loop() fonksiyonu da void ifadesiyle yazılır ve geriye değer döndürmez.

" // " ifadesinden sonra aynı satırda yer alan metin açıklama kısımlarıdır ve programın işleyişini etkilemez.

" /* " ve " */ " ifadeleri birden fazla satır içeren açıklama metni yazmak için kullanılır ve programın işleyişini etkilemez.



Görsel 1.16: Arduino yeni taslak

1.7. DİJİTAL GİRİŞ ÇIKIŞ İŞLEMLERİ

Arduino'da dijital işlemler 0 ve 1'in yani 0 V ile 5 V'luk iki seçeneğin olduğu işlemlerdir. Giriş işlemleri Arduino pinlerinin haricî bir devreden elektrik enerjisi alınmasıdır. Çıkış işlemleri ise Arduino pinlerinden haricî bir devreye elektrik enerjisi gönderilmesidir. Arduino'da dijital işlemlerde kullanılan üç temel fonksiyon vardır.

1.7.1. Port Tanımlama pinMode() Fonksiyonu

Arduino'da dijital işlemler 0 ve 1'in yani 0 V ile 5 V'luk iki seçeneğin olduğu işlemlerdir. Giriş işlemleri Arduino pinlerinin haricî bir devreden elektrik enerjisi alınmasıdır. Çıkış işlemleri ise Arduino pinlerinden haricî bir devreye elektrik enerjisi gönderilmesidir. Arduino'da dijital işlemlerde kullanılan üç temel fonksiyon vardır.

Ardino 0 - 13 numaralı dijital pinleriyle A0-A5 (14-19 numaralı) analog pinlerini (Dijital olarak kullanılabilir.) giriş (INPUT) veya çıkış (OUTPUT) olarak ayarlamak için kullanılan fonksiyondur.

pinMode(pin, mod)

pin: Bağlantı ucu numarası.

mod: INPUT, OUTPUT veya INPUT_PULLUP

Örnek: Dijital pinleri giriş veya çıkış olarak ayarlar.

```
pinMode(2, INPUT); // 2 numaralı pin giriş olarak ayarlandı.
pinMode(19, INPUT_PULLUP); // 19 numaralı pin giriş ve pull-up olarak ayarlandı.
pinMode(13, OUTPUT); // 13 numaralı pin çıkış olarak ayarlandı.
```

INPUT olarak yapılandırılmış Arduino (ATmega) pinleri yüksek empedansa ayarlanır. Pinlerin önünde 100 MΩ'luk bir seri direnç bağlanmış şeklinde düşünülebilir. Giriş olarak ayarlanan pinler sensör vb. okumak için kullanışlı hâle gelir. Girişte buton kullanılırsa 10 kΩ'luk pull-up veya pull-down direnci de kullanılmalıdır.

INPUT_PULLUP kullanıldığında Arduino içindeki dâhilî pull-up dirençleri (20 kΩ) aktif edilir. Devrede pull-up direnci kullanmaya gerek yoktur. Giriş sürekli 1'e çekili olduğundan aktif 0 çıkışlı sensörler kullanılmalı veya devre bu şekilde tasarlanmalıdır. INPUT veya INPUT_PULL-UP komutuyla giriş olarak yapılandırılan pinler, 0 V altındaki (negatif voltajlar) veya 5 V üzerindeki voltajlara bağlanırsa zarar görebilir veya bozulabilir. 13 numaralı pinde dâhilî LED ve ona bağlı seri direnç bulunduğundan INPUT_PULLUP kullanıldığında 5 V yerine 1,7 V'ta kalacağından kullanışlı değildir.

OUTPUT olarak yapılandırılan pinler, düşük empedansa ayarlanır. Arduino pinleri, diğer cihazlara 40 mA'e kadar akım sağlayabilir. Bu akım LED gibi düşük güçlü elemanlar için yeterli olsa da motor gibi yüksek güç isteyen elemanlar için araya yeterli akım ve gerilimi sağlayabilecek yükselteç (sürücü) devreleri ilave edilmelidir. OUTPUT olarak ayarlanan pinler, GND'ye veya 5 V'a bağlanırsa hasar görebilir veya bozulabilir.

NOT

pinMode() fonksiyonu loop() fonksiyonunun içinde kullanılarak bir pin, programın belli kısımlarında INPUT, belli kısımlarında OUTPUT olarak ayarlanabilir. Kullanılan dijital pin sayısını azaltmak için bu yönteme başvurulabilir.

1.7.2. Dijital Çıkış İşlemleri

digitalWrite() Fonksiyonu

pinMode() fonksiyonuyla OUTPUT olarak ayarlanan pinlere HIGH (5 V veya lojik 1) veya LOW (0 V veya lojik 0) vermek için kullanılır. Örneğin pine 5 V verildiğinde akım kaynağı olarak katodu seri bir dirençle GND'ye bağlı olan LED ışık verir. 0 V verildiğinde anodu seri bir dirençle 5 V'a bağlanmış LED ışık verir.

digitalWrite(pin, deger)

pin: Pin numarası

deger: HIGH veya LOW (1 veya 0)

Örnek: Dijital çıkışı 5 V veya 0 V yapar.

```
digitalWrite(13, HIGH); // 13 numaralı pin çıkışı 1 olur.
digitalWrite(13, LOW); // 13 numaralı pin çıkışı 0 olur.
```

Arduino UNO'da A0 – A5 olarak adlandırılan altı adet analog giriş pini, dijital pin olarak da kullanılabilir. Böylelikle Arduino UNO'da yirmi adet pin digital işlemler için kullanılabilir. İstisna olarak Arduino Nano, Pro Mini ve Mini'nin A6 ve A7 pinleri yalnızca analog giriş olarak kullanılmaktadır.

NOT

Bir pin pinMode() fonksiyonuyla OUTPUT olarak ayarlanmadığında ve o pine bağlı LED'e, digitalWrite() fonksiyonuyla HIGH gönderildiğinde LED sönük görünebilir. Bunun nedeni PinMode() fonksiyonu ayarlamadan digitalWrite() kullanılması, o pinde akım sınırlayıcı direnç gibi davranan dâhilî pull-up direncini etkinleştirmesidir.

delay() Fonksiyonu

Programı, parametre olarak belirtilen süre kadar (milisaniye cinsinden) duraklatır (1000 milisaniye 1 saniyedir.). Delay fonksiyonu işlemleri durduğu için diğer işlemler sensör okuma vb. yapılmayacaktır. delay() fonksiyonu, işletilen programı delay fonksiyonun olduğu satırda belirtilen süre boyunca bekletir. Bu sırada bir sonraki komuta geçilemez. Diğer hiçbir işlem yapılmaz. Delay fonksiyonu işlemleri durdurduğu için diğer işlemler sensör okuma vb. yapılmayacaktır. Gelişmiş bir programda delay() yerine milis() fonksiyonu kullanılır.

delay(ms); //Programı bekletme komutu yazılımı.

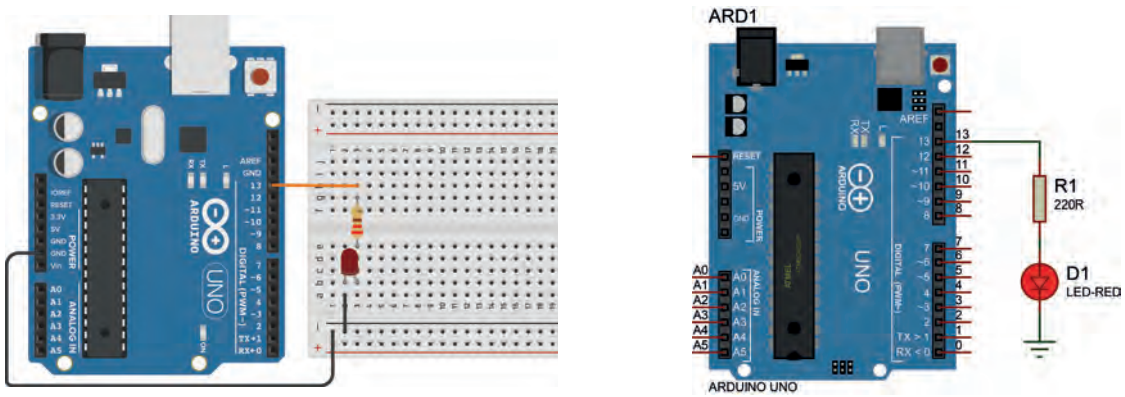
ms: Duraklatılacak sürenin milisaniye cinsinden sayısı. Veri tipi: unsigned long.

delay(500);//Programı 500 ms (yarım dakika) bekletir.

Uygulama Adı: LED'le Dijital Çıkış Uygulaması — No.: 1

Amaç: LED'le dijital çıkış uygulaması yapmak.

Görsel 1.17'de Arduino'nun 13. pininde bir LED ve LED'e seri bağlı direnç vardır. Arduino kodu LED'i 1 saniye aralıklarla yakıp söndürmektedir.



Görsel 1.17: LED'le dijital çıkış şeması ve devresi

LED'le dijital çıkış uygulaması programı aşağıdaki gibidir:

```
void setup() {          // Portların giriş-çıkış olarak tanımladığı fonksiyon. (Bir
kez çalışır.)
pinMode(13, OUTPUT);    // 13 numaralı pin çıkış olarak ayarlandı.
}
void loop() {           // Loop sonsuz döngüsü. (Sürekli çalışır.)
digitalWrite(13, HIGH); // 13 numaralı pinden 5 V (lojik 1) gönderildi.
delay(1000);            // Zaman gecikmesi // Bekle 1000 ms = 1 sn.
digitalWrite(13, LOW);  // 13 numaralı pin 0 V'a (GND) çekildi. (lojik 0)
delay(1000);            // Zaman gecikmesi // Bekle 1000 ms = 1 sn.
}
```

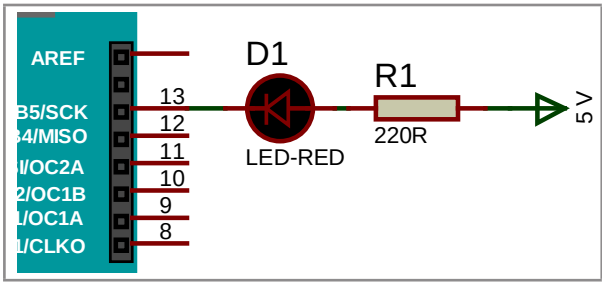
İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 1.17'deki devreyi kurunuz.
4. Programı yazıp Arduino'ya yükleyiniz. LED'in çalışmasını gözlemleyiniz.
5. digitalWrite(13, HIGH); komut satırını digitalWrite(13, 1); olarak, digitalWrite(13, LOW); komut satırını digitalWrite(13, 0); olarak değiştirip programı yükleyiniz.
6. Programdaki delay(1000); fonksiyonlarının parametresini değiştirip LED'in durumunu gözlemleyiniz.
7. A0 pinini kullanarak programı ve devreyi yeniden düzenleyiniz. Programı yükleyerek devrenin çalışmasını gözlemleyiniz.

8. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

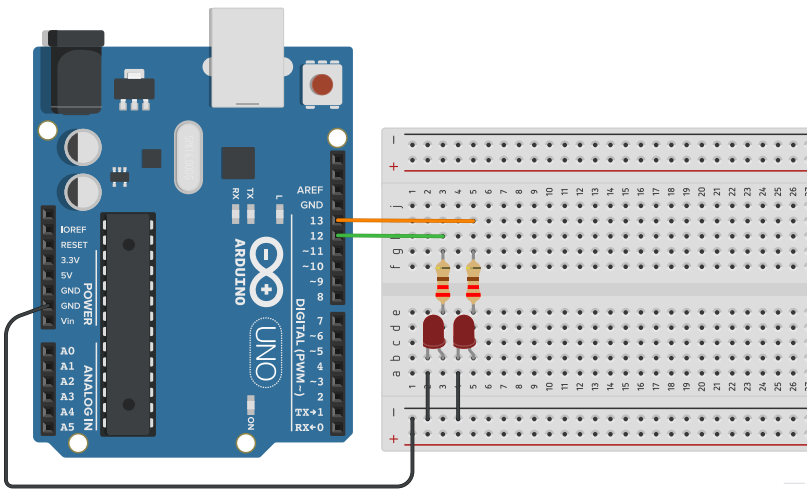
SIRA SİZDE

1. Devre bağlantısını altta verilen şekildeki gibi düzenleyip digitalWrite(13, 0); komutuyla lojik 0 göndererek LED'in yandığını gözlemleyiniz.




```
void setup() {
 pinMode(13, OUTPUT);
}
 void loop() {
 digitalWrite(13, 0);// LED'i 2 saniye yak.
 delay(2000);
 digitalWrite(13, 1);// LED'i 0,5 saniye söndür.
 delay(500);
}
```

2. Arduino'nun 12 numaralı pinine ikinci bir LED bağlayarak programı iki LED sırayla yanıp sönecek şekilde yeniden düzenleyiniz.



Sorular

1. delay(100); fonksiyonunun parametresi "100""den küçük değerler girildiğinde LED nasıl görünmektedir? Nedenini açıklayınız.

2. LED'e seri bağlı direnç niçin kullanılmıştır? Yazınız.

3. Arduino Uno'da kaç adet pin dijital giriş çıkış olarak ayarlanabilir? Belirtiniz.

Değişkenler

Değişkenler, içerisinde veri saklamak için kullanılan bellekte en az bir byte yer kaplayan ifadelerdir. int a = 5; komutuyla "a" isminde integer (tam sayı) bir değişken oluşturulmuş ve içerisine 5 verisi (sabiti) yüklenmiştir. "a" değişkeninin içeriği program içerisindeki çeşitli işlemlerle (matematiksel, port giriş bilgileri vb.) değişebilir. Değişkenler farklı veri tiplerinde tanımlanır. Veri tipine göre hafızada kapladıkları alan değişir.

Değişkenlere isim verme kuralları şunlardır:

1. Bir değişkenin ilk harfi rakam olamaz.

2. Değişken isimlerinde alt tire "_" hariç özel karakter bulunamaz.

3. Türkçe karakter kullanılamaz.

4. Boşluk içermez.

5. Arduino komutları (int, loop, float vb.) değişken adı olamaz.

6. Küçük büyük harf duyarlıdır. Örneğin "LED" ile "led" farklı değişken isimleridir.

Programın okunabilirliği için değişken isimleri anlamlı olmalıdır. Örneğin butonun durumunu kontrol etmek için butonDurum adında bir değişken oluşturulabilir. Değişken adı birden fazla kelimeden oluşuyorsa ya kelimeler arası alt çizgi kullanılır ya da daha çok tercih edilen ve kabul gören şekliyle baştaki kelime hariç diğer kelimelere büyük harfle başlanır. "ledPin", "triggerPinDurum", "potDegeri" gibi değişken isimleri Arduino söz dizimine benzer okunabilirliği yüksek örnek değişken isimleridir. Tablo 1.1'de değişken tipleri ve özellikleri görülmektedir.

Tablo 1.1: Değişken Tipleri ve Özellikleri

Değişken tipi	Hafızada kapladığı alan	Örnek	Açıklama
byte	1 byte	byte LED = 13; byte a = 0, b = 128, c=255;	0-255 arası işaretsiz sayılar

boolean (bool)	1 byte	bool butonDurum=0;
		boolean x = 0, y = 1;
		boolean kontrolBiti = True;
			0 (FALSE) veya 1 (TRUE) değerlerini alır.

char	1 byte	char karakter = 'A'; char karakter = 65; char karakterDizisi[] = "Arduino";	-128 ile 127 arasındaki değerleri alır.
unsigned char	1 byte	unsigned char a = 0, b = 255;	0-255 arası işaretsiz sayılar.
string	1 byte	string metin = "Arduino";	Metin işlemleri için kullanılır. (char dizisi)
int	2 byte	int a = 5, b = -618, c = 20140;	İşaretli sayılar için-32.768-32.767
unsigned int	2 byte	unsigned int sayim = 50000;	İşaretsiz sayılar için 0-65,535
word	2 byte	word sure = 60000;	İşaretsiz sayılar için 0-65,535

short	2 byte	short a = -272, b = 32000;	İşaretli sayılar için -32.768-32.767
long	4 byte	long zaman = -86400000;	İşaretli sayılar için -2.147.483.648-2.147.483.647
unsigned long	4 byte	unsigned long sure = 604800000;	İşaretsiz sayılar için 0-4.294.967.295
float	4 byte	float pi = 3.14; float oran = 0.618;	İşaretli ondalıklı sayılar -3,4028235E+38-3,4028235E+38
double	4 byte	double e = 2,718281; double f = 1.618;	İşaretli ondalıklı sayılar -3,4028235E+38-3,4028235E+38
			Due kartında 64 bitlik (8 byte) yer tutar.

NOT

Boolean veri tipinde "false" ifadesi sadece sıfır için geçerliyken " true" ifadesi sıfır dışındaki tüm sayılar için geçerlidir.

Değişkenlerin Dönüşümü

Değişken tipleri arasında işlem yapılırken bir değeri başka bir değere dönüştürmek gerekebilir. Bu dönüşüm işlemlerini derleyici bazen otomatik olarak yapsa da bazı zamanlarda dönüşümün programı yazan tarafından yapılması gerekebilir. Tablo 1.2'de değişkenlerin nasıl dönüştürüleceği görülmektedir.

Tablo 1.2: Değişkenlerin Dönüştürülmesi

char()	Parantez içerisine yazılan herhangi bir veriyi "char" veri türüne dönüştürür.
byte()	Parantez içerisine yazılan herhangi bir veriyi "byte" veri türüne dönüştürür.
int()	Parantez içerisine yazılan herhangi bir veriyi "int" veri türüne dönüştürür.
long()	Parantez içerisine yazılan herhangi bir veriyi "long" veri türüne dönüştürür.
float()	Parantez içerisine yazılan herhangi bir veriyi "float" veri türüne dönüştürür.

Değişken Niteleyiciler

Static: Sadece tanımlandığı fonksiyonda içeriği değişebilir. Diğer fonksiyonlar bu değişkenin içeriğini değiştiremez.

Const: Değişkenin içeriğinin hiçbir şekilde değiştirilemeyeceğini belirtir.

Volatile: Tanımlanan değişkenler Arduino'nun RAM bölgesine kaydedilir. Sadece interruptla değişir.

Değişkenlerin Kapsamı

Bir değişken tanımlandığı yere göre programın tamamında, belli bir fonksiyonda hatta tek bir döngüde faaliyet gösterebilir.

Global Değişkenler: Herhangi bir fonksiyonun içinde değil de en üstte tanımlanma blokunda tanımlanan değişkenlerdir. Program bitene kadar bellekten alınan alan geri verilmez ve değişkenin değeri korunmaya devam eder. Bütün fonksiyonlarda kullanılabilir. Yerel fonksiyonlarda aynı isimle farklı değişken oluşturulamaz.

Local Değişkenler: Bir blok veya fonksiyon içinde tanımlanır. Sadece tanımlandıkları alanda kullanılır. Bir fonksiyonda tanımlandılarsa o fonksiyon bittikten sonra alınan alan belleğe geri verilir. Farklı fonksiyonlar içinde aynı ada sahip yerel değişkenler tanımlamak mümkündür.

Örnek: Değişkenlerin faaliyet alanını gösterir.

```
int sayi;  // Tüm fonksiyonlar sayi değişkenini kullanabilir.
void setup() {
  // ...
}
void loop() {
  int i;    // “i” değişkeni sadece loop fonksiyonu içinde kullanılabilir.
  float f;  // “f” değişkeni sadece loop fonksiyonu içinde kullanılabilir. // ...
  for (int j = 0; j < 100; j++) {
    // “j” değişkeni sadece bu for döngüsü içinde kullanılır.
  }
}
```

Değişken Araçları

PROGMEM: Bilgiyi SRAM bellek yerine flash bellekte depolamaya yarar. Arduino programında pek çok char tanımlanırsa SRAM yetersiz kalabilir. Bu yüzden çok uzun metinler PROGMEM ile flash belleğe kaydedilir.

sizeof(): Değişkenlerin veya dizilerin byte sayısını verir.

Sabitler

Sabitler, Arduino dilinde önceden tanımlanmış ifadelerdir. Programların okunmasını kolaylaştırmak için kullanılır. Sabitler değişken ismi olarak kullanılamaz.

Mantıksal düzeyleri tanımlama: true ve false (Boolen sabitleri)

Pin seviyelerini tanımlama: HIGH ve LOW

Dijital pin modlarını tanımlama: INPUT, INPUT_PULLUP ve OUTPUT

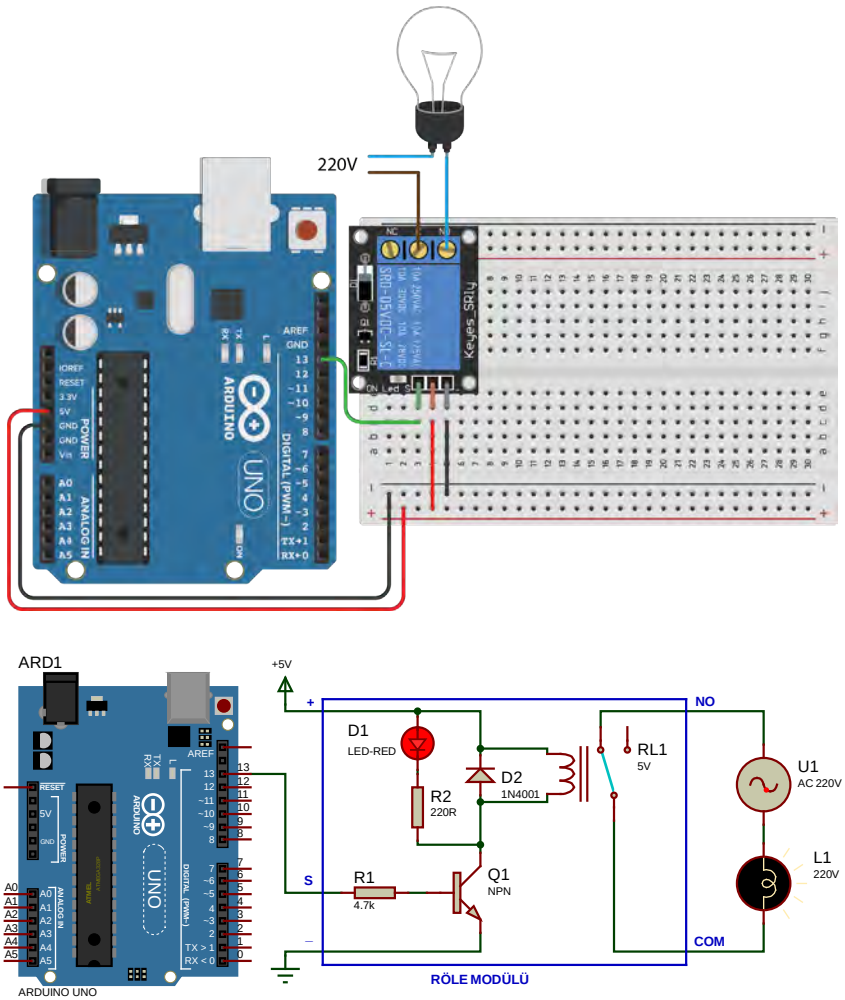
Dâhilî LED (13 numaralı pine bağlı): LED_BUILTIN

Uygulama Adı: Röleyle Dijital Çıkış Uygulaması No.: 2

Amaç: Röleyle dijital çıkış uygulaması yapmak.

Görsel 1.18'de röle modülünün beslemesi Arduino kartının 5 V ve GND uçlarından verilir. Modülün sinyal girişine Arduino'nun çıkış olarak ayarlanan 13 numaralı pini bağlanır. Modülün NO-COM uçlarına 220 V'la seri bağlanan lamba normalde sönüktür. Modülün sinyal girişi Arduino'nun 13 numaralı pininden lojik 1 (5 V) bilgisini aldığında röle içinde COM ucu NO kontağıyla temas eder ve lamba yanar. Modülün sinyal girişi Arduino'nun 13 numaralı pininden lojik 0 (GND) bilgisini aldığında röle içinde COM ucu NO kontağından ayrılır (NC kontağıyla temas eder.) ve lamba söner.

Arduino'nun 13 numaralı pini **byte** tipinde **role** değişkenine atanmıştır (**role** ifadesinin içine 13 sayısı yüklenmiştir.). Program içerisinde 13 sayısı yerine **role** ifadesi (değişkeni) kullanılır. **role** değişkeninin içeriği program içinde değişmesi istenmediğinden **const** ifadesi kullanılmıştır.



Görsel 1.18: Röleyle dijital çıkış uygulama şeması ve devresi

Röleyle dijital çıkış uygulama programı aşağıdaki gibidir:

```
const byte role = 13;      // Arduino'nun 13 nolu ucu röle değişkenine atandı.
void setup() {             // Setup ayarları // Portların giriş-çıkış olarak tanımladığı bölüm.
pinMode(role, OUTPUT);     // Role, yani 13 nolu pin çıkış olarak tanımlandı.
}
void loop() {              // Loop sonsuz döngüsü.
digitalWrite(role, HIGH); // Role değişkenini HIGH (1) yap.
delay(1000);               // Zaman gecikmesi // Bekle 1000 ms = 1 sn
digitalWrite(role, LOW);  // Role değişkenini LOW (0) yap.
delay(1000);               // Zaman gecikmesi // Bekle 1000 ms = 1 sn
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 1.18'deki devreyi kurunuz. Devre bağlantısında 220 V bulunduğundan devreye enerji vermeyiniz.
4. Programı yazıp Arduino'ya yükleyiniz.
5. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

DİKKAT Devre bağlantısında 220 V bulunduğundan devreye enerji vermeyiniz. Devreyi çalıştırmak için öğretmeninizden yardım isteyiniz.

3. Öğretmeniniz gözetiminde devrenin çalışmasını gözlemleyiniz.

SIRA SİZDE

1. delay(sure); şeklindeki bir komutu kullanabilmek için programın başındaki değişken tanımlamasını yapınız.
2. LED ile dijital çıkış uygulamasının programını değişken kullanarak yazınız.

Soru

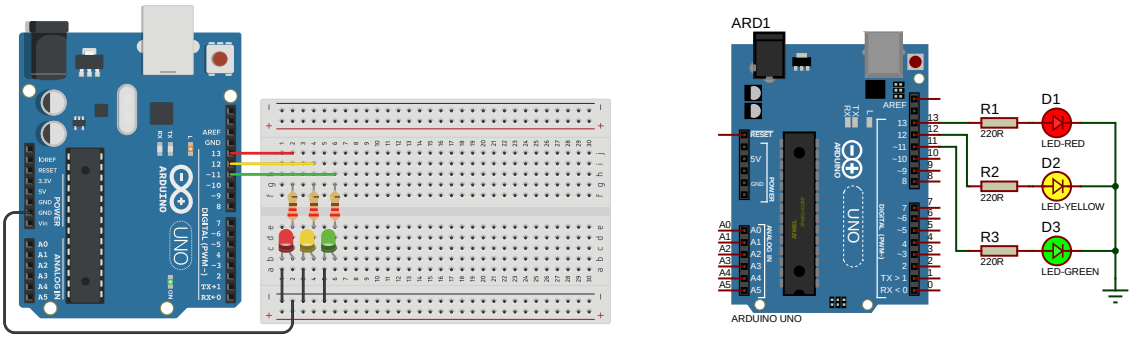
1. digitalWrite(13,1); komutu programdaki hangi satırla eş değerdir? Belirtiniz.

Uygulama Adı: Trafik Lambası Uygulaması — No.: 3

Amaç: Trafik lambası uygulaması yapmak.

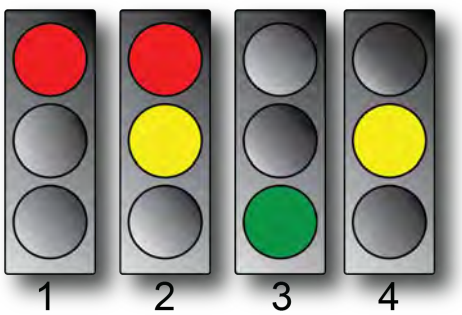
Görsel 1.18'de röle modülünün beslemesi Arduino kartının 5 V ve GND uçlarından verilir. Modülün Görsel 1.19'da 13, 12, 11 numaralı pinlere sırasıyla kırmızı, sarı ve yeşil LED bağlanarak programda çıkış olarak ayarlanmıştır. Her durum için yanan lambalara lojik 1 bilgisi gönderilirken sönmesi istenen lambalara lojik 0 bilgisi gönderilir.

Değişkenler (kirmizi, sari, yesil) **byte** (0-255 arası sayılar için) tipinde tanımlanarak **const** ifadesiyle içeriği değiştirilmeyecek şekilde ayarlanmıştır.



Görsel 1.19: Trafik lambası uygulama şeması ve devresi

Görsel 1.20'de trafik lambası çalışma sırası görülmektedir. Kırmızı ve yeşil ışık için 3 saniye, sarı ışık için 1 saniye yanma süresi verilmiştir.



Görsel 1.20: Trafik lambası çalışma sırası

Trafik lambası uygulama programı aşağıdaki gibidir:

```
const byte kirmizi = 13, sari = 12, yesil = 11;
void setup() {
  pinMode(kirmizi, OUTPUT); // Kırmızı LED.
  pinMode(sari, OUTPUT); // Sarı LED.
 pinMode(yesil, OUTPUT); // Yeşil LED.}
void loop() {
```

```
 digitalWrite(kirmizi, 1); // Kırmızı LED'i yak.
  digitalWrite(sari, 0);
  digitalWrite(yesil, 0);
  delay(3000);

  digitalWrite(kirmizi, 1); // Kırmızı LED'i yak.
  digitalWrite(sari, 1); // Sarı LED'i yak.
  digitalWrite(yesil, 0);
  delay(1000);

  digitalWrite(kirmizi, 0);
  digitalWrite(sari, 0);
  digitalWrite(yesil, 1); // Yeşil LED'i yak.
  delay(3000);

  digitalWrite(kirmizi, 0);
  digitalWrite(sari, 1); // Sarı LED'i yak.
  digitalWrite(yesil, 0);
  delay(1000);
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 1.19'daki devreyi kurunuz.
4. Programı yazıp Arduino'ya yükleyerek devrenin çalışmasını gözlemleyiniz.
5. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. boolean yak=1, sondur=0; şeklinde değişkenler tanımlayarak LED'e 1 ve 0 bilgisini "yak" ve "sondur" değişkenleriyle gönderiniz.

```
// …
void loop() {
  digitalWrite(kirmizi, yak); // Kırmızı LED'i yak.
  digitalWrite(sari, sondur);
  digitalWrite(yesil, sondur);
  delay(3000);
// …
```

Operatörler

İşlem yapmayı sağlayan işaretlere **operatör** denir. Tablo 1.3'te aritmetiksel, Tablo 1.4'te bileşik operatörler, Tablo 1.5'te karşılaştırma operatörleri, Tablo 1.6'da mantıksal (lojik) operatörler görülmektedir.

Tablo 1.3: Aritmetiksel Operatörler

İşlem	Operatör	Örnek	Açıklama
Toplama	+	a+b	a ile b'yi topla.
Çıkarma	-	a-b	a'dan b'yi çıkar.
Çarpma	*	a*b	a ile b'yi çarp.
Bölme	/	a/b	a'yı b'ye böl.
Mod	%	a%b	a'nın b'ye bölümünden kalan
Atama	=	a = b	b'yi a'ya ata (b'nin içeriğini a'ya yükle.).

Tablo 1.4: Bileşik Operatörler

İşlem	Operatör	Örnek	Açıklama
Bir arttır.	++	a++	a = a + 1
Bir azalt.	--	a--	a = a - 1
n arttır.	+=	a+= n	a = a + n
n azalt.	-=	a-= n	a = a-n
n'yle çarp.	*=	a*= n	a = a * n
n'ye böl.	/=	a/= n	a = a / n
n'ye göre mod	%=	a%= n	a = a % n

NOT

" ++" ve "--" operatörlerinin iki farklı kullanım şekli vardır.

```
y = x++; // x'i, y'ye yükler, sonra x'i bir artırır.
y = ++x; // x'i bir artırır, sonra x'i, y'ye yükler.
```

Örnek: ++'ın değişkenin önünde ve arkasında kullanımını gösterir.

```
x = 2;
y = ++x;  // x'in içeriği 3, y'nin içeriği 3.
y = x++;  // x'in içeriği 4 ancak y'nin içeriği hâlâ 3.
```

Tablo 1.5: Karşılaştırma Operatörleri

İşlem	Operatör	Örnek	Açıklama
Eşit mi?	==	a == b	a b'ye eşit mi?
Eşit değil mi?	!=	a != b	a b'ye eşit değil mi?
Büyük mü?	>	a > b	a b'den büyük mü?
Küçük mü?	<	a < b	a b'den küçük mü?
Büyük eşit mi?	>=	a >= b	a b'den büyük eşit mi?
Küçük eşit mi?	<=	a <= b	a b'den küçük eşit mi?

Tablo 1.6: Mantıksal (Lojik) Operatörler

İşlem	Operatör	Örnek	Açıklama
Ve (And)	&&	a>b && a>c	a, b'den ve c'den büyük mü?
Veya (Or)	||	a>b || a<c	a, b'den büyük veya c'den küçük mü?
Değil (Not)	!	!(a>b)	a, b'den büyük değil mi?

Bitsel Operatörler

Bitsel operatörler değişken ve veriler üzerinde bit seviyesinde ve, veya, değil, xor, bit kaydırma gibi işlemler yapmak için kullanılan operatörlerdir. Tablo 1.7'de a=60 ve b=13 ondalık sayıları için bit düzeyinde işlem örnekleri verilmiştir.

a = 111100 → 60 sayısının binary karşılığı

b = **001101** → 13 sayısının binary karşılığı

&_____

Q = 001100 → a & b = 12 sayısının binary karşılığı

Tablo 1.7: a=60 ve b=13 Ondalık Sayıları İçin Bit Düzeyinde İşlem Örnekleri

İşlem	Operatör	Örnek (a=60 b=13 için)	Açıklama
Ve (And)	&	(a & b) onluk 12 ikilik 0000 1100	Her iki bit 1 ise sonuç 1'dir.
Veya (Or)	|	(a | b) onluk 61 ikilik 0011 1101	Bitlerden biri 1 ise sonuç 1'dir.
Değil (Not)	~	(~a) onluk -60 ikilik 1100 0011	Sonuç her bitin tersidir.
Özel Veya (Xor)	^	(a ^ b) onluk 49 ikilik 0011 0001	Bitler farklıysa sonuç 1'dir.
Sola Kaydır (Shift Left)	<<	a << 2 onluk 240 ikilik 1111 0000	Operatörün sağındaki sayı kadar biti sola kaydırır (Sağa sayı kadar 0 eklenir.).
Sağa Kaydır (Shift Right)	>>	a >> 2 onluk 15 ikilik 0000 1111	Operatörün sağındaki sayı kadar biti sağa kaydırır (Sola sayı kadar 0 eklenir.).

Örnek: İki girişli lojik ifadeleri çıkışa bağlı LED'lerle gösterir.

```
boolean A; // A ve B iki girişli kapılar.
boolean B;
boolean AND;
boolean OR;
boolean A; // A ve B iki girişli kapılar.
boolean B;
boolean AND;
boolean OR;
boolean NAND;
```

```
boolean NOR;
boolean XOR;
boolean XNOR;
int butonA = 2; // Giriş butonları.
int butonB = 3;
int ledA = 4;// Giriş butonlarının durumunu gösteren LED'ler.
int ledB = 5;
int ledAND = 6;// Çıkış LED›leri.
int ledOR = 7;
int ledNAND = 8;
int ledNOR = 9;
int ledXOR = 10;
int ledXNOR = 11;
void setup() {
  pinMode(butonA, INPUT);
  pinMode(butonB, INPUT);
  pinMode(ledA, OUTPUT);
  pinMode(ledB, OUTPUT);
  pinMode(ledAND, OUTPUT);
  pinMode(ledOR, OUTPUT);
  pinMode(ledNAND, OUTPUT);
  pinMode(ledNOR, OUTPUT);
  pinMode(ledXOR, OUTPUT);
  pinMode(ledXNOR, OUTPUT);
}
void loop() {
  A = digitalRead(butonA);
  B = digitalRead(butonB);
  digitalWrite(ledA, A);
  digitalWrite(ledB, B);
 XNOR = !(A ^ B);
  XOR = A ^ B;
  NOR = !(A | B);
  OR = A | B;
  NAND = !(A & B);
  AND = A & B;
  digitalWrite(ledXNOR, XNOR);
  digitalWrite(ledXOR, XOR);
  digitalWrite(ledNOR, NOR);
  digitalWrite(ledOR, OR);
  digitalWrite(ledNAND, NAND);
  digitalWrite(ledAND, AND);
  delay(50);
```

```
boolean NOR;
boolean XOR;
boolean XNOR;
int butonA = 2; // Giriş butonları.
int butonB = 3;
int ledA = 4;// Giriş butonlarının durumunu gösteren LED'ler.
int ledB = 5;
int ledAND = 6;// Çıkış LED›leri.
int ledOR = 7;
int ledNAND = 8;
int ledNOR = 9;
int ledXOR = 10;
int ledXNOR = 11;
void setup() {
  pinMode(butonA, INPUT);
  pinMode(butonB, INPUT);
  pinMode(ledA, OUTPUT);
  pinMode(ledB, OUTPUT);
  pinMode(ledAND, OUTPUT);
  pinMode(ledOR, OUTPUT);
  pinMode(ledNAND, OUTPUT);
  pinMode(ledNOR, OUTPUT);
  pinMode(ledXOR, OUTPUT);
  pinMode(ledXNOR, OUTPUT);
}
void loop() {
  A = digitalRead(butonA);
  B = digitalRead(butonB);
  digitalWrite(ledA, A);
  digitalWrite(ledB, B);
 XNOR = !(A ^ B);
  XOR = A ^ B;
  NOR = !(A | B);
  OR = A | B;
  NAND = !(A & B);
  AND = A & B;
  digitalWrite(ledXNOR, XNOR);
  digitalWrite(ledXOR, XOR);
  digitalWrite(ledNOR, NOR);
  digitalWrite(ledOR, OR);
  digitalWrite(ledNAND, NAND);
  digitalWrite(ledAND, AND);
  delay(50);
}
```

Döngüler

Koşul sağlanana kadar tekrar edilmesi istenen komutlar için kullanılır. Döngülerde koşul her zaman boolean (true-false) bir ifadedir.

While Döngüsü

While döngüsünde program koşula bakar, koşul doğruysa döngü içine girer ve koşul yanlış olana kadar içerdeki işlemleri tekrarlar. Eğer içeride koşulu bozacak (false) değişiklikler yapılmazsa koşul sürekli doğru olacağından program sonsuz döngüye girer. Döngü içindeki kodlarda koşulu bozacak durum, bir artırım veya sensör bilgisi gibi haricî bir durum olur.

```
while(koşul) {
// Tekrarlanan işlemler.
}
```

Örnek: Program dâhilî LED'i beş kez yakıp söndürür.

```
void setup() {
  int sayac = 0; // Döngüden önce sayaç değişkeninin içeriği 0.
  while (sayac < 5) {
    digitalWrite(13, HIGH); // Yak.
    delay(200);
    digitalWrite(13, LOW); // Söndür.
    sayac++; // Sayaç değişkenini 1 artır.
  }
}
void loop() {
}
```

Örnek: Program döngü içinde sensörün durumunu okumaktadır.

```
while (sensorDurum) { // Sensörden gelen bilgi 1 olduğu sürece döngüde kal.
  digitalWrite(role, 1); // Röleye enerji ver.
  sensorDurum = digitalRead(sensor); /Sensörü oku.
}
```

do...while Döngüsü

While döngüsüyle benzerdir. Farkı do...while döngüsünde önce döngü içindeki işlem yapılır. Sonra koşula bakılır. Komutlar bir kez mutlaka yapılmış olur.

```
do {
// Tekrarlanan işlemler.
}
while(koşul);
```

Örnek: Program dâhilî LED'i on kez yakıp söndürür.

```
i = 1;
do {
digitalWrite(13, HIGH);
```

```
delay(1000);
digitalWrite(13, LOW);
delay(1000);
i++;
}while(i <= 10);
```

Örnek: Program döngü içinde sensörün durumunu okumaktadır.

```
int x = 0;
do {
   x = sensorOku();  // Sensörü kontrol et.
} while (x < 100);
```

For Döngüsü

For ifadesi koşul sağlanana kadar kodların tekrarlanması için kullanılır. Başlangıç değeri, koşul ve artırım değeri olarak üç bölümden oluşur.

```
for(Başlangıç; Koşul; Artırım) {
// Tekrarlanan işlemler.
}
```

Başlangıç: Başlangıçta bir kere çalışır. İhtiyaç yoksa boş bırakılabilir.

Koşul: Döngü koşul sağlandığı sürece tekrar eder. Koşul sağlanıyorsa (true) döngüye girilir ve komutlar icra edilir. Döngü tamamlandığında koşulu tekrar kontrol eder. Koşul sağlanmıyorsa (false) döngüden çıkılır.

Artırım: Koşul her sağlandığında artırım işlemi de gerçekleştirilir.

Örnek: Yürüyen ışık programıdır.

```
byte i;
void setup()
{ for (i = 0; i < 10; i++)
  {
    pinMode(i, OUTPUT); // 0 - 9 pinleri çıkış yap.
  }
}
void loop()
{
  for (i = 0; i < 10; i++) // 0. pinden 9. pine kadar olan LED'leri sırayla...
  {
    digitalWrite(i, HIGH); // Yak.
    delay(200);
    digitalWrite(i, LOW); // Söndür.
  }
}
```

Örnek: İç içe kullanılan for döngüsünde ilk beş LED üç kez yanıp söner. Sonra son beş LED üç kez yanıp söner. Bu işlem on kere tekrar eder.

```
void setup() {
  byte i, x;
  int sure = 100;
  for (i = 0; i < 10; i++) {
    pinMode(i, OUTPUT); // 0 - 9 pinleri çıkış yap.
  }
  for (byte j = 0; j < 10; j++) { // Alttaki tüm kodları 10 kez çalıştır.
    for (x = 0; x < 3; x++) { // İçerdeki iki for döngüsünü üç kez tekrarla.
      for (i = 0; i < 5; i++) { // İlk beş LED'i yak. (Hepsi aynı anda yanar görünür.)
        digitalWrite(i, 1);
      }
      delay(sure);
      for (i = 0; i < 5; i++) { // İlk beş LED'i söndür.
        digitalWrite(i, 0);
      }
      delay(sure);
    }
    for (x = 0; x < 3; x++) { // İçerdeki iki for döngüsünü üç kez tekrarla.
      for (i = 5; i < 10; i++) { // Son beş LED'i yak.
        digitalWrite(i, 1);
      }
      delay(sure);
      for (i = 5; i < 10; i++) { // Son beş LED'i söndür.
        digitalWrite(i, 0);
      }
      delay(sure);
    }
  }
}
void loop() {}
```

Break

Normal döngü koşulunu atlayarak for, while veya do…while döngüsünden çıkmak için kullanılır. Ayrıca bir switch case ifadesinden çıkmak için de kullanılır.

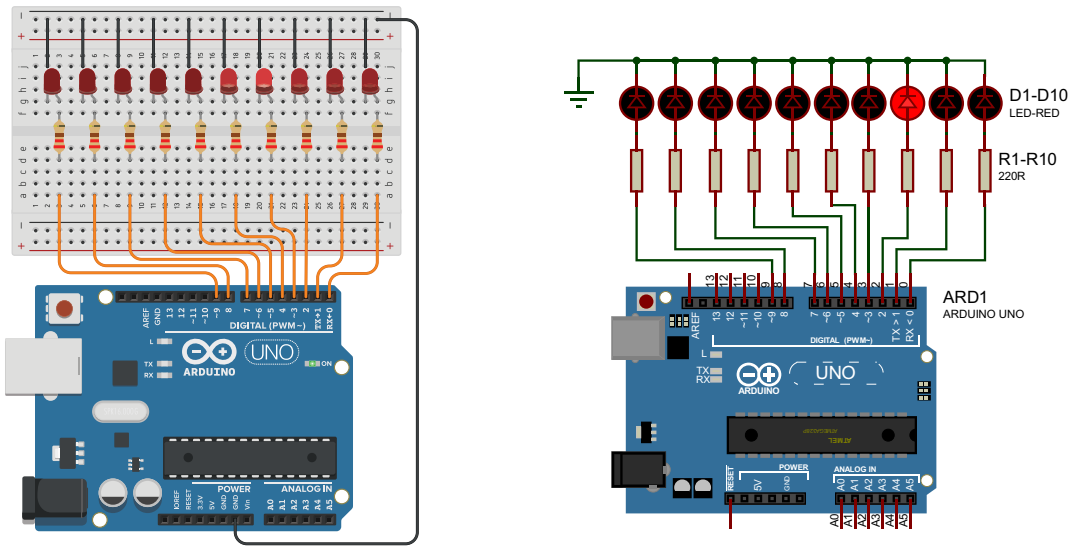
Örnek: Break kullanarak döngüden çıkar.

```
int esik = 40;
for (int x = 0; x < 255; x++) {
  analogWrite(PWMpin, x);
  deger = analogRead(sensorPin);
  if (deger > esik) {
    x = 0;
    break;
  }
  delay(50);}
```

Uygulama Adı: Kara Şimşek Uygulaması No.: 4

Amaç: **Kara şimşek uygulaması yapmak.**

Arduino'nun on adet çıkış pinine bağlanmış LED'i sırayla tek tek yakıp söndürmek için döngü komutları kullanılmıştır. Döngü komutları kullanılmazsa program tekrar eden komutlar nedeniyle çok uzun olacaktır. Uygulamada 0. pinden 9. pine kadar olan LED'ler sırayla yakılıp söndürülmektedir. Geriye dönüşte ise 8. pinden 1. pine kadar olan LED'ler sırayla yakılıp söndürülmektedir. "sure" değişkeninin içeriği değiştirilerek LED'lerin çalışma hızı ayarlanır (Görsel 1.21).



Görsel 1.21: Kara şimşek uygulaması şema ve devresi

Kara şimşek uygulama programı aşağıdaki gibidir:

```
byte i;
int sure = 100;

void setup(){
  for (i = 0; i <= 9; i++){
    pinMode(i, OUTPUT); // 0 - 9 pinleri çıkış yap.
  }
}

void loop(){
  for (i = 0; i <= 9; i++){
    digitalWrite(i, HIGH);
    delay(sure);
    digitalWrite(i, LOW);
  }

 for (i = 8; i > 0; i--){ // İlk ve son LED dönüşlerde bir kez yanar.
    digitalWrite(i, HIGH);
```

```
 delay(sure);
    digitalWrite(i, LOW);
  }
}
```

DİKKAT Arduino'nun 0. (Rx) ve 1. (Tx) pinleri USB kabloyla bilgisayara bağlıdır. Yükleme tuşuna basıldığında bu pinler kullanılır. 0. (Rx) pini devreye bağlıysa program karta yüklenirken hata verir. 0. pindeki **jumper** kablosu kaldırıldıktan sonra yükle düğmesine basılır. Program yüklendikten sonra **jumper** kablosu 0. pine tekrar bağlanır.

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 1.21'deki devreyi kurunuz.
4. Programı yazıp Arduino'ya yükleyerek devrenin çalışmasını gözlemleyiniz.
5. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Uygulamayı for döngüsü kullanmadan yazınız.

```
int sure = 100;
void setup() {
  pinMode(0, OUTPUT);
  pinMode(1, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
}
void loop() {
  digitalWrite(0, HIGH);
  delay(sure);
  digitalWrite(0, LOW);
  digitalWrite(1, HIGH);
  delay(sure);
  digitalWrite(1, LOW);
 digitalWrite(2, HIGH);
  delay(sure);
```

```
digitalWrite(2, LOW);

digitalWrite(3, HIGH);    delay(sure);
digitalWrite(3, LOW);
digitalWrite(4, HIGH);
delay(sure);
digitalWrite(4, LOW);
digitalWrite(5, HIGH);
delay(sure);
digitalWrite(5, LOW);
digitalWrite(6, HIGH);
delay(sure);
digitalWrite(6, LOW);
digitalWrite(7, HIGH);
delay(sure);
digitalWrite(7, LOW);
digitalWrite(8, HIGH);
delay(sure);
digitalWrite(8, LOW);
digitalWrite(9, HIGH);
delay(sure);
digitalWrite(9, LOW);
digitalWrite(8, HIGH);
delay(sure);
digitalWrite(8, LOW);
digitalWrite(7, HIGH);
delay(sure);
digitalWrite(7, LOW);
digitalWrite(6, HIGH);
delay(sure);
digitalWrite(6, LOW);
digitalWrite(5, HIGH);
delay(sure);
digitalWrite(5, LOW);
digitalWrite(4, HIGH);
delay(sure);
digitalWrite(4, LOW);
digitalWrite(3, HIGH);
delay(sure);
digitalWrite(3, LOW);
digitalWrite(2, HIGH);
delay(sure);
digitalWrite(2, LOW);
digitalWrite(1, HIGH);
delay(sure);
```

```
  digitalWrite(1, LOW);
  }
```

2. Uygulamayı for döngüsü kullanarak yirmi adet LED'le gerçekleştiriniz.

3. Aşağıdaki çakar uygulamasında ilk beş LED üç kez yanıp söner. Sonra son beş LED üç kez yanıp söner. Döngü sürekli tekrar eder.

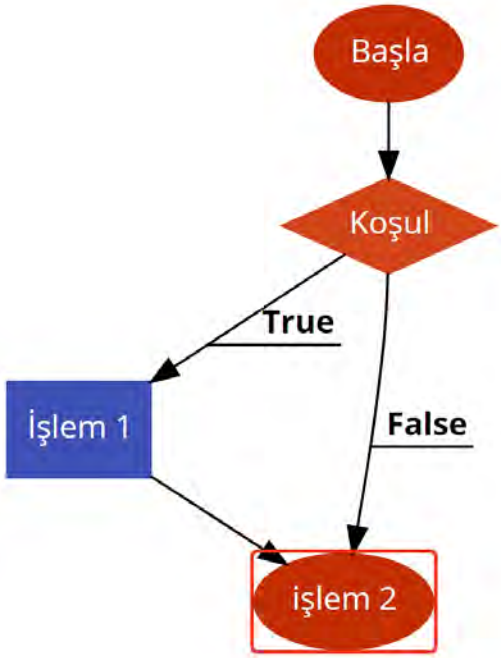
```
byte i, x;
int sure = 100;
void setup() {
  for (i = 0; i < 10; i++) {
    pinMode(i, OUTPUT); // 0 - 9 pinleri çıkış yap.
  }
}
void loop() {
  for (x = 0; x < 3; x++) { // İçerdeki iki for döngüsünü üç kez tekrarla.
    for (i = 1; i < 6; i++) { // İlk beş LED›i yak. (Hepsi aynı anda yanar görünür.)
      digitalWrite(i, 1);
    }
    delay(sure);

    for (i = 1; i < 6; i++) { // İlk beş LED›i söndür.
      digitalWrite(i, 0);
    }
    delay(sure);
  }
  for (x = 0; x < 3; x++) { // İçerdeki iki for döngüsünü üç kez tekrarla.
    for (i = 6; i < 11; i++) { // Son beş LED'i yak.
      digitalWrite(i, 1);
    }
    delay(sure);

    for (i = 5; i < 11; i++) { // Son beş LED'i söndür.
      digitalWrite(i, 0);
    }
    delay(sure);
  }
}
```

Karar Yapıları

Koşula bağlı olarak iki veya daha fazla seçenekten birine dallanma işlemi gerçekleştiren komutlardır. Program şartlı olarak hangi işlemi yapacağına karar verir. Koşullar karşılaştırma operatörleriyle sınanır.

if (eğer) Yapısı

Koşula bağlı olarak tek bir işlemi yerine getiren yapıdır. Koşul ya doğrudur (true) ya da yanlıştır (false). Eğer koşul yanlışsa herhangi bir işlem yapılmaz. Koşul boolean bir ifadedir (Görsel 1.22).



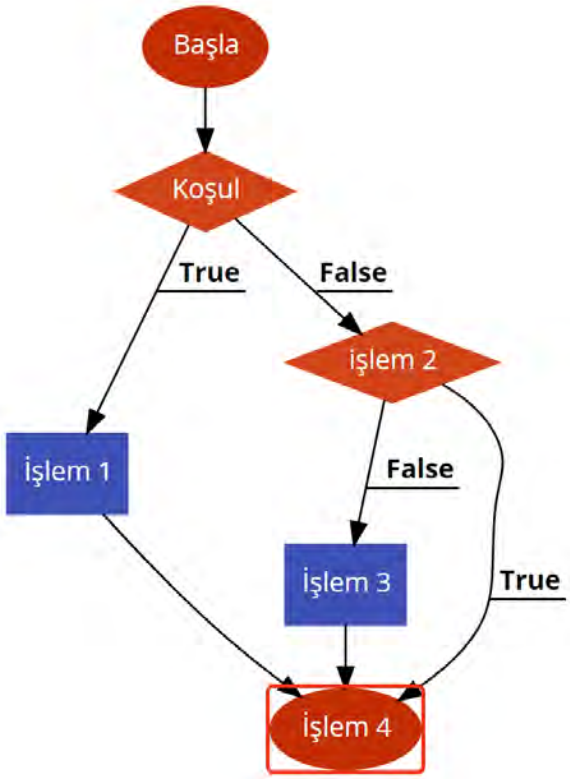
Görsel 1.22: Koşula bağlı işlem

Örnek: if yapısının kullanım şekilleri gösterilmiştir.

```
if (x > 120) digitalWrite(LED, HIGH);
if (x > 120)
digitalWrite(LEDpin, HIGH);

if (x > 120) {digitalWrite(LED, HIGH);}

if (x > 120) {
  digitalWrite(LED1, HIGH);
  digitalWrite(LED2, HIGH);
}
```

if-else Deyimi

Birden fazla koşulu test etmek için kullanılır. "else" komutuyla biten yapının sonunda koşul yoktur, varsayılan işlem vardır.



Görsel 1.23: Varsayılan koşula bağlı işlem

Görsel 1.23'te işlem 3 varsayılan işlem, işlem 4 koşul dışındaki koddur. Yapının sonunda "else" kullanılmazsa varsayılan işlem de kalkar.

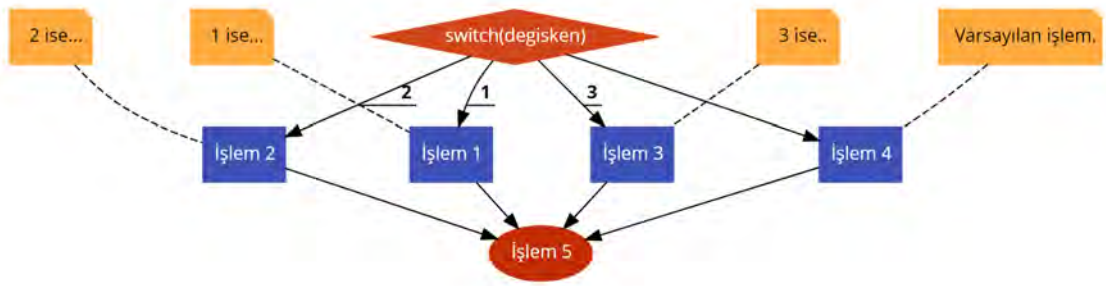
Örnek: if-else kullanımını gösterir.

```
if (sicaklik >= 70) {
  // Sistemi kapat.
}
else if (sicaklik >= 60) { // 60 <= sicaklik< 70
  // Dikkat riskli bölge.
}
else { // sicaklik< 60 (Koşul yok. Varsayılan işlem.)
  // Güvenli. Çalışmaya devam.
}
```

switch - case Deyimi

Sınanan değişkene göre işlemlerin seçiminde kullanılır. İç içe if-else yapısıyla benzerdir. Switch-case yapısı switch alanında belirtilen değişken değerini her bir case yapısındaki değerle karşılaştırır. Karşılaştırma sonucunda koşulu sağlanan case alanındaki kod çalıştırılır ve break komutuyla switch-case yapısından çıkılır. Hiçbir case koşulu sağlanmamışsa varsayılan (default) kodlar çalıştırılır.

Default kısmı isteğe bağlı olarak kullanılmayabilir (Görsel 1.24).



Görsel 1.24: Switch-case kullanımı

Örnek: switch-case kullanımını gösterir.

```
switch (degisken) {
case 1:
  // Değişken 1'e eşitse buradaki işlemler yapılır.
  break;
case 2:
  // Değişken 1'e eşitse buradaki işlemler yapılır.
  break;
default:
  // Hiçbir eşleşme olmazsa varsayılan işlemi yapar.
  // Varsayılan işlem tercihe bağlıdır.
  break;
}
```

NOT

Karar ve döngü işlemlerinde tek bir komut varsa { } (süslü parantezler) kullanılmasa da olur.

Diziler

Diziler indis numarasıyla ulaşılan değişken topluluğudur. Aşağıda dizi oluşturma örnekleri verilmiştir.

Örnek: Dizi tanımlamalarını gösterir.

```
int sayilar[6];
int pinler[] = {2, 4, 8, 3, 6};
int sensorDegerleri[6] = {2, 4, -8, 3, 2};
char mesaj[6] = "hello";
```

Diziler indis numarasıyla ulaşılan değişken topluluğudur. Aşağıda dizi oluşturma örnekleri verilmiştir.

Örnek: Dizi tanımlamalarını gösterir.

```
int sayilar[6];
int pinler[] = {2, 4, 8, 3, 6};
int sensorDegerleri[6] = {2, 4, -8, 3, 2};
char mesaj[6] = “hello”;
```

pnler[] dizisinde dizi uzunluğu (eleman sayısı) belirtilmemiştir. Derleyici elemanları sayarak uygun uzunluğu kendi oluşturur.

Dizilere Erişim

Diziler indis şeklindedir ve ilk indis 0'la başlar. Yukarıdaki örnekte sensorDegerleri[0] == 2'ye, sensorDegerleri[2] == -8'e eşittir. Aşağıdaki örnekte dizi elemanına değer atama görülmektedir.

```
sensorDegerleri[0] = 10 // Dizinin ilk elemanına 10 yüklenir.
x = sensorDegerleri[4] // x değişkenine 2 yüklenir.
```

Diziler for döngüsünde sıklıkla kullanılır. İçerik dizinin elemanlarına ulaşılacak şekilde yapılandırılır. Örnek programda ledler[] dizisinin elemanları LED'lerin bağlı olduğu pin numaralarını içermektedir. Arduino'nun 0. (Rx) ve 1. (Tx) pinleri yükleme esnasında “sök tak” yapmamak için tercih edilmemiştir.

Örnek: Dizi kullanılarak yapılmış kara şimşek programını gösterir.

```
byte i;
byte ledler[10] = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11}; // On adet LED’in bağlandığı pinler.
int sure = 100;

void setup() {
  for (i = 2; i <= 11; i++)
    pinMode(i, OUTPUT); // 2 - 11 pinleri çıkış yap.
}

void loop()
{
  for (i = 0; i <= 9; i++)  {
    digitalWrite(ledler[i], HIGH);
    delay(sure);
    digitalWrite(ledler[i], LOW);
  }
  for (i = 8; i > 0; i--) { // İlk ve son LED dönüşlerde bir kez yanar.
    digitalWrite(ledler[i], HIGH);
    delay(sure);
    digitalWrite(ledler[i], LOW);
  }
}
```

Çok Boyutlu Diziler

Her elemanı bir dizi olan dizilere **çok boyutlu dizi** denir. İki boyutlu dizilere **matris** denir. Matrisler satır ve sütunlardan oluşan tablolar gibidir.

Örnek: Çok boyutlu dizi tanımlamalarını gösterir.

```
int dizi[3][4] = {{10, 5, 12, 20}, {7, 3, 14, 22}, {8, 7, 8, 24}};

// Dizinin her elemanına atama yapıldı.

int matris[4][3] = {{25, 33, 12}, {87, 66, 63}, {45, 90, 34}, {22, 46, 14}};

int x = dizi[0][3]; // x'e 10 sayısı yüklendi.

int y = matris[3][0]; // y'ye 22 sayısı yüklendi.
```

Örnek: Matrisle yedi segment display uygulamasını gösterir.

```
const byte a = 6, b = 5, c = 4, d = 3, e = 2, f = 1, g = 0 ; // Segmentlerin bağlandığı pinler.
byte rakamSegmentleri[10][7] = { // Her rakam için segment durumları.

  // g, f, e, d, c, b, a
  { 0, 1, 1, 1, 1, 1, 1 }, // = 0
  { 0, 0, 0, 0, 1, 1, 0 }, // = 1
  { 1, 0, 1, 1, 0, 1, 1 }, // = 2
  { 1, 0, 0, 1, 1, 1, 1 }, // = 3
  { 1, 1, 0, 0, 1, 1, 0 }, // = 4
  { 1, 1, 0, 1, 1, 0, 1 }, // = 5
  { 1, 1, 1, 1, 1, 0, 1 }, // = 6
  { 0, 0, 0, 0, 1, 1, 1 }, // = 7
  { 1, 1, 1, 1, 1, 1, 1 }, // = 8
  { 1, 1, 0, 1, 1, 1, 1 }  // = 9
};

void setup() {
  for (byte i = 0; i <= 6; i++) { // Yedi adet pin çıkış olarak ayarlandı.
    pinMode(i, OUTPUT); // 0 - 6 pinleri çıkış yap.
  }
}

void loop() {

for (byte rakam = 0; rakam <= 9; rakam++) {
for (byte segment = 0; segment < 7; segment++) {
      digitalWrite(segment , rakamSegmentleri[rakam][segment]);
    }
    delay(200);
  }
}
```

Fonksiyonlar

Program içinde pek çok yerde tekrar eden kod bloku varsa okunabilirliği arttırmak için bu kod bloku fonksiyon olarak diğer fonksiyonların dışında tanımlanır ve istenilen yerde bu fonksiyon çağrılarak kod bloku çalıştırılır. Dört çeşit fonksiyon tanımlaması mevcuttur.

Değer Almayan ve Değer Döndürmeyen Fonksiyonlar

Bu tip fonksiyonların başına void ifadesi yazılır. Değer almayan fonksiyonların parantezlerinin içi boştur. Aşağıda blink (yanıp sönen LED) uygulamasının fonksiyona dönüştürülmüş şekli verilmiştir.

Örnek: Değer almayan ve değer döndürmeyen blink() fonksiyonunu gösterir.

```
const byte LED = 13;
void setup() {
  pinMode(LED, OUTPUT);
}
void loop() {
  blink(); // Blink fonksiyonunu çağır.
}
void blink() { // Değer almayan ve değer döndürmeyen.
  digitalWrite(LED, HIGH);
  delay(1000);
  digitalWrite(LED, LOW);
  delay(1000);
}
```

Değer Alan ve Değer Döndürmeyen Fonksiyonlar

Bu tip fonksiyonların başına void ifadesi yazılır. Fonksiyon parantezlerinin içine belirtilen değişken tipinde veri girilir.

Örnek: Değer alan ve değer döndürmeyen blink() fonksiyonunu gösterir.

```
const byte LED = 13;
void setup() {
 pinMode(LED, OUTPUT);
}
void loop() {
  blink(500); // 500 değerini almış blink foksiyonunu çağır.
}
void blink(int sure) { // int tipinde değer alan, değer döndürmeyen.
  digitalWrite(LED, HIGH);
  delay(sure);
  digitalWrite(LED, LOW);
  delay(sure);
}
```

Örnek: Yedi segment display uygulamasıdır.

```
const byte a = 6, b = 5, c = 4, d = 3, e = 2, f = 1, g = 0;
// Segmentlerin bağlandığı pinler.
int sure = 200;

void setup() {
  for (byte i = 0; i <= 6; i++) {
    pinMode(i, OUTPUT); // 0 - 6 pinleri çıkış yap.
  }
}

void loop() {
  for (byte rakam = 0; rakam <= 9; rakam++) { // Rakamları ileri say.
    yazdir(rakam); // Girilen rakamı yazdir() fonksiyonuna gönder.
    delay(sure);
  }
  for (byte rakam = 8; rakam >= 1; rakam--) { // Rakamları geri say.
    yazdir(rakam);
    delay(sure);
  }
}

void yazdir(byte rakam) { // byte tipinde değer alan, değer döndürmeyen.
  switch (rakam) { // Rakam değişkenini seçeneklerden bul.
    case 0 :
      digitalWrite(a, HIGH);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, HIGH);
      digitalWrite(e, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, LOW);
      break;
    case 1:
      digitalWrite(a, LOW);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, LOW);
       digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
      break;
    case 2:
      digitalWrite(a, HIGH);
      digitalWrite(b, HIGH);
       digitalWrite(c, LOW);
```

```
   digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
  break;
case 3:
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
  break;
case 4:
  digitalWrite(a, LOW);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
  break;
case 5:
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
  break;
case 6:
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
   digitalWrite(g, HIGH);
  break;
case 7:
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
```

```
      digitalWrite(c, HIGH); digitalWrite(LED, HIGH);
  delay(1000);
  digitalWrite(LED, LOW);
  delay(1000);
  int saniye = millis() / 1000; // Programın milisaniye cinsinden çalışma süresini 1000'e böl.
 return saniye; // saniye değişkenini döndür.
}
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
      break;
    case 8:
      digitalWrite(a, HIGH);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, HIGH);
      digitalWrite(e, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
      break;
    case 9:
      digitalWrite(a, HIGH);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
```


Değer Almayan ve Değer Döndüren Fonksiyonlar

Fonksiyonun değer döndürmesi (return) bir çıktı oluşturduğunu gösterir. Değer döndüren fonksiyonlar bir değişkene atanabilir. Değer döndüren fonksiyonların başına döndürülen verinin değişken tipi yazılır. Değer almadığı için fonksiyon parantezlerinin içi boştur.

Örnek: Değer almayan ve değer döndüren blink() fonksiyonunu gösterir.

```
const byte LED = 13;
void setup() {
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  blink(); // blink fonksiyonunu çağır.
  int zaman = blink(); // Değer döndüren fonksiyonlar bir değişkene atanabilir.
  Serial.print(zaman);
  Serial.println(" saniyedir program çalışmaktadır.");
}
int blink() { // Değer almayan ancak int tipinde değer döndüren.
```

```
      digitalWrite(d, HIGH);
      digitalWrite(e, LOW);
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
      break;
  }
}
```

Değer Alan ve Değer Döndüren Fonksiyonlar

Bu tip fonksiyonların başına döndürülen verinin değişken tipi yazılır. Fonksiyon parantezlerinin içine belirtilen değişken tipinde veri girilir.

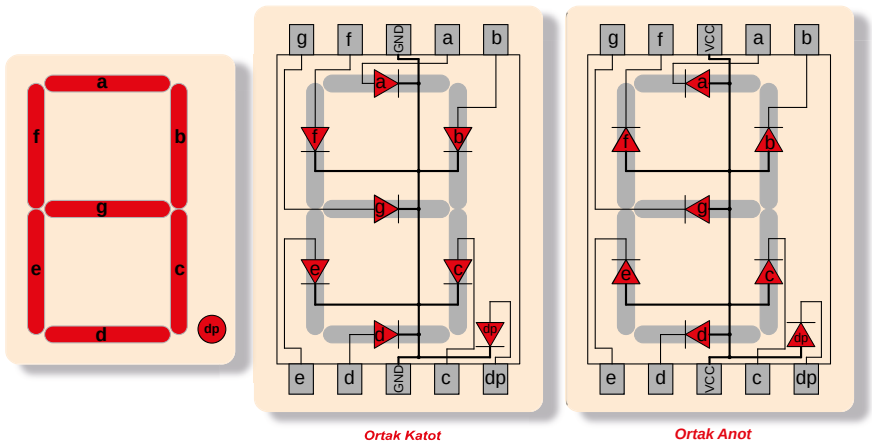
Örnek: Değer alan ve değer döndüren blink() fonksiyonunu gösterir.

```
const byte LED = 13;
void setup() {
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  blink(100); // 100 değerini almış blink foksiyonunu çağır.
  Serial.print(blink(100)); // Blink fonksiyonundan dönen değeri yazdır.
  Serial.println(" saniyedir program çalışmaktadır.");
}
int blink(int sure) {
  digitalWrite(LED, HIGH);
  delay(sure);
  digitalWrite(LED, LOW);
  delay(sure);
  return millis() / 1000; // millis() / 1000 işleminin sonucunu döndür.
}
```

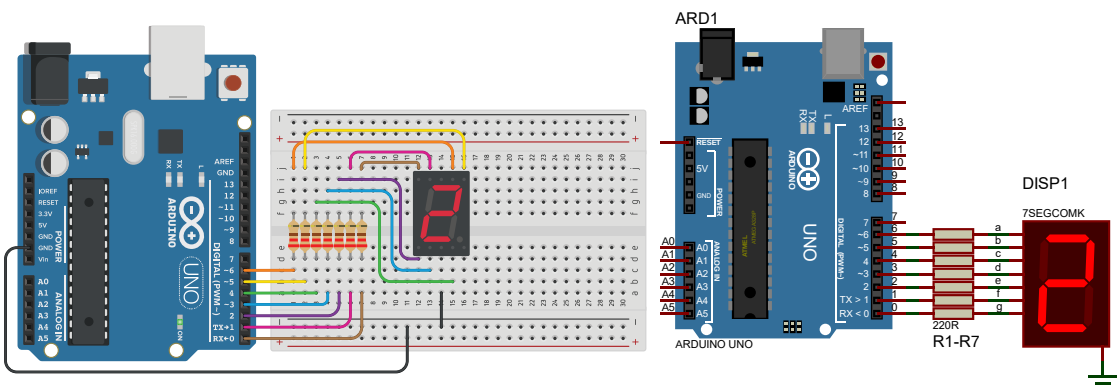
Uygulama Adı: Segment Displayle Dijital Çıkış Uygulaması No.: 5

Amaç: 7 segment displayle dijital çıkış uygulaması yapmak.

Görsel 1.25a'da ortak katotlu ve ortak anotlu 7 segment displayin yapısı, Görsel 1.25b'de ise dijital çıkış şeması ve devresi görülmektedir. Displayin a, b, c, d, e, f, g segmentleri sırasıyla Arduino'nun 6, 5, 4, 3, 2, 1, 0 numaralı pinlerine bağlıdır. Ortak katot Arduino'nun GND ucuna bağlıdır. Karar yapısı olarak switch case kullanılmıştır. Rakam değişkenin içeriğine göre ilgili komut kümesi çalıştırılarak dijital çıkışların durumuna göre rakamlar displayde yanmaktadır. For döngüsüyle 0,2 saniyede bir rakamlar artırılmaktadır. Tablo 1.8'de ortak katot display giriş kodları verilmiştir.



Görsel 1.25a: Ortak katotlu ve ortak anotlu 7 segment displayin yapısı



Görsel 1.25b: 7 segment displayle dijital çıkış şeması ve devresi

Tablo 1.8: Ortak Katot Display Giriş Kodları

Pin	6	5	4	3	2	1	0
Rakam	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

7 segment displayle dijital çıkış uygulama programı aşağıdaki gibidir:

```
const byte a = 6, b = 5, c = 4, d = 3, e = 2, f = 1, g = 0;
void setup() {
  for (byte i = 0; i <= 6; i++)
  {
    pinMode(i, OUTPUT); // 0 - 6 pinleri çıkış yap.
  }
}
void loop() {
  for (byte rakam = 0; rakam <= 9; rakam++) { // Rakamları yukarı say.
    switch (rakam) { // Rakam değişkenini seçeneklerden bul.
      case 0 :
        digitalWrite(a, HIGH);
        digitalWrite(b, HIGH);
        digitalWrite(c, HIGH);
        digitalWrite(d, HIGH);
        digitalWrite(e, HIGH);
        digitalWrite(f, HIGH);
        digitalWrite(g, LOW);
        break;
      case 1:
        digitalWrite(a, LOW);
        digitalWrite(b, HIGH);
        digitalWrite(c, HIGH);
   digitalWrite(d, LOW);
        digitalWrite(e, LOW);
        digitalWrite(f, LOW);
        digitalWrite(g, LOW);
        break;
```

```
case 2:
       digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, LOW);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
  break;
case 3:
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
  break;
case 4:
  digitalWrite(a, LOW);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
  break;
case 5:
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
  break;
case 6:
   digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
  break;
```

```
                    case 7:
      digitalWrite(a, HIGH);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
      break;
    case 8:
      digitalWrite(a, HIGH);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, HIGH);
      digitalWrite(e, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
      break;
    case 9:
      digitalWrite(a, HIGH);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, HIGH);
      digitalWrite(e, LOW);
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
      break;
  } // switch case bitti.
 } // for döngüsü bitti.
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 1.25'teki devreyi kurunuz.
4. Programı yazıp Arduino'ya yükleyiniz. Displayin çalışmasını gözlemleyiniz.
5. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

Bir programı yazmanın pek çok yolu vardır. Bu uygulamanın programı switch case, iki boyutlu dizi vb. kullanılarak yazılabilir. Bunlardan biri olan port komutlarıyla yapılmış programı deneyiniz.

Aşağıdaki Arduino kodunda DDRD port registerla portları giriş çıkış olarak tanımlama yapılmıştır. DDRD = B01111111; komutuyla port register'a 8 bitlik binary sayı atayarak 0 ile 7. port giriş, 1 ile 0-6. portlar çıkış olarak ayarlanmıştır.

PORTD = B01111110; komutuyla displaye 0 sayısı yazdırılır. Bu komutta a, b, c, d, e, f segmentlerine (6, 5, 4, 3, 2, 1 pinleri) 1 bilgisi gönderilirken g segmentine (0. Pin) 0 gönderilmiştir. Programda 0, 1, 2 sayıları yazdırılmıştır. 3, 4, 5, 6, 7, 8, 9 sayılarını programa ekleyerek çalışmasını gözlemleyiniz.

```
void setup() {
DDRD = B01111111; // 0-6 numaralı pinler çıkış, 7 numaralı pin giriş olarak ayarlandı.
}
void loop() {
PORTD = B01111110; // 0-5 numaralı pinlere 1, 6. pine 0 bilgisi gönderildi. 0 rakamı yazdırıldı.
delay(1000);
PORTD = B00110000; // 1 rakamını yaz.
delay(1000);}
PORTD = B01101101; // 2 rakamını yaz.
delay(1000);
}
```

Sorular

1. pinMode() fonksiyonuyla tek bir pin giriş veya çıkış olarak ayarlanırken DDRD registerla kaç pin giriş veya çıkış olarak ayarlanmaktadır? Belirtiniz.
2. Display ortak anotlu olsaydı PORTD = B00000001; komutuyla displayde hangi sayı görünürdü? Açıklayınız.
3. 7 segment display uygulamasını matris kullanarak yazınız.

1.7.3. Dijital Giriş İşlemleri

digitalRead() Fonksiyonu

pinMode() fonksiyonuyla INPUT olarak ayarlanan pinin HIGH (5 V) veya LOW (0 V) durumunu okur. Çıktı (dönüş) olarak HIGH veya LOW (1 veya 0) olma durumunu döndürür.

digitalRead(pin)

pin: Pin numarası.

Örnek: Dijital girişin 5 V veya 0 V'ta olma durumunu okur.

```
void setup() {
pinMode(2,INPUT); // 2 numaralı pin giriş.
pinMode(13,OUTPUT); // 13 numaralı pin çıkış.
}
void loop() {
digitalWrite(13,digitalRead(2)); // 2 numaralı pini oku. Gelen 0 veya 1 bilgisini
// 13 numaralı pine gönder.
}
```

Örnek: Tüm dijital çıkışlı sensörler için temel programdır.

```
const byte giris = 2; // Giriş pini.
const byte cikis = 13; // Çıkış pini.


void setup() {
  pinMode(giris, INPUT_PULLUP); // Çıkışı aktif "0" olan sensörler için.
  // pinMode(giris, INPUT); // Çıkışı aktif "1" olan sensörler için kullanılır.
  pinMode(cikis, OUTPUT);
}

void loop() {
  bool girisDurum = digitalRead(giris); // Giriş bilgisini oku.
  if (girisDurum == 0) { // Sensör aktifse yapılacaklar...
    digitalWrite(cikis, HIGH);
  }
  else { // Sensör pasifse yapılacaklar...
    digitalWrite(cikis, LOW);
  }
}
```

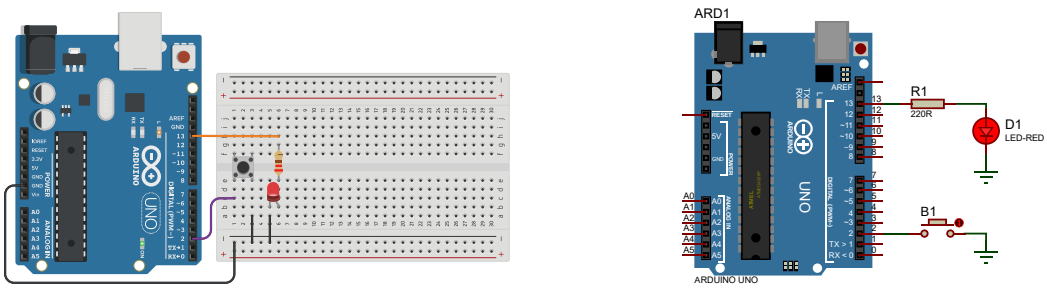
Uygulama Adı: Butonla Dijital Giriş Uygulaması — No.: 6

Amaç: Butonla dijital giriş uygulaması yapmak.

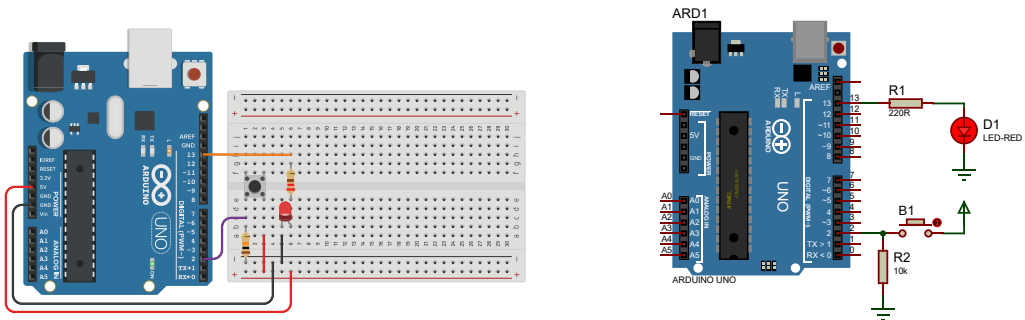
Görsel 1.26'da dâhilî pull-up direnci pinMode(buton, INPUT_PULLUP); komutuyla aktif edilmiştir. Butonla dijital giriş uygulanmıştır. Görsel 1.26'da dâhilî pull-up direncinden dolayı butona basılmadığı sürece 2 numaralı giriş pininde lojik 1 bilgisi mevcuttur. Butona basıldığında 2 numaralı giriş pinine GND'den lojik 0 bilgisi verilir.

Pull-down direnci kullanılacaksa buton ve direnç Görsel 1.27'deki şekilde bağlanır.10 kΩ'luk pull-down direncinden dolayı butona basılmadığı sürece 2 numaralı giriş pininde lojik 0 bilgisi mevcuttur. Butona basıldığında 2 numaralı giriş pinine 5 V beslemeden lojik 1 bilgisi verilir.

Booelan tipinde tanımlanan butonDurum değişkeni sadece 0 (false) veya 1 (true) değerini alır. if karar yapısıyla butonun durumu kontrol edilir. Buton basılıyken GND'den 2 numaralı girişe 0 bilgisi gelir. Bu durumda LED yakılır. Butondan el çekildiğinde INPUT_PULLUP'tan dolayı butonDurum değişkeni 1 olur ve LED söner.



Görsel 1.26: Butonla dijital giriş uygulaması şema ve devresi (Dâhilî pull-up direnci kullanılmıştır.)



Görsel 1.27: Butonla dijital giriş uygulaması şema ve devresi (Pull-down direnci kullanılmıştır.)

Butonla dijital giriş uygulama programı aşağıdaki gibidir:

```
const byte LED = 13; // 13 numaralı pini LED değişkenine ata.
const byte buton = 2; // 2 numaralı pini buton değişkenine ata.
boolean butonDurum; // butonDurum adında boolean değişken oluştur.
void setup() {
  pinMode(buton, INPUT_PULLUP); // 2 numaralı girişteki pull-up direncini aktif yap.
  pinMode(LED, OUTPUT); // LED'in bağlı olduğu pini çıkış olarak ayarla.
}
void loop() {
  butonDurum = digitalRead(buton); // Butona basıldı (0), basılmadı (1) bilgisini oku.
  if (butonDurum == 0) // Buton basılıysa...
    digitalWrite(LED, HIGH); // LED'i yak.
    else digitalWrite(LED, LOW); // LED'i söndür.
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 1.26'daki devreyi kurunuz.
4. Programı yazıp Arduino'ya yükleyiniz.
5. Butona basıp bırakarak LED'in çalışmasını gözlemleyiniz.
6. Görsel 1.27'deki devreyi kurunuz.
7. Programı pull-down direncine göre düzenleyerek Arduino'ya yükleyiniz.
8. Butona basıp bırakarak LED'in çalışmasını gözlemleyiniz.
9. Programı hiç değişken kullanmadan tekrar yazınız.
10. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

Aşağıdaki programla Arduino'ya yüklenip çalışması gözlemlenen program aynı işi görmektemi - dir? Aralarındaki farklar nelerdir? Açıklayınız.

```
void setup() {
pinMode(2,INPUT_PULLUP);
pinMode(13,OUTPUT);
}
void loop() {
digitalWrite(13,!digitalRead(2));
}
```

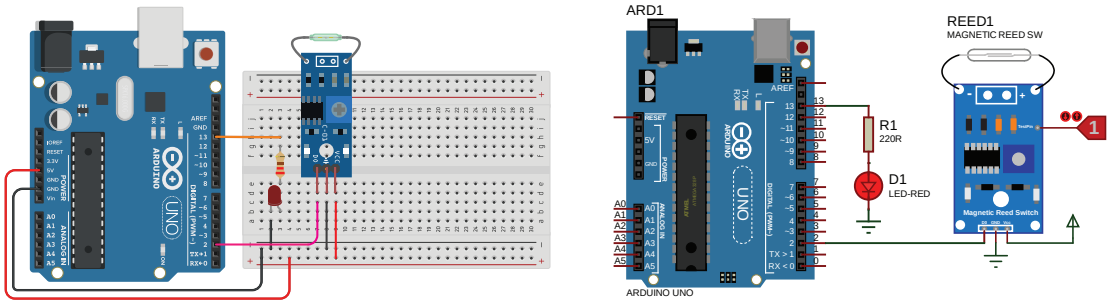
Soru

1. Arduino'da haricî pull-up direncine neden ihtiyaç yoktur? Açıklayınız.

Uygulama Adı: Reed Switch'le Dijital Giriş Uygulaması No.: 7

Amaç: Reed switch'le dijital giriş uygulaması yapmak.

Görsel 1.28'de kullanılan reed röle modülüne mıknatıs yaklaştırıldığında veya manyetik alan içine girdiğinde D0 çıkışı lojik 1 olmaktadır. Modül üzerinde besleme (5 V), GND ve D0 dijital çıkış pini yer almaktadır. Mıknatıs yaklaştırıldığında D0 pininden 5 V (1) çıkış vermektedir ve aynı zamanda üzerindeki LED de yanmaktadır. Bazı modüllerde A0 analog çıkış pini de bulunmaktadır.



Görsel 1.28: Reed switch'le dijital giriş şeması ve devresi

Reed switch'le dijital giriş uygulama programı aşağıdaki gibidir:

```
const byte LED = 13; // 13 numaralı pini LED değişkenine ata.
const byte reedRole = 2; // 2 numaralı pini reedRole değişkenine ata.
void setup() {
pinMode(reedRole, INPUT); // 2 numaralı pini giriş olarak ayarla.
pinMode(LED, OUTPUT); // LED'in bağlı olduğu pini çıkış olarak ayarla.
}
void loop() {
boolean reedRoleDurum = digitalRead(reedRole);
/* Değişkenler fonkisyon içinde de tanımlanabilir.
Ancak sadece tanımlandığı fonksiyon içinde kullanılabilir.
Mıknatıs yaklaştı (1), Mıknatıs uzaklaştı (0) bilgisini oku.*/
if (reedRoleDurum == 1) // reedRoleDurum değişkeni içindeki bilgi 1'se...
digitalWrite(LED, reedRoleDurum); // reedRoleDurum değişkenindeki bilgiyi (1) LED'e yaz.
else // reedRoleDurum değişkeni içindeki bilgi 0'sa...
digitalWrite(LED, reedRoleDurum); // reedRoleDurum değişkenindeki bilgiyi (0) LED'e yaz.
// if kontrolünden sonra tek bir komut varsa { } süslü parantezler kullanılmasa da olur.
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 1.28'deki devreyi kurunuz.

4. Programı yazıp Arduino'ya yükleyiniz.

5. Mıknatısı reed röleye yaklaştırınız LED'in yandığını gözlemleyiniz.

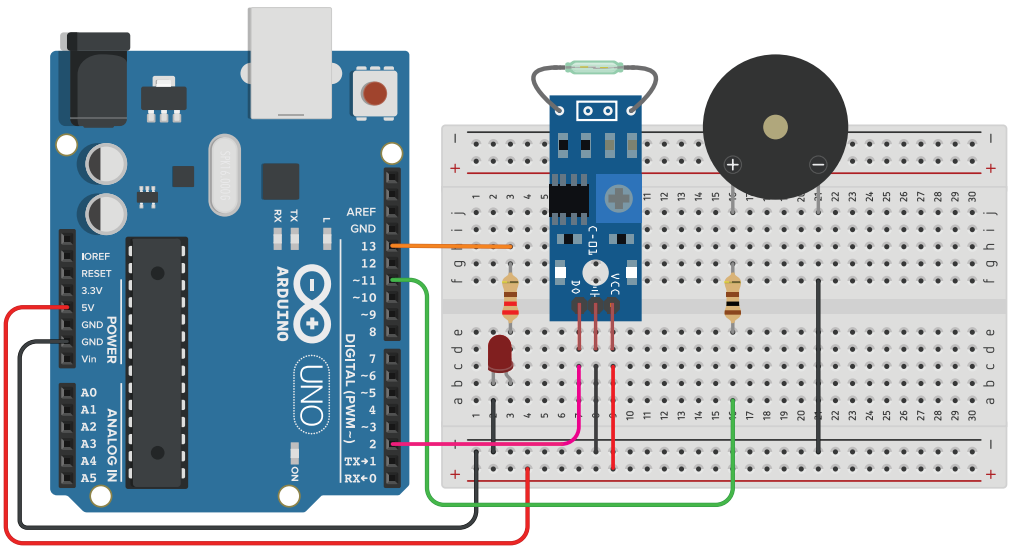
6. Mıknatısı reed röleden uzaklaştırınız. LED'in söndüğünü gözlemleyiniz.

7. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Mıknatıs yaklaştığında LED sönecek şekilde yazılımı düzenleyiniz.

2. Mıknatıs yaklaştığında sesli ikaz verecek donanımsal değişikliği yapınız.



Sorular

1. Program da if(1) ise LED'in durumu nedir? Açıklayınız.

2. Program da if(0) ise LED'in durumu nedir? Açıklayınız.

3. if(x) ifadesinde x her zaman boolean bir ifade midir? Belirtiniz.

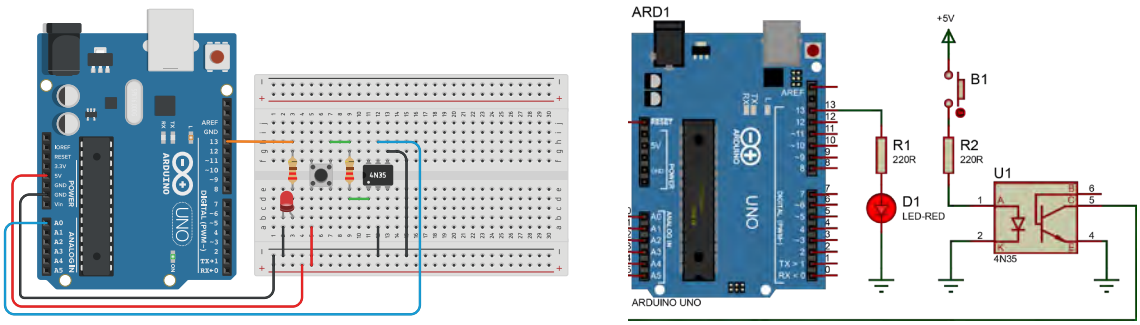
4. Program if yapısı kullanılmadan yazılabilir mi? Açıklayınız.

Uygulama Adı: Optokuplörle dijital giriş uygulaması No.: 8

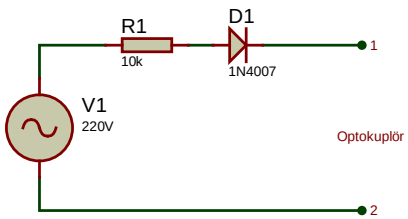
Amaç: Optokuplörle dijital giriş uygulaması yapmak.

Görsel 1.29.a'daki devrede optokuplör A0'a (14 numaralı pin) bağlanmıştır. Bu uygulamada A0 pini dijital giriş olarak kullanılmıştır. Optokuplör iletime geçtiğinde normalde pull-up'tan dolayı 1 olan A0 C-E üzerinden GND'ye bağlanır (0 olur.). Programda üç adet boolean (bool) değişken yan yana tanımlanmış ve ikisine değer atanmıştır. Programda HIGH veya 1 kullanılan yerlerde içinde 1 bilgisi olan **yak** değişkeni kullanılmıştır.

Optokuplörün 1 ve 2 numaralı uçlarına Görsel 1.29.b'deki devre bağlanarak 220 V gerilim Arduino'ya zarar vermeden test edilebilir.



Görsel 1.29a: Optokuplörle dijital giriş şeması ve devresi



Görsel 1.29b: Optokuplör ve Arduino ile 220 V kontrol

Optokuplörle dijital giriş uygulama programı aşağıdaki gibidir:

```
const byte LED = 13; // 13 numaralı pini LED değişkenine ata.
const byte opto = 14; // 14 numaralı pin A0 pinidir.
bool optoDurum, yak = 1, sondur = 0;
void setup() {
pinMode(opto, INPUT_PULLUP); // 14 numaralı pin 5 V'a (1) çekildi.
pinMode(LED, OUTPUT); // LED'in bağlı olduğu pini çıkış olarak ayarla.
}
void loop() {
optoDurum = digitalRead(opto);
if (optoDurum == 0) // optoDurum 0'a eşitse..
digitalWrite(LED, yak); // 13 numaralı pine 1 gönder.
else
digitalWrite(LED, sondur); // 13 numaralı pine 0 gönder.
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 1.29a'daki devreyi kurunuz.
4. Programı yazıp Arduino'ya yükleyiniz.
5. Butona bastığınızda LED'in yandığı gözlemleyiniz.
6. Görsel 1.29b'deki **devreyi test etmek için öğretmeninizden yardım alınız.**
7. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Programı değişken ve if kullanmadan yazınız (Butonla dijital giriş uygulamasına bakınız.).

Sorular

1. Yüksek gerilimde oluşacak bir aksaklıkta Arduino kartını koruyan eleman hangisidir? Belirtiniz.
2. Aşağıdaki program bu uygulamadaki programla aynı işlevi görür mü?

```
const byte giris = 2; // Giriş pini.
const byte cikis = 13; // Çıkış pini.


void setup() {
  pinMode(giris, INPUT_PULLUP); // Çıkışı aktif "0" olan sensörler için.
  pinMode(cikis, OUTPUT);
}

void loop() {
  bool girisDurum = digitalRead(giris); // Giriş bilgisini oku.
  if (girisDurum == 0) { // Sensör aktifse yapılacaklar...
    digitalWrite(cikis, HIGH);
  }
  else { // Sensör pasifse yapılacaklar...
    digitalWrite(cikis, LOW);
  }
}
```

1.7.4. Zaman Gecikmesi Komutları

Arduino kodları içerisinde farklı işlevleri olan program durdurma komutları vardır.

delay() Fonksiyonu

Programı, parametre olarak belirtilen süre kadar (milisaniye cinsinden) duraklatır (1000 milisaniye 1 saniyedir.). delay fonksiyonu işlemleri durduğu için süre sonuna kadar diğer işlemler (sensör okuma vb.) yapılmayacaktır. Gelişmiş bir programda delay() yerine milis() fonksiyonu kullanılır.

delay(ms)

ms: Duraklatılacak milisaniye sayısı. Veri tipi: unsigned long.

delayMicroseconds() Fonksiyonu

Programı, parametre olarak belirtilen süre kadar (mikrosaniye cinsinden) duraklatır (1000000 mikrosaniye 1 saniyedir.).

delayMicroseconds(us)

us: Duraklatılacak mikrosaniye sayısı. Veri tipi: unsigned int.

Örnek: 12 numaralı pinden 10 mikrosaniyelik kare dalga gönderilir.

```
void setup(){
pinMode(12,OUTPUT);
}
void loop(){ // 12 numaralı pinden 10 mikrosaniyelik kare dalga gönderiliyor.
digitalWrite(12,0);
delayMicroseconds(10);
digitalWrite(12,1);
delayMicroseconds(10);
}
```

millis() Fonksiyonu

Arduino kartının mevcut programı çalıştırmaya başlamasından itibaren geçen milisaniye sayısını döndürür. Yaklaşık 50 gün sonra bu sayı sıfıra dönerek tekrar eder.

zaman = millis();

Giriş parametresi yoktur. Unsigned long veri tipinde geri döndürdüğü sayı bir değişkene atanarak kullanılabilir.

Örnek: Program başladıktan sonra geçen zamanı milisaniye cinsinden ekrana yazar.

```
unsigned long gecenZaman;
void setup() {
  Serial.begin(9600);
}
void loop() {
 Serial.print("Geçen zaman: "); // Arduino'daki reset butonunuyla zaman sıfırlanır.
```

```
  gecenZaman = millis();
 Serial.println(gecenZaman);// Program başladıktan sonra geçen zamanı (ms) ekrana yaz.
  delay(1000); // 1 saniye bekle. (Saniyede bir ekrana yazdır.)
```

micros() Fonksiyonu

Arduino kartının mevcut programı çalıştırmaya başlamasından itibaren geçen mikrosaniye sayısını döndürür. Yaklaşık 70 dakika sonra bu sayı sıfıra dönerek tekrar eder.

zaman = micros();

Giriş parametresi yoktur. Unsigned long veri tipinde geri döndürdüğü sayı bir değişkene atanarak kullanılabilir.

Örnek: Program başladıktan sonra geçen zamanı mikrosaniye cinsinden ekrana yazar.

```
unsigned long gecenZaman;
void setup() {
  Serial.begin(9600);
}
void loop() {
 Serial.print("Geçen zaman: ");// Arduino'daki reset butonunuyla zaman sıfırlanır.
  gecenZaman = micros();
 Serial.println(gecenZaman);// Program başladıktan sonra geçen zamanı (us) ekrana yaz.
  delay(1000); // 1 saniye bekle. (Saniyede bir ekrana yazdır.)
}
```

1.7.5. Gelişmiş Giriş Çıkış İşlemleri

Arduino'nun pinlerden belirli bir frekansta kare dalga üretmek, dalga süresini hesaplamak veya kaydırmalı giriş çıkış (shift register) için kullanılan fonksiyonlar vardır.

tone() Fonksiyonu

tone() fonksiyonu Arduino dijital pini üzerinden belirtilen frekansta (%50 duty cycle) kare dalga üretmek için kullanılır. Süre belirtilebilir. Süre belirtilmezse noTone() fonksiyonuna ulaşılıncaya kadar kare dalga sinyal üretmeye devam eder. Pin çıkışına pizeo buzzer (pasif buzzer) veya hoparlör bağlanabilir. Üç adet parametre alır. Geriye değer döndürmez. Aynı anda birden fazla pinden çıkış veremez. tone() fonksiyonu (Mega dışındaki kartlar için) 3 ve 11 No.lu pinlerdeki PWM çıkışını engeller. tone() fonksiyonuyla en az 31 Hz değerinde pulse çıkış elde edilebilir.

```
tone(pin, frekans)

tone(pin, frekans, sure)

pin: Pin numarası.

frekans: Frekans.

sure: Frekansı üretme süresi.
```

Örnek: Ambulans sesi çıkarır.

```
byte buzzer = 7;
int sure = 500;
void setup() {
}
void loop() {
  tone(buzzer, 700);
  delay(sure);
  tone(buzzer, 900);
  delay(sure);
}
```

Örnek: Siren sesi çıkarır.

```
byte buzzer = 7; // Buzzer 7 numaralı pine bağlı.
void setup() {
}
void loop() {
  for (int frekans = 500; frekans < 2000; frekans += 10) { // Frekansı artır.
    tone(buzzer, frekans);
    delay(10);
  }
  for (int frekans = 2000; frekans > 500; frekans -= 10) { // Frekansı azalt.
    tone(buzzer, frekans);
    delay(10);
  }
}
```

noTone() Fonksiyonu

tone() fonksiyonuyla tetiklenen kare dalga sinyali durdurmak için kullanılır. Sinyali durdurulmak istenen pin numarası parametre olarak alınır. Geriye herhangi bir değer döndürmez. Birçok pinde tone() kullanılcaksa noTone() kullanıldıktan sonra diğerine geçilir.

```
noTone(pin)
```

pin: Pin numarası.

Örnek: Bazı notaları sırayla çalar.

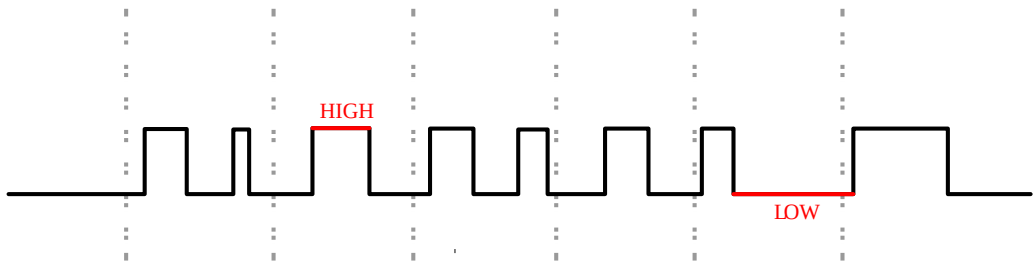
```
void setup() {
}
void loop() {
  noTone(8);
  tone(6, 440, 200);
  delay(200);
  noTone(6);
  tone(7, 494, 500);
  delay(500);
```

```
  noTone(7);
  tone(8, 523, 300);
  delay(300);
}
```

pulseIn() Fonksiyonu

Giriş olarak ayarlanan pin üzerine uygulanan sinyal darbesinin kaç mikrosaniye süreyle HIGH ya da LOW durumunda kaldığını belirlemek için kullanılır. Geriye unsigned long türünde mikrosaniye cinsinden değer döndürür. Örnek olarak HIGH parametresi almışsa pin üzerine HIGH geldiğinde zamanlayıcıyı çalıştırır. Sinyal LOW olunca zamanlayıcıyı durdurur. HIGH seviyesinde kalma süresini mikrosaniye cinsinden unsigned long veri tipinde döndürür. Zaman aşımıyla belirtilen süre içerisinde pals sinyali konum değiştirmemişse geriye değer olarak 0 döndürür (Görsel 1.30).

```
pulseIn(pin, durum)

pulseIn(pin, durum, zamanAsimi)

pin: Pin numarası.

durum: Okunacak pals tipi (HIGH veya LOW)

zamanAsimi: Darbe sinyalinin başlamasını bekleme süresi. Varsayılan süre 1 saniye
```



Görsel 1.30: HIGH veya LOW seviyelerinde kalma süresi

Örnek: Program butona basılma süresini vermektedir.

```
int pin = 7; // Butona bağlı.
unsigned long sure;
void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT_PULLUP); // Butonun diğer ucu GND'ye bağlı.
}
void loop() {
  sure = pulseIn(pin, LOW, 60000000); // Zaman aşımı 1 dakika ayarlandı.
  Serial.print(sure / 1000); // Butona basılma süresini milisaniye olarak göster.
Serial.println(" ms.");
}
```

Örnek: pulseIn() fonksiyonuyla yapılmış debounce (buton arkını önleme) programıdır.

```
int pin = const byte buton = 2;
unsigned long sure;
void setup() {
  Serial.begin(9600);
  pinMode(buton, INPUT_PULLUP);
}
void loop() {
  sure = pulseIn(buton, LOW) / 1000;
  if ( sure > 60)
    Serial.println("Butona basıldı.");
}
```

Örnek: Tek butonla LED'i yakıp söndürür.

```
const byte buton = 2;
const byte LED = 13;
unsigned long sure;
bool tersle = 1; // ilk basışta LED'i yakmak için.
void setup() {
  pinMode(buton, INPUT_PULLUP);
  pinMode(LED, OUTPUT);
}
void loop() {
  sure = pulseIn(buton, LOW) / 1000;
  if ( sure > 60) { // İlk 60 millisaniye içindeki bounce (ark) etkisinden kurtul.
    digitalWrite(LED, tersle); // Çıkışın konumunu değiştir.
    tersle = !tersle;
  }
}
```

pulseInLong() Fonksiyonu

Kullanımı pulseIn() fonksiyonuyla aynı olan pulseInLong(), uzun darbe sürelerini algılama ve kesmelerde (intterrupt) etkilenen senaryoları işlemede daha iyi olan bir alternatiftir.

Örnek: Ultrasonik sensörle mesafe ölçer.

```
const byte trig = 12; // Sinyal çıkışı.
const byte echo = 11; // Sinyal girişi.
unsigned long sure;
int uzaklik;
void setup() {
 pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  Serial.begin(9600);
}
void loop() {
  digitalWrite(trig, 0); // Kare dalga gönder.
```

```
  delayMicroseconds(10);
  digitalWrite(trig, 1);
  delayMicroseconds(10);
  sure = pulseInLong(echo, 1); // Gelen sinyalin HIGH'da kalma süresini oku.
  uzaklik = (0.0343 * sure) / 2;
  Serial.print(uzaklik);
  Serial.println(" cm");
}
```

shiftIn() Fonksiyonu

Bir byte veride her seferinde bir bit kaydırır. En yüksek değerlikli veya en düşük değerlikli bitten başlar. Her bit için clock pini HIGH olur. Bir sonraki bit veri hattından okunur ve ardından clock pini LOW olur. Dijital giriş sayısını arttırmak için kullanılır.

```
shiftIn(dataPin, clockPin, bitSirasi
```

`dataPin:` Her bir biti okuyan giriş pini.

`clockPin:` Clock sinyalini gönderen pin.

`bitSirasi:` Bit okuma sırası. MSBFIRST veya LSBFIRST.

shiftOut() Fonksiyonu

Her saat darbesinde bir byte verinin bir **bit**ini kaydırarak çıkışa gönderir. Dijital çıkış sayısını arttırmak için kullanılır.

```
shiftOut(dataPin, clockPin, bitSirasi, veri)
```

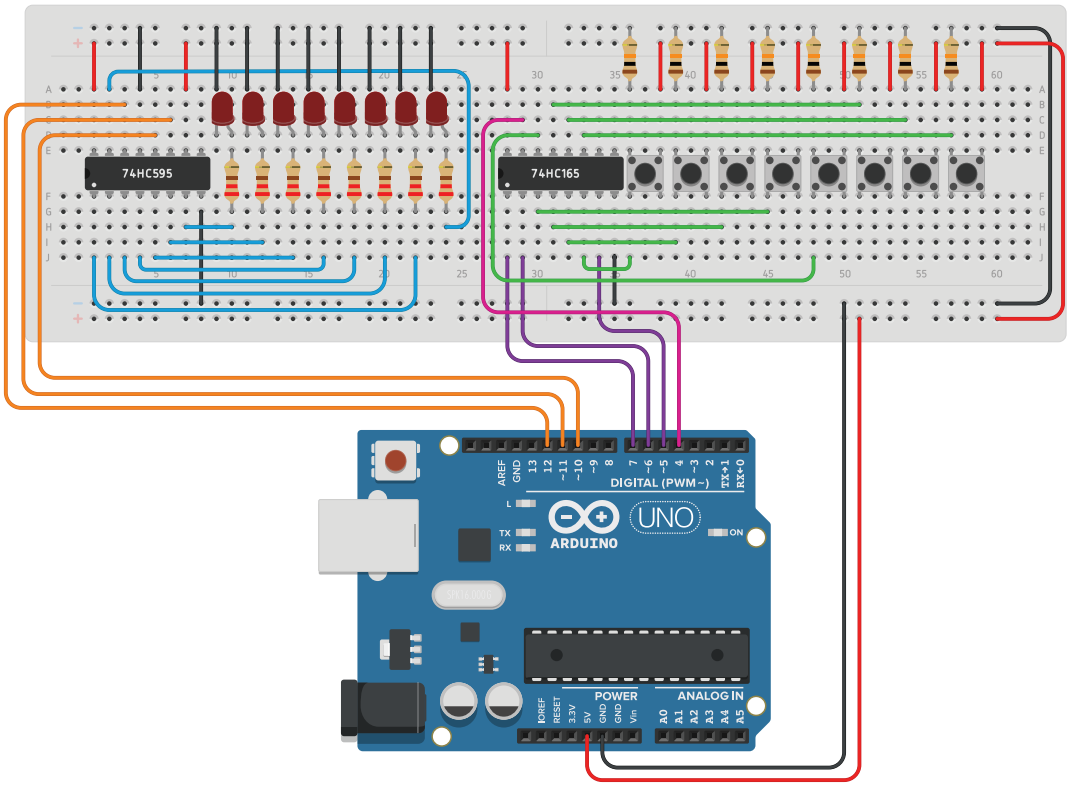
`dataPin:` Her bir biti gönderen çıkış pini.

`clockPin:` Clock sinyalini gönderen pin.

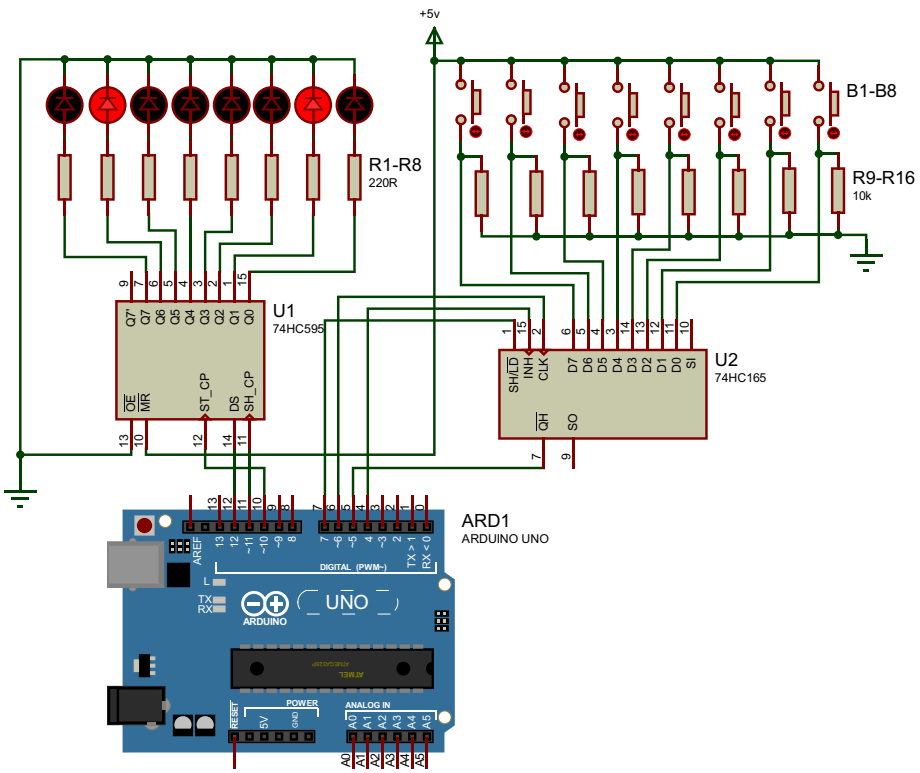
`bitSirasi:` Bit okuma sırası. MSBFIRST veya LSBFIRST.

`veri:` Kaydırılarak çıkışa aktarılacak bir byte veri.

Örnek: 74HC165 ve 74HC595 ile dijital giriş ve çıkış sayısını arttırmayı gösterir. Dijital giriş ve çıkış sayısını arttırma devresi Görsel 1.31'de, şeması ise Görsel 1.32'de görülmektedir.



Görsel 1.31: 74HC165 ve 74HC595 ile dijital giriş ve çıkış sayısını artırma devresi



Görsel 1.32: 74HC165 ve 74HC595 ile dijital giriş ve çıkış sayısını artırma şeması

```
int yukleme = 7;   // 74HC165 bağlantıları.
int clockAktif = 4;
int veriGiris = 5;
int clockGiris = 6;

const int bekle = 10; // 74HC595 bağlantıları.
const int clockPin = 11;
const int veri = 12;
int ledler[8];
void setup () {
  pinMode(yukleme, OUTPUT); // 74HC165 pinleri.
  pinMode(clockAktif, OUTPUT);
  pinMode(clockGiris, OUTPUT);
  pinMode(veriGiris, INPUT);

  pinMode(bekle, OUTPUT); // 74HC595 pinleri.
  pinMode(clockPin, OUTPUT);
  pinMode(veri, OUTPUT);
}
void loop() {
  digitalWrite(yukleme, LOW); // Girişleri oku.
  delayMicroseconds(5);
  digitalWrite(yukleme, HIGH);
  delayMicroseconds(5);
 digitalWrite(clockGiris, HIGH); // 74HC165'ten bilgiyi al.
  digitalWrite(clockAktif, LOW);
  byte gelenVeri = shiftIn(veriGiris, clockGiris, LSBFIRST);
  digitalWrite(clockAktif, HIGH);

  switch (gelenVeri) {
    case B11111110:
      ledler[0] = B11111111;
      ledler[1] = B01111110;
      ledler[2] = B10111101;
      ledler[3] = B11011011;
      ledler[4] = B11100111;
      ledler[5] = B11011011;
      ledler[6] = B10111101;
      ledler[7] = B01111110;
      break;
    case B11111101:
      ledler[0] = B00000001;
       ledler[1] = B00000010;
      ledler[2] = B00000100;
      ledler[3] = B00001000;
```

```
    ledler[4] = B00010000;
    ledler[5] = B00100000;
    ledler[6] = B01000000;
    ledler[7] = B10000000;
    break;
  case B11111011:
    ledler[0] = B10000001;
    ledler[1] = B01000010;
    ledler[2] = B00100100;
    ledler[3] = B00011000;
    ledler[4] = B00000000;
    ledler[5] = B00100100;
    ledler[6] = B01000010;
    ledler[7] = B10000001;
    break;
  case B11110111:
    ledler[0] = B10101010;
    ledler[1] = B01010101;
    ledler[2] = B10101010;
    ledler[3] = B01010101;
    ledler[4] = B10101010;
    ledler[5] = B01010101;
    ledler[6] = B10101010;
    ledler[7] = B01010101;
    break;
  case B11101111:
    ledler[0] = B10000000;
    ledler[1] = B00000001;
    ledler[2] = B01000000;
    ledler[3] = B00000010;
    ledler[4] = B00100000;
    ledler[5] = B00000100;
    ledler[6] = B00010000;
    ledler[7] = B00001000;
    break;
 case B11011111:
    ledler[0] = B11000000;
    ledler[1] = B01100000;
    ledler[2] = B00110000;
    ledler[3] = B00011000;
   ledler[4] = B00001100;
    ledler[5] = B00000110;
    ledler[6] = B00000011;
    ledler[7] = B10000001;
    break;
```

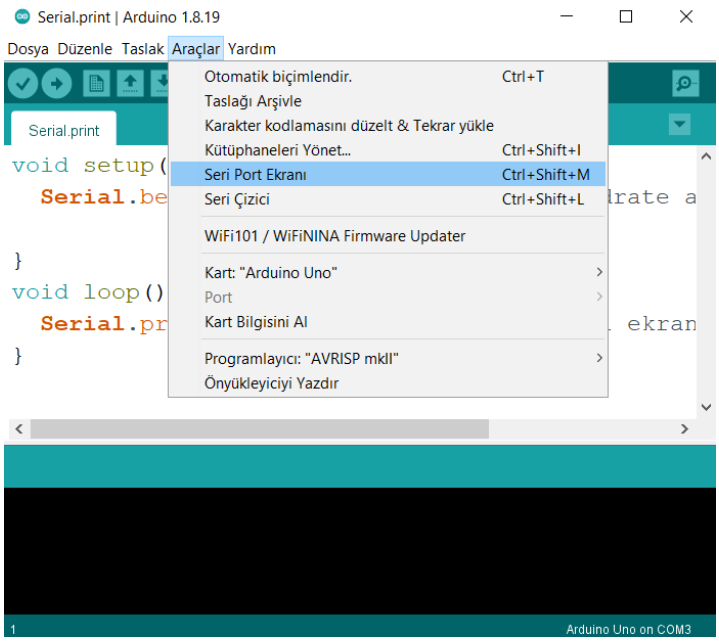
```
      case B10111111:
        ledler[0] = B11100000;
        ledler[1] = B01110000;
        ledler[2] = B00111000;
        ledler[3] = B00011100;
        ledler[4] = B00001110;
        ledler[5] = B00000111;
        ledler[6] = B10000011;
        ledler[7] = B11000001;
        break;
      case B01111111:
        ledler[0] = B10001000;
        ledler[1] = B01000100;
        ledler[2] = B00100010;
        ledler[3] = B00010001;
        ledler[4] = B10001000;
        ledler[5] = B01000100;
        ledler[6] = B00100010;
        ledler[7] = B00010001;
        break;
      default:
        break;
    }
    for (int i = 0; i < 8; i++) { // LED'lere gönder.
      digitalWrite(bekle, LOW);
      shiftOut(veri, clockPin, MSBFIRST, ledler[i]);
      digitalWrite(bekle, HIGH);
      delay(200);
    }
}
```

1.8. SERİ PORT İŞLEMLERİ

Arduino kartıyla bilgisayar veya diğer cihazlar arasındaki iletişim için kullanılır. Tüm Arduino kartlarında en az bir seri bağlantı noktası (UART veya USART olarak da bilinir.) bulunur.

Arduino Uno, Nano, Mini ve Mega'da bilgisayarla iletişim için 0 (Rx) ve 1 (Tx) pinleri kullanılır. Bu pinlere herhangi bir şey bağlanırsa program yükleme aşamasında hata verir.

Arduino'nun USB kablosu üzerinden (Rx ve Tx pinleri) bilgisayara veri aktarmak için Serial() fonksiyonu kullanılır. Görsel 1.33'te görüldüğü gibi Araçlar→Seri Port Ekranı seçeneğiyle açılır.



Görsel 1.33: Seri port ekranını açma

Serial.begin() Fonksiyonu

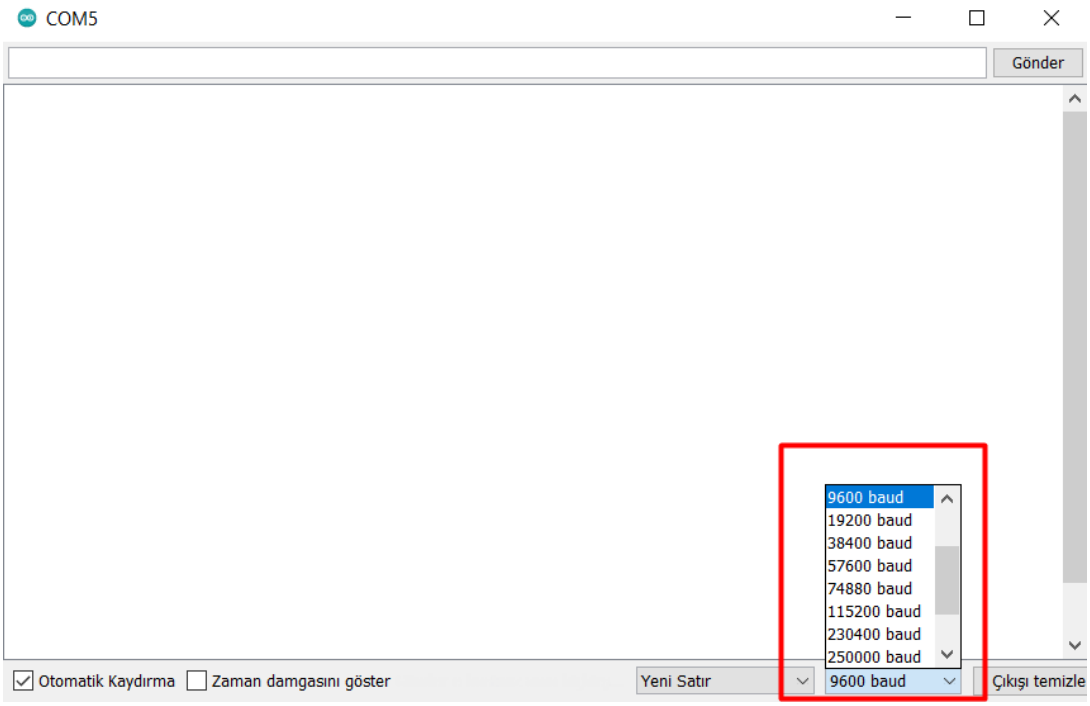
Serial.begin(9600) fonksiyonuyla seri port haberleşme hızı ayarlanır. 9600 bps (bit per second-saniyedeki bit sayısı) buadrate olarak isimlendirilen veri transfer hızıdır. Serial.begin() fonksiyonu içine yazılan değerle Görsel 1.34'te gösterilen seri port ekranındaki değer aynı şekilde ayarlanmalıdır.

NOT Programda Serial.begin() fonksiyonu kullanılıyorsa devrede 0. ve 1. pinlerde bağlantı olmamasına dikkat edilmelidir.

```
Serial.begin(hiz)
Serial.begin(hiz, ayar)
```

Hız: Seri portun saniyede göndereceği ve alacağı bit sayısıdır. Seri portun haberleşeceği cihazla aynı olmak zorundadır.

Ayar: Veri, eşlik ve durdurma bitlerini ayarlar.



Görsel 1.34: Seri port ekranı

Serial.print() Fonksiyonu

Seri port ekranına veri yazdırmak için kullanılır.

`Serial.print(deger)`

`deger`: herhangi bir veri tipindeki değer.

Örnek: Bazı veri tiplerini ekrana yazdırır.

```
Serial.print(35); // “35”
Serial.print(1.23456); // “1.23”
Serial.print(‘N’); // “N”
Serial.print(“Arduino.”); // “Arduino.”
```

Serial.print(deger, format)

format: Yazdırma formatı.

Örnek: Sayıları formatlı yazdırır.

```
Serial.print(78, BIN); // “1001110”
Serial.print(78, OCT); // “116”
Serial.print(78, DEC); // “78”
Serial.print(78, HEX); // “4E”
Serial.print(1.23456, 0); // “1”
Serial.print(1.23456, 2); // “1.23”
Serial.print(1.23456, 4); // “1.2345”
```

Serial.println() Fonksiyonu

Serial.print() fonksiyonuyla aynı özelliklere sahiptir. Tek farkı satır başı yapmasıdır.

F()makro Kullanımı: Serial.print("Arduino."); kullanıldığında yazdırılacak metin (dizi) normal olarak RAM'e kaydedilir. Progam (taslak) seri ekrana çok fazla şey yazdırıyorsa RAM hızlıca dolar. Boş flash bellek (programın yüklendiği hafıza) alanı varsa Serial.print(F("Arduino.")); söz dizimi kullanılarak **metin flash**a kaydedilir.

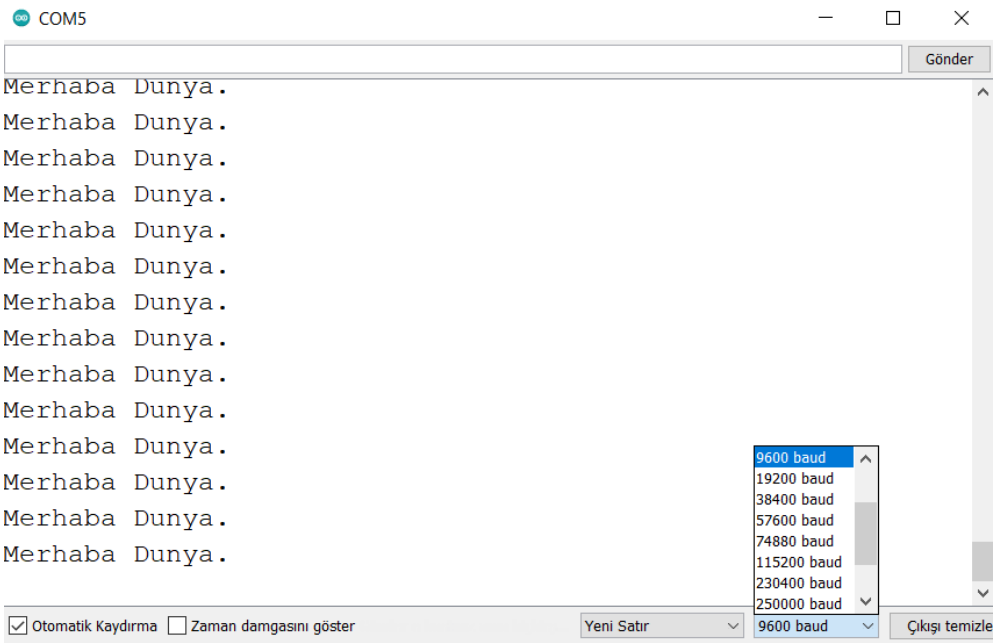
NOT DEFTERİ

Uygulama Adı: Seri Port Ekran Uygulaması — No.: 9

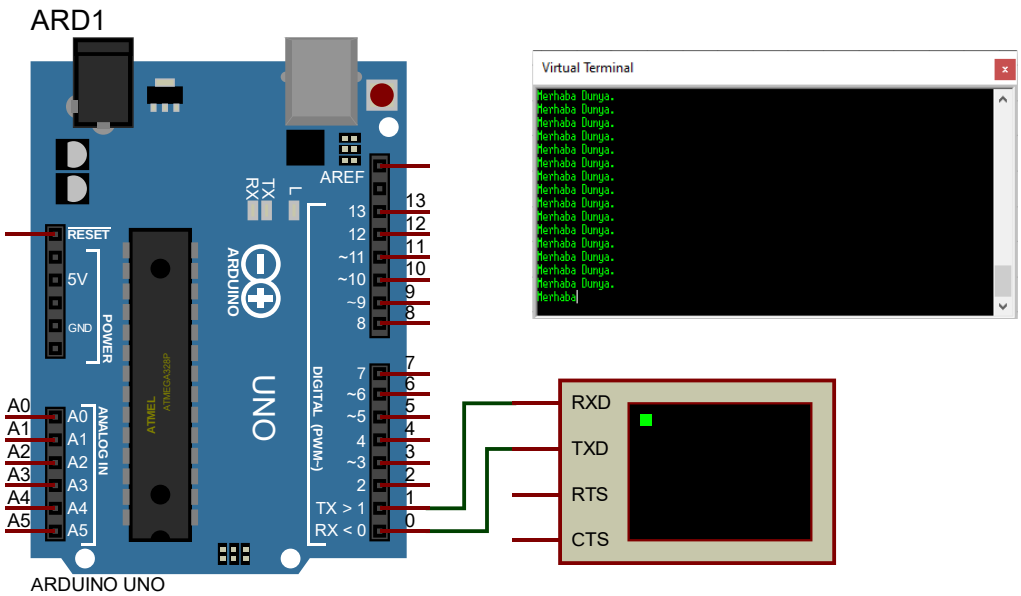
Amaç: **Seri port ekran uygulaması yapmak.**

Görsel 1.35a ve 1.35b'deki çıktılarda seri port ekranına Serial.println() fonksiyonuyla "Merhaba Dünya" yazdırılmaktadır.

İkinci programda butonla dijital giriş uygulamasında butonun durumu seri port ekranına yazdırılmaktadır (Görsel 1.36).



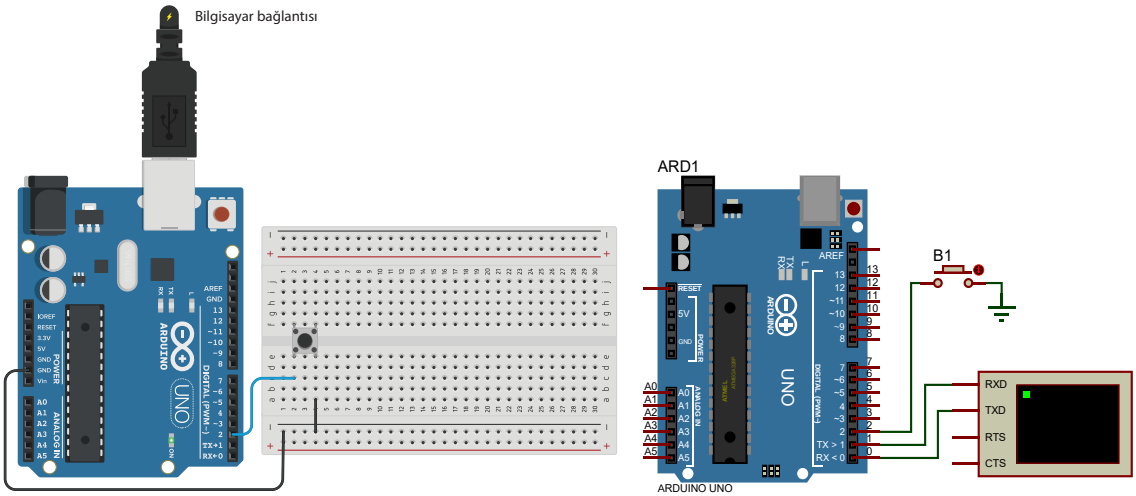
Görsel 1.35a: Seri port ekranı



Görsel 1.35b: Seri port ekranı simülasyon devresi

Seri port ekran uygulama programı aşağıdaki gibidir:

```
void setup() {Serial.begin(9600); // Seri iletişim baudrate ayarı yapıldı.
}
void loop() {
  Serial.println("Merhaba Dünya."); // Seri ekrana metin yazdırıldı.
}
```



Görsel 1.36: Buton durumunu seri port ekranına yazdırma devresi

Buton durumunu seri port ekranına yazdırma uygulaması programı aşağıdaki gibidir:

```
const  byte buton = 2;
void setup() {
  Serial.begin(9600); // Seri iletişim 9600 bps olarak ayarlandı.
  pinMode(buton, INPUT_PULLUP);
}
void loop() {
  boolean butonDurum = digitalRead(buton);
  Serial.println(butonDurum); // butonDurum değişkeni içindeki bilgi seri ekrana yazdırıldı.
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. "Merhaba Dünya." programını yazıp Arduino'ya yükleyiniz. Seri port ekranına yazdırınız.
4. Seri port ekranındaki **baudrate'i** programdakinden farklı seçip sonuçları gözlemleyiniz.
5. Buton durumu programını yazıp Arduino'ya yükleyiniz. Buton durumunu seri port ekranına yazdırınız.
6. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Butonun durumunu **if** kullanarak seri port ekranına "Butona basıldı.", "Butona basılmadı." şeklinde yazdırınız.

Sorular

1. Programdaki **buadrate** hızıyla seri port ekranında seçili olan **buadrate** hızı farklı olursa ne olur? Açıklayınız.

2. Digital giriş çıkış uygulamalarında 0 ve 1 numaralı pinler neden tercih edilmemektedir? Açıklayınız.

3. Serial.print() fonksiyonuyla Serial.println() fonksiyonları arasındaki fark nedir? Açıklayınız.

4. Aşağıdaki çıktıyı veren programı yazınız.

E, dec: 69, hex: 45, oct: 105, bin: 1000101
F, dec: 70, hex: 46, oct: 106, bin: 1000110
G, dec: 71, hex: 47, oct: 107, bin: 1000111
H, dec: 72, hex: 48, oct: 110, bin: 1001000
I, dec: 73, hex: 49, oct: 111, bin: 1001001
J, dec: 74, hex: 4A, oct: 112, bin: 1001010
K, dec: 75, hex: 4B, oct: 113, bin: 1001011
L, dec: 76, hex: 4C, oct: 114, bin: 1001100
M, dec: 77, hex: 4D, oct: 115, bin: 1001101
N, dec: 78, hex: 4E, oct: 116, bin: 1001110
O, dec: 79, hex: 4F, oct: 117, bin: 1001111
P, dec: 80, hex: 50, oct: 120, bin: 1010000
Q, dec: 81, hex: 51, oct: 121, bin: 1010001
R, dec: 82, hex: 52, oct: 122, bin: 1010010
S, dec: 83, hex: 53, oct: 123, bin: 1010011
T, dec: 84, hex: 54, oct: 124, bin: 1010100
U, dec: 85, hex: 55, oct: 125, bin: 1010101
V, dec: 86, hex: 56, oct: 126, bin: 1010110
W, dec: 87, hex: 57, oct: 127, bin: 1010111
X, dec: 88, hex: 58, oct: 130, bin: 1011000
Y, dec: 89, hex: 59, oct: 131, bin: 1011001
Z, dec: 90, hex: 5A, oct: 132, bin: 1011010

Serial.available() Fonksiyonu

Seri iletişim arabelleğine (64 bayt) ulaşmış ve depolanmış olan verilerin byte (karakter) sayısını verir. Parametre almaz. Gelen karakter sayısını döndürür.

Örnek: Gelen verinin byte sayısını yazdırır.

```
void setup() {
  Serial.begin(9600); // Seri iletişim başlatıldı.
}
void loop() {
  if (Serial.available() > 0) { // Eğer buffer'da veri varsa...
    byte uzunluk = Serial.available(); // Gelen byte sayısını uzunluk değişkenine ata.
    Serial.println(uzunluk); // Uzunluk değişkeninin içeriğini yazdır.
    delay(1000);
  }
}
```

Serial.read() Fonksiyonu

Parametre almaz. Seri porttan gelen karakteri okur. Gelen karakter bir değişkene atanabilir.

Örnek: Seri porttan karakter okur.

```
void setup() {
  Serial.begin(9600); // Seri iletişim başlatıldı.
}
void loop() {
  if (Serial.available()>0) { // Seri iletişimde veri mevcutsa 1 yoksa 0.
  char karakter = Serial.read();// Karakter değişkenine seri hattan gelen veriyi yükle.
    Serial.print("Gelen karakter "); // Ekrana "Gelen karakter " yaz.
    Serial.println(karakter); // Karakter değişkeninin içeriğini yazdır.
  }
}
```

Serial.readString() Fonksiyonu

Seri porttan gelen verileri string (metin) olarak okur. Parametre almaz. Değişkene atanabilir.

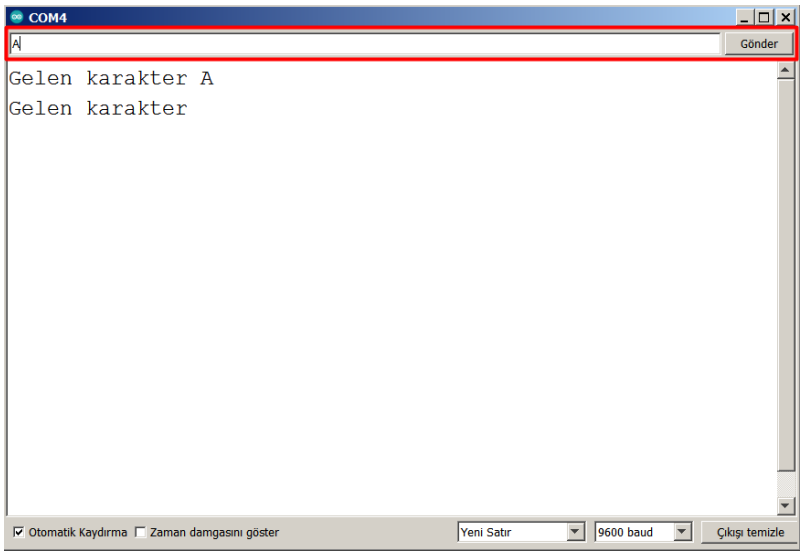
Örnek: Seri porttan karakter dizisi okuyarak seri port ekranına yazar.

```
String metin;
void setup() {
  Serial.begin(9600); // Seri iletişimi başlat
}
void loop() {
  if (Serial.available()) { // Veri geliyorsa...
    metin = Serial.readString(); // Gelen veriyi metin değişkenine ata.
    Serial.print("Gelen metin--->  ");
    Serial.println(metin);
  }
}
```

Uygulama Adı: Bilgisayardan Arduino'ya Veri Gönderme No.: 10

Amaç: Bilgisayardan Arduino'ya veri gönderme uygulaması yapmak.

Bu uygulamada Serial.read() fonksiyonuyla USB kablo üzerinden bilgisayardan gelen veriler okunarak seri port ekranına yazdırılmıştır. Bilgisayardan Arduino'ya veri göndermek için seri port ekranının üst kısmındaki kutuya karakter yazıp gönder düğmesine tıklanır (Görsel 1.37). Çok karakterli metin bilgisini okumak için Serial.readString() fonksiyonu kullanılır.



Görsel 1.37: Bilgisayardan Arduino'ya veri gönderme

Bilgisayardan Arduino'ya veri (tek karakter) gönderme programı aşağıdaki gibidir:

```
oid setup() {
  Serial.begin(9600); // Seri iletişim başlatıldı.
}
void loop() {
  if (Serial.available()>0) { // Seri iletişimde veri mevcutsa 1 yoksa 0.
    char karakter = Serial.read(); // Karekter değişkenine seri hattan gelen veriyi yükle.
    Serial.print("Gelen karakter "); // Ekrana "Gelen karakter " yaz.
    Serial.println(karakter); // Karakter değişkeninin içeriğini yazdır.
  }
}
```

Bilgisayardan Arduino'ya veri (metin) gönderme programı aşağıdaki gibidir:

```
String metin;
void setup() {
  Serial.begin(9600); // Seri iletişimi başlat.
}
void loop() {
  if (Serial.available()) { // Veri geliyorsa...
```

```
    metin = Serial.readString(); // Gelen veriyi metin değişkenine ata.
    Serial.print("Gelen metin--->  ");
     Serial.println(metin);
  }
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Bilgisayardan Arduino'ya veri (tek karakter) gönderme programını yazınız.
4. Seri port ekranına A karakterini yazıp enter tuşuna basınız.
5. Bilgisayardan Arduino'ya veri (metin) gönderme programını yazınız.
4. Seri port ekranına adınızı yazıp enter tuşuna basınız.

SIRA SİZDE

1. LED'le dijital çıkış uygulamasındaki LED'i bilgisayardan A karakteri göndererek yakıp B karakteri göndererek söndürünüz.

```
char karakter;
void setup() {
  pinMode(13, OUTPUT);
  Serial.begin(9600);// Seri iletişimi başlat.
}
void loop() {
  if (Serial.available()) { // Veri geliyorsa...
    karakter = Serial.read(); // Gelen veriyi karakter değişkenine ata.
    if (karakter == 'A') // Gelen karakter A'sa...
      digitalWrite(13, 1); // LED'i yak.
    if (karakter == 'B') // Gelen karakter B'se...
      digitalWrite(13, 0); // LED'i söndür.
  }
}
```

2. String (metin); değişkeni tanımlayarak, Serial.readString() fonksiyonuyla "yak" ve "söndür" metinlerini arka arkaya göndererek LED'i yakıp söndürünüz.

İPUCU Yukarıdaki programı kullanarak Arduino'ya USB kablo yerine bluetooth modül bağlayıp, kablosuz iletişimle LED'i yakıp söndürebilirsiniz.

Sorular

1. Karakter kabul eden programa metin bilgisi gönderildiğinde nasıl işlem yapar? Açıklayınız.
2. Bilgisayarda A ile a farklı karakterler midir? Belirtiniz.
3. "Arduino öğreniyorum." metninde kaç karakter vardır? Boşluk karakteri klavyedeki hangi tuşla yazılır? Belirtiniz.

Serial.end() Fonksiyonu

Seri portu devre dışı bırakarak RX ve TX pinlerinin dijital giriş çıkış olarak kullanılmasına izin verir. Seri portu tekrar etkinleştirmek için Serial.begin() fonksiyonu kullanılır. Serial.end() fonksiyonu parametre almaz ve değer döndürmez.

```
Serial.end()
```

Serial.readBytes() Fonksiyonu

Seri porttan gelen verilerin hepsini veya bir kısmını char veya **byte** dizisine atar.

```
Serial.readBytes(veriDizisi, uzunluk)
```

veriDizisi: Arabellekteki (buffer) karakterlerin (**byte**) atandığı dizi. Veri tipi char veya **byte** dizisi.

uzunluk: Dizinin uzunluğu. Veri tipi int.

Örnek: Program seri porttan Arduino'ya gönderilen verinin ikinci karakterini ekrana yazdırır.

```
char karakter;
void setup() {
  Serial.begin(9600); // Seri iletişim başlatıldı.
}
void loop() {
  if (Serial.available() > 0) { // Eğer buffer'da veri varsa...
    int uzunluk = Serial.available(); // Gelen byte sayısını uzunluk değişkenine ata.
   char gelenVeri[uzunluk]; // Gelen veri uzunluğunda "gelenVeri" adında dizi oluştur.
  Serial.readBytes(gelenVeri, uzunluk);// Verinin tamamını, "gelenVeri" dizisine aktar.
karakter = gelenVeri[1];// Dizinin ikinci karakterini (ASCII) karakter değişkenine ata.
  }
Serial.println(karakter);// Karakter değişkenin içindeki char tipindeki karakteri yaz.
  delay(1000);
}
```

Örnek: Verinin ilk üç **byte**ı okunur.

```
void loop() {
  if (Serial.available() > 0) { // Eğer buffer'da veri varsa...
   char gelenVeri[2]; // 3 byte uzunluğunda "gelenVeri" adında dizi oluştur.
Serial.readBytes(gelenVeri, 2);// Verinin ilk üç karakterini, "gelenVeri" dizisine al.
}
```

Serial.readBytesUntil() Fonksiyonu

Seri porttan gelen verileri belli bir karaktere kadar okur. Belirlenen karaktere ulaşıldığında okuma işlemi sona erer ve belirtilen karakter arabellekten temizlenir.

```
Serial.readBytesUntil(karakter, veriDizisi, uzunluk)
```

karakter: Gelen veride aranan karakter. Veri tipi char.

veriDizisi: Arabellekteki karakterlerin atandığı dizi. Veri tipi char veya **byte** dizisi.

uzunluk: Dizinin uzunluğu. Veri tipi int.

Örnek: Program seri porttan Arduino'ya gönderilen verinin A karakterine kadar olan karakter sayısını ekrana yazdırır.

```
int kacKarakter;
void setup() {
  Serial.begin(9600);
}
void loop() {
  if (Serial.available() > 0) { // Eğer buffer'da veri varsa...
    int uzunluk = Serial.available(); // Gelen byte sayısını uzunluk değişkenine ata.
    char gelenVeri[uzunluk]; // Gelen veri uzunluğunda "gelenVeri"  dizisi oluştur.
    kacKarakter = Serial.readBytesUntil('A', gelenVeri, uzunluk);
    // Gelen veride A karakterine kadar kaç karakter var?
    Serial.println(kacKarakter); // A karakterine kadar olan karakter sayısını yazdır.
    delay(1000);
  }
}
```

Serial.readStringUntil() Fonksiyonu

Seri porttan gelen metni belirlenen karaktere kadar okur. Belirlenen karaktere kadar olan metni döndürür.

```
Serial.readStringUntil(karakter)
```

karakter: Gelen metinde aranan karakter. Veri tipi char.

Örnek: A karakterine kadar olan kısmı alır.

```
String metin;
void setup() {
  Serial.begin(9600); // Seri iletişimi başlat.
}
void loop() {
  if (Serial.available()) { // Veri geliyorsa...
    metin = Serial.readStringUntil('A'); // A karakterine kadar olan kısmı metin de-
ğişkenine ata.
    Serial.print("Gelen metin--->  ");
    Serial.println(metin);
  }
}
```

Serial.write() Fonksiyonu

Seri porta binary veri gönderir. Gönderilen veri **byte** veya **byte** dizisi şeklindedir.

```
Serial.write(deger)
Serial.write(metin)
Serial.write(veriDizisi, uzunluk)
```

deger: Gönderilen bir byte veri.

metin: Gönderilen birçok byte veri.

veriDizisi: Dizi şeklinde gönderilen veri.

uzunluk: Dizinin byte uzunluğu.

Örnek: Seri porttan bir **byte** veri gönderir.

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.write(45); // 45 değerindeki bir byte veri gönder.
  int uzunluk = Serial.write("Arduino");
  // Arduino metnini gönder ve metnin byte uzunluğunu değişkene ata.
}
```

Serial.availableForWrite() Fonksiyonu

Serial.write() işlemini engellemeden seri arabelleğe yazmak için kullanılabilecek **byte** (karakter) sayısını verir.

```
Serial.availableForWrite()
```

Örnek: Seri port arabelleğinde kalan boşluk (**byte** sayısı) seri port ekranına yazdırır.

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.write("Arduino");
  int kalanByteUzunlugu = Serial.availableForWrite();
  Serial.println(kalanByteUzunlugu);
  // Seri porta "Arduino" verisini gönderdikten sonra 64 byte hafızadan kalanı yazdır.
  delay(1000);
}
```

Serial.find() Fonksiyonu

Seri porttan gelen veride belirlenen karakteri arar. Char veri tipinde parametre alır. Fonksiyon karakteri bulursa true, bulamazsa false sonucunu döndürür.

```
Serial.availableForWrite()
Serial.find(karakter)
Serial.find(karakter, uzunluk)
```

karakter: Aranan karakter. Veri tipi char.

uzunluk: Belirtilen uzunlukta (sayıda) karakter içinde arama yapar.

```
void setup() {
  Serial.begin(9600);
```

```
}
void loop() {
  if (Serial.available() > 0) {
    bool ara = Serial.find('a');// Gelen veri içinde a karakterini ara.
    Serial.println(ara);// Gelen veride a karakteri varsa 1 yoksa 0 yazar.
  }
}
```

Serial.findUntil() Fonksiyonu

Seri porttan gelen veride sonlandırıcı karaktere kadar belirlenen karakteri arar. Fonksiyon karakteri bulursa true, bulamazsa false sonucunu döndürür.

```
Serial.findUntil(karakter, sonlandirici)
```

karakter: Aranan karakter. Veri tipi char.

sonlandırıcı: Aramanın sonlanacağı karakter. Veri tipi char.

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  if (Serial.available() > 0) {
    bool ara = Serial.findUntil("a","."); // '.' (nokta) karakterine kadar 'a' karakterini ara.
    Serial.println(ara);// 'a' karakteri varsa 1 yoksa 0 yazar.
  }
}
```


Serial.flush() Fonksiyonu

Seri porttan giden verinin iletiminin tamamlanmasını beklemek için kullanılır. Parametre almaz ve geriye değer döndürmez.

Serial.parseInt() Fonksiyonu

Seri porttan gelen veriler arasında tamsayı olanları alır. Rakam dışındaki ASCII karakterler atlanır. Long veri tipinde tamsayı döndürür. Rakam bulunmazsa 0 döndürür.

```
Serial.parseInt()
```

Örnek: Seri porttan gelen veriler arasında tam sayı olanları alır.

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  if (Serial.available() > 0) {
    int sayi = Serial.parseInt();// Gelen veride tamsayı varsa sayi değişkenine ata.
    Serial.print("Gelen veri içindeki sayı: ");
    Serial.println(sayi);
  }
}
```

Örnek: Bir kısmı verilen Arduino bluetooth araç programında, akıllı telefon uygulamasından bluetooth ile gönderilen karakterlerden a,b,c ve d yönü belirlerken tam sayı gönderildiğinde **hiz** değişkenine atanmaktadır.

```
void loop() {
  if (bluetooth.available()) {
     karakter = bluetooth.read();// Seri iletişimden gelen karakteri oku.
     hiz = bluetooth.parseInt();// Gelen veri tamsayıysa hiz değişkenine ata.
    if (karakter == 'a') { // Gelen karakter a'sa…
      // ileri
      analogWrite(IN1, hiz);
      analogWrite(IN2, 0);
      analogWrite(IN3, hiz);
      analogWrite(IN4, 0);
    } else if (karakter == 'b') {
      // geri
          // …
```

Serial.parseFloat() Fonksiyonu

Seri porttan gelen veriler arasında ondalıklı sayıları alır. Float veri tipinde ondalıklı sayı döndürür. Ondalıklı sayı bulunmazsa 0 döndürür.

```
Serial.parseInt()
```

Örnek: Ondalıklı sayıyı bulur.

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  if (Serial.available() > 0) {
    float sayi = Serial.parseFloat();// Ondalıklı sayıyı bul sayi değişkenine ata.
    Serial.print("Gelen veri içindeki sayı: ");
    Serial.println(sayi); // sayi değişkeni float veri tipindedir.
  }
}
```

Serial.setTimeout() Fonksiyonu

Seri porttan gelecek olan verinin en fazla ne kadar süre bekleneceği milisaniye cinsinden ayarlanır. Varsayılan ayar 1000 milisaniyedir.

```
Serial.setTimeout(zaman)
```

zaman: Milisaniye cinsinden beklenecek süre.

Serial.setTimeout() fonksiyonuyla ayarlanan süreyi aşağıdaki fonksiyonlar kullanır.

```
Serial.find()
Serial.findUntil()
Serial.parseInt()
```

```
Serial.parseFloat()
Serial.readBytes()
Serial.readBytesUntil()
Serial.readString()
Serial.readStringUntil()
```

Örnek: Program metin girişi için 10 saniye bekler.

```
void setup() {
  Serial.begin(9600);
  Serial.println("Lütfen adınızı giriniz");
  Serial.setTimeout(10000);
  String metin = Serial.readString(); // 10 saniye içinde gelen metni oku.
  if (metin.length() > 0) { // Metin değişkeninde veri varsa…
    Serial.print("Gelen metin--->  ");
    Serial.println(metin);
  } else  Serial.print("Belirlenen süre içinde giriş yapmadınız.");
}
void loop() {
  // Giriş bilgisi setup fonksiyonunda bir kez istendi.
}
```

Serial.peek() Fonksiyonu

Serial.read() foksiyonuyla benzerdir. Gelen verinin sadece ilk **byte**ını okur. Okunan veri arabellekten silinmez. Parametre almaz. Gelen verinin ilk **byte**ını döndürür.

```
Serial.peek()
void setup() {
  Serial.begin(9600);
}
void loop() {
  if (Serial.available() > 0) {
    char karakter = Serial.peek();// Karakter değişkenine seri hattan gelen veriyi yükle.
    Serial.print("Gelen karakter "); // Ekrana "Gelen karakter " yaz.
    Serial.println(karakter); // Karakter değişkeninin içeriğini yazdır.
  }
}
```

SerialEvent() Fonksiyonu

Ana programdan bağımsız olarak (kesme gibi) çalışır. Seri porttan veri geldiğinde aktif olur ve fonksiyon içine yazılan komutları çalıştırır. Parametre almaz. Değer döndürmez.

```
void serialEvent() {
  // Seri porttan veri geldiği anda buradaki komutlar çalışır.
}
```

Örnek: Program blink uygulamasını seri porttan A karakteri geldiğinde çalıştırır, B karakteri geldiğinde durdurur.

```
char karakter;
bool kontrolBiti;
void setup() {
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  if (karakter == 'A')
    kontrolBiti = 1;
  if (karakter == 'B')
    kontrolBiti = 0;
  if (kontrolBiti == 1)
  {
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
  }
}
void serialEvent() { // Seri porttan gelen veri varsa aşağıdaki kodları çalıştır.
  karakter = Serial.read(); // Gelen karakteri oku.
}
```

Karakter Fonksiyonları

Harf, rakam, noktalama işaretleri ve sembollerden oluşan karakterler üzerinde işlem yapan fonksiyonlardır. Bu fonksiyonların tümü parametre olarak tek karakter alır ve geriye true veya false döndürür.

isAlpha(): Karakter harf ise true döndürür.

```
isAlpha(karakter)
```

karakter: Char veri tipinde değişken.

Örnek: Karakter bulur.

```
char karakter1 = 'A';
char karakter2 = 9;
void setup() {
  Serial.begin(9600);
  if (isAlpha(karakter1)) {
    Serial.println("karakter1 harftir");
  } else {
    Serial.println("karakter1 harf değildir");
  }
  if (isAlpha(karakter2)) {
    Serial.println("karakter2 harftir");
```

```
  } else {
    Serial.println("karakter2 harf değildir");
  }
}
void loop() {}
```

isAlphaNumeric(): Karakter alfanümerikse (harf veya rakam) true döndürür.

Örnek: Alfanümerik karakter bulur.

```
char karakter1 = 'A';
char karakter2 = 9;
char karakter3 = '9';
void setup() {
  Serial.begin(9600);
  if (isAlphaNumeric(karakter1)) {
    Serial.println("karakter1 alfanümerik karakterdir.");
  } else {
    Serial.println("karakter1 alfanümerik karakter değildir.");
  }
  if (isAlphaNumeric(karakter2)) {
    Serial.println("karakter2 alfanümerik karakterdir.");
  } else {
    Serial.println("karakter2 alfanümerik karakter değildir.");
  }
  if (isAlphaNumeric(karakter3)) {
    Serial.println("karakter3 alfanümerik karakterdir.");
  } else {
    Serial.println("karakter3 alfanümerik karakter değildir.");
  }
}
void loop() {}
```

isAscii(): Karakter ASCII'se [American Standard Code for Information Interchange (ımerıkın standırd kod for informeyşın intırçeync)] true döndürür. ASCII karakterler 7 bitlik bir karakter kümesidir. 33 tane basılmayan (ekranda çıkmayan) kontrol karakteri ve 95 tane basılan karakter bulunur.

```
isAscii(karakter)
```

karakter: Char veri tipinde değişken.

Örnek: ASCII karakter bulur.

```
char karakter1 = 'A';
char karakter2 = 9;
char karakter3 = '9';
char karakter4 = '#';
char karakter5 = '©'; // Klavyeden Alt+3 giriniz. (3'ü Num Lock'tan giriniz.)
```

```
void setup() {
  Serial.begin(9600);
  if (isAscii(karakter1)) {
    Serial.println("karakter1 ASCII karakterdir.");
  } else {
    Serial.println("karakter1 ASCII karakter değildir.");
  }
  if (isAscii(karakter2)) {
    Serial.println("karakter2 ASCII karakterdir.");
  } else {
    Serial.println("karakter2 ASCII karakter değildir.");
  }
  if (isAscii(karakter3)) {
    Serial.println("karakter3 ASCII karakterdir.");
  } else {
    Serial.println("karakter3 ASCII karakter değildir.");
  }
  if (isAscii(karakter4)) {
    Serial.println("karakter4 ASCII karakterdir.");
  } else {
    Serial.println("karakter4 ASCII karakter değildir.");
  }
  if (isAscii(karakter5)) {
    Serial.println("karakter5 ASCII karakterdir.");
  } else {
    Serial.println("karakter5 ASCII karakter değildir.");
  }
}
void loop() {}
```

isControl(): Karakter kontrol karakteriyse (ASCII'nin ilk 33 karakteri) true döndürür.

`isControl(karakter)`

karakter: Char veri tipinde değişken.

Örnek: Kontrol karakteri bulur.

```
char karakter1 = 'A';
char karakter2 = 9;
void setup() {
  Serial.begin(9600);
  if (isControl(karakter1)) {
    Serial.println("karakter1 kontrol karakteridir.");
  } else {
    Serial.println("karakter1 kontrol karakteri değildir.");
```

```
  }
  if (isControl(karakter2)) {
    Serial.println("karakter2 kontrol karakteridir.");
  } else {
    Serial.println("karakter2 kontrol karakteri değildir.");
  }
}
void loop() {}
```

isDigit(): Karakter rakamsa true döndürür.

`isDigit(karakter)`

karakter: Char veri tipinde değişken.

Örnek: Rakam bulur.

```
char karakter1 = '9';
char karakter2 = 9;
void setup() {
  Serial.begin(9600);
  if (isDigit(karakter1)) {
    Serial.println("karakter1 rakam karakteridir.");
  } else {
    Serial.println("karakter1 rakam karakteri değildir.");
  }
  if (isDigit(karakter2)) {
    Serial.println("karakter2 rakam karakteridir.");
  } else {
    Serial.println("karakter2 rakam karakteri değildir.");
  }
}
void loop() {}
```

isGraph(): Karakter yazdırılabilir karakterse true döndürür.

`isGraph(karakter)`

karakter: Char veri tipinde değişken.

Örnek: Yazdırılabilir karakter bulur.

```
char karakter1 = 'A';
char karakter2 = 9;
void setup() {
  Serial.begin(9600);
  if (isGraph(karakter1)) {
    Serial.println("karakter1 yazdırılabilir karakterdir.");
  } else {
```

```
    Serial.println("karakter1 yazdırılabilir karakter değildir.");
  }
  if (isGraph(karakter2)) {
    Serial.println("karakter2 yazdırılabilir karakterdir.");
  } else {
    Serial.println("karakter2 yazdırılabilir karakter değildir.");
  }
}
void loop() {}
```

isHexadecimalDigit(): Karakter heksadesimal rakamsa (A-F, 0-9) true döndürür.

isHexadecimalDigit(karakter)

karakter: Char veri tipinde değişken.

Örnek: Heksadesimal rakam bulur.

```
char karakter1 = '11A';
char karakter2 = '5G';
void setup() {
  Serial.begin(9600);
  if (isHexadecimalDigit(karakter1)) {
    Serial.println("karakter1 heksadesimal karakterdir.");
  } else {
    Serial.println("karakter1 heksadesimal karakter değildir.");
  }
  if (isHexadecimalDigit(karakter2)) {
    Serial.println("karakter2 heksadesimal karakterdir.");
  } else {
    Serial.println("karakter2 heksadesimal karakter değildir.");
  }
}
void loop() {}
```

isLowerCase(): Karakter küçük harfse true döndürür.

isLowerCase(karakter)

karakter: Char veri tipinde değişken.

isUpperCase(): Karakter büyük harfse true döndürür.

isUpperCase(karakter)

karakter: Char veri tipinde değişken.

Örnek: Küçük veya büyük karakter bulur.

```
char karakter1 = 'a';
char karakter2 = 'A';
void setup() {
  Serial.begin(9600);
```

```
  if (isLowerCase(karakter1)) {
    Serial.println("karakter1 küçük harftir.");
  } else {
    Serial.println("karakter1 küçük harf değildir.");
  }
  if (isUpperCase(karakter2)) {
    Serial.println("karakter2 büyük harftir.");
  } else {
    Serial.println("karakter2 büyük harf değildir.");
  }
}
void loop() {}
```

isPrintable(): Karakter yazıcıda basılabilirse true döndürür.

`isPrintable(karakter)`

karakter: Char veri tipinde değişken.

Örnek: Yazıcıda basılabilir karakter bulur.

```
char karakter1 = 65; // 'A' ile aynıdır.
char karakter2 = 29;
void setup() {
  Serial.begin(9600);
  if (isPrintable(karakter1)) {
    Serial.println("karakter1 basılabilir.");
  } else {
    Serial.println("karakter1 basılamaz.");
  }
  if (isPrintable(karakter2)) {
    Serial.println("karakter2 basılabilir.");
  } else {
    Serial.println("karakter2 basılamaz.");
  }
}
void loop() {}
```

isPunct(): Karakter noktalama işaretiyse true döndürür.

`isPunct(karakter)`

karakter: Char veri tipinde değişken.

Örnek: Noktalama işareti bulur.

```
char karakter1 = '?';
char karakter2 = 'A';
void setup() {
  Serial.begin(9600);
```

```
  if (isPunct(karakter1)) {
    Serial.println("karakter1 noktalama işaretidir.");
  } else {
    Serial.println("karakter1 noktalama işareti değildir.");
  }
  if (isPunct(karakter2)) {
    Serial.println("karakter2 noktalama işaretidir.");
  } else {
    Serial.println("karakter2 noktalama işareti değildir.");
  }
}
void loop() {}
```

isSpace(): Karakter boşluk karakteriyse true döndürür. Boşluk karakterleri space, form beslemesi (‹\f›), yeni satır (‹\n›), satır başı (‹\r›), yatay sekme (‹\t›) veya dikey sekme (‹\v›) karakterleridir.

`isSpace(karakter)`

karakter: Char veri tipinde değişken.

Örnek: Boşluk karakterleri bulur.

```
char karakter1 = ' ';
char karakter2 = 'A';
void setup() {
  Serial.begin(9600);
  if (isSpace(karakter1)) {
    Serial.println("karakter1 boşluk karakteridir.");
  } else {
    Serial.println("karakter1 boşluk karakteri değildir.");
  }
  if (isSpace(karakter2)) {
    Serial.println("karakter2 boşluk karakteridir.");
  } else {
    Serial.println("karakter2 boşluk karakteri değildir.");
  }
}
void loop() {}
```

isWhitespace(): Karakter boşluk karakteri (space decimal 32) veya tab karakteriyse (\t) true döndürür.

`isWhitespace(karakter)`

karakter: Char veri tipinde değişken.

Örnek: Boşluk karakterleri bulur.

```
char karakter1 = ' ';
char karakter2 = 'A';
```

```
void setup() {
  Serial.begin(9600);
  if (isSpace(karakter1)) {
```

Bit ve Byte Fonksiyonları

Bit ve **byte** fonksiyonları bit seviyesinde işlem yapmak için kullanılan fonksiyonlardır.
bit(): Girilen bitin onluk değerini döndürür.

```
Sebit(n)
n: Hesaplanacak bit.
```

Örnek: On basamaklı binary sayının onluk basamaktaki sınırını verir.

```
void setup() {
Serial.begin(9600);
Serial.println(bit(10)); // 1024 döndürür.
}
void loop() {}
```

bitClear(): Sayının bir **bit**ini siler (0 yapar.). Yeni sayıyı döndürür.

isbitClear(x, n)

x: Biti silinecek sayı.

n: Silinecek **bit** numarası. En düşük anlamlı (sağ) **bit**ten başlar.

Örnek: x ve n int tipinde iki sayıdır. 6'nın ikilik karşılığı 0110'dır. n=1 olduğundan sağdan ikinci **bit** 0 olur. İkilik sayı 0100 olur. Onluk karşılığı 4'tür.

```
void setup() {
  Serial.begin(9600);
  int x = 6;
  int n = 1;
  Serial.print(bitClear(x, n)); // Sonuç 4.
}

void loop() {}
```

bitSet(): Sayının bir **bit**ini 1 yapar. Yeni sayıyı döndürür.

bitSet(x, n)

x: Bitlerinden biri 1 yapılacak sayı.

n: 1 yapılacak **bit** numarası. En düşük anlamlı (sağ) **bit**ten başlar.

Örnek: Sekiz LED'li kara şimşek programıdır.

```
byte i;
int sure = 100;
byte sayi = 255; // İkilik B11111111›dir.
void setup() {
  DDRD = sayi; // 0-7 numaralı pinler çıkış olarak ayarlandı.
}
```

```
void loop() {
 for (i = 0; i <= 7; i++)  {
    PORTD = bitSet(sayi, i); // Sayının bitlerini sırayla 1 yap.
    delay(sure);
    PORTD = bitClear(sayi, i); // Sayının bitlerini sırayla 0 yap.
  }
  for (i = 6; i > 0; i--) { // İlk ve son LED dönüşlerde bir kez yanar.
    PORTD = bitSet(sayi, i);
    delay(sure);
    PORTD = bitClear(sayi, i);
  }
}
```

bitRead(): Sayının bir **bit**ini okur. **Bit**in değerini (0 veya 1) döndürür.

bitRead(x, n)

x: Biti okuncak sayı.

n: Okunacak **bit** numarası. En düşük anlamlı (sağ) **bit**ten başlar.

Örnek: Sekiz LED'li animasyon programıdır.

```
byte j, i;
int sure = 100;
byte sayilar[] = {129, 66, 36, 24, 24, 36, 66, 129};
void setup() {
  for (i = 0; i <= 7; i++)  {
    pinMode(i, OUTPUT); // 0 - 7 numaralı pinleri çıkış yap.
  }
}
void loop()
{
  for (j = 0; j <= 7; j++) {
    for (i = 0; i <= 7; i++) {
      digitalWrite(i, bitRead(sayilar[j], i));// Sıradaki sayının i'inci bitini çıkışa yaz.
    }
    delay(sure);
  }
}
```

bitWrite(): Sayının bir **bit**ini değiştirir. **Bit**in değerini (0 veya 1) döndürür.

bitWrite(x, n, b)

x: Biti değiştirilecek sayı.

n: Değiştirilecek **bit** numarası. En düşük anlamlı (sağ) **bit**ten başlar.

b: Bitin değeri (0 veya 1) .

```
void setup() {
  Serial.begin(9600);
  byte x = 0b10000000;
  Serial.println(x, BIN); // 10000000
```

```
  bitWrite(x, 0, 1);  // x sayısının 0. bitine 1 yaz.
  Serial.println(x, BIN); // 10000001
}
void loop() {}
```

highByte(): İki **byte**lık sayının soldaki yüksek değerlikli **byte**ını verir. İki **byte**tan büyük sayılarda sağdan (LSB) ikinci **byte**ı verir.

`bithighByte(x)`

x: Herhangi bir veri tipindeki sayı.

lowByte(): İki **byte**lık sayının sağdaki düşük değerlikli **byte**ını verir. İki **byte**tan büyük sayılarda en sağdaki (LSB) **byte**ı verir.

`lowByte(x)`

x: Herhangi bir veri tipindeki sayı.

1.9. ANALOG GİRİŞ ÇIKIŞ İŞLEMLERİ

Arduino analog girişler için 10 **bit**lik (0 -1023) A0-A5 pinlerini kullanırken analog çıkışlar için dijital pinlerden ~ işareti olanları kullanır.

1.9.1. Analog Giriş İşlemleri

Analog işlemlerde kullanılan komutlar okuma, yazma ve referans olmak üzere üç çeşittir.

AnalogRead() Fonksiyonu

A0-A5'le belirtilen analog pinlerin değerini okur. Arduino kartları 10 **bit** analogdan dijitale dönüştürücü (ADC) içerir. 0 V ile 5 V arasındaki giriş voltajlarını 0 ile 1023 arasındaki tam sayı değerlerine eşler. Her bir adım başına 5/1024=0,0049 V (4,9 mV) düşmektedir. Fonksiyon girişteki gerilime göre 0-1023 arasında int veri tipinde değer döndürür. analogReference(EXTERNAL) fonksiyonuyla AREF pininden verilen 5V'tan düşük voltaj tepe değerini oluşturmaktadır. Örneğin AREF 4 V ise 1023 4 V'a eşlenir.

Uno, Nano, Mini, Mega'da bir analog girişi okumak yaklaşık 100 mikrosaniye (0,0001 sn.) sürer, bu nedenle maksimum okuma hızı saniyede yaklaşık 10.000 defadır.

`analogRead(pin)`
pin: Analog giriş pininin adı. Uno'da A0 - A5, Mini ve Nano'da A0 - A7, Mega'da A0 - A15.

Örnek: Tüm analog çıkışlı sensörler için kullanılan temel program.

```
const byte cikisPini = 13;
void setup() {
  Serial.begin(9600); //Seri iletişim başlatıldı.
  pinMode(cikisPini, OUTPUT);}
void loop() {
  int analogGirisDegeri = analogRead(A0); //A0 girişinden okunan 0-1023 arası veriyi
analogGirisDegeri değişkenine yükle.
  Serial.println(analogGirisDegeri); //analogGirisDegeri değişkeninin içeriğini yazdır.
  if (analogGirisDegeri < 800) { // Karşılaştırma satırı.
    digitalWrite(cikisPini, HIGH); // Karşılaştırma geçerliyse yapılacaklar.}
  else { // Karşılaştırma geçerliyse değilse yapılacaklar.
    digitalWrite(cikisPini, LOW);}  }
```

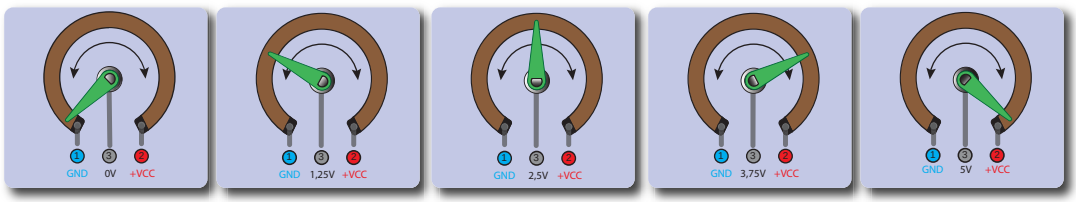
Uygulama Adı: Potansiyometreyle Analog Giriş Uygulaması No.: 11

Amaç: Potansiyometreyle analog giriş uygulaması yapmak.

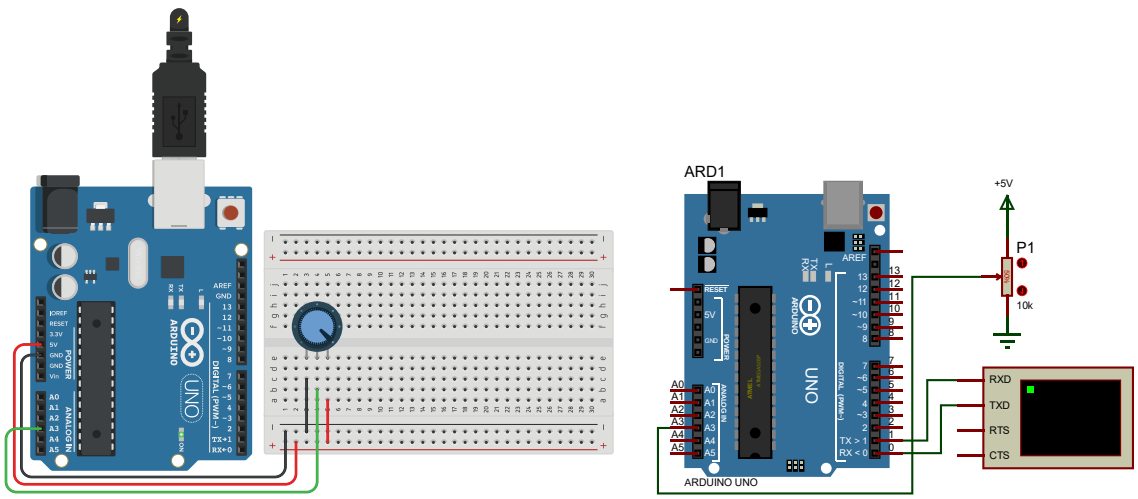
Görsel 1.38'de potansiyometrenin pin yapısı ve yapılan ayara göre orta ucundan alınan gerilim değerleri görülmektedir. Potansiyometrenin bir kenardaki ucu GND'ye, diğer kenardaki ucuysa 5 V'a bağlanmıştır. Gerilim bölücü olarak çalışan bu potansiyometrenin orta ucundan 0-5 V arasında bir gerilim elde edilir. Bu gerilim Arduino'nun A3 analog girişine uygulanmıştır (Görsel 1.39).

NOT

Analog girişleri pinMode() fonksiyonuyla giriş olarak tanımlamaya gerek yoktur.



Görsel 1.38: Potansiyometre pin yapısı



Görsel 1.39: Potansiyometreyle analog giriş şeması ve devresi

Potansiyometreyle analog giriş uygulama programı aşağıdaki gibidir:

```
void setup() {
  Serial.begin(9600); // Seri iletişim başlatıldı.
// Analog pinleri giriş olarak tanımlamaya gerek yoktur.
}
void loop() {

  int pot = analogRead(A3); // A3 girişinden okunan 10 bitlik veriyi pot değişkenine yükle.
```

Potansiyometreyle analog giriş uygulama programı aşağıdaki gibidir:

```
void setup() {
  Serial.begin(9600); // Seri iletişim başlatıldı.
// Analog pinleri giriş olarak tanımlamaya gerek yoktur.
}
void loop() {

  int pot = analogRead(A3); // A3 girişinden okunan 10 bitlik veriyi pot değişkenine yükle.
  Serial.println(pot); // Pot değişkeninin içeriğini yazdır.
}
```

Voltmetre uygulama programı aşağıdaki gibidir:

```
void setup() {
  Serial.begin(9600);
}
void loop() {

  int pot = analogRead(A3); /* A3 pininden 0 - 1023 arasında okunan sayıyı pot değişkenine yükle. */

   float gerilim = (5.0/1024.0) * pot; /* pot değerini ~0,0049 değeriyle çarp gerilim değişkenine yükle. 0.0 - 5.0 arasında küsuratlı sayılar elde edeceğimiz için gerilim değişkeni float olarak tanımlandı. */
 Serial.print(gerilim);
  Serial.println(" Volt");
}
```

DİKKAT İngilizcede küsuratlı sayılarda nokta (3.14) kullanılırken Türkçede virgül (3,14) kullanılmaktadır.

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Görsel 1.39'daki devreyi kurunuz.
3. Programı yazıp Arduino'ya yükleyiniz.
4. Potansiyometrenin GND'ye bağlı ucuyla orta ucunu, potansiyometrenin konumunu değiştirerek voltmetreyle ölçüm değerlerini gözlemleyiniz.

SIRA SİZDE

1. Blink uygulamasındaki LED'in yanma süresini potansiyometreyle değiştirecek şekilde programı yazınız.

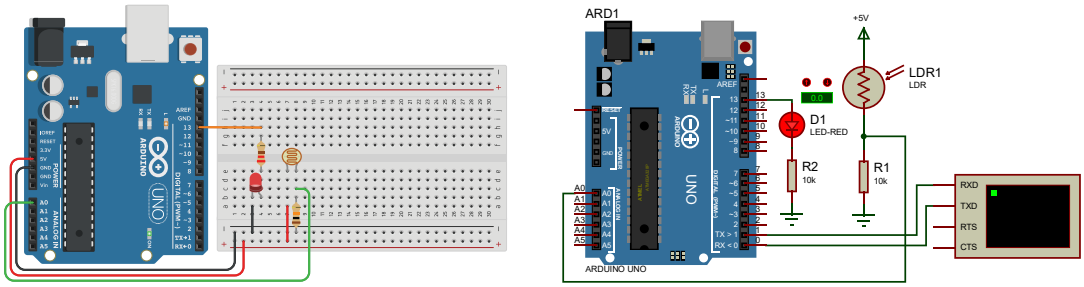
Sorular

1. Pot değişkeni neden **byte** yerine int tipinde tanımlanmıştır? Açıklayınız.
2. 30 V'luk gerilim kaynağı Arduino ile nasıl ölçülür? Belirtiniz.

Uygulama Adı: LDR'yle Analog Giriş Uygulaması — No.: 12

Amaç: **LDR'yle analog giriş uygulaması yapmak.**

LDR, ışıkla direnci ters orantılı olarak değişen elemandır. Arduino'nun 5 V gerilimi üzerine uygulanan LDR'nin direnç-gerilim değişiminin anlaşılabilmesi için 10 kΩ'luk bir direnç seri bağlanarak gerilim bölücü oluşturulur. Bağlantı noktasının gerilim değişimi Arduino'nun analog girişine verilir (Görsel 1.40).



Görsel 1.40: LDR'yle analog giriş şeması ve devresi

LDR'yle analog giriş uygulama programı aşağıdaki gibidir:

```
const byte LED = 13; // LED değişkeninin içeriği 13'le sabitlendi.
int LDR;
void setup() {
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  LDR = analogRead(A0); // Analog giriş bilgisini LDR değişkenine yükle.

  if (LDR < 800)  // 800 değeri aydınlık/karanlık durumuna göre istenilen değerle değiştirilebilir.
    digitalWrite(LED, 1);
  else
    digitalWrite(LED, 0);
  Serial.println(LDR); // Seri port ekranından LDR'nin aldığı değer gözlenebilir.
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 1.40'taki devreyi kurunuz.
4. Seri port ekranını açınız.
5. Elinizle LDR'nin üzerini kapatarak seri ekrandaki değeri ve devredeki LED'i gözlemleyiniz.
6. LED'i yakıp söndüren en uygun değeri programda güncelleyerek tekrar deneyiniz.
7. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Programı aydınlıkta LED yanacak şekilde düzenleyiniz.

2. Aşağıdaki program LDR'nin algıladığı ışık değişimini ses değişimi olarak vermektedir.

```
const byte buzzer = 13; // Buzzer 13 numaralı pine bağlı.
void setup() {
  pinMode(buzzer, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  int LDR = analogRead(A0); // Analog giriş bilgisini LDR değişkenine yükle.

  int frekans = LDR * 1; // LDR içindeki değeri 1'le çarp. (2, 3... ile çarpılarak frekans artırılabilir.)

  tone(buzzer, frekans); // 13 numaralı pinde frekans değişkeninin değerinde kare dalga (%50 duty cycle) oluşturur.

  Serial.println(frekans); // Seri port ekranından frekans gözlenebilir.
}
```

DİKKAT

Analog giriş uygulamalarında analog çıkışlı sensör modülleri de kullanılabilir.

Sorular

1. 1023 değeri 5 V'sa 512 değeri kaç volttur? Hesaplayınız.

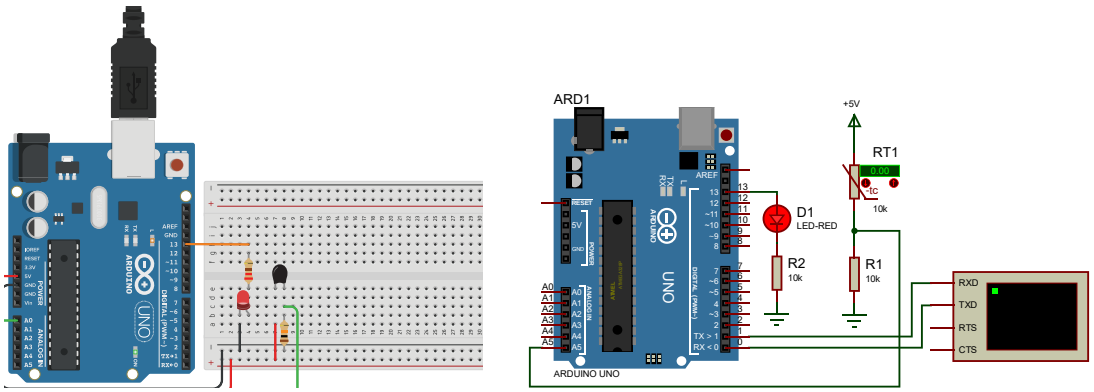
2. LDR değişkeninin değeri neden belli bir değerin altına inememektedir? Belirtiniz.

3. 10 kΩ'luk direnç 4,7 kΩ ile değiştirilirse gerilim bölücünün orta ucunda oluşan referans gerilimi 0 V ile 5 V'tan hangisine yaklaşır? Açıklayınız.

Uygulama Adı: NTC'yle Analog Giriş Uygulaması — No.: 13

Amaç: NTC'yle analog giriş uygulaması yapmak.

Görsel 1.41'deki devrede NTC ve NTC'ye seri bağlı direnç gerilim bölücü olarak çalışmaktadır. Oda sıcaklığında (yaklaşık 25 °C) NTC'nin direnci yaklaşık 8 kΩ-10 kΩ'dur. NTC'nin sıcaklığı arttırıldığında direnci ve üzerine düşen gerilim azalır. Böylelikle 10 kΩ'luk sabit direnç üzerindeki gerilim artar. Direnç üzerindeki bu gerilim analog giriş tarafından algılanarak değeri 1023'e doğru yaklaşır. Bu değer belirlenen referans değerini aştığında uyarı LED'i yanar.



Görsel 1.41: NTC'yle analog giriş şeması ve devresi

NTC'yle analog giriş uygulama programı aşağıdaki gibidir:

```
const byte LED = 13, ntcPin = 19; // A5'e (19 numaralı pin) NTC bağlandı.
int ntcDegeri;
void setup() {
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  ntcDegeri = analogRead(ntcPin); // A5'ten gelen veriyi ntcDegeri değişkenine yükle.
  if (ntcDegeri > 800)  // 800'den büyükse...
    digitalWrite(LED, HIGH);
  else
    digitalWrite(LED, LOW);
  Serial.println(ntcDegeri); // Seri port ekranına ntcDegeri yazdır.
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 1.41'deki devreyi kurunuz. Programı yükleyiniz.
4. NTC'yi ısıtınız.
5. Seri ekrandan analog değeri gözlemleyiniz.
6. Referans değerini farklı bir değerle değiştirerek uygulamayı tekrarlayınız.

SIRA SİZDE

1. PNTC'yle ortam sıcaklığını ölçmek mümkün müdür? Araştırınız.

```
#include <math.h> // Matematik kütüphanesini çağır.
const byte LED = 13;

void setup() {
  Serial.begin(9600); // Seri iletişimi başlat.
  pinMode(LED, OUTPUT); // 13 numaralı pini çıkış yap.
}
void loop() {

  int analogGirisDegeri = analogRead(A0); // A0'dan analog giriş değerini oku.

  // NTC için analog değer - sıcaklık dönüşüm formülü.
  double sicaklik = log(((10240000 / analogGirisDegeri) - 10000));
  sicaklik = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * sicaklik * sicaklik))
* sicaklik);
  sicaklik = sicaklik - 273.15;
  Serial.println(sicaklik); // Sıcaklığı ekrana yazdır.

  if (sicaklik > 30) { // Sıcaklık 30 dereceden yüksekse...
    digitalWrite(LED, HIGH);
  } else {
    digitalWrite(LED, LOW);
  }
  delay(500);
}
```

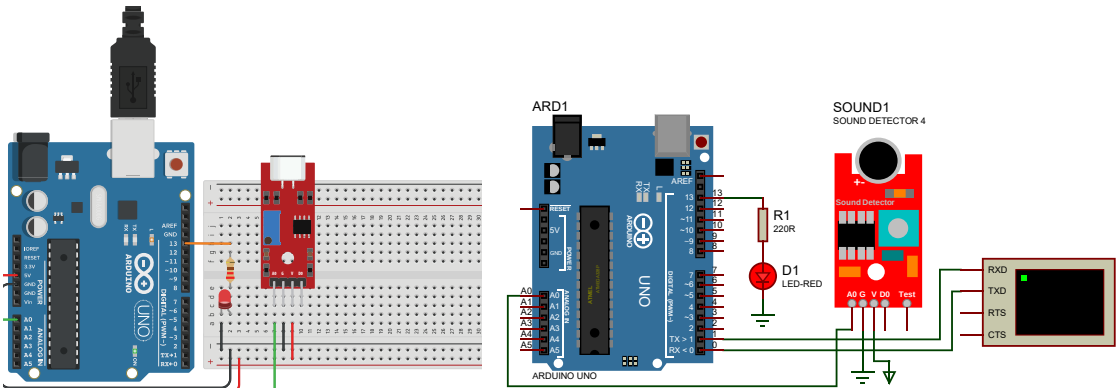
Soru

1. Analog giriş değeri 800'ü aştığında çıkıştaki motoru durdurma bilgisi göndermek için programda nasıl değişiklik yapılmalıdır? Belirtiniz.

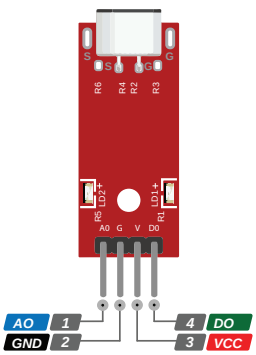
Uygulama Adı: Alkışla Çalışan LED Uygulaması — No.: 14

Amaç: **Alkışla çalışan LED uygulaması yapmak.**

Görsel 1.42a'da alkışla çalışan LED uygulaması görülmektedir. Analog çıkışlı mikrofon modülünden gelen sinyal belli bir seviyeyi aştığında çıkıştaki LED yanmaktadır. Seviye tekrar aşıldığında LED sönmektedir. Görsel 1.42b'de ise mikrofon devresinin pin sıralaması verilmiştir.



Görsel 1.42a: Alkışla çalışan LED şeması ve devresi



Görsel 1.42b: Mikrofon modülü pin yapısı

A0: Analog veri çıkışı.
GND: Toprak pini.
VCC: 5 V besleme.
D0: Dijital veri çıkışı.

Alkışla çalışan LED uygulama programı aşağıdaki gibidir:

```
const byte led = 12;
int mic;
bool degistir = 1; // Boolean tipinde oluşturulan değiştir isimli değişkene 1 yüklendi.
void setup() {
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}
void loop() {
```

```
  mic = analogRead(A0); // Analog girişten gelen veriyi (0 - 1023) mic değişkenine yükle.
  if (mic > 59) { // Seri ekrandan uygun referans noktası görülebilir.
    digitalWrite(led, degistir); // ilk seferde LED'i yakar.
    degistir = !degistir; // Boolean değişken olan degistir'i tersle.
    delay(100); // Debounce gecikmesi (Çok hızlı-kararsız yanıp sönmeyi engellemek için.)
  }
  Serial.println(mic); // Analog bilginin değerini göster.
}
    digitalWrite(LED, HIGH);
  else
    digitalWrite(LED, LOW);
  Serial.println(ntcDegeri); // Seri port ekranına ntcDegeri yazdır.
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 1.42a'daki devreyi kurunuz. Programı yükleyiniz.
4. Seri ekrandan analog girişin aldığı değerleri gözlemleyiniz.
5. Alkış ve yüksek seslerde LED'in sırayla yanıp söndüğünü gözlemleyiniz.
6. Donanımınızı ses şiddetine göre referans değerini değiştirip tekrar deneyiniz.
7. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Alkışla yanan LED beş saniye sonra sönecek şekilde programı düzenleyiniz.

Sorular

1. `digitalWrite(13, !digitalRead(13));` komutu ne yapar? Araştırınız.
2. `digitalWrite(13, digitalRead(13)^1);` komutu ne yapar? Araştırınız.

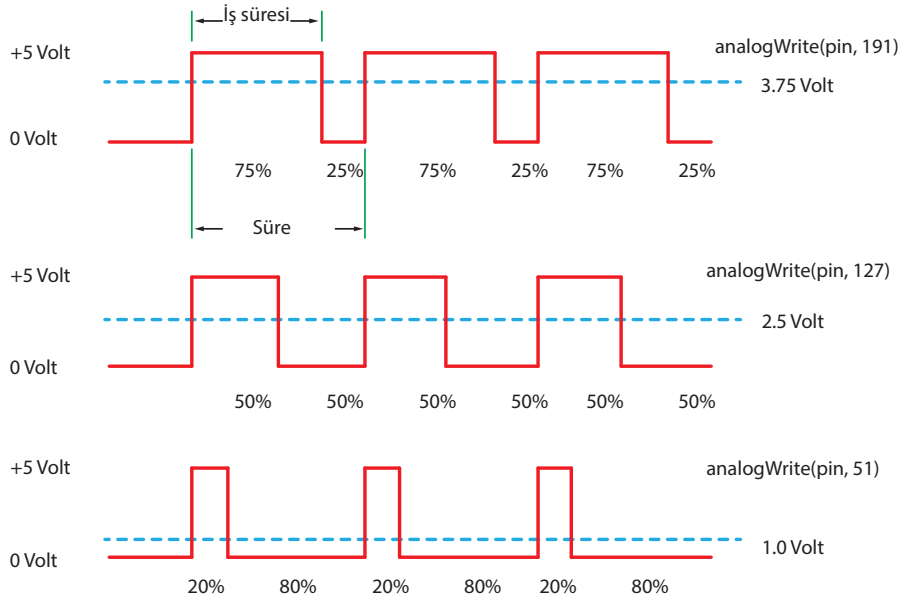
1.9.2. Analog Çıkış İşlemleri

Her mikrodenetleyicinin çıkışından gerçek anlamda analog gerilim alınamaz. Analog çıkış gerilimleri genellikle sayısal PWM sinyalleri ile elde edilir.

analogWrite() Fonksiyonu

Seçilen pinde PWM sinyalinin üretilmesini sağlar. PWM sinyalinin görev zamanı (duty cycle) fonksiyona verilen değerlerle belirlenir. Bir LED'i değişen parlaklıklarda yakmak veya bir motoru çeşitli hızlarda sürmek için kullanılabilir.

PWM [Pulse Width Modulation (puls wîdth mecıleyşın)] darbe genişliği modülasyonu, dijital bir sinyal olup (sürekli değişen 0-5 V), anahtarlamayla 5 V'ta kalma süresi (duty cycle) ayarlanarak kısmi analog sinyal oluşturmaktır. Arduino'da dijital analog dönüştürücü (DAC) 8 bit olduğundan çıkıştaki analog sinyal 256 (0-255) adımdan oluşur. Görsel 1.43'te analogWrite(pin, 51) verildiğinde duty cycle %20 ve çıkış 1 V'tur. analogWrite(pin, 127) verildiğinde duty cycle %50 ve çıkış 2,5 V'tur. analogWrite(pin, 191) verildiğinde duty cycle %75 ve çıkış 3,75 V'tur. Her bir adım 5/256=0,019 V'tur.



Görsel 1.43: PWM sinyali

analogWrite() fonksiyonu Uno , Nano, Mini'de 3, 5, 6, 9, 10 ve 11 numaralı pinlerde kullanılır. PWM frekansı 490 Hz'dir (5 ve 6. pinlerde 980 Hz.). Bu pinler pinMode() fonksiyonuyla çıkış olarak ayarlanmadan da analogWrite() fonksiyonuyla kullanılabilir.

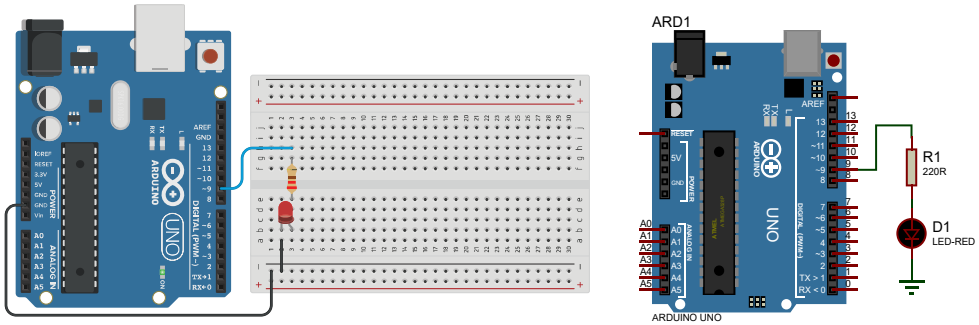
```
analogWrite(pin,deger)
```

pin: PWM özellikli pinlerden biri. Veri tipi int.
deger: 0-255 arası PWM değeri. Veri tipi int.

Uygulama Adı: Analog Çıkış Uygulaması — No.: 15

Amaç: Analog çıkış uygulaması yapmak.

Görsel 1.44'teki uygulamada LED 9 numaralı PWM özellikli pine bağlanmıştır. LED'in parlaklığını artırıp azaltmak için analogWrite() fonksiyonun ikinci parametresi 0-255 arası değiştirilir. 255'e yaklaştıkça parlaklık artmakta, 0'a yaklaştıkça parlaklık azalmaktadır.



Görsel 1.44: Analog çıkış şeması ve devresi

Analog çıkış uygulama programı aşağıdaki gibidir:

```
const byte ledPin = 9;     // LED 9 numaralı pine bağlı.
void setup() {
  // PWM pinleri çıkış olarak ayarlamaya gerek yoktur.
}
void loop() {
  for (int pwm = 0 ; pwm <= 255; pwm += 5) { // 0'dan 255'e 5'er artır.
    analogWrite(ledPin, pwm);
    delay(30);
  }
  for (int pwm = 255 ; pwm >= 0; pwm -= 5) { // 255'ten 0'a 5'er azalt.
    analogWrite(ledPin, pwm);
    delay(30);
  }
}
```

İşlem Basamakları

1. Görsel 1.44'teki devreyi kurunuz.
2. Programı yükleyerek LED'in parlaklığını gözlemleyiniz.
3. PWM'i birer adım arttırıp azaltarak deneyiniz.

SIRA SİZDE

1. Benzer bir programı for döngüsü kullanmadan yazınız.

```
const byte led = 9;
byte x, y = 0;
```

```
void setup() {
}
void loop() {
  if (y == 0) {
    x = x + 1;
    analogWrite(led, x);
    delay(10);
  }
  if (x == 255)
  {
    y = y - 1;
    analogWrite(led, y);
    delay(10);
  }
}
```

2. RGB LED'le 255^3 renk gösterimi (Ortak anot veya ortak katot kullanılabilir.).

```
const byte kirmizi=9, yesil=10, mavi=11; // PWM pinleri.
void setup() {
}
void loop() {
  for (int a = 0; a <= 255; a++) {
    analogWrite(kirmizi, a);
    for (int b = 0; b <= 255; b++) {
      analogWrite(yesil, b);
      for (int c = 0; c <= 255; c++) {
        analogWrite(mavi, c);
      }
    }
  }
// Kodu buraya yazınız.
}
```

3. **Önceki programda 255^3** renk gösterimi bittiğinde 1 saniye aralıklarla **kırmızı**, **yeşil** ve **mavi** yakacak kodu ekleyiniz.

Sorular

1. Hangi PWM değeri 1.25 V çıkış verir? Belirtiniz.

2. Analog sinyalde 0 V ile 5 V arasında sonsuz değer varken 8 **bit**lik çıkışta 0 V ile 5 V arasında kaç değer vardır? Açıklayınız.

Matematiksel Fonksiyonlar

abs(): Bir sayının mutlak değerini verir. Bir sayının mutlak değeri sıfıra olan uzaklığıdır. Parametre olarak sayı alır. Sayı sıfır veya pozitifse sayının kendisini, sıfırdan küçükse (negatifse) sayıyı pozitif olarak döndürür.

`abs(x)`

`x:` Sayı.

Örnek: Kod parçasında bluetooth üzerinden gelen negatif veri PWM çıkışlarına gönderilmektedir.

```
if(data[0] < -50)// geri
   {
     analogWrite(in1,0);
     analogWrite(in2,abs(data[0]));
     analogWrite(in3,0);
     analogWrite(in4,abs(data[0]));
   }
```

constrain(): Aralık sınırlamak için kullanılır. Üç adet parametre alır. Değişken sınırlar içindeyse kendisini, alt sınır altındaysa alt sınırı, üst sınır üstündeyse üst sınır değerini döndürür.

`bsconstrain(degisken, altSinir, ustSinir)`

`degisken:` Sınırlandırılacak değişken.

`altSinir:` Alt sınır değeri.

`ustSinir:` Üst sınır değeri.

Örnek: A0 girişinden gelen 0-1023 arasındaki değerlerin 50-150 arasındaki değerlerini kullanır.

```
void void setup() {
  Serial.begin(9600);
}
void loop() {
  int pot = analogRead(A0);
  int aralik = constrain(pot, 50, 150);
  Serial.println(aralik);
}
}
```

map(): Belli bir aralıktaki değeri başka bir aralığa dönüştürür. Beş adet parametre alır. Dönüştürülmüş aralıktaki değerleri int veri tipinde döndürür.

`map(deger, altDeger, ustDeger, yeniAltDeger, yeniUstDeger)`

`deger:` Dönüştürülecek değer.

`altDeger:` Dönüştürülmek istenen verinin alt değeri.

`ustDeger:` Dönüştürülmek istenen verinin üst değeri.

`yeniAltDeger:` Dönüştürülmüş alt değer.

Örnek: Aralığı tersten verir.

```
y = map(x, 1, 50, 50, 1);
```

Örnek: İvme sensöründen gelen -16384 ile 16384 arasındaki değerler -255 ile 255 aralığına dönüştürülmektedir.

```
data[0] = map(x, -17000, 17000, -255, 255 ); // X düzleminin verisi (ileri-geri)
data[1] = map(y, -17000, 17000, -255, 255); // Y düzleminin verisi (sağ-sol)
```

NOT

Eğer ondalık sayılarda dönüştürme yapılmak istenirse aşağıdaki map() fonksiyonunun matematiksel eş değeri olan işlem kodlarına dönüştürülmelidir.

$$y = \frac{(x - y_{min}) * (x_{max} - x_{min})}{(y_{max} - y_{min}) + x_{min}}$$

Örnek: 100'lük not sisteminde 70'in karşılığını 4'lük sistemde verir.

```
double donustur(double deger, double altDeger, double ustDeger, double yeniAltDeger, double yeniUstDeger)
{
  return (deger - altDeger) * (yeniUstDeger - yeniAltDeger) / (ustDeger - altDeger) + yeniAltDeger;
}
void setup() {
  Serial.begin(9600);
  Serial.println(map(70.0, 0.0, 100.0, 0.0, 4.0)); // Sonuç 2,80'dir.
  // Float ve double veri tiplerinde sayılar noktalı girilir.
}
void loop() {}
```

max(): İki sayıdan büyük olanını seçer (Döndürür.).

```
max(x, y)
```

x: Birinci sayı.

y: İkinci sayı.

Örnek: İki analog girişten büyük olanı seçer. 50'nin altındaki değerleri eler.

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  int sensor1 = analogRead(A0);
  int sensor2 = analogRead(A1);
  int deger = max(sensor1, sensor2); // İki sensör değerinden büyük olanını seç.
  int sonuc = max(deger, 50); // Sonuç 50 veya üstü olur.
  Serial.println(sonuc);
}
```

min(): İki sayıdan küçük olanını seçer.
min(x, y)
x: Birinci sayı.
y: İkinci sayı.

Örnek: İki analog girişten küçük olanı seçer. 800'ün üstündeki değerleri eler.

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  int sensor1 = analogRead(A0);
  int sensor2 = analogRead(A1);
  int deger = min(sensor1, sensor2); // İki sensör değerinden küçük olanını seç.
  int sonuc = min(deger, 800); // Sonuç 800 veya altı olur.
  Serial.println(sonuc);
}
```

pow(): Sayının üssünü alır.

```
pow(taban,üs)
```

taban: Taban sayısı.

üs: Üs sayısı.

Örnek: 2^3 işleminin sonucunu verir.

```
sonuc = pow(2, 3); // Sonuç 8 olur.
```

sq(): Sayının karesini alır (Kendisiyle çarpar.).
sq(x)
x: Karesi alınacak sayı.

Örnek: 10^2 işleminin sonucunu verir.

```
sonuc = sq(10); // Sonuç 100 olur.
```

sqrt(): Sayının karekökünü verir.

sqrt(x)

x: Karekökü alınacak sayı.

Örnek: 16 işleminin sonucunu verir.

```
sonuc = sq(16); // Sonuç 4 olur.
```

random(): Rastgele sayı üretir. Long veri tipinde sayı döndürür.

```
random(max)
random(min, max)
```

min: Üretilecek sayının en küçük değeri (İsteğe bağlı.).

max: Üretilecek sayının en büyük değeri.

Örnek: Rastgele sayı üretir.

```
long rastgeleSayi;
void setup() {
  Serial.begin(9600);
}
void loop() {
  rastgeleSayi = random(300); // 0 - 299 arası rastgele sayı üret.
  Serial.println(rastgeleSayi);
  rastgeleSayi = random(10, 20); // 10 - 19 arası rastgele sayı üret.
  Serial.println(rastgeleSayi);
  delay(500);
}
```

randomSeed(): Yukarıdaki örnek seri port ekranını her kapatıp açmada aynı sırada rastgele sayı üretir. randomSeed() içine sabit bir sayı girildiğinde de sonuç aynıdır. Ancak randomSeed() her seferinde farklı bir değer alırsa üretilen rastgele sayı düzeni de farklı olacaktır. Parametre olarak boşta olan bir analog giriş seçilebilir. Analog girişler boştayken ortama göre farklı değerler alır.

`randomSeed(seed)`

seed: Rastgele sayı düzenini başlatacak sayı. Veri tipi unsigned long.

Örnek: Başlama düzenini analog girişten rastgele seçer.

```
long rastgeleSayi;
void setup() {
  Serial.begin(9600);
  randomSeed(analogRead(0));
}
void loop() {
  rastgeleSayi = random(300); // 0 - 299 arası rastgele sayı üret.
  Serial.println(rastgeleSayi);
  delay(500);
}
```

Örnek: LED'lerle rastgele animasyon programıdır.

```
byte i;
int sure = 100;
void setup() {
  for (i = 0; i <= 7; i++)  {
    pinMode(i, OUTPUT); // 0 - 7 pinleri çıkış yap.
  }
  randomSeed(analogRead(0));
}
void loop() {
  for (i = 0; i <= 7; i++) {
    digitalWrite(i, bitRead(random(256), i));
  }
  delay(sure);
}
```

Trigonometrik Fonksiyonlar

cos(): Açının kosinüs değerini radyan olarak verir. -1 ile 1 arasında double veri tipinde değer döndürür.

Örnek: Kosinüsün radyan değerini verir.

```
sonuc = cos(90); // Sonuc -0,45 olur.
```

sin(): Açının sinüs değerini radyan olarak verir. -1 ile 1 arasında double veri tipinde değer döndürür.

Örnek: Sinüsün radyan değerini verir.

```
sonuc = sin(90); // Sonuç 0,89 olur.
```

tan(): Açının tanjant değerini radyan olarak verir.

Örnek: Tanjantın radyan değerini verir.

```
sonuc = tan(90); // Sonuç -2 olur.
```

Örnek: Açı-radyan dönüşümlerini verir.

```
void setup() {
  Serial.begin(9600);
  Serial.println(cos(90)); // -0,45
  Serial.println(sin(90)); // 0,89
  Serial.println(tan(90)); // -2,00
}
void loop() {}
```

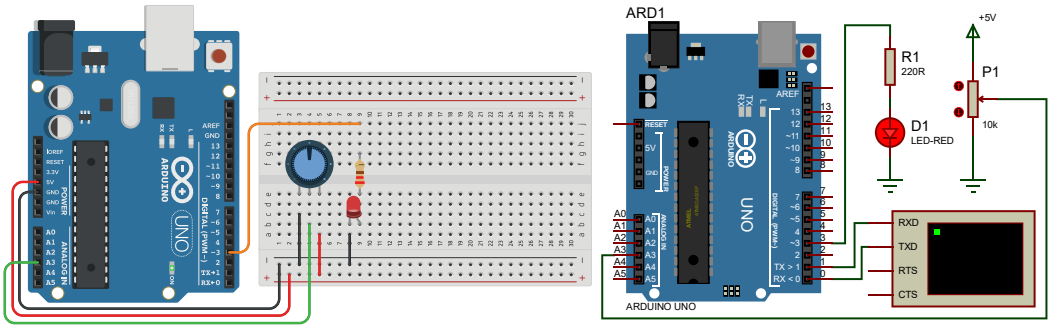
DİKKAT Matematiksel fonksiyonların içinde işlem yapılması yanlış sonuçlar vermesine neden olabilir.

```
abs(a++); // Bu kullanımdan kaçınılmalı. Yanlış sonuçlar verebilir.
// Bu kullanım tercih edilmelidir.
abs(a);
a++;  // İşlemler fonksiyon dışında yapılmalıdır.
```

Uygulama Adı: Analog Giriş Çıkış Uygulaması — No.: 16

Amaç: Analog giriş çıkış uygulaması yapmak.

Görsel 1.45'teki devrede potansiyometre 10 bitlik analog girişe bağlıyken LED 8 bitlik PWM çıkışına bağlıdır. Potansiyometreden alınan 0 -1023 arası değer doğrudan PWM çkışına verilirse (analogWrite(3, pot)) 8 bitlik PWM çıkışı potansiyometrenin bir tam dönüşünde dört tur dönecektir. map(pot, 0, 1023, 0, 255) fonksiyonu pot değişkeninden gelen 0 -1023 aralığını 0-255 aralığına eşler.



Görsel 1.45: Analog giriş çıkış şema ve devresi

Analog giriş çıkış uygulama programı aşağıdaki gibidir:

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  int pot = analogRead(A3);
  byte pwm = map(pot, 0, 1023, 0, 255);
  analogWrite(3, pwm);
  Serial.print("Analog giriş değeri = ");
  Serial.print(pot);
  Serial.print("\t Analog çıkış değeri = ");
  Serial.println(pwm);
}
```

İşlem Basamakları

1. Görsel 1.45'teki devreyi kurunuz. Programı yükleyiniz.
2. Potansiyometreyi baştan sona çevirerek LED'in durumunu ve seri port ekranını gözlemleyiniz.

SIRA SİZDE

1. analogWrite(3, pwm) komutunda **pwm** yerine pot yazınız. LED'in durumunu gözlemleyiniz.

Soru

1. Arduino'da PWM frekansı 980 Hz'den yüksek frekanslara ayarlanabilir mi? Araştırınız.

1.10. KESME İŞLEMLERİ

Kesme işlemi, mikrodenetleyicilerde önceden belirlenen görevleri ana programdan bağımsız şekilde kontrol ederek koşul sağlandığı anda programı durdurur ve belirlenen kesme fonksiyonun çalışmasını sağlar. Kesme fonksiyonundaki işlemler tamamlandığında program kaldığı yerden çalışmaya devam eder.

Arduino Uno kartında 2 adet dış kesme ve 3 adet zamanlayıcı kesmesi bulunmaktadır. Dış kesmeler Arduino UNO kartının 2 numaralı dijital pininde bulunan INT0 ve 3 numaralı dijital pininde bulunan INT1 kesmeleridir.

attachInterrupt() Fonksiyonu

Dış kesme kullanılırken kesme pini **dijital giriş** olarak ayarlanır. Kesmeyi başlatmak için setup() fonksiyonu içinde kesme fonksiyonu tanımlanır.

Kesme fonksiyonu tanımlama `attachInterrupt()` fonksiyonuyla yapılır. Haricî kesme kullanımındaki amaç ana programdaki işlemler devam ederken sensör vb. giriş bilgisini kaçırmamaktır.

Arduino UNO 'da kesme fonksiyonu için iki farklı söz dizimi mevcuttur.

```
attachInterrupt(kesmeNumarası, kesmeFonksiyonu, mod)
attachInterrupt(digitalPinToInterrupt(kesmePini), kesmeFonksiyonu, mod)
```

`kesmeNumarası:` Kesme numarası. Arduino Uno'da 2 numaralı pin için 0, 3 numaralı pin için 1 yazılır.

`kesmePini`: 2 girildiğinde digitalPinToInterrupt(kesmePini) fonksiyonu 0 döndürür, 3 girildiğinde 1 döndürür. Bu kullanımda kesme numarasıyla değil pin numarasıyla işlem yapılır.

`kesmeFonksiyonu:` Kesme oluştuğunda çağrılan fonksiyon. Parametre almaz. Değer döndürmez.

`mod:` Kesmenin tetiklenme modudur. Dört çeşittir.

LOW: Pindeki voltaj 0 V'sa kesme oluşur.

CHANGE: Pinde oluşacak voltaj değişimlerinde kesme gerçekleşir.

RISING: Yükselen kenarlarda kesme gerçekleşir. Pindeki voltaj değeri 0 V'tan 5 V'a çıkarken kesme gerçekleşir.

FALLING: Düşen kenarlarda kesme gerçekleşir. Pindeki voltaj değeri 5 V'tan 0 V'a düşerken kesme gerçekleşir.

Örnek: Ana programda blink uygulaması devam ederken butona basıldığında seri port ekranına "Butona basıldı." yazdırılmaktadır. Benzer işlem ana programda if kullanılarak da yapılabilir. Ancak butona basma anı if dışındaki komutlar icra edildiği zamana gelirse işlem gerçekleşmez veya gecikmeli olarak gerçekleşir.

```
const byte LED = 13;
const byte kesmePini = 2;
void setup() {
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
```

```
   pinMode(kesmePini, INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(kesmePini), kesmeFonksiyonu, FALLING);
}
void loop() { // Blink uygulaması
  digitalWrite(LED, HIGH);
  delay(1000);
  digitalWrite(LED, LOW);
  delay(1000);
}

void kesmeFonksiyonu() { // Kesme gerçekleştiğinde buradaki işlemler yapılır.
  Serial.println("Butona basıldı");
}
```

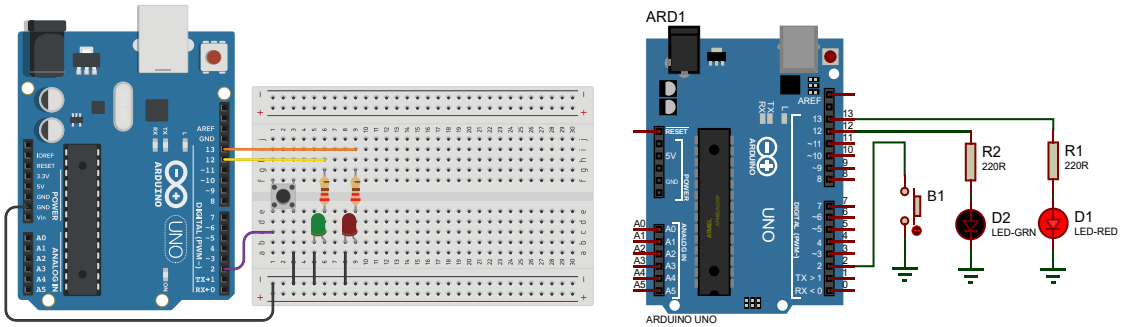
Örnek: Butona üç kez basıldığında kesme aktif edilmektedir.

```
const byte dahiliLED = 13;
const byte hariciLED = 12;
const byte kesmePini = 2; //Butona bağlı.
byte sayac;
void setup() {
  Serial.begin(9600);
  pinMode(dahiliLED, OUTPUT);
  pinMode(hariciLED, OUTPUT);
  pinMode(kesmePini, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(kesmePini), kesmeFonksiyonu, FALLING);
}

void loop() { //Blink uygulaması
  digitalWrite(dahiliLED, HIGH);
  delay(1000);
  digitalWrite(dahiliLED, LOW);
  delay(1000);

  digitalWrite (hariciLED, LOW);//Harici LED'i söndür.
  if (sayac > 3)
    detachInterrupt(digitalPinToInterrupt(kesmePini));
}

void kesmeFonksiyonu() { //Kesme gerçekleştiğinde buradaki işlemleri yap.
  digitalWrite (hariciLED, HIGH); //Harici LED'i yak.
  sayac++;
}
```

Uygulama Adı: Kesme Uygulaması — No.: 17

Amaç: **Kesme uygulaması yapmak.**

1.46'daki programda ana döngüde blink uygulaması çalışırken butona basıldığında kesme aktif edilmekte ve haricî LED anında yanmaktadır. Haricî LED'in sönme komutu ise ana döngüde verildiğinden sönme işlemi ana programda icra edilen komut satırına bağlı olarak 0 ile 2 saniye arasında süre alır.



Görsel 1.46: Kesme uygulaması şema ve devresi

Kesme uygulaması programı aşağıdaki gibidir:

```
const byte dahiliLED = 13;
const byte hariciLED = 12;
const byte kesmePini = 2; // Butona bağlı.
void setup() {
  Serial.begin(9600);
  pinMode(dahiliLED, OUTPUT);
  pinMode(hariciLED, OUTPUT);
  pinMode(kesmePini, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(kesmePini), kesmeFonksiyonu, FALLING);
}
void loop() { // Blink uygulaması
  digitalWrite(dahiliLED, HIGH);
  delay(1000);
  digitalWrite(dahiliLED, LOW);
  delay(1000);
  digitalWrite (hariciLED, LOW);// Haricî LED'i söndür.
}
```

İşlem Basamakları

1. Görsel 1.46'daki devreyi kurunuz. Programı yükleyiniz.

2. Blink uygulaması çalışırken butona basınız. LED'lerin çalışmasını gözlemleyiniz.

SIRA SİZDE

1. Haricî kesme tetiklendiğinde seri port ekranına "Butona basıldı." yazdırınız.

Soru

1. Haricî kesme tetiklendiğinde loop döngüsü içindeki komutlar niçin icra edilmemektedir? Açıklayınız.

2. Aşağıdaki programla yazdığınız programın farkı nedir?

```
const byte dahiliLED = 13;
const byte hariciLED = 12;
const byte kesmePini = 2; //Butona bağlı.
volatile bool a = 1;

void setup() {
  Serial.begin(9600);
  pinMode(dahiliLED, OUTPUT);
  pinMode(hariciLED, OUTPUT);
  pinMode(kesmePini, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(kesmePini), kesmeFonksiyonu, FALLING);
}

void loop() { //Blink uygulaması
  digitalWrite(dahiliLED, HIGH);
  delay(1000);
  digitalWrite(dahiliLED, LOW);
  delay(1000);
}

void kesmeFonksiyonu() { //Kesme gerçekleştiğinde buradaki işlemleri yap.
  delay(100);
  if (a)
    digitalWrite (hariciLED, HIGH); //Harici LED'i yak.
  else  digitalWrite (hariciLED, LOW);//Harici LED'i söndür.
  a = !a;
}
```

detachInterrupt() Fonksiyonu

Belirtilen kesmeyi sonlandırır.

`detachInterrupt(kesmeNumarası)`

`detachInterrupt(digitalPinToInterrupt(kesmePini))`

kesmeNumarası: Kesme numarası. Arduino Uno 'da 2 numaralı pin için 0, 3 numaralı pin için 1 yazılır.
kesmePini: Arduino Uno 'da 2 veya 3 numaralı pinlerdir.

Aşağıdaki programda ana döngüde blink uygulaması çalışırken butona basıldığında kesme aktif edilerek haricî LED yanmaktadır. Kesme üç kez çalıştıktan sonra sonlandırılmıştır.

```
const byte hariciLED = 12;
const byte kesmePini = 2; // Butona bağlı.
byte sayac;
void setup() {
  Serial.begin(9600);
  pinMode(dahiliLED, OUTPUT);
  pinMode(hariciLED, OUTPUT);
  pinMode(kesmePini, INPUT_PULLUP); // 2 numaralı giriş 0 olduğunda kesme aktif olur.
  attachInterrupt(digitalPinToInterrupt(kesmePini), kesmeFonksiyonu, FALLING);
}
void loop() { // Blink uygulaması
  digitalWrite(dahiliLED, HIGH);
  delay(1000);
  digitalWrite(dahiliLED, LOW);
  delay(1000);

  digitalWrite (hariciLED, LOW);// Haricî LED'i söndür.
  if (sayac > 3)
    detachInterrupt(digitalPinToInterrupt(kesmePini)); // 2 numaralı pindeki kesmeyi sonlandır.
}
void kesmeFonksiyonu() { // Kesme gerçekleştiğinde buradaki işlemleri yap.
  digitalWrite (hariciLED, HIGH); // Haricî LED'i yak.
  delay(1000); // Harici LED'i 1 saniye yak.
  sayac++; // Sayaç değişkenini 1 artır.
}
```

Zamanlayıcı Kesmeleri

Zaman kesmesi (timer interrupt), belirli zaman aralıklarında istenilen görevleri ana programdan bağımsız gerçekleştirmek için kullanılır. Blink, kara şimşek vb. uygulamalar zaman kesmesiyle çalıştırılırken ana döngüde farklı bir program çalışmaya devam eder.

Arduino'da bulunan 16 MHz'lik işlemci, sayaç kaydedicisini [counter register (kauntır recîstır)] her saat darbesinde (clock) arttırır. Frekans bölücü (prescalar) kullanılarak daha düşük saat frekansları elde edilir. Uno'da üç adet zamanlayıcı bulunur. delay() ve millis() fonksiyonlarının da kullanıldığı Timer0 8 bit (255), servo kütüphanesinin kullandığı Timer1 16 bit (65535), tone() fonksiyonunun kullandığı Timer2 8 **bit**tir. Zamanlayıcılar her saat darbesinde bir artarak belirli

sayılara ulaşınca taşma (overflow) gerçekleşir. Taşma değerine ulaştığında zamanlayıcı sıfırlanır ve kesme oluşur. Bu kesme ana programdaki işlemleri durdurarak ISR(TIMERx_OVF_vect) fonksiyonu içindeki komutların icra edilmesini sağlar. TIMERx ifadesinde x yerine 0, 1, 2'den biri yazılarak ilgili timer seçilir. Taşma modu haricinde karşılaştırma modu da mevcuttur.

Karşılaştırma zamanlayıcısında taşma oluşmasını (255 veya 65535 sayılarını) beklemeksizin istenen sayıya ulaşıldığında kesme işlemi gerçekleşir ve ISR(TIMERx_COMPA_vect) fonksiyonu içindeki komutlar çalışır. Karşılaştırma değeri

$$\text{Karşılaştırma değeri} = \left(\frac{\text{İşlemci frekansı}}{\text{Frekans} * \text{Ölçek}}\right) - 1$$

formülüyle hesaplanır. Elde etmek istenen frekans yerine 1/T yazılarak saniye cinsinden süre de yazılabilir. Zamanlayıcı kesmesini karşılaştırma modunda çalıştırmak için 8 **bit**lik **TCCR1A**, **TCCR1B**, **TIMSK1** ve 16 **bit**lik **OCR1A** kaydedicilerin ayarlanması gerekir.

Zamanlayıcı Kesmesini Karşılaştırma Modunda Ayarlama

TCCR1A Kaydedicisi

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
COM1A1	COM1A0	COM1B1	COM1AB0				

Karşılaştırma modunu aktif yapmak için TCCR1A kaydedicisi 0 yapılır (TCCR1A = 0).

TCCR1B Kaydedicisi

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
					CS12	CS11	CS10

TCCR1B kaydedicisinin ilk üç **bit**i ölçeği belirler. Tablo 2.9'da ölçeklendirme seçenekleri yer almaktadır.

Tablo 1.9: Clock Sinyalini Ölçeklendirme (Prescalar)

CS12	CS11	CS10	Açıklama
0	0	0	Zamanlayıcı/Sayıcı kapalı
0	0	1	Clock/1 (16 MHz)
0	1	0	Clock/8
0	1	1	Clock/64
1	0	0	Clock/256
1	0	1	Clock/1024
1	1	0	T1 pinden haricî clock sinyali. Düşen kenar tetikleme.
1	1	1	T1 pinden haricî clock sinyali. Yükselen kenar tetikleme.

Öncelikle TCCR1B kaydedicisi sıfırlanır. (TCCR1B = 0) Örneğin 256 ölçek için TCCR1B |= B00000100 binary sayısıyla **bit**sel veya (|) işlemi yapılarak kurulum yapılır. (Aynı kurulum TCCR1B |= (1 << CS12) şeklinde de yapılabilir. **Bit** numarasını içinde barındıran CS12 sabiti (içeriği 2) lojik 1'i iki **bit** sola kaydırarak **bit**sel veya (|) işlemine tabi tutulur.)

TIMSK1 Kaydedicisi

BIT	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
						OCIE1B	OCIE1A	TOIE1

Taşma zamanlayıcısı kesmesini aktifleştirmek için TOIE1 **bit**i (0. Bit), karşılaştırma zamanlayıcısı kesmelerini aktifleştirmek için OCIE1A ve OCIE1B **bit**leri 1 yapılır.

Son olarak 16 **bit**lik OCR1A kaydedicisine karşılaştırma değeri yüklenir (OCR1A = 62499).

Setup fonksiyonunda zamanlayıcı kesmesinin ayarları yapılmadan önce cli() fonksiyonuyla kesmeler durdurulur. Ayarlar yapıldıktan sonra sei() fonksiyonuyla kesmeler aktif edilir.

Örnek: Blink uygulamasının zamanlayıcı kesmesiyle karşılaştırma modunda çalıştırılır.

```
void setup() {
  pinMode(13, OUTPUT);
pinMode(12, OUTPUT);
  cli();                      // Ayarlar yapılana kadar kesmeleri durdur.
  TCCR1A = 0;                 // Karşılaştırıcı çıkış modunu aktif yap.
  TCCR1B = 0;                 // TCCR1B'yi sıfırla.
  TCCR1B |= B00000100;        // CS12'yi 1 yap. Ölçek 256 ayarlandı.
  TIMSK1 |= B00000010;        // OCIE1A'yı 1 yap.

  // 1 Hz 256 ölçek ile:  Karşılaştırma değeri = [16000000 / (256 * 1)] - 1 = 62499
  OCR1A = 62499;              // Karşılatırma kaydedicisini 62499'e ayarla.
  sei();                      // Kesmeleri aktif yap.
}

void loop() {// Zamanlayıcı kesmesi blink uygulmasını çalıştırırken buraya yazılacak kodlar da
çalışır.


digitalWrite(12, !digitalRead(12)); // 12 No.lu pin çıkışı değiştir.
}

// Karşılaştırıcı kesme fonksiyonu saniyede bir kez tetiklenecek şekilde setup fonksiyonu içinde
ayarlandı.
ISR(TIMER1_COMPA_vect) {
  digitalWrite(13, !digitalRead(13)); // 13 No.lu pin çıkışı değiştir.
}
```

1.11. EEPROM İŞLEMLERİ

Arduino kartlar üzerindeki mikrodenetleyicide EEPROM bulunur. EEPROM kartın enerjisi kesildiğinde (küçük bir sabit sürücü gibi) değerleri tutulan bellektir. EEPROM.h kütüphanesi, **byte**ları EEPROM hafızaya yazmayı ve buradan okumayı sağlar. Farklı kartlar farklı EEPROM hafıza alanına sahiptir. Uno'da EEPROM hafıza 1 kB'tır. EEPROM belleğinin belirli bir (100,000 kez) yazma ömrü vardır.

1.11.1. EEPROM Yazma İşlemleri

EEPROM.write() Fonksiyonu

EEPROM'da belirtilen adrese 1 **byte** veri yazar. Değer döndürmez.

`EEPROM.write(adres, deger)`

`adres:` Verinin yazılacağı adres. Veri tipi int (1 kB hafıza için 0-1023 arası adres.).

`deger:` 1 **byte**lık veri. Veri tipi **byte** (0-255 arası değer.).

Örnek: Program A0 analog girişinden (pot, sensör vb.) aldığı bilgileri tüm hafızaya sırayla yazmaktadır.

```
#include <EEPROM.h>
byte deger;
void setup() {
  int hafizaBoyutu = EEPROM.length(); // Kartın hafıza boyutunu al.
  for (int adres = 0; adres < hafizaBoyutu; adres++) {
    int deger = analogRead(A0) / 4; // 0 -1023 arası gelen veriyi 0 - 255 arası değer olarak
ata.
    EEPROM.write(adres, deger); // Analog veriyi tüm hafıza sırayla kaydet.
    delay(10); // 10 milisaniye bekle.1 kB hafiza yaklaşık 20 saniyede dolar.
  }
}
void loop() {
  /*100.000 yazma/okuma ömrünü daha uzun süre kullanabilmek için
    program loop döngüsüne yazılmamıştır.*/
}
// Yazılan değerleri görmek için EEPROM.read örneğini yükleyin.
```

1.11.2. EEPROM Okuma İşlemleri

EEPROM.read() Fonksiyonu

EEPROM'dan bir **byte** okur. Girilen adresteki **byte**ı döndürür.

`EEPROM.read(adres)`

`adres:` Verinin okunacağı adres. Veri tipi int.

Örnek: Program tüm EEPROM hafızasını sırayla seri port ekranına yazmaktadır.

```
#include <EEPROM.h>
byte deger;
void setup() {
  Serial.begin(9600);
  int hafizaBoyutu = EEPROM.length(); // Kartın hafıza boyutunu al.
  for (int adres = 0; adres < hafizaBoyutu; adres++) {
    deger = EEPROM.read(adres); // Analog veriyi 10 ms aralıklarla tüm hafızadan sırayla oku.
    Serial.print("Adres = ");
```

EEPROM.read() Fonksiyonu

EEPROM'dan bir **byte** okur. Girilen adresteki **byte**ı döndürür.

`EEPROM.read(adres)`

adres: Verinin okunacağı adres. Veri tipi int.

Örnek: Program tüm EEPROM hafızasını sırayla seri port ekranına yazmaktadır.

```
#include <EEPROM.h>
byte deger;
void setup() {
  Serial.begin(9600);
  int hafizaBoyutu = EEPROM.length(); // Kartın hafıza boyutunu al.
  for (int adres = 0; adres < hafizaBoyutu; adres++) {
    deger = EEPROM.read(adres); // Analog veriyi 10 ms aralıklarla tüm hafızadan sı-
rayla oku.
    Serial.print("Adres = ");
    Serial.print(adres);
    Serial.print("\t Değer = ");
    Serial.println(deger);
    delay(10);
  }
}
void loop() {}
```

NOT EEPROM[] ifadesi bir dizi gibi EEPROM hücrelerine ulaşmayı sağlar.

```
x = EEPROM[0]; // İlk EEPROM hücresini oku.
 EEPROM[0] = x; // İlk EEPROM hücresine yaz.
```

EEPROM.read() Fonksiyonu

EEPROM.read() ve EEPROM.write() fonksiyonlarının birleşiminden oluşmuştur. Adresteki değeri önce okur, yeni yazılacak değer kaydedilmiş olandan farklıysa yazma işlemi gerçekleştirilir. Böylelikle yazma ömründen tasarruf edilir.

`EEPROM.update(adres, deger)`

adres: Verinin güncelleneceği adres. Veri tipi int.

deger: 1 **byte**lık veri. Veri tipi **byte**.

Örnek: EEPROM.update kullanımını gösterir.

```
#include <EEPROM.h>
void setup(){
  for (int i = 0; i < 255; i++) {
    EEPROM.update(i, i);
  }
  for (int i = 0; i < 255; i++) {
    // 3 numaralı adrese 12 bilgisi bir kez yazılır. Diğer 254 tekrarda işlem yapılmaz.
```

```
    EEPROM.update(3, 12);
  }
}
```

EEPROM.put() Fonksiyonu

EEPROM'a herhangi bir veri tipini yazmak için kullanılır.

`EEPROM.put(adres, deger)`

adres: Verinin yazılacağı adres. Veri tipi int.

deger: Herhangi bir veri tipindeki değer.

Örnek: EEPROM hafızaya veri kaydeder.

```
#include <EEPROM.h>
int adres = 0;
struct urun {
  String ad;
  float agirlik;
  byte uzunluk;
};
void setup() {
  Serial.begin(9600);
  urun urun1 = { // Kaydedilecek veriler.
    "Ürün1",
    57.29,
    168
  };
  EEPROM.put(adres, urun1);
  Serial.print("Veriler sonraki örnekte EEPROM.get() fonksiyonuyla okunacaktır.");
}
void loop() {}
```

EEPROM.get() Fonksiyonu

EEPROM'dan herhangi bir veri tipini okumak için kullanılır.

`EEPROM.get(adres, ozelNesne)`

adres: Verinin okunacağı adres. Veri tipi int.

ozelNesne: EEPROM'dan okunan farklı tiplerdeki verilerin atanacağı, önceden oluşturulmuş nesne.

Örnek: EEPROM hafızadan veri okur.

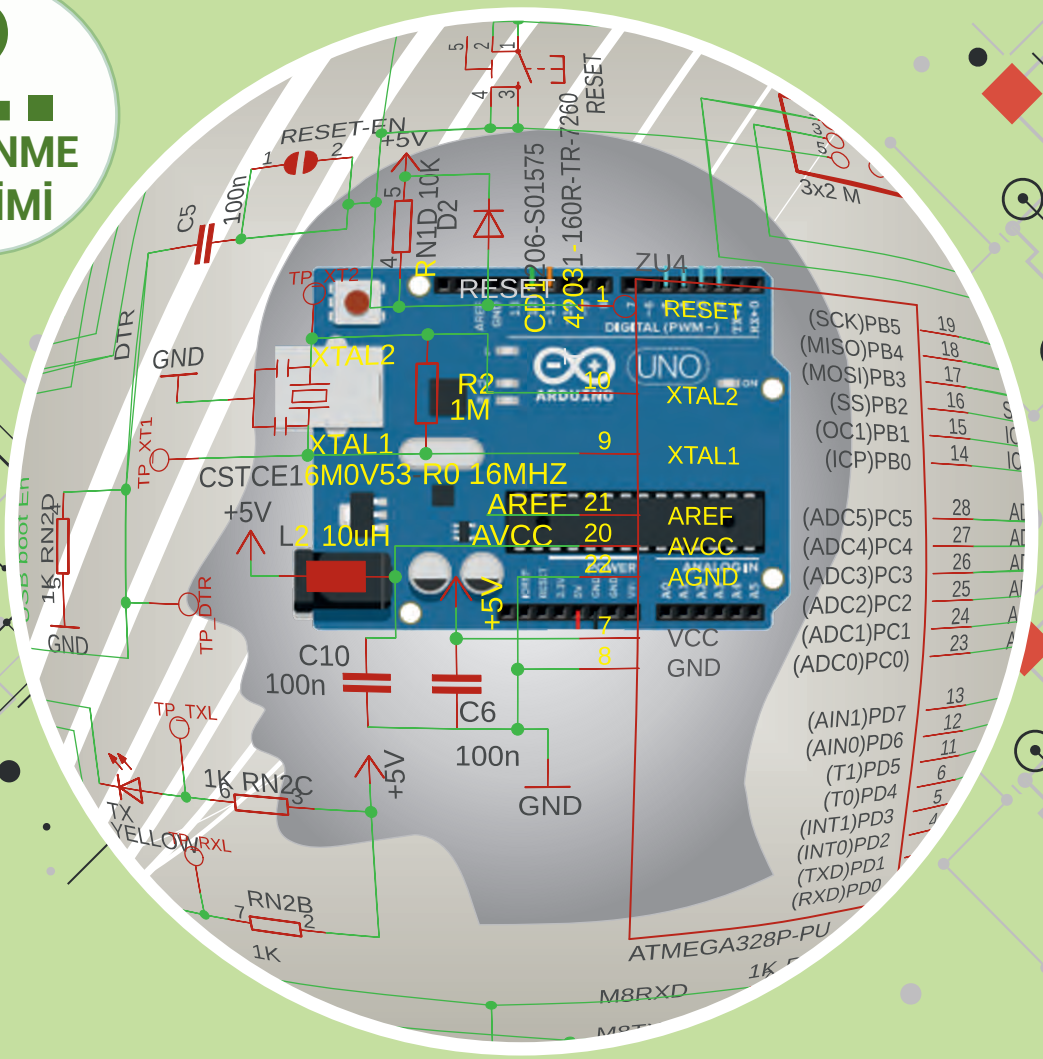
```
#include <EEPROM.h>
int adres = 0;
struct urun { // Ürün isimli nesne oluşturuldu.
  String ad;
  float agirlik;
  byte uzunluk;
```

```
};
void setup() {
  Serial.begin(9600);
  urun urun1; // ÜrÜn nesnesiyle ürün1 isimli değişken oluşturuldu.
  EEPROM.get(adres, urun1); // Ürün1'e ait bilgileri hafızadan al.
  Serial.println(urun1.ad);
  Serial.println(urun1.agirlik);
  Serial.println(urun1.uzunluk);
}
void loop() {}
```

Örnek: Tüm EEPROM hafızayı okur veya siler.

```
#include <EEPROM.h>
void setup() {
  Serial.begin(9600);
  eepromOku();
  //eepromReset();
}
void eepromOku() {
  int hafizaBoyutu = EEPROM.length(); // Kartın hafıza boyutunu al.
  for (int adres = 0; adres < hafizaBoyutu; adres++) {
    byte deger = EEPROM.read(adres);

    Serial.print(adres);
    Serial.print(" ");
    Serial.println(deger);
    delay(1);
  }
}
void eepromReset() {
  Serial.println("Hafıza sıfırlanıyor. Lütfen bekleyiniz...");
  int hafizaBoyutu = EEPROM.length(); // Kartın hafıza boyutunu al.
  for (int adres = 0; adres < hafizaBoyutu; adres++) {
    EEPROM.write(adres, 0); // Tüm hafızayı sıfırla.
  }
  Serial.println("Tüm hafıza silindi.");
}
void loop() {
}
```

2.
ÖĞRENME BİRİMİ



MİKRODENETLEYİCİ UYGULAMALARI

NELER ÖĞRENECEKSİNİZ ?

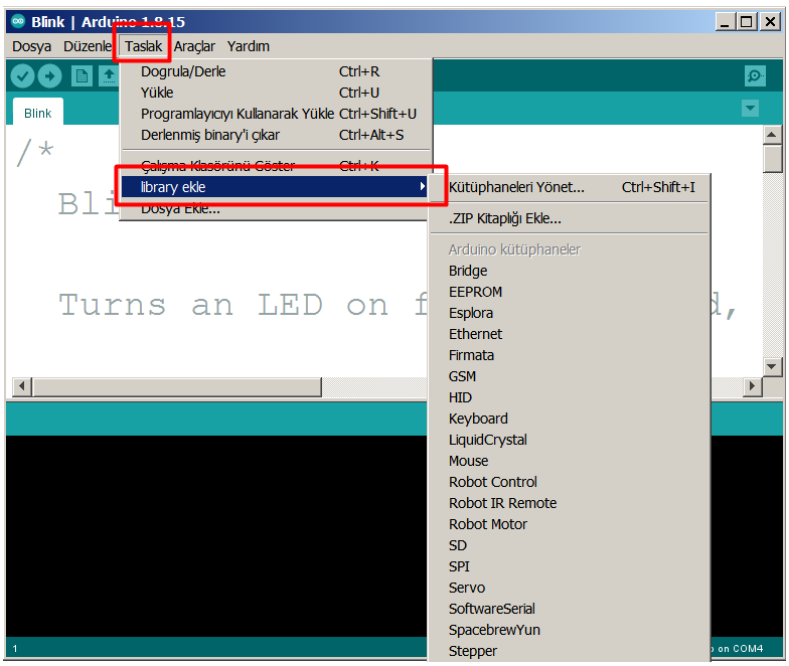
- Kütüphane Dosyası Yükleme
- Keypad Uygulamaları
- LCD Ekran Uygulamaları
- Sensör Uygulamaları
- Elektrik Motor Uygulamaları
- Haberleşme Uygulamaları

TEMEL KAVRAMLAR

kütüphane
nesnelerin interneti
radyo frekans
haberleşme
aplikasyon
sensör
motor
PWM
robot
PİD

2.1. KÜTÜPHANE DOSYASI YÜKLEME

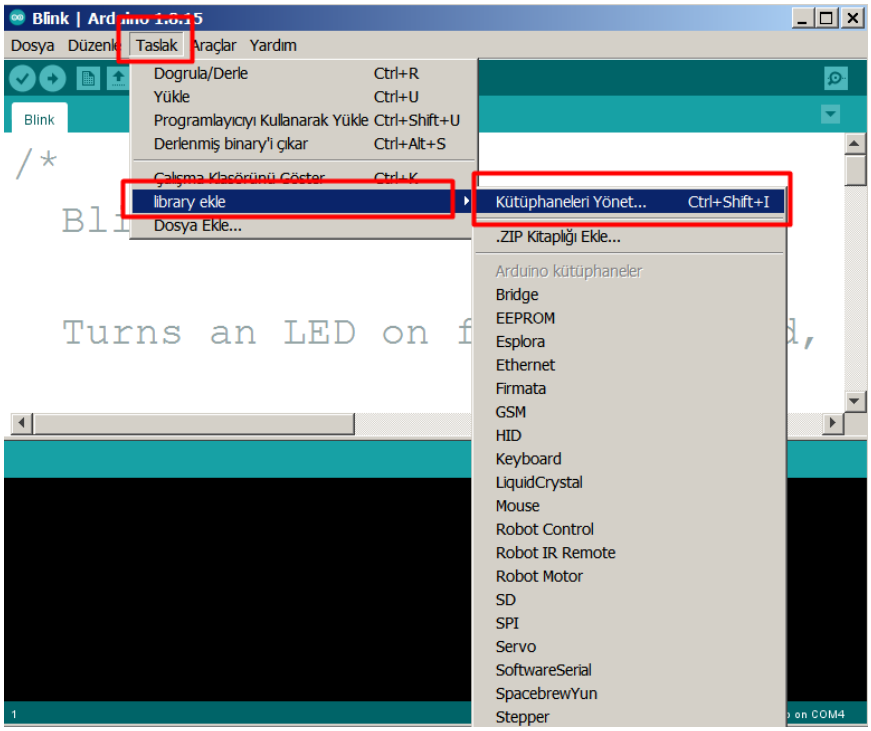
Arduino ortamı, çoğu programlama platformunda olduğu gibi kütüphaneler kullanılarak genişletilebilir. Kütüphaneler hazır fonksiyonlarıyla pek çok işlemi kolaylaştırır. Kütüphaneler fonksiyonların dosyalanmış şekli olarak da düşünülebilir. Örneğin dâhilî Liquid Crystal kütüphanesi LCD ekrana kolayca karakter yazmaya yardımcı olur. Bir taslağa kütüphane eklemek için Görsel 2.1'de görüldüğü gibi "Taslak" menüsünden "library ekle" komutu seçilir. Arduino IDE ile beraber bazı kütüphaneler gelmektedir. Bunun yanında kütüphane yöneticisiyle yeni kütüphaneler internetten indirilebilir veya bir kütüphane yazılabilir.



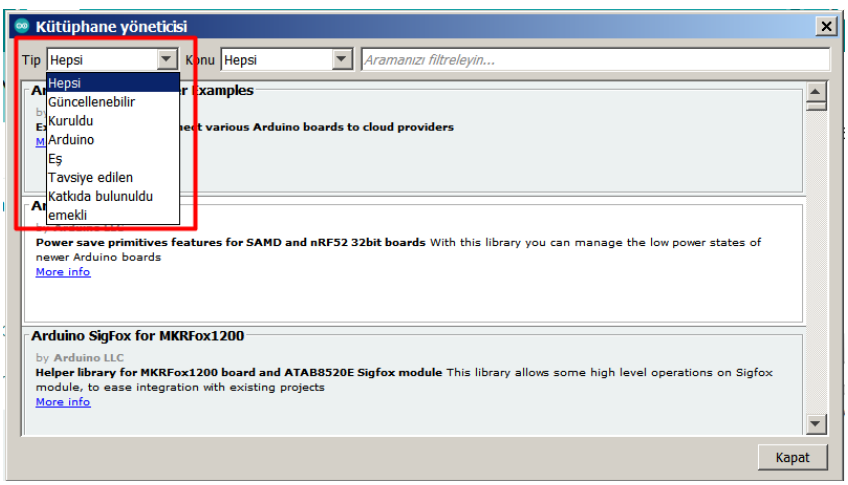
Görsel 2.1: Kütüphane ekleme

İlave Kütüphane Yükleme

Yeni bir kütüphane yüklemek için Görsel 2.2'de görüldüğü gibi "Taslak → Library Ekle →Kütüphaneleri Yönet" yolu seçilir. Kütüphane yöneticisi açıldığında yüklü kütüphaneler ile yüklenebilecek kütüphanelerin listesi görülür (Görsel 2.3). Yüklemek istenilen kütüphanenin üzerindeki "kur" düğmesine basıldığında kütüphane yüklenir. Artık üzerinde "yüklendi" ifadesi yer alır. "Taslak → Library Ekle" menüsünde yüklenen yeni kütüphane görülür.



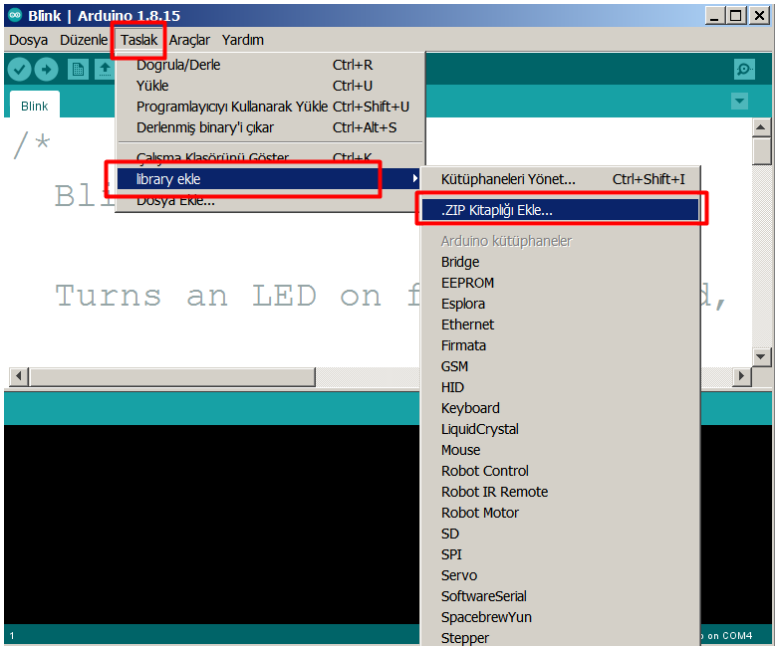
Görsel 2.2: Yeni kütüphane ekleme



Görsel 2.3: Kütüphane yöneticisi

Sıkıştırılmış Dosya (.zip) Kitaplığını İçe Aktarma

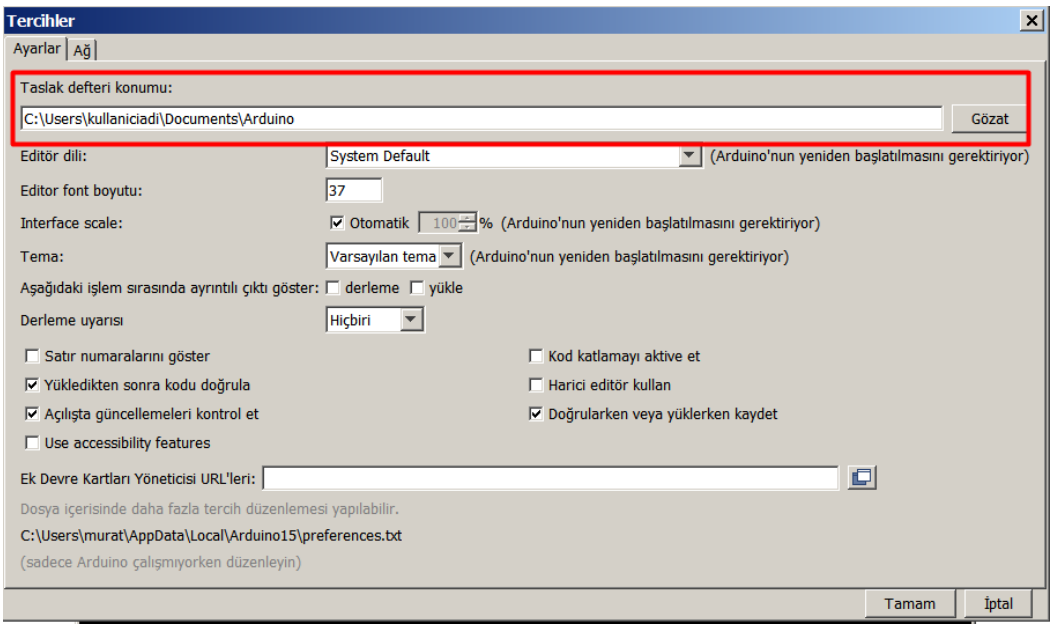
Kütüphaneler genellikle bir **.zip** dosyası veya klasörü olarak **github.com** vb. web sitelerinde dağıtılır. Klasörün adı genelde kütüphanenin adıdır. Klasörün içinde bir **.cpp** dosyası, bir **.h** dosyası ve genellikle bir **keywords.txt** dosyası, örnekler klasörü ve kütüphanenin gerektirdiği diğer dosyalar bulunur. Sıkıştırılmış dosyayı (.zip) içe aktarmak için Görsel 2.4'te "Taslak → Library Ekle → .ZIP Kitaplığı Ekle" seçilir. Dosya yüklendiğinde "Taslak → Library Ekle" menüsünde yüklenen yeni kütüphane görülür.



Görsel 2.4: ZIP kitaplığı ekleme

Kütüphane Dosyalarını Elle Yükleme

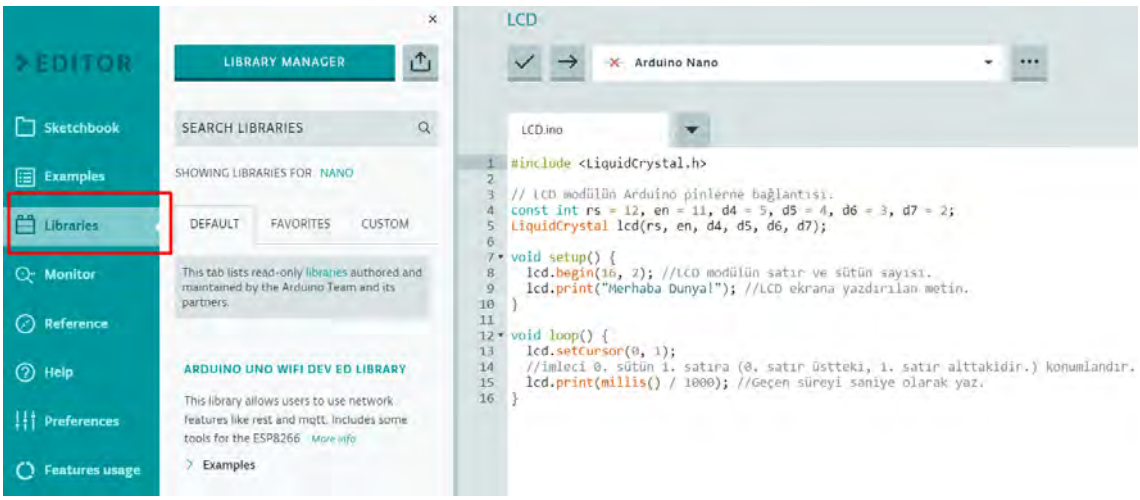
Taslak klasörünün konumuna "Dosya→ Tercihler → Taslak Defteri Konumu" bölümünden ulaşılır ve bu konum istenirse değiştirilebilir (Görsel 2.5). Daha sonra kütüphanenin **.zip** dosyasının indirildiği dizine gidilir. **.zip** dosyası tüm klasör yapısıyla birlikte geçici bir klasöre çıkarılır, ardından kütüphanenin adına sahip olması gereken ana klasör seçilerek "C:\Users\kullaniciadi\Documents\Arduino" klasörüne kopyalanır. Daha sonra "Taslak → Library Ekle" menüsünde yüklenen yeni kütüphane kontrol edilir.



Görsel 2.5: Taslak defteri konumu

Arduino Web Editörde Kütüphaneler

Taslak web editöre (https://create.arduino.cc/editor) yüklendiğinde veya web editöre de yazıldığında Arduino IDE kütüphane yöneticisinde yer alan tüm kütüphaneler kurulu olduğundan kütüphane kurulumu gerekmez. Ancak Arduino kütüphane yöneticisinde bulunmayan farklı bir yerden edinilmiş kütüphane dosyaları web editöre yüklenerek kullanılır (Görsel 2.6). Web editörde tüm kütüphanelerin son (güncel) versiyonu yer aldığından eski versiyon kütüphane komutlarıyla yazılmış taslaklar derleme sırasında hata verir.



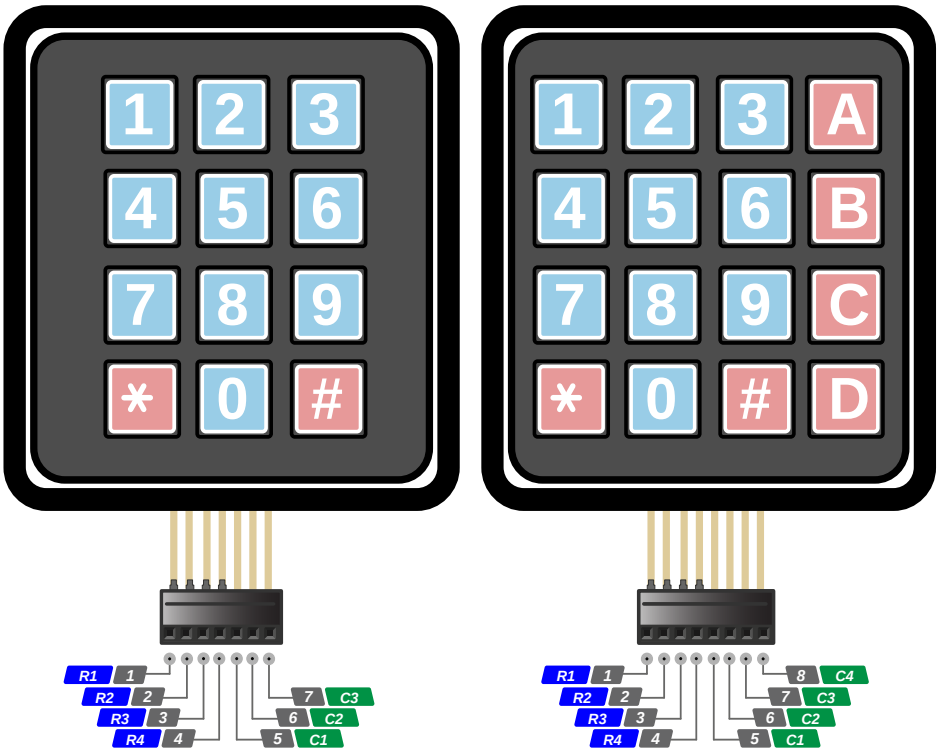
Görsel 2.6: Web editörde kütüphaneler

DİKKAT

Kitapta yer alan kütüphanelerin versiyonları kütüphane komutunun yanında yazmaktadır. Bazı kütüphanelerin güncel versiyonlarında fonksiyonların yapıları tamamen değiştiğinden derleyici hata verebilir. Kitapta yazılmış programların ilk satırlarında belirtilen kütüphane versiyonları kullanılmalı veya yeni versiyona göre taslak yeniden düzenlenmelidir.

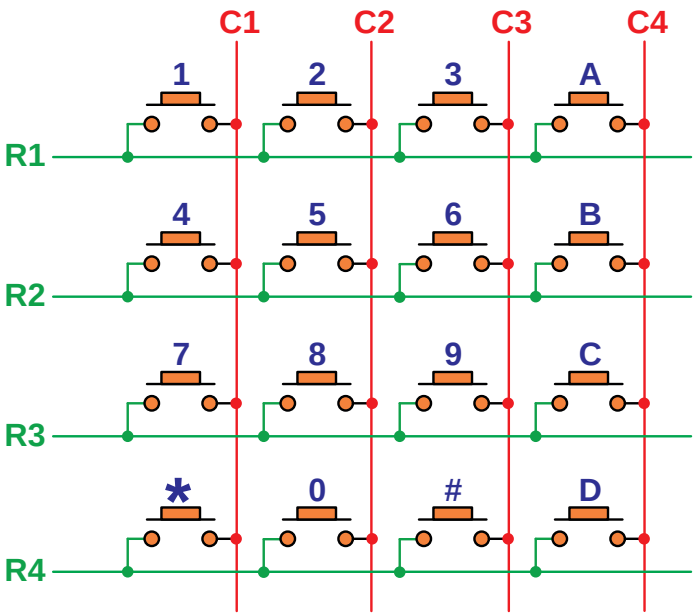
2.2. KEYPAD UYGULAMALARI

Tuş takımı, veri girişi gerektiren projelerde kullanılır. Tuş takımı bir buton grubudur. Satır ve sütunlardan oluşan buton gruplarının push buton (keyboard) veya membran (keypad) gibi farklı malzemelerden üretilmiş çeşitleri mevcuttur. Görsel 2.7'de 4x3 ve 4x4 membran tuş takımları girilmektedir.



Görsel 2.7: 4x3 ve 4x4 membran tuş takımı pinleri

Görsel 2.8'de 4x4 tuş takımının iç yapısı ve bağlantıları verilmiştir. Butonlardan birine basıldığında buton, bağlantılı olduğu satır ve sütun hattını birleştirir. Her buton için bir satır ve bir buton bilgisi aktif olur. Bir numaralı butona basıldığında R1: C1, iki numaralı butona basıldığında R1: C2 vb. şeklinde 16 farklı seçenek oluşur.



Görsel 2.8: 4x4 tuş takımının iç yapısı ve bağlantıları

Uygulama Adı: Keypad Uygulaması — No.: 1

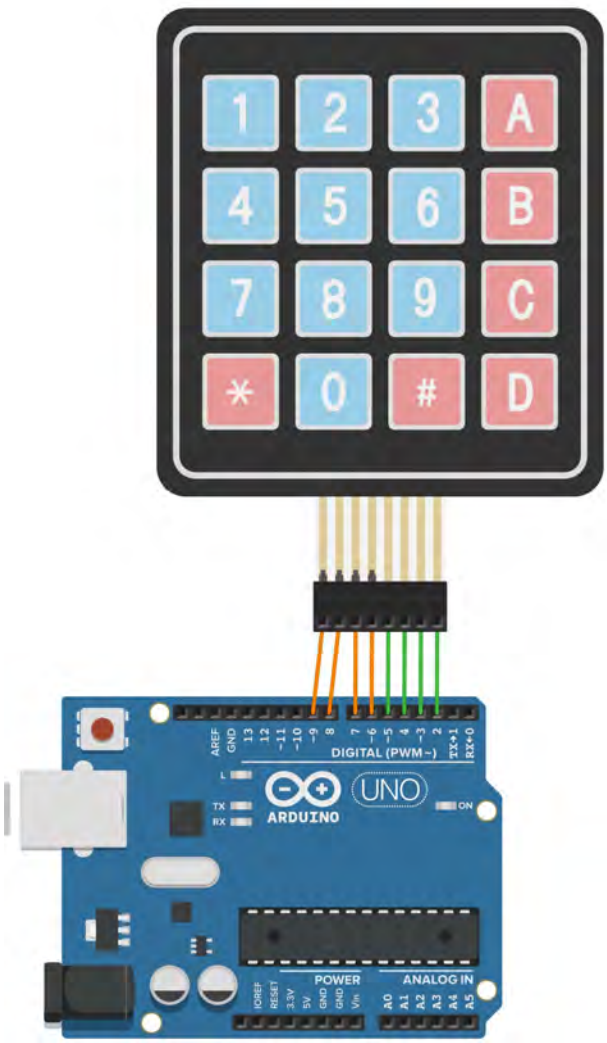
Amaç: Keypad uygulaması yapmak.

Görsel 2.9'daki devrede tuş takımı uygulaması verilmiştir. Keypad kütüphanesini kurmak için menülerden "Taslak → Library Ekle → Kütüphaneleri Yönet" seçeneğinden arama kutusuna "Keypad" yazılıp Keypad (Keypad by Mark Stanley and Alexander Brevig) kütüphanesi bulunarak kurulur. Daha sonra "Taslak → Library Ekle → Keypad" seçilerek programın başına eklenir. Bu programın bir örneğine menülerden "Dosya → Örnekler → Keypad → CustomKeypad" adımları izlenerek de ulaşılır.

Kullanılan tuş takımının satır ve sütun bilgileri, tuş dizilimi ve Arduino'ya bağlanan pinlerin bilgileri belirlenen değişkenlere atanır. Daha sonra kütüphane komutuyla tuş haritası oluşturulur.

```
Keypad tusTakimi = Keypad( makeKeymap(tusDizilimi), satirPinleri, sutunPinleri, satir, sutun);
```

Oluşturulan "`tusTakimi`" nesnesiyle kütüphane fonksiyonları çağrılır. `tusTakimi.getKey()` fonksiyonu, basılan tuşa ait karakteri döndürür.



Görsel 2.9: Tuş takımı uygulaması

Tuş takımı uygulaması programı aşağıdaki gibidir:

```
#include <Keypad.h> //Tuş takımı kütüphanesi. V3.1.1
const byte satir = 4; //Dört satır.
const byte sutun = 4; //Dört sütun.
char tusDizilimi[satir][sutun] = { //Tuş takımındaki sembollerin dizilimini tanımlama.
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte satirPinleri[satir] = {9, 8, 7, 6};
//{R1, R2, R3, R4} Satır pinlerinin Arduino pinlerine bağlantısı.
byte sutunPinleri[sutun] = {5, 4, 3, 2};
//{C1, C2, C3, C4} Sütun pinlerinin Arduino pinlerine bağlantısı.
//tusTakimi adında nesne oluşturma.
Keypad tusTakimi = Keypad( makeKeymap(tusDizilimi), satirPinleri, sutunPinleri, satir, sutun);
void setup() {
  Serial.begin(9600);
}
void loop() {
  char tus = tusTakimi.getKey(); //Basılan tuşu oku.
  if (tus) {
  Serial.println(tus); //Basılan tuşu seri ekrana yaz.
  }
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.9'daki devreyi kurunuz.
4. Taslaktaki tuş takımı dizilimini elinizdeki tuş takımına göre düzenleyiniz.
5. Programı yükleyiniz.
6. Basılan tuşu seri port ekranında gözlemleyiniz.
7. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Doğru şifre girildiğinde kapı kilidini açan uygulamayı gerçekleştiriniz.

```
#include <Keypad.h> //Tuş takımı kütüphanesi.
```

```
byte hane = 0, LED = 13; //Selenoid kapı kilidi bağlanabilir.
char sifre[] = "123ABC"; // Kullanılacak şifre.
char giris[sizeof(sifre)]; //Şifre uzunluğu kadar dizi oluştur.//---Tuş takımı ayarları.---
const byte satir = 4; //Dört satır.
const byte sutun = 4; //Dört sütün.
char tusDizilimi[satir][sutun] = {//Tuş takımındaki sebollerin dizilimini tanımlama.
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte satirPinleri[satir] = {9, 8, 7, 6};
//{R1, R2, R3, R4} Satır pinlerinin Arduino pinlerine bağlantısı.
byte sutunPinleri[sutun] = {5, 4, 3, 2};
//{C1, C2, C3, C4} Sütün pinlerinin Arduino pinlerine bağlantısı.

//tusTakimi adında nesne oluşturma.
Keypad tusTakimi = Keypad( makeKeymap(tusDizilimi), satirPinleri, sutunPinleri, satir, sutun);
void setup() {
  Serial.begin(9600);
  Serial.println("Lütfen şifrenizi giriniz: ");
}
void loop() {
  char tus = tusTakimi.waitForKey(); //Tuşa basılana kadar bekle.
  Serial.print(tus); //Basılan tuşu ekrana yazdır.
  giris[hane] = tus; //Sırayla her tuşu giris dizisine yükle.
  hane++;

  if (hane == sizeof(sifre) - 1) { //Şifre uzunluğu kadar tuşa basıldığında...
  if (!strcmp(giris, sifre)) { //İki diziyi karşılaştır.
   Serial.println("\nŞifre doğru. Kapı açıldı");
   digitalWrite(LED, HIGH);
   delay(1000);
   Serial.println("Lütfen şifrenizi giriniz: ");
  } else Serial.println("\nŞifre yanlış. Lütfen tekrar deneyiniz.");
  hane = 0;
  }
}
```

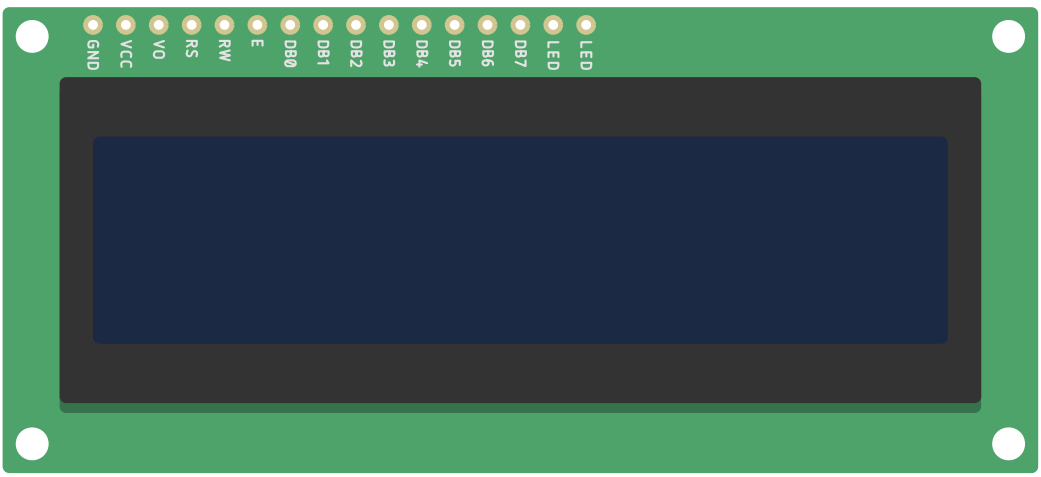
Sorular

1. Tuş takımında "1" rakamına basıldığında farklı bir karakter yazdırılabilir mi? Nedenini açıklayınız.
2. Kütüphane ile fonksiyon arasındaki fark nedir? Belirtiniz.

2.3. LCD EKRAN UYGULAMALARI

LCD [Liquid Crystal Display (likwıd kristıl displey) Sıvı Kristal Ekran], elektrikle kutuplanan sıvının ışığı tek fazlı geçirmesi ve önüne eklenen bir kutuplanma filtresiyle gözle görülebilmesi ilkesine dayanan bir görüntü teknolojisidir. Sıvı kristal ekranlar, düşük enerji tüketimine sahiptir. Görsel 2.10'daki LCD ekran, 2 satır ve 16 sütundan oluşmaktadır. Bir karakter ise 5x7 pikselden oluşmaktadır.

LCD modülde sekiz veri girişi bulunsa da dört girişli mod sıklıkla kullanılır. Dört girişli modda, veriler bir seferde yarım **bayt** olarak gönderilir. Bu nedenle dört veri bağlantısı yapılır. Veri girişinin üst yarısı (D4 - D7) kullanılırken diğer dört pin kullanılmamaktadır.



Görsel 2.10: LCD modül pinleri

1. GND: Topraklama pinidir. Bazı modüllerde VSS olarak etiketlenmiştir.

2. VCC: 5 V'luk güç bağlantısıdır. Bazı modüllerde VDD olarak etiketlenmiştir.

3. V0: Ekran parlaklığını kontrol etmek için 0 ile 5 V arasında değişen parlaklık kontrol voltajı girişidir.

4. RS: Register select pinidir. Giriş verilerinin LCD'de görüntülenip görüntülenemeyeceğini veya kontrol karakterleri olarak kullanılıp kullanılamayacağını kontrol eder.

5.RW: LCD'yi okuma veya yazma moduna geçirir. Genelde GND'ye bağlıdır. Okuma modundadır.

6. E: Etkinleştir pini. Lojik 1 olduğunda veri pinlerine uygulanan verileri okur. Lojik 0 olduğunda komutları yürütür veya verileri görüntüler.

7. D0: Veri girişi 0.

8. D1: Veri girişi 1.

9. D2: Veri girişi 2.

10. D3: Veri girişi 3.

11. D4: Veri girişi 4.

12. D5: Veri girişi 5.

13. D6: Veri girişi 6.

14. D7: Veri girişi 7.

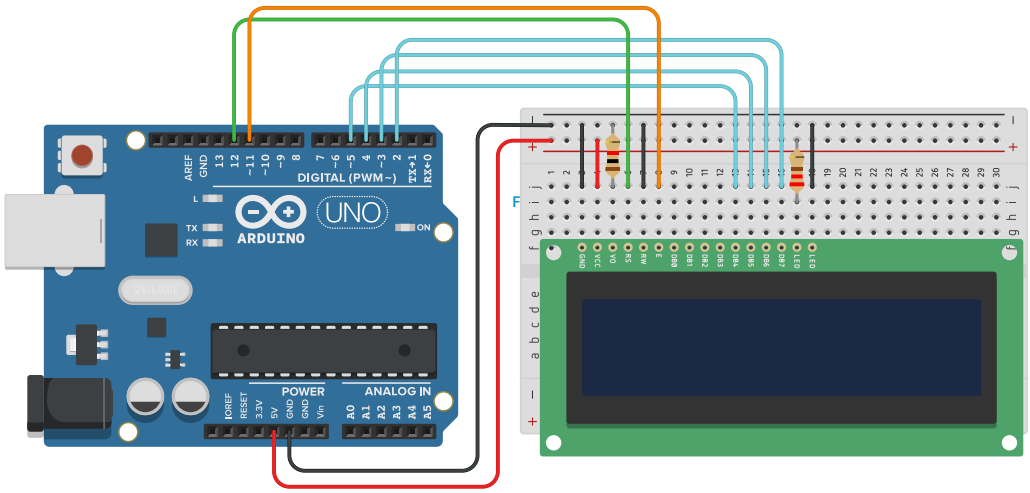
15. LED A: Arka ışık LED'ine anot (pozitif voltaj) bağlantısı.

16. LED K: Arka ışık LED'ine katot (GND) bağlantısı.

Uygulama Adı: LCD Uygulaması — No.: 2

Amaç: LCD uygulaması yapmak.

Görsel 2.11'deki LCD uygulamasında V0 ucuna bağlı dirençle ekran parlaklığı ayarlanmıştır. Bu pine potansiyometre bağlanarak da LCD ekranın parlaklığı ayarlanabilir. LCD kütüphanesi dâhilî kütüphanelerde mevcuttur. Menülerden "Taslak → Library Ekle → LiquidCrystal" seçilerek kütüphane taslakta kullanılır. Aşağıdaki program LCD ekrana "Merhaba dünya." yazdırmaktadır. Bu programın bir örneğine menülerden "Dosya → Örnekler → LiquidCrystal → HelloWorld" adımlarını izleyerek de ulaşılır.



Görsel 2.11: LCD uygulaması

LCD uygulaması programı aşağıdaki gibidir:

```
#include <LiquidCrystal.h> //v1.0.7
// LCD modülün Arduino pinlerine bağlantısı.
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
void setup() {
  lcd.begin(16, 2); //LCD modülün satır ve sütun sayısı.
  lcd.print("Merhaba Dunya!"); //LCD ekrana yazdırılan metin.
}
void loop() {
  lcd.setCursor(0, 1);
  //imleci 0. sütun 1. satıra (0. satır üstteki, 1. satır alttakidir.) konumlandır.
  lcd.print(millis() / 1000); //Geçen süreyi saniye olarak yaz.
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.11'deki devreyi kurunuz. Programı yükleyiniz.
4. Ekrana adınızı yazdırınız.

SIRA SİZDE

1. Ekrana kayan yazı yazdıran uygulamayı gerçekleştiriniz. Programa "Dosya → Örnekler → LiquidCrystal → Scroll" adımlarını izleyerek ulaşabilirsiniz.

```
#include <LiquidCrystal.h>
// LCD modülün Arduino pinleme bağlantısı.
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
void setup() {
  lcd.begin(16, 2); //LCD modülün satır ve sütun sayısı.
  lcd.print("Merhaba Dunya!"); //LCD ekrana yazdırılan metin.
  delay(1000);
}
void loop() {
  // 15 hücre sola kaydır. (Metin uzunluğu kadar.)
  for (int i = 0; i < 15; i++) {
  lcd.scrollDisplayLeft();
  delay(150);
  }
  // 31 hücre sağa kaydır. (Metin uzunluğu + ekran uzunluğu kadar.)
  for (int i = 0; i < 31; i++) {
  lcd.scrollDisplayRight();
  delay(150);
  }
  for (int i = 0; i < 16; i++) { // 16 hücre sola kaydır.(Ekran uzunluğu kadar.)
  lcd.scrollDisplayLeft();
  delay(150);
  }
  delay(1000);
}
```

2. Ekrana özel karakter yazdıran uygulamayı gerçekleştiriniz. Programa "Dosya → Örnekler → LiquidCrystal → CustomCharacter" adımlarını izleyerek ulaşabilirsiniz.

```
#include <LiquidCrystal.h>
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

/* Bir taslakta en fazla sekiz karakter oluşturulabilir. Bir karakter 5 x 8 pikseldir.*/
byte kalp[8] = {
  0b00000,
  0b01010,
  0b11111,
  0b11111,
  0b11111,
  0b01110,
```

```
  0b00100,
  0b00000
};
byte gulenYuz[8] = {
  0b00000,
  0b00000,
  0b01010,
  0b00000,
  0b00000,
  0b10001,
  0b01110,
  0b00000
};
byte kollarAsagi[8] = {
  0b00100,
  0b01010,
  0b00100,
  0b00100,
  0b01110,
  0b10101,
  0b00100,
  0b01010
};
byte kollarYukari[8] = {
  0b00100,
  0b01010,
  0b00100,
  0b10101,
  0b01110,
  0b00100,
  0b00100,
  0b01010
};
void setup() {
  lcd.begin(16, 2);
  lcd.createChar(0, kalp);
  lcd.createChar(1, gulenYuz);
  lcd.createChar(2, kollarAsagi);
  lcd.createChar(3, kollarYukari);
  lcd.setCursor(0, 0); // İmleci sol üste konumlandır.
  lcd.print("I ");
  lcd.write(byte(0)); // 0. karakter byte olarak çağırılır.
  lcd.print(" Arduino! ");
  lcd.write(1);
}
```

```
void loop() {
  int pot = analogRead(A0); // Pot bağlanabilir, boş da bırakılabilir.
  int sure = map(pot, 0, 1023, 200, 1000); // Pot'la animasyon hızını ayarla.
  lcd.setCursor(7, 1); // İmleci alt ortaya konumlandır.
  lcd.write(2); // Kollar aşağı.
  delay(sure);
  lcd.setCursor(7, 1);
  lcd.write(3); // Kollar yukarı.
  delay(sure);
}
```

3. Aşağıdaki kod nasıl çalışmaktadır?

```
#include <LiquidCrystal_I2C.h> // v1.1.2

LiquidCrystal_I2C lcd(0x3F, 16, 2); // I2C adresi 0x3F. 16x2 LCD ekran.

void setup() {
  lcd.init();
  lcd.clear();
  lcd.backlight();

  lcd.setCursor(2, 0);  // 0. satır 2. karakter.
  lcd.print("Merhaba Dünya!");

  lcd.setCursor(2, 1);  // 1. satır 2. karakter.
  lcd.print("I2C modül.");
}

void loop() {
}
```

Sorular

1. LCD ekrana I2C modülü kullanılarak nasıl yazı yazdırılır? Araştırınız.

2. Aşağıdaki kod nasıl çalışmaktadır?

```
#include <LiquidCrystal_I2C.h> // v1.1.2
LiquidCrystal_I2C lcd(0x3F, 16, 2); // I2C adresi 0x3F. 16x2 LCD ekran.
void setup() {
  lcd.init();
  lcd.clear();
  lcd.backlight();
  lcd.setCursor(2, 0);  // 0. satır 2. karakter.
  lcd.print("Merhaba Dünya!");
  lcd.setCursor(2, 1);  // 1. satır 2. karakter.
  lcd.print("I2C modül.");
}
void loop() {}
```

2.4. SENSÖR UYGULAMALARI

LCD sensörler; etraftaki ısı, ışık, nem, ses, basınç, kuvvet, elektrik, uzaklık vb. fiziksel büyüklüklerin değişimini algılar. Sensörlerden elde edilen veriler elektronik bir modülle elektrik sinyaline dönüştürüldükten sonra mikrodenetleyiciye giriş bilgisi olarak verilir. Aktif sensörler, GND ve VCC bağlantı uçları olan ve çıkışta analog veya dijital uçları bulunan modüler yapıdaki sensörlerdir. Pasif sensörler ise iki ucuna uygulanan potansiyel farkı sensörün algıladığı fiziksel büyüklüğün durumuna göre sensörün iki ucu arasında minimum ile maksimum arası akım geçirecek şekilde çalışır (Buton, anahtar, potansiyometre, NTC/PTC, basınç sensörleri, LDR, fototransistör, fotodiyot, mikrofon vb.). Tablo 2.1'de sensör çeşitleri verilmiştir.

Tablo 2.1: Algıladığı Fiziksel Büyüklüklere Göre Sensör ve Transdüser Çeşitleri

Fiziksel Değişken	Sensör, Transdüser
Sıcaklık	Termokupl (ısıl çift) Termistor (NTC, PTC) Termostat RTD [resistive temperature detectors (resistiv temperıtur dıtektırs)]
Ses	Dinamik mikrofon Kapasitif mikrofon Hoparlör Piezo kristal
Işık (Optik)	LDR Foto diyot Foto transistör Fotovoltaik pil (solar cell)
Basınç (Gerilme)	Gerilme ölçer [strain gauge (sitreyn geyç)] Basınç anahtarı (pressure switch) Yük hücresi (loadcells)
Manyetizma	Reed röle Hall Sensörü
Yaklaşım	Endüktif sensörler Kapasitif sensörler Optik sensörler

Uygulama Adı: Isı Sensörü Uygulaması No.: 3

Amaç: Isı sensörü uygulaması yapmak.

Görsel 2.12'de LM35 sıcaklık sensörü pin yapısı, Görsel 2.13'te ise uygulama devresi görülmektedir. LM35 sıcaklık sensörü 4-20 V'la beslenebilmektedir. Çıkış ucu ise 25 °C'de 250 mV gerilim vermektedir. Bu değer her 1 °C'de 10 mV değişim gösterir.

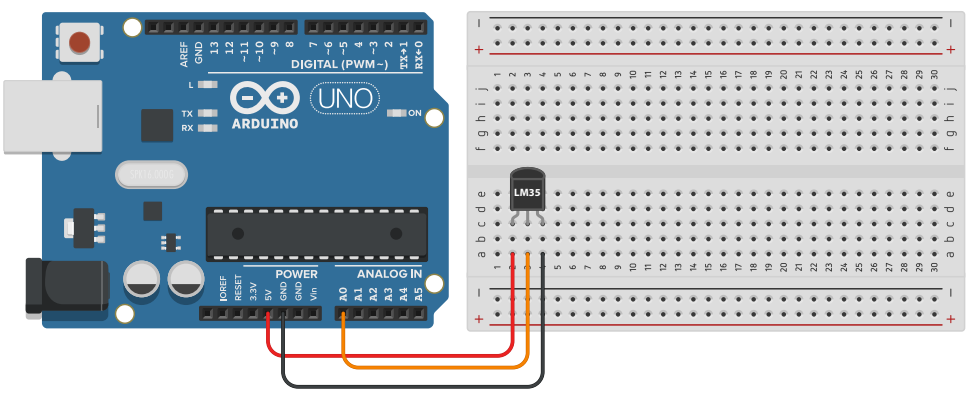
10 bitlik analog giriş 0-5 V için 0-1023 arası değer almaktadır. Analog girişteki gerilim (5/ 1023) *A0 işlemiyle volt cinsinden bulunur. Volt değeri 1000'le çarpıldığında mV değeri elde edilir. LM35'te her 10 mV'ta sıcaklık değeri 1 °C değiştiğinden A0 girişindeki mV değeri 10'a bölündüğünde sıcaklık değeri elde edilir. Tek formüle indirgenirse Sıcaklık = (5/1024) *A0 formülü elde edilir.

Görsel 2.12: LM35 pin yapısı

+V: 5 V besleme.

OUT: Analog veri çıkışı.

GND: Topraklama pini.



Görsel 2.13: LM35 uygulaması

Isı sensörü uygulaması programı aşağıdaki gibidir:

```
void setup() {
  Serial.begin(9600);
}
```

```
void loop() {
  int deger = analogRead(A0); // Analog girişi oku.
  float gerilim = (5000.0 / 1023.0) * deger; // Analog girişteki gerilim (mV).
  float  sicaklik = gerilim / 10.0; // Sıcaklık °C
  Serial.print(sicaklik);
  Serial.println(“ °C”);
  delay(1000);
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.13'teki devreyi kurunuz. Programı yükleyiniz.
4. LM35'i ısıtarak sıcaklık değişimini seri port ekranından gözlemleyiniz.
5. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Sıcaklığı LCD ekrana yazdıran uygulamayı gerçekleştiriniz.

```
#include <LiquidCrystal.h>
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  lcd.begin(16, 2); //LCD modülün satır ve sütün sayısı.
}
void loop() {
  lcd.setCursor(0, 0);
  lcd.print((500.0 / 1023.0) * A0); //Sıcaklık değerini yaz.
  lcd.setCursor(4, 0);
  lcd.print(“°C”);
}
```

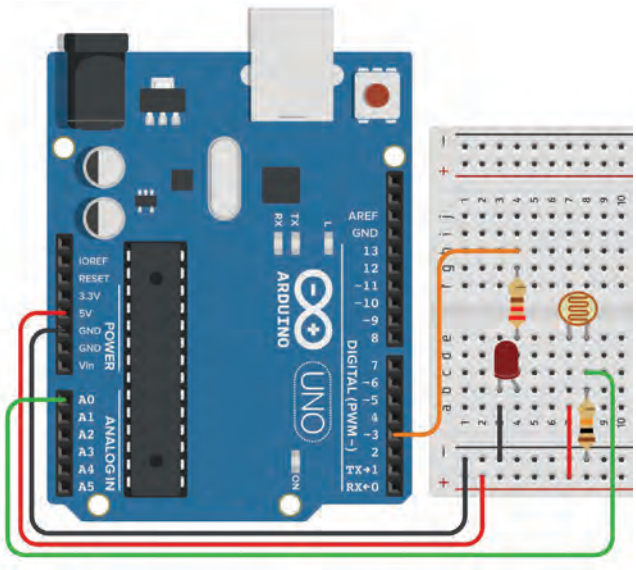
Soru

1. LM35 sıcaklık sensöründe sıcaklık - direnç ilişkisi nasıldır? Belirtiniz.

Uygulama Adı: Işık Sensörü Uygulaması — No.: 4

Amaç: Işık sensörü uygulaması yapmak.

Görsel 2.14'teki devrede Arduino'nun 5 V gerilimi üzerine uygulanan LDR'nin direnç-gerilim değişiminin anlaşılabilmesi için 10 kΩ'luk bir direnç seri bağlanarak gerilim bölücü oluşturulur. Bağlantı noktasının gerilim değişimi Arduino'nun analog girişine verilir. Aydınlık ortamda LDR'nin direnci 10 kΩ'un altındadır. Karanlıkta (LDR elle kapatıldığında) LDR direnci MΩ değerine doğru artar. LDR üzerine düşen gerilim artacağı için LDR ucuna bağlı A0 girişinden alınan bu bilgi 10 bitlik ADC'le 1023 değerine yaklaşır. Gelen bilginin ¼'ü 3 numaralı PWM çıkışından LED'e verilir. Böylelikte karanlık arttıkça LED'in parlaklığı da artar.



Görsel 2.14: LDR uygulaması

LDR uygulaması programı aşağıdaki gibidir:

```
const byte LED = 3;
void setup() {
  pinMode(LED, OUTPUT);
}
void loop() {
  // Analog giriş bilgisini 1/4'ü LDR değişkenine yükle.
  byte LDR = analogRead(A0) / 4;
  analogWrite(LED, LDR);
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.14'teki devreyi kurunuz. Programı yükleyiniz.

4. Elinizi LDR'ye yaklaştırıp uzaklaştırarak LED'in durumunu gözlemleyiniz.
5. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Akşam olduğunda lambayı yakan uygulamanın programını yazınız.

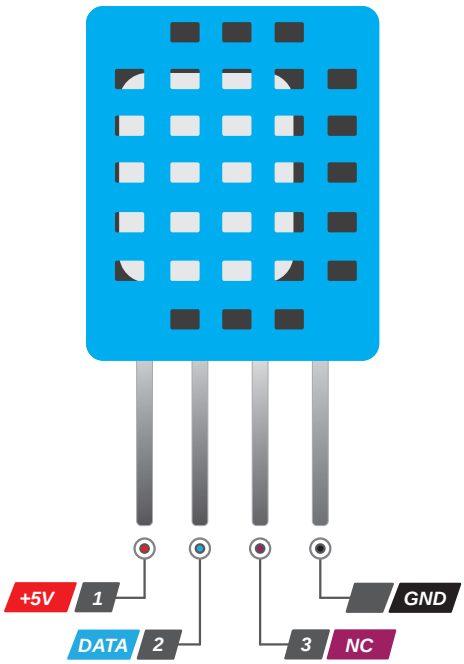
```
const byte lambaPini = 13;

void setup() {
  Serial.begin(9600); //Seri iletişim başlatıldı.
  pinMode(lambaPini, OUTPUT);
}

void loop() {
  int LDR = analogRead(A0); //A0 girişinden okunan 0-1023 arası veriyi LDR değişkenine yükle.
  Serial.println(LDR); //LDR değişkeninin içeriğini yazdır.
  if (LDR < 500) { // 500'den küçükse...
    digitalWrite(lambaPini, HIGH); // Lambayı yak.
  }
  else { // Lambayı söndür.
    digitalWrite(lambaPini, LOW);
  }
}
```

Soru

1. PWM çıkışına analogWrite(LED, 382); fonksiyonuyla 382 sayısı gönderilirse LED'e kaç volt verilmiş olur? Hesaplayınız.

Nem Sensörü Uygulaması

DHT sensörleri, kapasitif nem sensörü ve termistör (NTC) olmak üzere iki parçadan oluşur. İçinde analog dijital dönüştürü vardır. Görsel 3.15a'da DHT11 ve DHT22'nin pin yapısı görülmektedir. DHT11 ve DHT22 aynı pin çıkışına sahiptir ancak bunların farklı özellikleri vardır. DHT11 0 ile 50 °C arasındaki sıcaklığı ±2 °C doğrulukla verirken DHT22 -40 ile 80 °C arasındaki sıcaklığı ±0,5 °C doğrulukla verir. DHT11 saniyede bir örnekleme yaparken DHT22 saniyede iki örnekleme yapar.



Görsel 2.15a: DHT11 ve DHT22'nin pin yapısı

VCC: 5 V besleme.

Data: Dijital veri çıkışı.

NC: Kullanılmayan pin.

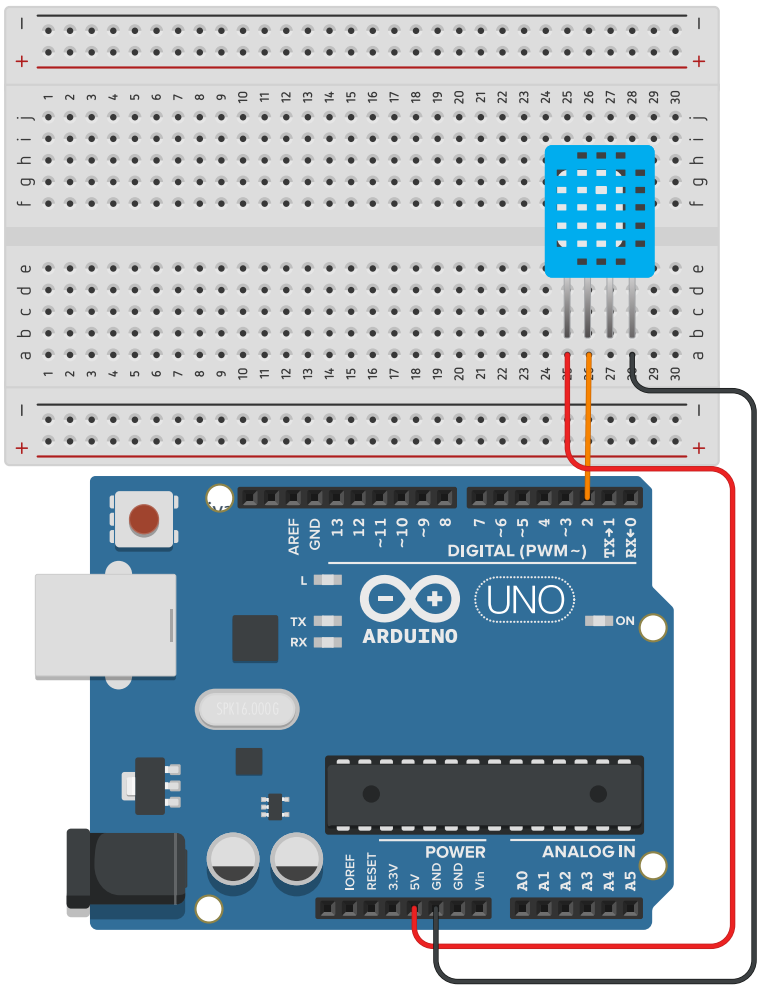
GND: Topraklama pini.

Menülerden "Taslak→Library Ekle → Kütüphaneleri Yönet" seçeneğinden arama kutusuna "DHT" yazılıp "Adafruit DHT Sensör Library" kütüphanesi bulunarak versiyon 1.2.3 kurulur (Kütüphanenin son versiyonunda aşağıdaki kod çalışmaz.). Daha sonra "Taslak → Library Ekle → DHT Sensör Library" seçilerek programın başına eklenir. Bu programın bir örneğine menülerden "Dosya →Örnekler →DHT Sensör Library→DHT Tester" adımlarını izleyerek de ulaşılır.

Uygulama Adı: Nem Sensörü Uygulaması — No.: 5

Amaç: Nem sensörü uygulaması yapmak.

Görsel 2.15b'deki devre uygulama ortamının sıcaklık ve nem bilgisini verir.



Görsel 2.15b: DHT11 ve DHT22 uygulaması

Nem sensörü uygulaması programı aşağıdaki gibidir:

```
#include “DHT.h” // Adafruit DHT Sensor Library v1.2.3
#define DHTpin 2     // Sensörün bağlandığı pin.
#define DHTtipi DHT11   // Sensör tipi: DHT11, DHT21, DHT22.

DHT dht(DHTpin, DHTtipi); //dht isimli nesne oluşturuldu.
void setup() {
  Serial.begin(9600);
  dht.begin();
```

```
}
void loop() {
  delay(1000);
 int nem = dht.readHumidity(); // Nem bilgisini oku.
  int sicaklik = dht.readTemperature(); // Sıcaklık bilgisini oku.

  if (!(sicaklik || nem)) { // Sıcaklık ve nem bilgisi yoksa...
  Serial.println("DHT sensor okunamadı!");
  return; // Başa dön.
  }

  Serial.print("Sıcaklık: ");
  Serial.print(sicaklik);
  Serial.print(" °C \t");
  Serial.print("Nem: %");
  Serial.println(nem);
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.15b'deki devreyi kurunuz. Programı yükleyiniz.
4. Ortamın nem ve sıcaklık bilgisini seri port ekranından gözlemleyiniz.
5. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Sıcaklık ve nem bilgisini LCD ekrana yazdıran uygulamayı gerçekleştiriniz.

Soru

1. Nem sensörünün içinde hangi tip termistör kullanılmaktadır? Belirtiniz.
2. DHT11 ve DHT22 arasındaki farklar nelerdir? Belirtiniz.

Hareket Sensörü

Maddeler sahip oldukları ısıdan dolayı insanların görebileceği ışık aralığının altında kızılötesi ışık yayar. Normal sıcaklığındaki insan vücudu (36,5 °C) 10 mikrometre dalga boyunda ışıma yapar. Hareket sensörü bulunduğu ortamdaki kızılötesi ışık dalgalarını içindeki özel kristal malzemeyle (piroelektrik sensör) elektriğe dönüştürür. Ortamdaki kızılötesi ışık miktarı sabitken bir canlı ortama girdiğinde kızılötesi ışık miktarında artış olur. Hareket sensörü bu artışı algılayarak çıkış verir. Hareket sensörünün diğer adı PIR (pasif infrared sensor-pasif kızılötesi) sensördür. PIR, kızılötesi ışık gönderip yansımasını almadığı için pasif moddadır yani sadece ortamdaki kızılötesi ışık değişimini algılar.

Görsel 2.16'da yaygın olarak kullanılan HC-SR501 hareket sensörünün pin yapısı ve ayarları görülmektedir. HC-SR501 sensörü temel olarak **piroelektrik sensör** ve kızılötesi sinyalleri piroelektrik sensöre odaklayan **fresnel lens** (sensörün dışındaki plastik kısım) olmak üzere iki ana bölümden oluşur.Kartta iki potansiyometre vardır. Biri hareketin algılanabileceği en fazla mesafeyi ayarlar. Bu mesafe 3 metre ile yaklaşık 7 metre arasında değişmektedir. Diğer potansiyometre ise hareket algılandıktan sonra çıktının ne kadar süreyle lojik 1'de kalacağını belirler. Bu süre de en az 3 saniye, en fazla 5 dakikadır.

Hareket sensörünün arkasında bulunan **jumper**la iki ayardan biri seçilir.

H: Bu konumda HC-SR501, hareketi algılamaya devam ettiği sürece lojik 1 sinyal vermeye devam eder.

L: Bu konumda çıkış, zaman potansiyometresi ayarıyla ayarlanan süre boyunca lojik 1'de kalacaktır.

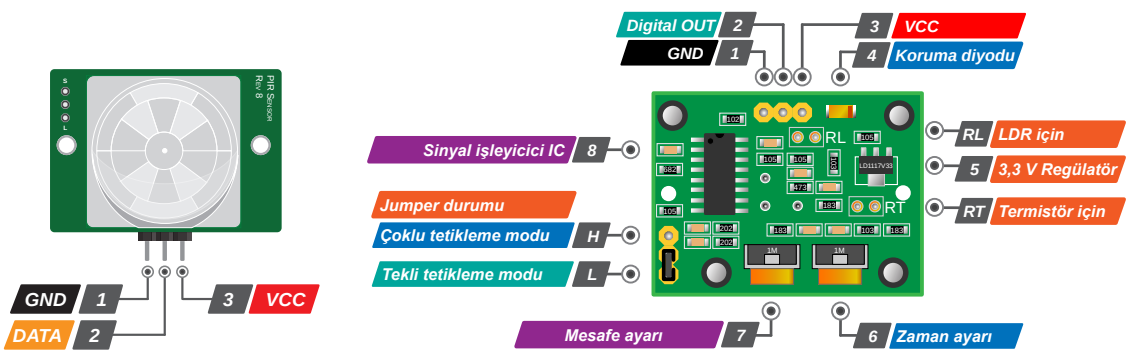
Hareket sensörünün çıkışındaki lojik 1 seviyesi 3,3 V'tur. 3,3 V gerilim, giriş olarak ayarlanan Arduino pininde lojik 1 seviyesi olarak algılanır çünkü Arduino'da dijital giriş olarak ayarlanmış bir pinde 3 V üzerindeki gerilimler lojik 1 olarak algılanır.

HC-SR501'e güç verildiğinde ortamdaki kızılötesi enerjiye alışması 30 ila 60 saniye sürer. Ayrıca sensörün bir okuma yaptıktan sonra yaklaşık 5 veya 6 saniye süren "sıfırlama" süresi vardır. Bu süre zarfında sensör herhangi bir hareket algılamayacaktır.

HC-SR501 devre kartında iki ek bileşen için lehim yuvaları bulunur.

RT: Sıcaklığa duyarlı direnç (termistör) içindir. Termistör eklemek, HC-SR501'in istenen sıcaklıklarda kullanılmasına izin verir ve dedektörün doğruluğunu bir dereceye kadar artırır.

RL: Işığa bağımlı direnç (LDR) içindir. LDR eklendiğinde HC-SR501 yalnızca karanlıkta çalışır.



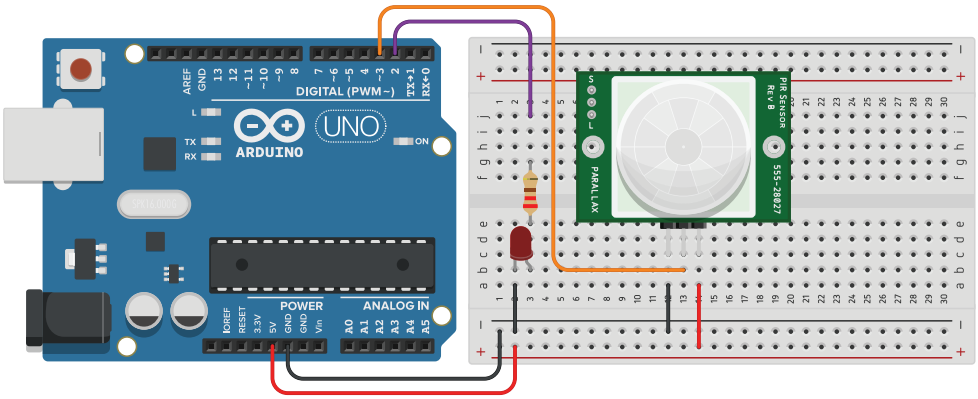
Görsel 2.16: Hareket sensörü pin yapısı ve ayarları

VCC: 5 V - 12 V besleme. **DATA:** Dijital veri çıkışı. (3,3 V) **GND:** Topraklama pini.

Uygulama Adı: Hareket Sensörü Uygulaması No.: 6

Amaç: Hareket sensörü uygulaması yapmak.

Görsel 2.17'deki devrede hareket sensörü, ortamda bir hareket algıladığında LED'i yakmakta ve seri port ekranından bilgi vermektedir.



Görsel 2.17: Hareket sensörü uygulaması

Hareket sensörü uygulaması programı aşağıdaki gibidir:

```
const byte LED = 2, PIR = 3;
void setup() {
  pinMode(LED, OUTPUT);
  pinMode(PIR, INPUT);
  Serial.begin(9600);
}
void loop() {
  if (digitalRead(PIR))  { // PIR çıkış veriyorsa...
  digitalWrite(LED, HIGH);
  Serial.println("Hareket algılandı.");
  }
  else  { // PIR çıkış vermiyorsa...
  digitalWrite(LED, LOW);
  Serial.println("Hareket algılanmadı.");
  }
```

İşlem Basamakları

1. Görsel 2.17'deki devreyi kurunuz. Programı yükleyiniz.
2. Hareketsiz bir ortamda HC-SR501 sensörünün önüne geçerek çalışmasını gözlemleyiniz.

SIRA SİZDE

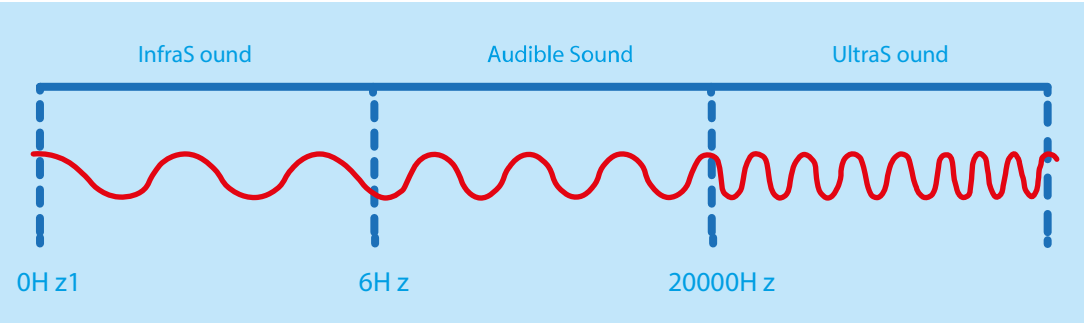
1. Uygulamayı hareket algılandığında sesle uyarı verecek şekilde düzenleyiniz.

Soru

1. Maddeler hangi sıcaklıkta kızılötesi ışınım yaymaz? Araştırınız.

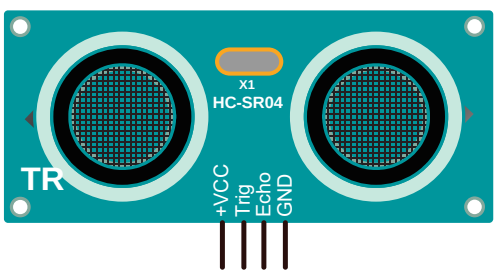
Ultrasonik Sensör

İnsanlar, frekansı 16 Hz ile 20 kHz arasında olan sesleri duyabilmektedir. Ultrasonik ses, insanın işitme sınırından daha yüksek frekanslardaki ses dalgalarıdır. Görsel 2.18'de ses frekans aralıkları görülmektedir.



Görsel 2.18: Ses frekans aralıkları

Görsel 2.19'da HC-SR04 ultrasonik sensör ve pin yapısı görülmektedir. HC-SR04 ultrasonik sensör üzerinde alıcı ve verici olmak üzere iki yüzey bulunur. Verici yüzeyden 15 derece açıyla ortama 40 kHz frekansta ultrasonik ses dalgası salınır. Alıcı, iletilen darbeleri dinler. Yayılan ses dalgası bu alanda bulunan bir cisme çarptığında ses dalgası cismin yüzeyinden sensöre geri yansır. Yansıyan dalganın giriş yüzeyine gelene kadar katettiği mesafenin belirlenebilmesi için genişliği kullanılabilen bir çıkış darbesi üretilir. Üretilen bu dalganın süresi ölçülerek hız formülü yardımıyla cismin uzaklığı hesaplanır. Sensör, 2 cm ile 400 cm arasındaki uzaklıkları 3 mm hassasiyetle ölçebilmektedir.



Görsel 2.19: HC-SR04 ultrasonik sensör ve pin yapısı

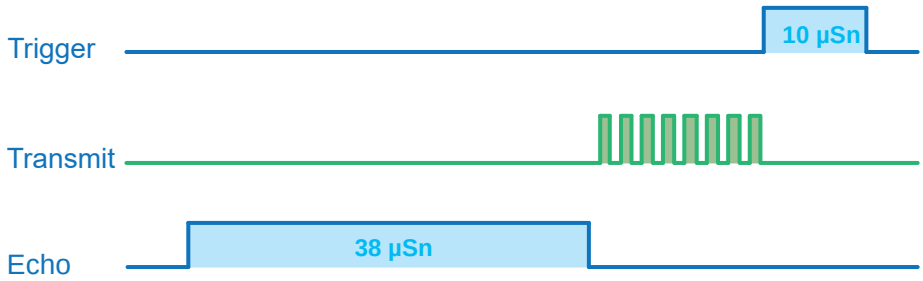
VCC: 5 V besleme.

Trig: Ultrasonik ses darbelerini tetiklemek için kullanılır.

Echo: Yansıyan sinyal alındığında darbe üreten pindir. Darbenin uzunluğu, iletilen sinyalin algılanması için geçen süreyle orantılıdır.

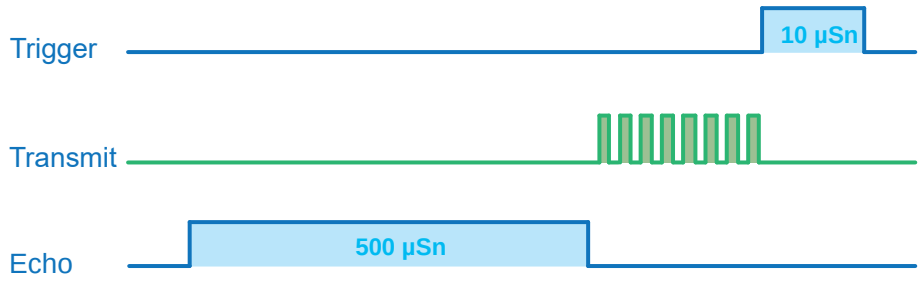
GND: Topraklama pini.

Trig pinine en az 10 µS (10 mikrosaniye) süreli bir darbe uygulandığında sensör, 40 kHz'de sekiz darbelik bir ses kalıbını ortama gönderir. Bu sekiz darbeli model, alıcının iletilen modeli ortamdaki ultrasonik gürültüden ayırt etmesini sağlar. Bu sinyal gönderildikten sonra **Echo** pini lojik 1 konumuna geçer ve gönderilen sinyalin dönmesini bekler. Eğer ses 38 milisaniye (sensörün menzil süresi) içinde geri dönmezse engel yoktur ve işlem tekrar eder (Görsel 2.20).



Görsel 2.20: Ultrasonik sensör engel algılamadığında

Darbeler geri yansıtılırsa sinyal alınır alınmaz Echo pini lojik 0 olur. Geçen süreye bağlı olarak **Echo** pini tarafından genişliği 150 µs ile 25 ms arasında değişen bir darbe üretilir (Görsel 2.21).



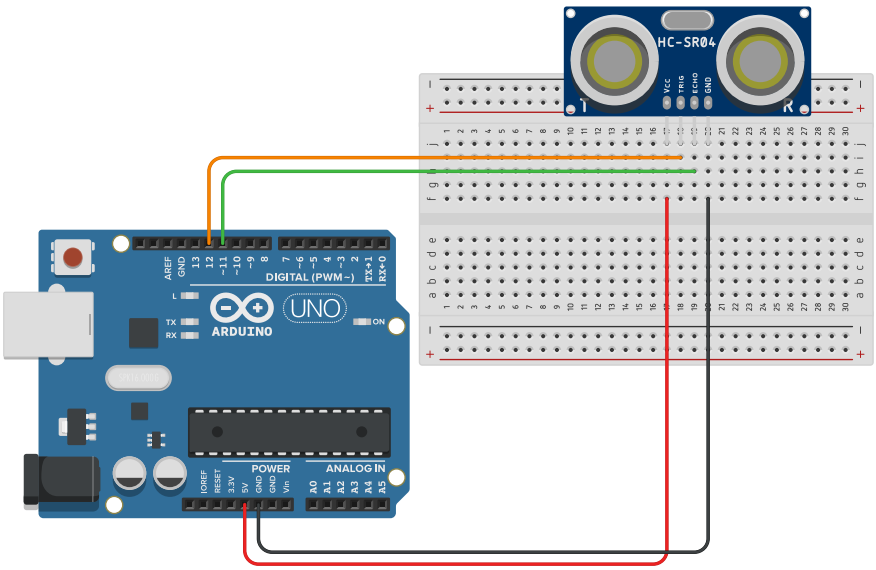
Görsel 2.21: Ultrasonik sensör engel algıladığında

Echo pinin lojik 1 seviyesinde kalma süresi ses dalgasının engele çarpıp geri dönme süresidir. **"Yol= hız x zaman"** formülünde bu süre kullanılarak mesafe hesaplanır. Sesin hızı ortamdaki nem ve sıcaklığa göre değişmektedir. 20 °C'de sesin yayılma hızı **344 m/s'dir**. DHT22 gibi bir sensör ilave edilerek ve ses hızı m/s = 331.4 + (0.606 * sıcaklık) + (0.0124 * nem) formülü programda kullanılarak ölçüm doğruluğu artırılabilir. Mesafe, süre, ortalama süre vb. değerleri veren NewPing.h kütüphanesi HC-SR04 uygulamalarında kullanılmaktadır.

Uygulama Adı: Ultrasonik Sensör Uygulaması No.: 7

Amaç: Ultrasonik sensör uygulaması yapmak.

Görsel 2.22'deki uygulamada kullanılan NewPing kütüphanesini kurmak için menülerden "Taslak → Library Ekle → Kütüphaneleri Yönet" seçeneğinden arama kutusuna "NewPing" yazılır. "NewPing" kütüphanesi bulunarak kurulur. Daha sonra "Taslak → Library Ekle → NewPing" seçilerek programın başına eklenir. Bu programın bir örneğine menülerden "Dosya → Örnekler → NewPing → NewPingExample" adımları izlenerek de ulaşılır. NewPing kütüphanesi, **trig** ve **echo** pinlerini aynı pin numarasında sırayla kullanarak birden fazla sensör kullanımında pin sayısından tasarruf eder.



Görsel 2.22: Ultrasonik sensör uygulaması

Ultrasonik sensör uygulaması programı aşağıdaki gibidir:

```
#include <NewPing.h> // v1.9.1
const byte trig = 12, echo = 11, sinir = 200;
NewPing olcum(trig, echo, sinir); // olcum isimli nesne oluşturuldu.
void setup() {
  Serial.begin(9600);
}
void loop() {
  int uzaklik = olcum.ping_cm(); // cm cinsinden uzaklık.
  Serial.print(uzaklik);
  Serial.println(" cm");
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.22'deki devreyi kurunuz. Programı yükleyiniz.

4. Ultrasonik sensörün önüne kitap, defter vb. engel koyunuz.
5. Engeli ileri geri hareket ettirip mesafeyi seri ekrandan gözlemleyiniz.
6. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Ultrasonik sensör uygulamasını kütüphane kullanmadan gerçekleştiriniz.

```
const byte trig = 12, echo = 11;
unsigned long sure;
int uzaklik;
void setup() {
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  Serial.begin(9600);
}
void loop() {
  digitalWrite(trig, 0); // Darbe sinyali gönder.
  delayMicroseconds(10);
  digitalWrite(trig, 1);
  delayMicroseconds(10);
  sure = pulseInLong(echo, 1); // Gelen sinyalin HIGH'da kalma süresini oku.
  uzaklik = (0.0343 * sure) / 2; // Mesafeyi hesapla.
  Serial.print(uzaklik);
  Serial.println(" cm");
```

2. 2 numaralı pine **buzzer** ekleyerek uygulamayı park sensörüne dönüştürünüz.

```
#include <NewPing.h>
const byte buzzer = 2, trig = 12, echo = 11, sinir = 200;
NewPing olcum(trig, echo, sinir); // olcum isimli nesne oluşturuldu.
void setup() {
  pinMode(buzzer, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  byte uzaklik = olcum.ping_cm(); // cm cinsinden uzaklık. (En fazla 200.)
  Serial.print(uzaklik);
  Serial.println(" cm");
  digitalWrite(buzzer, 1);
  delay(5 * uzaklik); // Mesafe azaldıkça bip sesini hızlandır.
  digitalWrite(buzzer, 0);
  delay(5 * uzaklik);
}
```

3. 3 adet ultrasonik sensörün mesafe bilgisini ekrana yazdırınız.

```
#include <NewPing.h>
byte sinir = 200;
NewPing sensor[3] = {   // Üç adet sensor nesnesi oluşturuldu.
  NewPing(2, 2, sinir), // Trig ve echo pinleri aynı pine bağlı.
  NewPing(3, 3, sinir),
  NewPing(4, 4, sinir)
};
void setup() {
  Serial.begin(115200); // Seri ekran  115200 baudrate'e ayarlandı.
}
void loop() {
  for (byte i = 0; i < 3; i++) { // Üç sensörün bilgileri göster.
  delay(50); // En az 29 ms beklenmeli.
  Serial.print(i);
  Serial.print("=");
  Serial.print(sensor[i].ping_cm());
  Serial.print("cm ");
  }
  Serial.println();
```

4. Mesafeyi LCD ekrana yazdıran uygulamayı gerçekleştiriniz.

```
#include <LiquidCrystal.h>
const byte trigEchoTekPin = 10; // Trig ve Echo pinleri 10 numaralı pine bağlı.
unsigned long sure;

// LCD modülün Arduino pinlerne bağlantısı.
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
void setup() {
  lcd.begin(16, 2); // LCD modülün satır ve sütun sayısı.
  lcd.print("Mesafe");
}
void loop() {
  pinMode(trigEchoTekPin, OUTPUT); // 10 numaralı pini çıkış yap.
  digitalWrite(trigEchoTekPin, 0); // Darbe sinyali gönder.
  delayMicroseconds(10);
  digitalWrite(trigEchoTekPin, 1);
  delayMicroseconds(10);
  pinMode(trigEchoTekPin, INPUT); // 10 numaralı pini giriş yap.

  sure = pulseInLong(trigEchoTekPin, 1); // Gelen sinyalin HIGH'da kalma süresini oku.
  lcd.setCursor(0, 1);
  lcd.print((0.0343 * sure) / 2); // Mesafeyi yazdır.
  lcd.setCursor(4, 0);
  lcd.print("cm");
}
```

5. Trig ve echo pinlerini tek bir pinde kullanınız.

```
const byte trig = 12, echo = 12;
unsigned long sure;
int uzaklik;

void setup() {
  Serial.begin(9600);
}

void loop() {
  pinMode(trig, OUTPUT); // 12 numaralı pini çıkış yap.
  digitalWrite(trig, 0); // Darbe sinyali gönder.
  delayMicroseconds(10);
  digitalWrite(trig, 1);
  delayMicroseconds(10);
  pinMode(echo, INPUT); // 12 numaralı pini giriş yap.

  sure = pulseInLong(echo, 1); // Gelen sinyalin HIGH'da kalma süresini oku.
  uzaklik = (0.0343 * sure) / 2; // Mesafeyi hesapla.

  Serial.print(uzaklik);
  Serial.println(" cm");
}
```

Sorular

1. Kütüphane kullanılarak yazılmış bir program, kütüphane kullanılmadan da yazılabilir mi? Açıklayınız.
2. Program yazılırken kütüphane kullanımı neden tercih edilir? Belirtiniz.
3. NewPing kütüphanesi engel algılamadığında (engel >400 cm) hangi değeri döndürmektedir? Belirtiniz.

2.5. ELEKTRİK MOTOR UYGULAMALARI

Elektrik enerjisini hareket enerjine çevirmeye yarayan mekanik aygıtlara motor denir. Elektrik motorları AC ve DC olarak iki gerilim türünde çalışacak şekilde üretilir. Diğer birçok motor modeli bu iki motor türünden geliştirilmiştir. AC motorlar genel olarak asenkron motor olarak iki türde üretilir. Bunlar bir fazlı ve üç fazlı motorlar olmak üzere iki türdedir. Bu motorların dönüş yönünü değiştirmek için farklı uygulamalar mevcuttur.

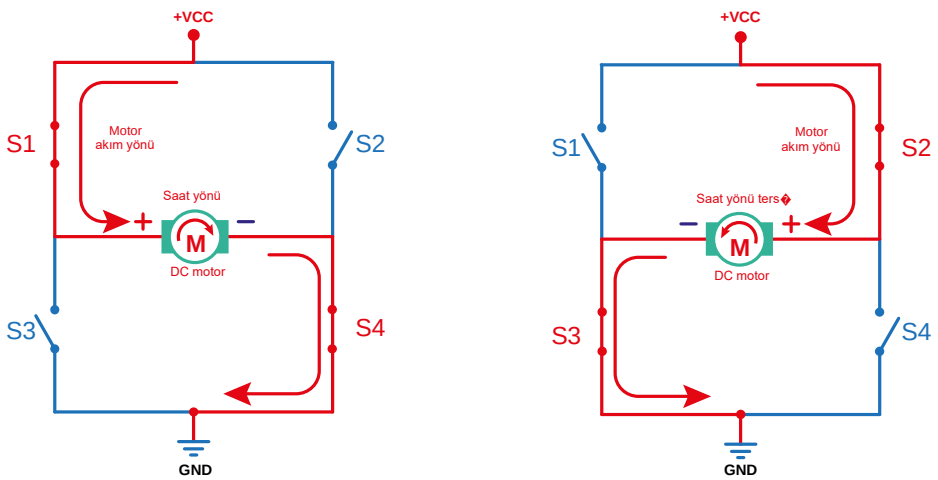
DC motorlar ise senkron motorlar, step motorlar, fırçalı ve fırçasız motorlar, sabit mıknatıslı motorlar olmak üzere üretilmektedir. Fırçalı DC motorlarda motor yönünü değiştirmek için besleme gerilim uçlarının yerini değiştirmek yeterlidir. Step motorlar, fırçasız motorlar ve sabit mıknatıslı motorlarda ise motor yönünü değiştirmek için farklı motor sürme teknikleri uygulanmaktadır.

Bu uygulamada fırçalı DC motor dönüş yönünü değiştirmek için dört transistörlü H tipi köprü devresi kullanılmaktadır.

DC Motor Uygulaması

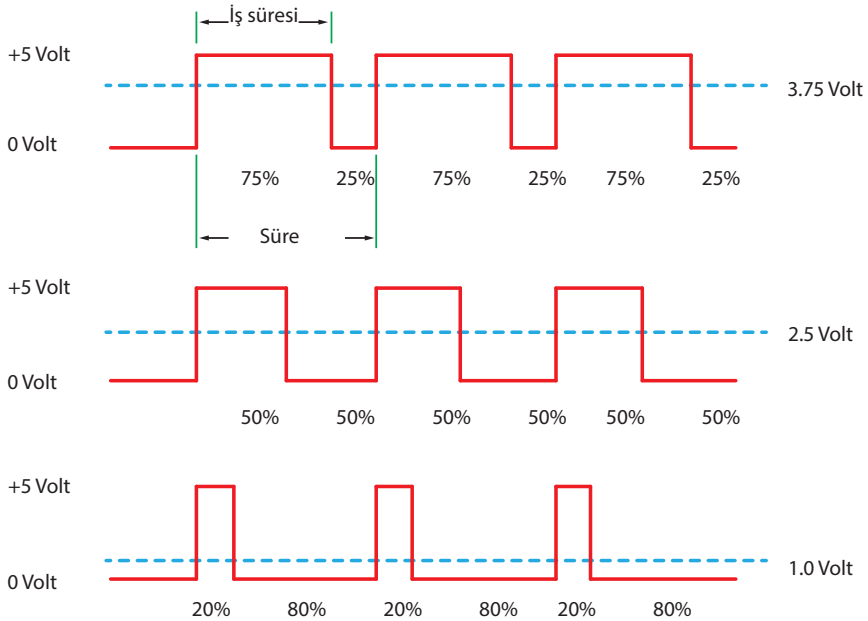
DC motor, doğru akım elektrik enerjisini mekanik enerjiye dönüştüren elektrik motorudur. Motorun içinde yer alan sargılara elektrik akımı uygulandığında manyetik kuvvetin etkisiyle motor hareket eder. DC motorda + ve - kutupların yerleri değiştirilerek dönüş yönü değiştirilirken uygulanan gerilim değiştirilerek de motorun hızı kontrol edilir.

DC motorun dönüş yönünü değiştirmek için H-köprüsü (H-Bridge) kullanılır. H köprüsü devresi, merkezinde motor bulunan ve "H" harfi benzeri bir düzenleme oluşturan dört anahtar içerir. Görsel 2.23'te çapraz iki anahtarın aynı anda kapatılması, motora uygulanan voltajın polaritesini tersine çevirir ve motorun dönüş yönünün değişmesini sağlar.



Görsel 2.23: DC motor yönü değiştirme (H-köprüsü)

Görsel 2.24'te motorun hızını ayarlamak için motora uygulanan gerilim değişiminin mikrodenetleyicilerde PWM kullanılarak yapılışı gösterilmiştir.

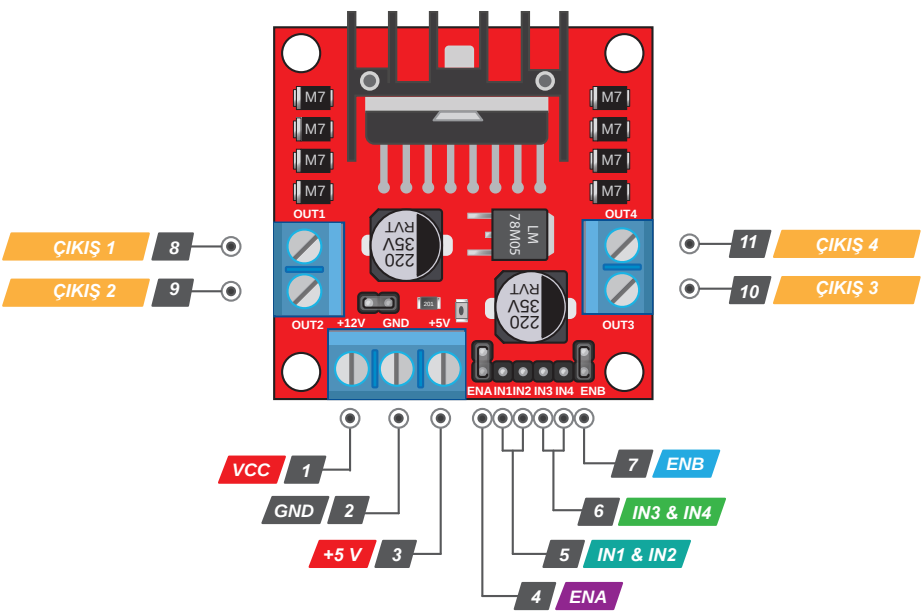


Görsel 2.24: DC motor gerilimini değiştirme (PWM tekniği)

Arduino pininden sağlanan 5 V, 40 mA (0,2 W) değerleri, motorları (mikro DC motorlar hariç) çalıştırmak için yeterli değildir. Bunun için Arduino ile motor arasında motor sürücü adı verilen güç üniteleri kullanılır.

L298N Motor Sürücü

H-Köprüsü oluşturmak için dört ayrı transistör kullanılabildiği gibi entegre devreler de kullanılabilir. Görsel 2.25'te H-Köprüsü içeren L298N motor sürücünün pin yapısı verilmiştir. Benzer özellikleri olan ancak daha küçük boyutlara sahip TB6612FNG motor sürücüsü de uygulamalarda tercih edilebilir.



Görsel 2.25: L298N pin yapısı

VCC: 5 V – 35 V besleme.
GND: Topraklama pini.

5 V: 5 V enable **jumper** yerindeyse bu pin çıkış görevi görür ve Arduino'u beslemek için kullanılır. 5V enable **jumper** çıkmamışsa modül üzerindeki 5 V regülatör pasif bırakılır. L298N içindeki anahtarlama mantığı devresinin gücü kesilir. Bu durumda 5 V pini Arduino üzerindeki 5 V pininden güç alacak şekilde haricî besleme bulunmayan mikro DC motor uygulamasında kullanılabilir.

ENA: A motorunun hız kontrolünü yapar. **Jumper** takılıysa 5 V sağladığından tam güç verilir. **Jumper**ı çıkarılıp bu pin PWM çıkışına bağlandığında A motorunun hızı kontrol edilir.

IN1 ve **IN2:** İki pinin kutuplanmasına göre (lojik 0 veya 1) A motorunun dönüş yönü belirlenir.

IN3 ve **IN4:** İki pinin kutuplanmasına göre (lojik 0 veya 1) B motorunun dönüş yönü belirlenir.

ENB: B motorunun hız kontrolünü yapar. **Jumper** takılıysa 5 V sağladığından tam güç verilir. **Jumper**ı çıkarılıp bu pin PWM çıkışına bağlandığında B motorunun hızı kontrol edilir.

OUT1-OUT2: A motorunun bağlandığı pinler (Tam güçte VCC kadar voltaj çıkışı verir. Çıkış akımı en fazla 2 A olur.).

OUT3-OUT4: B motorunun bağlandığı pinler.

NOT

ENA ve ENB **jumper**ları takılıyken IN1, IN2, IN3 ve IN4 girişleri Arduino PWM çıkışlarına bağlanarak analogWrite() fonksiyonuyla da kullanılabilir. Bu kullanımda kod ve bağlantıdan tasarruf edilir.

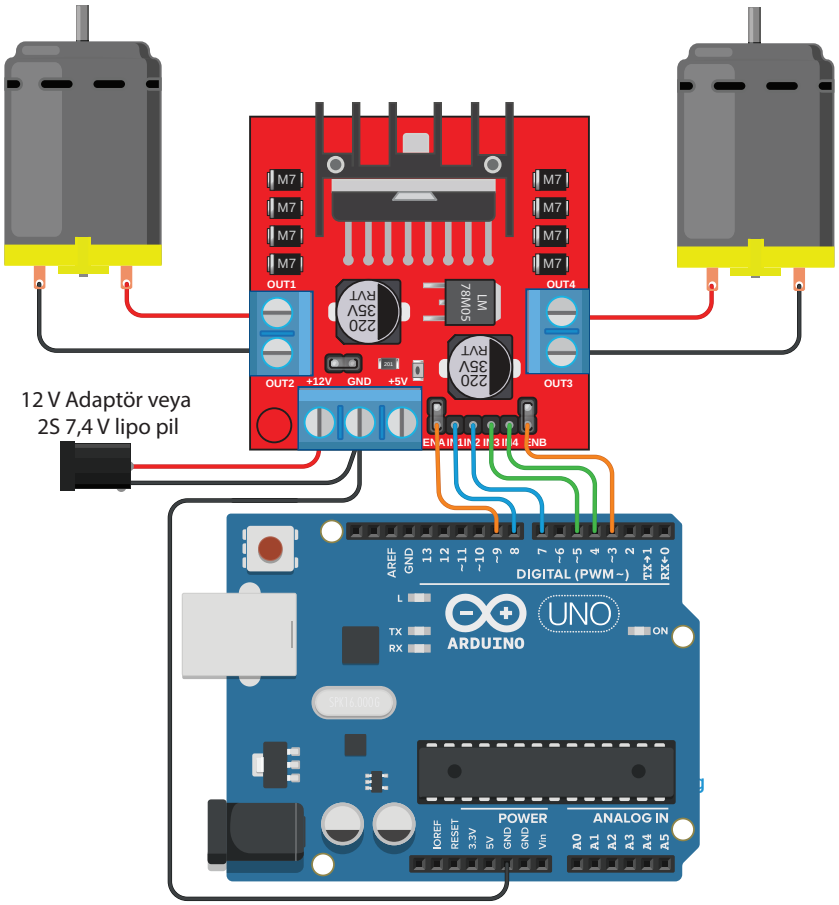
Örnek: L298N motor sürücünün IN1, IN2, IN3 ve IN4 girişlerinde analogWrite() kullanımı.

```
const byte IN1 = 5, IN2 = 6, IN3 = 9, IN4 = 10; //Tümü PWM çıkışı.
//enA ve enB jumperları kaldırılmamıştır.
void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}

void loop() {
  analogWrite(IN1, 200);
  analogWrite(IN2, 100);
  analogWrite(IN3, 200);
  analogWrite(IN4, 100);
}
```

Uygulama Adı: DC Motor Uygulaması — No.: 8

Amaç: DC motor uygulaması yapmak.

Görsel 2.26'daki devredeki DC motorlar, L298N motor sürücüsüyle sürülmektedir. Uygulamada motorlar birer saniye aralıklarla ileri, geri, sola ve sağa döndürülmektedir. Motorlara uygun tekerlek ve şase takılarak sonuçlar gözlemlenebilir.



Görsel 2.26: L298N'le DC motor sürme

DC motor sürme uygulaması programı aşağıdaki gibidir:

```
const byte IN1 = 8, IN2 = 7, IN3 = 5, IN4 = 4;
const byte enA = 9, enB = 3;
byte hiz = 255; // PWM.
void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}
```

```
void loop() {
  // İleri.
  digitalWrite(IN1, 1);
  digitalWrite(IN2, 0);
  digitalWrite(IN3, 1);
  digitalWrite(IN4, 0);
  analogWrite(enA, hiz); // Tam güç.
  analogWrite(enB, hiz);
  delay(1000);
  // Geri.
  digitalWrite(IN1, 0);
  digitalWrite(IN2, 1);
  digitalWrite(IN3, 0);
  digitalWrite(IN4, 1);
  analogWrite(enA, hiz / 4);
  analogWrite(enB, hiz / 4);
  delay(1000);
  // Sol.
  digitalWrite(IN1, 0);
  digitalWrite(IN2, 1);
  digitalWrite(IN3, 1);
  digitalWrite(IN4, 0);
  analogWrite(enA, hiz / 2); //Yarım güç.
  analogWrite(enB, hiz / 2);
  delay(1000);
  // Sağ.
  digitalWrite(IN1, 1);
  digitalWrite(IN2, 0);
  digitalWrite(IN3, 0);
  digitalWrite(IN4, 1);
  analogWrite(enA, hiz / 2);
  analogWrite(enB, hiz / 2);
  delay(1000);
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.26'daki devreyi kurunuz. Programı yükleyiniz.
4. Motorların dönüş yönünü gözlemleyiniz.
5. Yön sürelerinde ve hızlarında değişiklik yaparak farkı gözlemleyiniz.
6. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Görsel 2.26'daki uygulamada motorları bir saniye durdurmak için komutlar nasıl düzenlenir? Belirtiniz.
2. A0 ve A1 analog girişlerine potansiyometre veya **joystick** bağlayarak dört yön kontrolü yapınız.

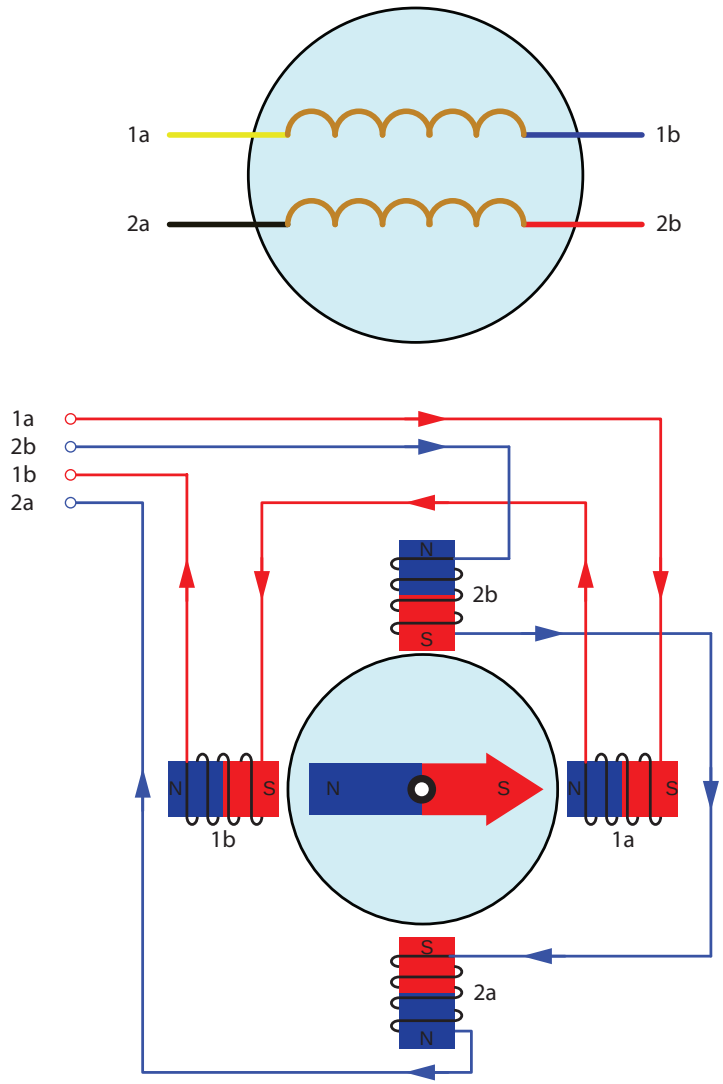
```
const byte IN1 = 5, IN2 = 6, IN3 = 9, IN4 = 10; // Tümü PWM çıkışı.
// enA ve enB jumperları kaldırılmamıştır.
void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}
void loop() {
  byte ileri = map(analogRead(A0), 0, 460, 0, 255); // 460-564 arası dur.
  byte geri = map(analogRead(A0), 564, 1023, 0, 255);
  byte sol = map(analogRead(A1), 0, 460, 0, 255);
  byte sag = map(analogRead(A1), 564, 1023, 0, 255);
  // İleri.
  analogWrite(IN1, ileri);
  analogWrite(IN2, 0);
  analogWrite(IN3, ileri);
  analogWrite(IN4, 0);
  // Geri.
  analogWrite(IN1, 0);
  analogWrite(IN2, geri);
  analogWrite(IN3, 0);
  analogWrite(IN4, geri);
  // Sol.
  analogWrite(IN1, 0);
  analogWrite(IN2, sol);
  analogWrite(IN3, sol);
  analogWrite(IN4, 0);
  // Sağ.
  analogWrite(IN1, sag);
  analogWrite(IN2, 0);
  analogWrite(IN3, 0);
  analogWrite(IN4, sag);
}
```

Sorular

1. Dinamik frenleme (durdurma) nedir? Araştırınız.
2. Dinamik frenleme Görsel 2.26'daki uygulamada nasıl uygulanır? Belirtiniz.
3. enA = 9, enB = 3 pinleri setup fonksiyonunda neden çıkış olarak tanımlanmamıştır? Belirtiniz.

Step Motor Uygulaması

Step motorlar, adımlarla hareket eden DC motorlardır (Görsel 2.27). Faz adı verilen gruplar hâlinde organize edilmiş çoklu bobinleri vardır. Her faza sırayla enerji verildiğinde motor her seferinde bir adım olacak şekilde döner. Çift kutuplu step motorlar, iki bobinden oluşur ve genellikle bobin başına iki olmak üzere dört bağlantıya sahiptir. Tek kutuplu bir step motor elektriksel olarak iki ayrı bobinden oluşur. Her bobinin ortadan bağlantısı olduğu için her bobinde üç bağlantı vardır. Bobindeki orta uçların hepsi birleştiğinden bobin beş bağlantıya sahiptir.

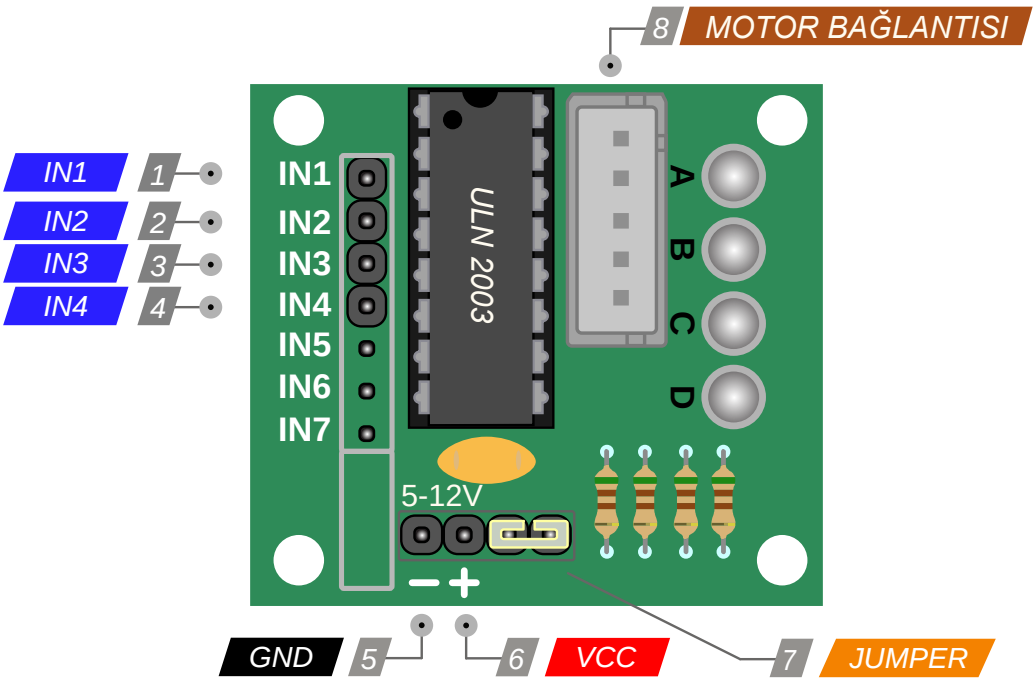


Görsel 2.27: Bipolar step motorun kavramsal modeli

Uygulama 9'da Kullanılan Step Motor ve Sürücü Kartın Özellikleri

28BYJ-48 Step Motor: 5 voltta çalışan 5 telli tek kutuplu bir step motordur. Motorun güç tüketimi 240 mA civarındadır. Motor tam adım modunda çalıştığında her adım 11,25°'lik bir dönüşe karşılık gelir. Bu durum devir başına 32 adım olduğu anlamına gelir (360°/11,25° = 32). Ayrıca motorda 1/64 redüksiyon dişli seti bulunur. (Aslında 1/63,68395) Bunun anlamı, devir başına aslında 32 x 63,68395 =~ 2038 adım demektir.

ULN2003 Sürücü Kartı: ULN2003, yedi darlington transistör çiftinden oluşan motor sürücü kartıdır. Her bir çift 500 mA ve 50 V'a kadar yükleri çalıştırabilir. Kartın, 28BYJ-48 step motor kablolarına uygun konnektörü vardır. Kartta, kontrol giriş hattında adım durumunu gösteren dört adet LED bulunur. Görsel 2.28'de ULN2003 sürücü kartı pin yapısı görülmektedir.



Görsel 2.28: ULN2003 sürücü kartı pin yapısı

IN1-IN4: Motoru sürmek için Arduino'daki dijital çıkış pinlerine bağlanır.

GND: Topraklama pini.

VCC: 5 V-12 V haricî besleme.

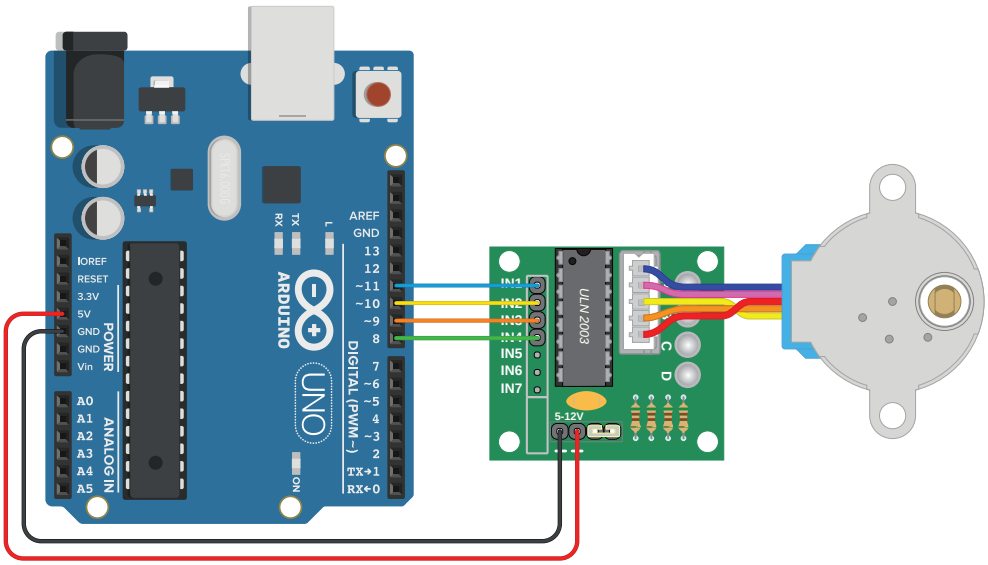
Motor Balants: Step motor konnektörün bağladığı yerdir (Tek yönlü bağlanabilir.).

Jumper: Jumper çıkarıldığında motora enerji gitmez.

Uygulama Adı: Step Motor Uygulaması — No.: 9

Amaç: Step motor uygulaması yapmak.

Görsel 2.29'daki uygulamada step motor önce saat yönünde bir tur, sonra saat yönünün tersinde bir tur dönmektedir. Step motor başlığına çubuk benzeri bir aparat takılarak dönüşler daha iyi gözlemlenir.



Görsel 2.29: Step motor uygulaması

Step motor uygulaması programı aşağıdaki gibidir:

```
int x, adimSayisi = 2038, sure = 2;
void setup() {
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
}
void loop() {
  for (x = 0; x < adimSayisi/4; x++) {
  for (byte i = 8; i <= 11; i++) { //Sağa dön.
   digitalWrite(i, 1);
   delay(sure);
   digitalWrite(i, 0);
  }
  }
  for (x = 0; x < adimSayisi/4; x++) {
  for (byte i = 11; i >= 8; i--) { //Sola dön.
 digitalWrite(i, 1);
   delay(sure);
```

```
   digitalWrite(i, 0);
  }
  }
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.29'daki devreyi kurunuz. Programı yükleyiniz.
4. Step motorun çalışmasını gözlemleyiniz.
5. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Step motoru iki tur sağa, bir tur sola döndürünüz.
2. Aşağıdaki kod step motoru nasıl çalıştırır?

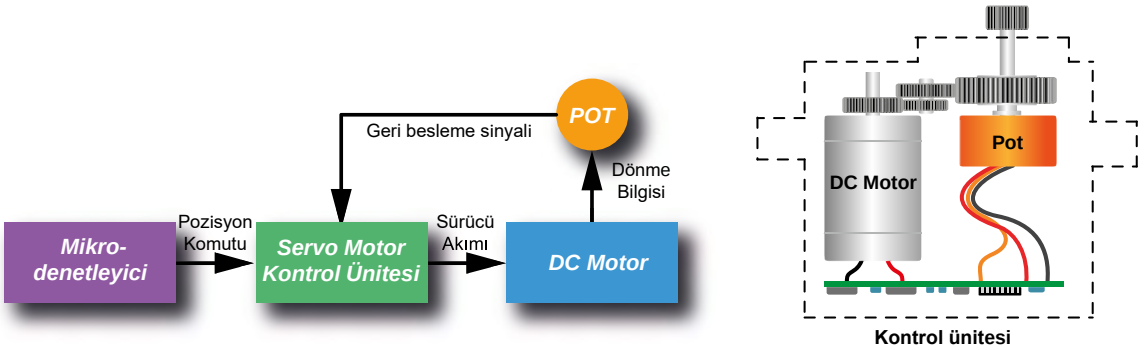
```
#include <Stepper.h>

const int adimSayisi = 200;  // Bir devirdeki adım sayısı

Stepper stepMotor(adimSayisi, 8, 9, 10, 11);

void setup() {
  stepMotor.setSpeed(60); //Hız 60 rpm ayarlandı.
  Serial.begin(9600);
}

void loop() {
  Serial.println("Sağa dön.");
  stepMotor.step(adimSayisi);
  delay(500);

  Serial.println("Sola dön.");
  stepMotor.step(-adimSayisi);
  delay(500);
}
```

Soru

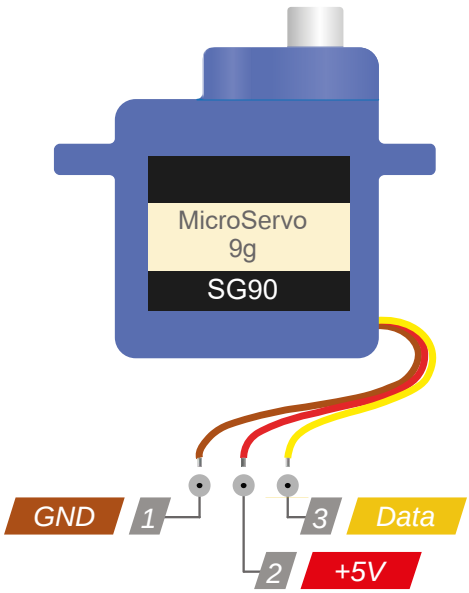
1. 90°lik dönüşte step motor kaç adım atar?

Servo Motor Uygulaması

Servo, kapalı çevrim (geri besleme) kontrol sistemi anlamına gelir. Kapalı çevrim kontrol sistemi, istenen sonucu elde etmek yani motorun hızını ve yönünü ayarlamak için geri besleme sinyalini kullanır. Görsel 2.30'da servo motor geri besleme blok şeması ve kontrol ünitesi görülmektedir. DC motor, dişliler aracılığıyla çıkış miline bağlıdır ki bu mil potansiyometrenin milidir. Potansiyometre, motorun mevcut konumunun hedef konumla karşılaştırıldığı servo kontrol ünitesine konum geri bildirimi sağlar. Hataya göre kontrol ünitesi motorun gerçek konumunu, hedef konumla eşleşecek şekilde düzeltir. Görsel 2.31'de ise mikro servo pin yapısı görülmektedir.



Görsel 2.30: Servo motor geri besleme blok şeması ve kontrol ünitesi



Görsel 2.31: Mikro servo pin yapısı

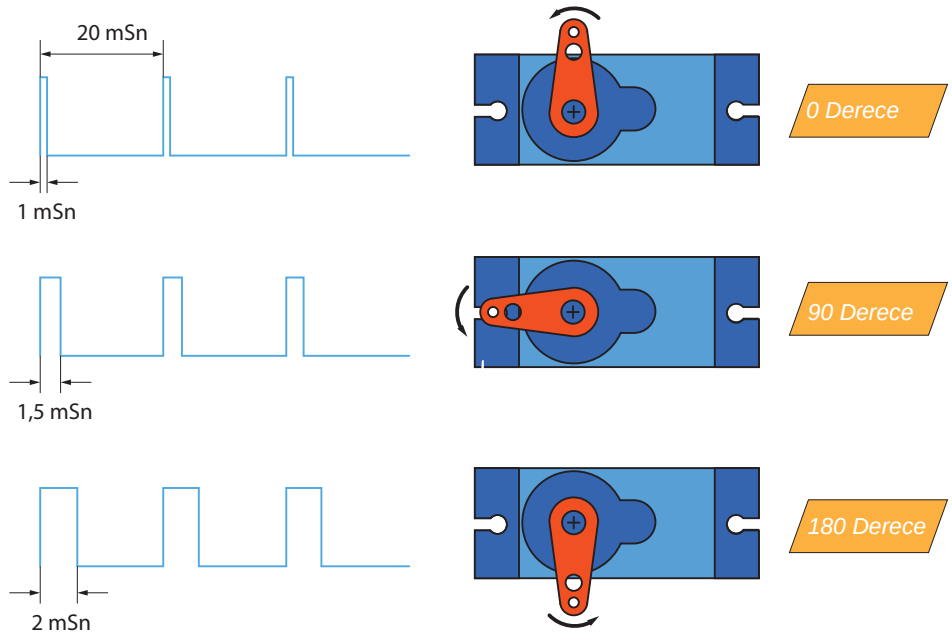
VCC: 5 V besleme.

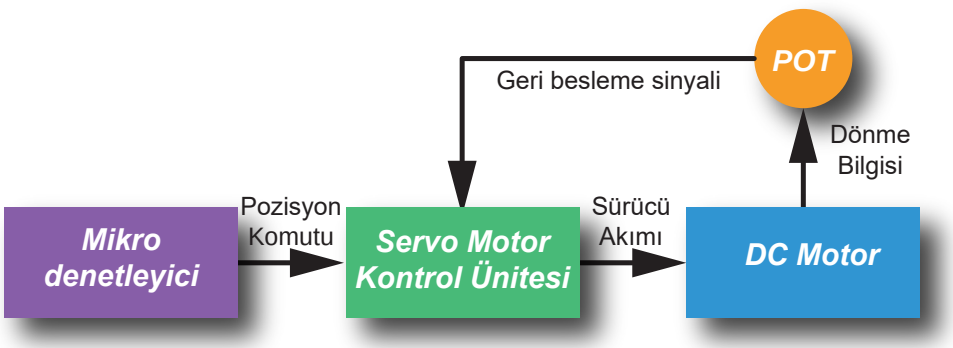
DATA: PWM girişi

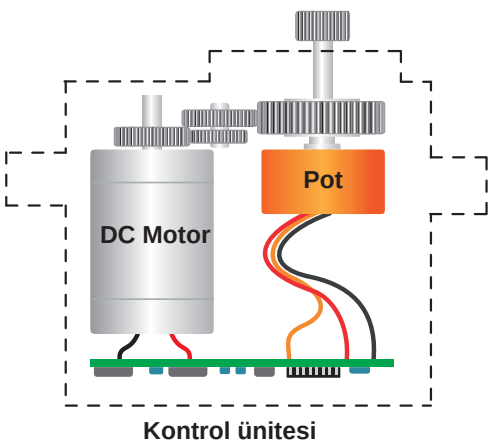
GND: Topraklama pini.

Data hattına PWM sinyali gönderilerek servo motoru kontrol edilir. Görsel 2.32'deki servo motor darbe genişliği ile açı ilişkisinde görüldüğü gibi 20 milisaniyede bir (50 Hz) gönderilen darbenin genişliğine göre açı pozisyonu belirlenir.

Darbe genişliği (lojik 1'de kalma süresi) 1 ms ile 2 ms arasındaki değişim servo motorda 0° ile 180° arasına değişimle eşleşir. Servo motor boşta 10 mA, hareket hâlindeyken 100 mA-250 mA arası akım çeker.







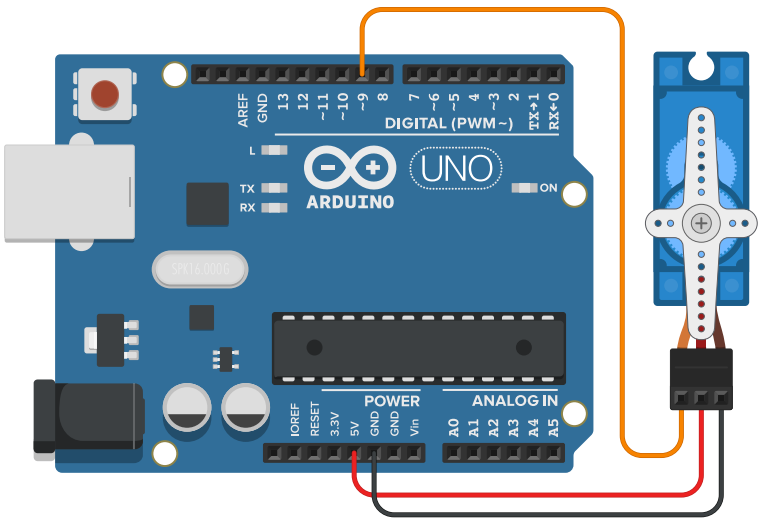
Görsel 2.32: Servo motor darbe genişliği ile açı ilişkisi

Uygulama Adı: Servo Motor Uygulaması — No.: 10

Amaç: Servo motor uygulaması yapmak.

Görsel 2.33'teki uygulamada servo motor 0°den 180°ye birer derece açıyla gidip aynı şekilde geri döner. 0° ile 180° arası mesafe, bu servo motorda 300 ms sürmektedir. 300 ms'den hızlı komutlarda servo motor doğru çalışmaz.

Menülerden ”Taslak → Library Ekle → Servo” seçilerek programın başına eklenir. Bu programların örneğine menülerden “Dosya → Örnekler → Servo → Sweep ve Knob” adımları izlenerek de ulaşılabilir.



Görsel 2.33: Mikro servo uygulaması

Mikro servo uygulaması programı aşağıdaki gibidir:

```
#include <Servo.h> // v1.1.7
Servo servo; // Servo isimli nesne oluşturuldu.
byte derece = 0; // Servo derecesini tutan değişken.

void setup() {
  servo.attach(9); // Servo motor 9 numaralı PWM çıkışına bağlı.
}
void loop() {
  for (derece = 0; derece <= 180; derece += 1) { // 0°'den 180°'ye birer derece git.
   servo.write(derece); // Servo motorun gideceği derece.
    delay(15); // 15 ms bekle.
  }
  for (derece = 180; derece >= 0; derece -= 1) { // 180°'den 0°'ye birer derece git
  servo.write(derece);
  delay(15); // 15 x 180 = 2700 ms'de 0°'den 180°'ye tamamlar.
  }
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.33'teki devreyi kurunuz. Programı yükleyiniz.
4. Servo motorun çalışmasını gözlemleyiniz.
5. delay(15); komut satılarını // yorum satırına dönüştürerek programı tekrar yükleyiniz. Programın çalışmasındaki değişikliği gözlemleyiniz.
6. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. A0 girişine potansiyometre veya **joystick** bağlayarak servo motorun pozisyonunu kontrol eden uygulamayı gerçekleştiriniz.

```
#include <Servo.h>
Servo servo; // Servo isimli nesne oluşturuldu.
void setup() {
  servo.attach(9); // Servo motor 9 numaralı PWM çıkışına bağlı.
}
void loop() {
  int pot = analogRead(A0); // Potansiyometrenin konumunu oku. (0 - 1023)
  byte derece = map(pot, 0, 1023, 0, 180);/* Servo motorun kullanacağı 0 - 180 aralığına dönüştür. */
  servo.write(derece); // Servo motorun gideceği derece.
  delay(15);
}
```

Soru

1. Mikro servo motor 1°lik açıyı kaç saniyede tamamlar? Belirtiniz.

2.6. HABERLEŞME UYGULAMALARI

Arduino ve benzeri birçok mikrodenetleyici sıklıkla UART, I2C ve SPI haberleşme standartlarını kullanmaktadır.

Seri İletişim: Bitlerin verinin tek bir iletim yolu üzerinden sırasıyla aktarılmasıdır.

Paralel İletişim: Verinin tüm bitlerinin aynı anda transfer edilmesidir. Paralel veri iletiminde iletilen bilginin her biti için ayrı bir yol (bus) bağlantısı vardır.

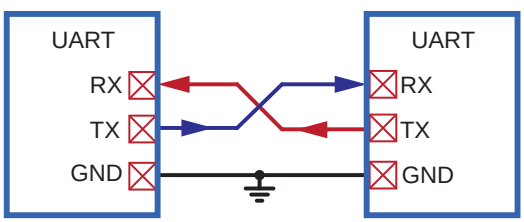
Senkron İletişim: Gönderici clock sinyalinin her yükselen / düşen kenarında verilerle birlikte bir saat sinyali gönderir ve veri değeri alıcı tarafından okunur.

Asenkron İletişim: Saat sinyali yoktur. Alıcı ve gönderici önceden belirlenmiş bir hızda (baud hızı) iletişim kurmaktadır.

Uart/Usart

Arduino ve benzeri birçok mikrodenetleyici üzerinde bulunan seri iletişim birimidir. Bu birimler vasıtasıyla seri iletişim oluşturulur (Görsel 2.34). **UART** [Universal Asynchronous Receiver Transmitter (yunıvırsıl eysingkrınıs risivır transmitır)] Evrensel asenkron alıcı-verici seri iletişim türüdür. **USART** [(Universal Synchronous and Asynchronous Receiver Transmitter (yunıvırsıl singkrınıs end eysingkrınıs risivır transmitır)] ise hem senkron hem de asenkron seri iletişim türüdür.

UART iletişimde veriler **byte** hâlinde iletilir. Her bir **byte** için bir start bir de stop bitleri eklenir. Bir **byte** verinin iletilebilmesi için 10 bit gönderilir. Arduino programlanırken UART kullanılır. Setup() fonksiyonu içerisinde Serial.begin(9600) fonksiyonuyla iletişim hızı (**baudrate**) ayarlanır. Seri haberleşmede iletişim hızı "baud" adı verilen değerle ifade edilir. Bu değer saniyede gönderilen bit sayısını (bps - bits per second) ifade eder. Alıcı ve verici tarafında aynı hız ayarlanmalıdır.



Görsel 2.34: UART haberleşme

I²C Protokolü

I²C, Inter Integrated Circuit (intır intıgreytıd sekıt)] ifadesinin kısaltmasıdır. Aynı zamanda "IIC" olarak da adlandırılır. I²C, düşük hızlı iki hatla kullanılan bir seri protokoldür. Güvenilir iletimin en fazla mesafesi hız arttıkça azalır. En düşük hızda (100 Kbaud veya 100 KHz saat hızı) en fazla mesafe yaklaşık bir metredir.

I²C ve bazı daha yüksek hız modlarına sahiptir. Tüm I²C cihazları şu modları desteklemez:

Hızlı Mod-400 KHz.

Yüksek Hızlı Mod-3,4 MHz.

Ultra Hızlı Mod-5 MHz.

I²C veri yolu, VCC ve GND bağlantısıyla birlikte iki sinyale sahiptir:

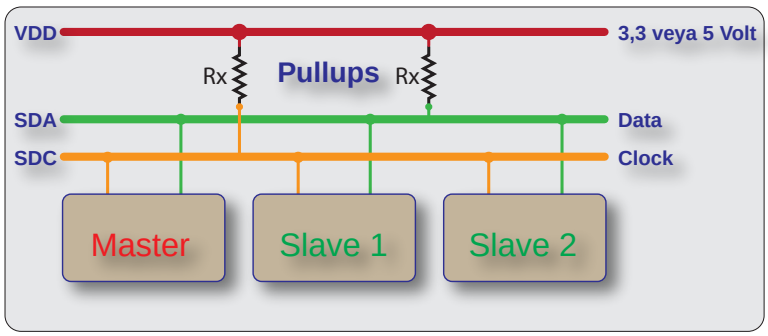
SDA (serial data)-Çift yönlü veri hattıdır.

SCL (serial clock)-Saat sinyalidir.

Her bir sinyal hattına bağlı iki adet pull-up rezistörü vardır (Görsel 2.35). Arduino türlerine göre SDA ve SCL pinleri Tablo 2.2'de gösterilmiştir.

Tablo 2.2: Arduino Türlerine Göre SDA ve SCL Pinleri

Arduino 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



Görsel 2.35: I²C haberleşme

I²C veri yoluna ara birim oluşturan **master** ve **slave** isimli iki tür cihaz vardır. **Master** cihaz veri yolunu kontrol eder ve saat sinyalini sağlar. **Slave** cihazlardan tek tek veri ister. Veri yolu üzerinde birden fazla **master** cihaz olabilir ancak herhangi bir anda sadece bir tanesi aktif **master** olabilir. **Master** cihazların kendilerine atanmış bir adresi yoktur. Slave cihazların bir adresi vardır ve bu adresin veri yolunda benzersiz olması gerekir. 7 bitlik adresleme kullanıldığından I²C veri yolunda 128'e kadar slave cihaz olabilir. Yeni nesilde 10 bitlik adresleme de kullanılmaktadır.

Arduino'da I2C protokolüyle haberleşmek için yerleşik Wire.h kütüphanesi kullanılır.

start() - Kütüphaneyi başlatır ve Arduino'yu **master** veya **slave** olacak şekilde ayarlar.

requestFrom() - **Master** tarafından bir **slave**dan veri istemek için kullanılır.

startTransmission() - **Master** tarafından belirtilen bir **slave**a veri göndermek için kullanılır.

endTransmission() - **Master** tarafından startTransmission fonksiyonuyla başlatılan bir iletimi sonlandırmak için kullanılır.

write() - I2C veri yoluna veri göndermek için hem **master** hem de **slave** tarafından kullanılır.

available() - Alınan verinin **bayt** sayısını belirlemek için hem **master** hem de **slave** tarafından kullanılır. read() - I2C veri yolundan bir **bayt** veri okur.

SetClock() - **Master** tarafından belirli bir saat frekansını ayarlamak için kullanılır.

onReceive() - **Master** cihazdan veri geldiğinde **slave** cihaz tarafından bir fonksiyon çalıştırılır.

onRequest() - **Master** cihaz veri istediğinde **slave** cihaz tarafından bir fonksiyon çalıştırılır.

SPI Protokolü [Serial Peripheral Interface (siriıl pırifrıl intırfeys)]

Senkron haberleşme protokollerinden bir tanesidir. **Full duplex** (eş zamanlı çift yönlü

çalışabilen) olarak çalışabilir. Haberleşme gerçekleştirilecek cihazlar arasında **master - slave** ilişkisi vardır. Birden fazla **slave** cihazla haberleşme sağlanabilir. Senkron olarak çalıştığı için clock sinyali kullanılır. Kısa mesafeli iletimde tercih edilir (Görsel 2.36).

Master ve çevresel cihazlara bağlanan **MISO** (Master In Slave Out), **MOSI** (Master Out Slave In) ve **SCK** (Serial Clock) olmak üzere üç adet SPI hattı bulunur.

MISO: Çevresel cihazlardan (**slave**) yollanan verilerin **master** cihaza aktarıldığı hattır.

MOSI: Master cihazdan yollanan verilerin çevresel cihazlara aktarıldığı hattır.

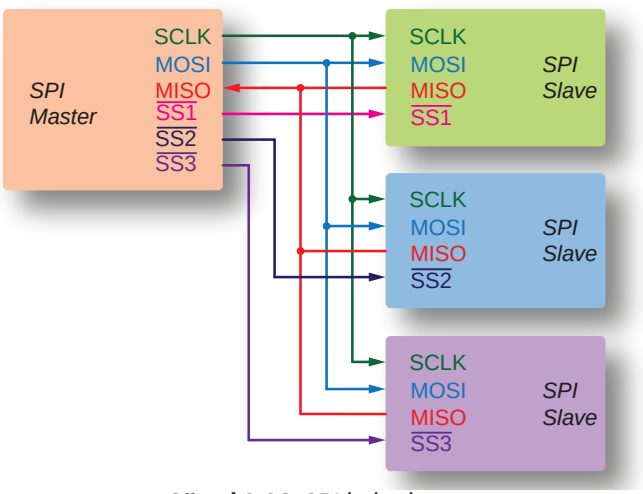
SCK: SPI haberleşmesinde senkronu sağlayan saat sinyalinin bulunduğu hattır. Saat sinyali **master** cihaz tarafından üretilir.

SPI protokolünde I²C'den farklı olarak veri hatları tek yönlüdür. **Slave** cihazların adreslerinin olmasına gerek yoktur. Her **slave** cihazın seçim ayağı bulunur. Bu ayağa **SS** (Slave Select) denir. Bu hattın sayısı kullanılan slave cihazların sayısı kadardır. Her cihaz için **master** cihazından ayrı SS hattı çıkar. SS hattı LOW (0 volt) düzeyinde olan **slave** cihaz, **master** cihazla iletişime başlar.

Arduino türlerine göre MOSI, MISO ve SCK pinleri Tablo 2.3'te gösterilmiştir. SS için herhangi bir dijital çıkış kullanılabilir.

Tablo 2.3: Arduino Türlerine Göre MOSI, MISO ve SCK Pinleri

Arduino türü	MOSI	MISO	SCK
Arduino Uno	11	12	13
Arduino Nano	11	12	13
Arduino Mega	51	50	52



Görsel 2.36: SPI haberleşme

Arduino'da SPI protokolüyle haberleşmek için yerleşik SPI.h kütüphanesi kullanılır.

SPI.begin(): SPI haberleşmesini başlatır ve SPI pinlerini başlangıç konumlarına alır.

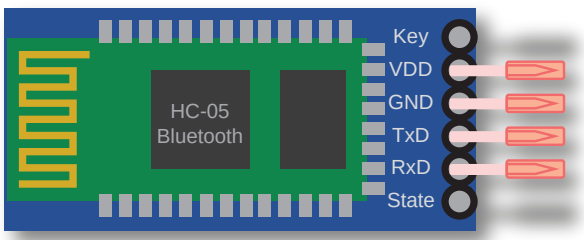
SPI.setClockDivider(): SPI haberleşmesinin saati ayarlanır. Fonksiyon, değer olarak saat değişkenlerini almaktadır. Eğer hiçbir değişiklik yapılmazsa SPI saati "SPI_CLOCK_DIV4" olarak çalışır. Fonksiyonun alabileceği değişkenler; SPI_CLOCK_DIV4, SPI_CLOCK_DIV8, SPI_CLOCK_DIV16,

SPI_CLOCK_DIV32, SPI_CLOCK_DIV64, SPI_CLOCK_DIV128'dir.

SPI.transfer(): SPI hattına veri yollamak veya veri almak için bu fonksiyon kullanılır.

Bluetooth Haberleşme

Bluetooth, veri iletmek ve almak için 10 metre (Bulunduğu ortamdan etkilenir.) menzilli bir kablosuz teknolojidir. 2,400 ila 2,485 GHz arasında ISM (Endüstri, Bilim ve Medikal) bandında çalışır. Bluetooth haberleşme için **UART** protokolü kullanılır. Varsayılan haberleşme hızı 9600 bps'dir. 115,200 **baud**a kadar hızlarda çalışabilir. Modüller ayrıca "AT" metin dizisiyle başlayan standart bir komutlar seti olan AT komut setini de destekler. Görsel 2.37'de HC-05 bluetooth modülünün pin yapısı görülmektedir.



Görsel 2.37: HC-05 pin yapısı

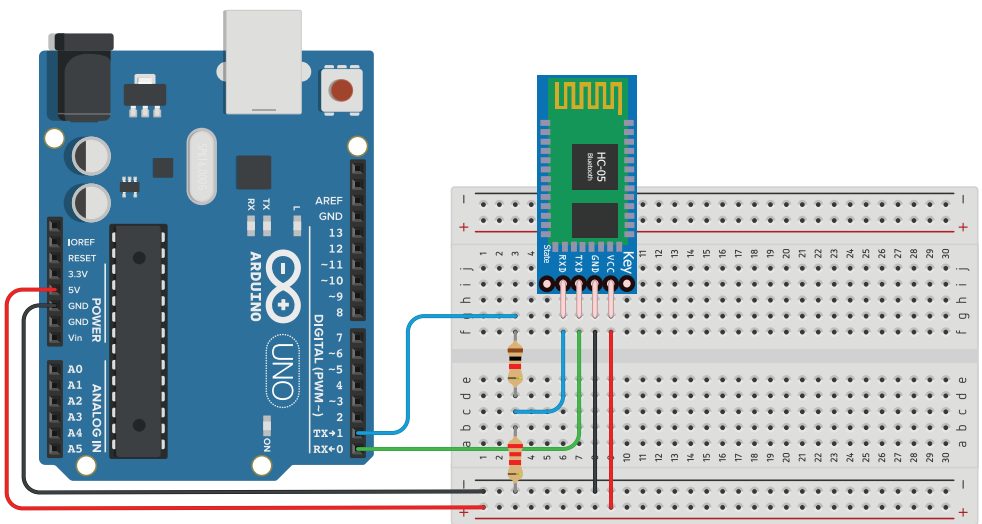
VCC: 3,3 V-5 V besleme.

GND: Topraklama pini.

TXD: Veri çıkışı (3,3 V).

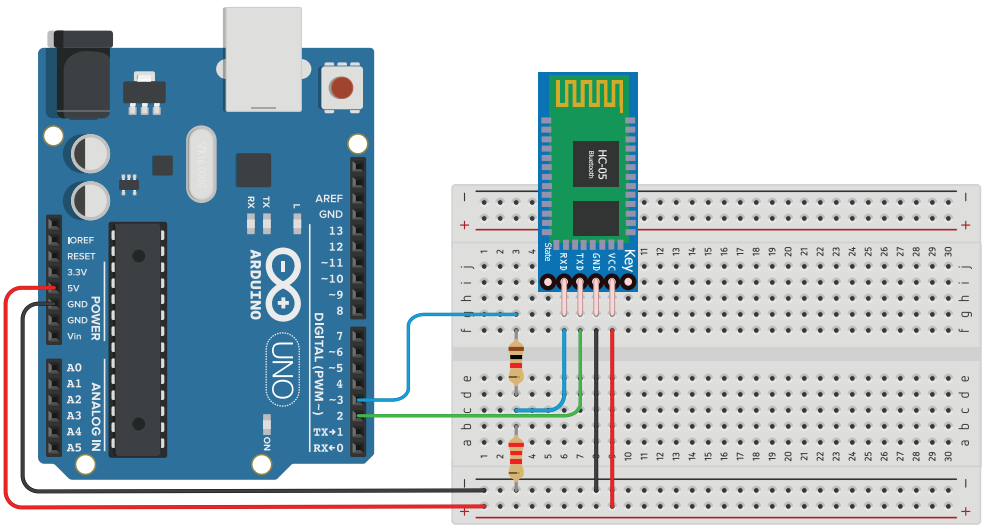
RXD: Veri girişi (3,3 V).

DİKKAT Modülün TX ucu Arduino'nun RX (0. pin) ucuna, modülün RX ucu Arduino'nun TX (1. pin) ucuna bağlanmalıdır. Ancak modülün TX ucundan çıkan 3,3 V Arduino RX ucunda lojik 1 olarak kabul edilirken Arduino'nun TX ucundan çıkan 5 V, modülün RX ucuna doğrudan uygulandığında bozulmasına neden olacaktır. Bu nedenle Arduino'nun TX ucundan çıkan 5 V'u 3,3 V'a düşürmek için Görsel 2.38'deki gibi gerilim bölücü dirençler kullanılmalıdır.



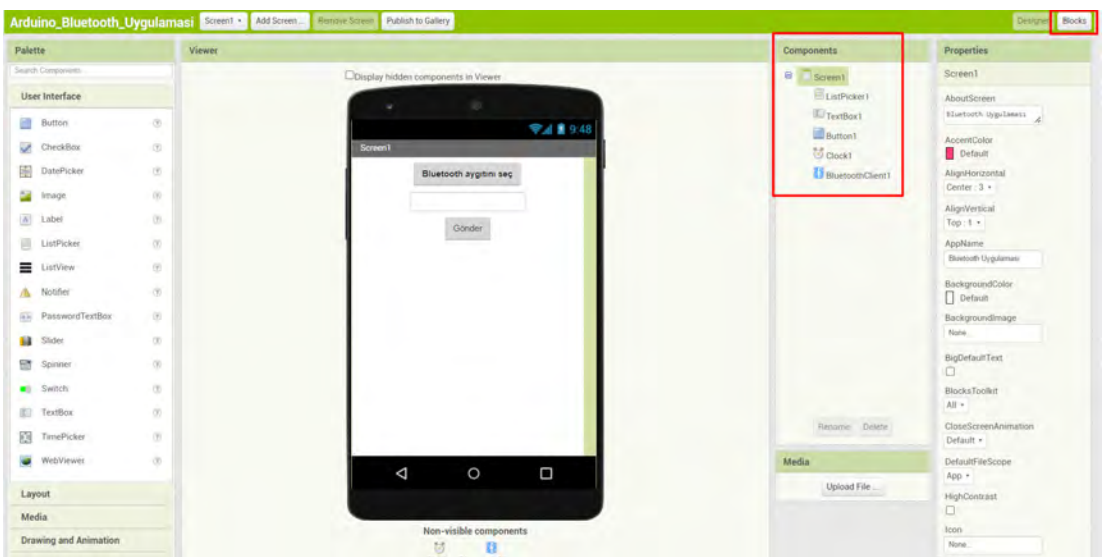
Görsel 2.38: Gerilim bölücü dirençlerle bluetooth modülün Rx ucuna 3,3 V verme

Bluetooth modülü doğrudan Arduino'nun 0. ve 1. pinlerine bağlandığında program yüklenirken (program yükleme bu pinlerden yapıldığı için) hata verir. Modül sökülerek program yüklenir ve modül tekrar takılır. Sök tak işlemini yapmamak için SoftwareSerial.h kütüphanesi kullanılarak istenen dijital pinler (Görsel 2.39'da 2 ve 3 numaralı pinler seçilmiştir.) TX ve RX olarak ayarlanır. Menülerden "Taslak → Library Ekle → SoftwareSerial" seçilerek programın başına eklenir.



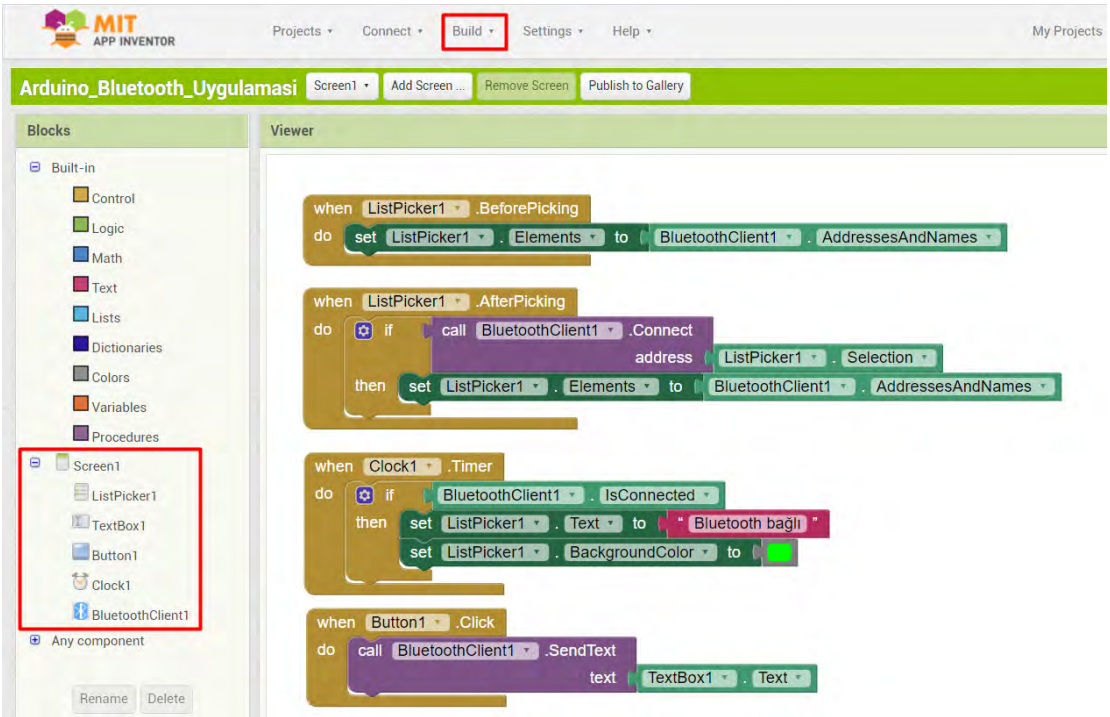
Görsel 2.39: SoftwareSerial.h kütüphanesi ile TX ve RX'i farklı pinlerde kullanma

Devreye enerji verildiğinde bluetooth modülle haberleşmek için Android telefondan bluetooth bağlantısı açılarak hc-05 cihazı bulunur. Varsayılan şifre "1234" veya "0000"dır. Android telefondan veri gönderebilmek için **play store**dan "Arduino bluetooth controller" benzeri uygulamalar kullanılır veya http://ai2.appinventor.mit.edu adresinden basit bir Android uygulaması yapılarak karakterler bluetooth üzerinden gönderilir. Uygulamayı yapmak için Görsel 2.40'taki "Components" bölümünde verilen elemanlar soldaki listeden seçilir. "Bloks" düğmesine basılarak uygulamanın kodları yazılır.



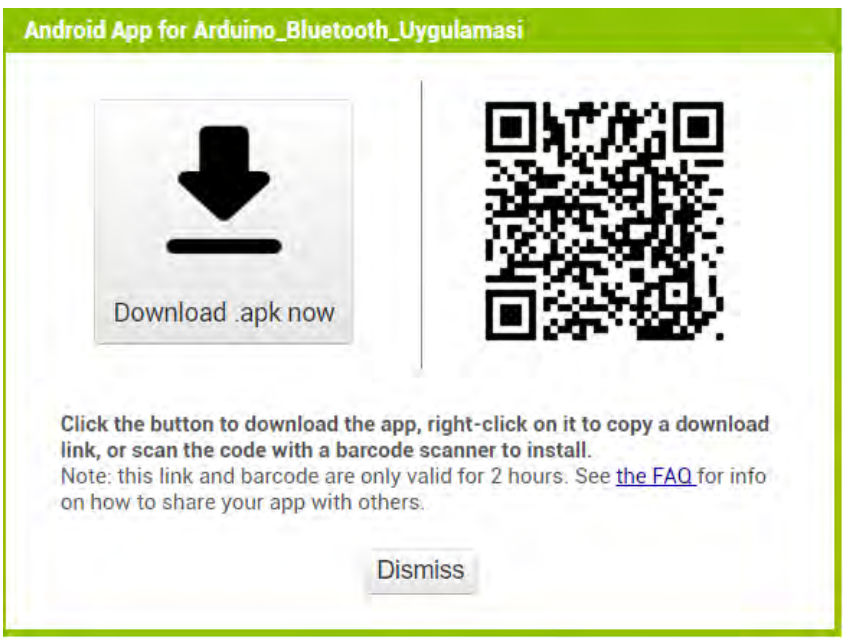
Görsel 2.40: MIT app invertor ile Android için uygulama oluşturma

Görsel 2.41'de görüldüğü gibi soldaki menüden elemanların üzerine tıklanarak ilgili komutlar seçilerek kodlar yazılır. "Build" düğmesine tıklanarak Android App (.apk) oluşturulur.



Görsel 2.41: Uygulama kodları

Görsel 2.42'deki gibi oluşturulan karekod (Kısa bir süre geçerlidir.) okutularak uygulama telefona indirilir. Uygulama yüklenirken "Bilinmeyen kaynaklardan yükle." seçeneği açılmalıdır.



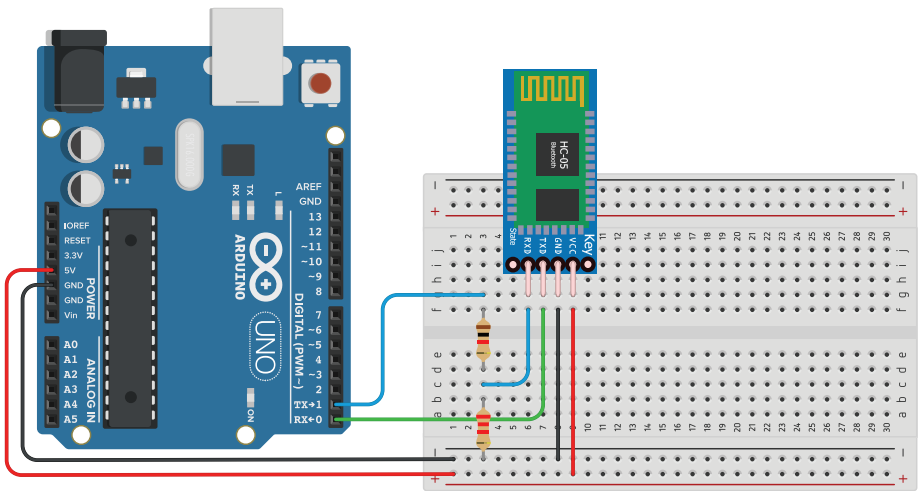
Görsel 2.42: Karekodla akıllı telefona uygulama indirme

Uygulama Adı: Bluetooth Uygulaması — No.: 11

Amaç: Bluetooth uygulaması yapmak.

Bluetooth modülle haberleşmek için telefondan bluetooth bağlantısı açılarak hc-05 cihazı bulunur ve "Bluetooth aygıtını seç" düğmesiyle hc-05 seçilir. Varsayılan şifre "1234" veya "0000" dır. Bluetooth ile "A" harfi gönderilerek LED yakılır, "B" harfi gönderilerek LED söndürülür (Görsel 2.43). Bluetooth uygulama devresi Görsel 2.44'te görülmektedir.

Görsel 2.43: Uygulamayla bluetooth üzerinden karakter gönderme



Görsel 2.44: Bluetooth uygulaması

Bluetooth uygulaması programı aşağıdaki gibidir:

```
void setup() {
  pinMode(13, OUTPUT);
  Serial.begin(9600); // Bluetooth'la seri iletişimi başlat.
}
void loop() {
  if (Serial.available()) { // Veri geliyorsa...
  char karakter = Serial.read(); // Gelen veriyi karakter değişkenine ata.
  if (karakter == 'A') // Karakter A ise...  (a farklı bir karakterdir.)
   digitalWrite(13, 1); // LED'i yak.
  else if (karakter == 'B') // Karakter B ise...
   digitalWrite(13, 0); // LED'i söndür.
  }
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.44'teki devreyi kurunuz. Programı yükleyiniz.
4. Telefondan buluetooh açarak modülle eşleştiriniz.
5. Telefondan uygulamayla A karakteri gönderek LED'in yandığını gözlemleyiniz.
6. Telefondan uygulamayla B karakteri gönderek LED'in söndüğünü gözlemleyiniz.
7. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. SoftwareSerial.h kütüphanesini kullanarak PWM çıkışına bağlı LED'in parlaklığını değiştiriniz. Telefondan 0-255 arası sayı gönderiniz.

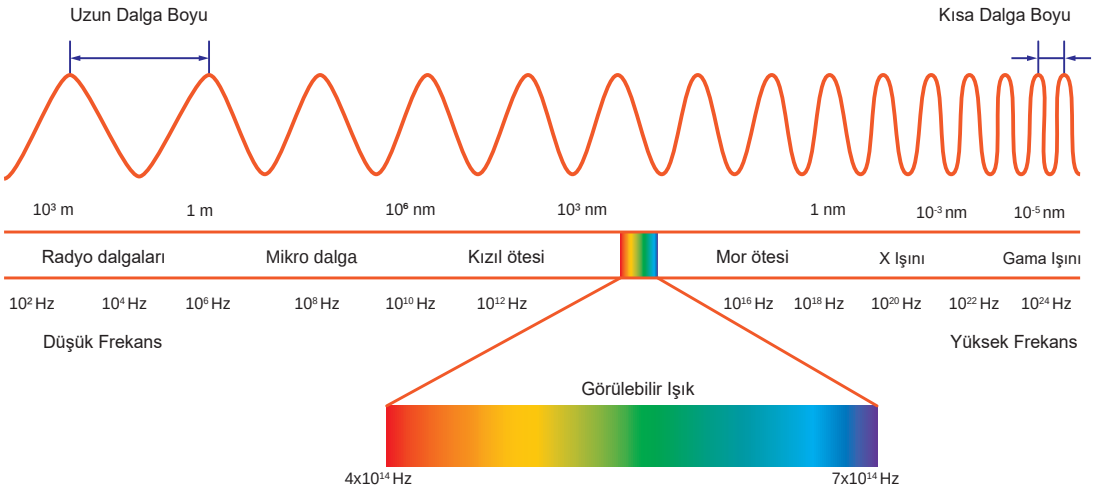
```
#include <SoftwareSerial.h> // Dahili olarak Arduino IDE'de bulunur.
SoftwareSerial bluetooth(10, 11); // Arduino 10. pin RX, 11. pin TX olarak ayarladı.
byte LED = 3; // LED 3 numaralı PWM özellikli pine bağlı.
void setup() {
  bluetooth.begin(9600); // Bluetooth ismiyle oluşturulan nesne başlatıldı.
}
void loop() {
  if (bluetooth.available()) { // 10. pinden veri geliyorsa...
  String veri = bluetooth.readString(); // 0 - 255 arası gelen karakterleri oku.
  int PWM = veri.toInt(); // Gelen metni integer veri tipine dönüştür.
  analogWrite(LED, PWM);
  }
}
```

Soru

1. Gelen veriden hem açma kapama bilgisi hem de PWM bilgisi nasıl ayırt edilerek okunur? Açıklayınız.

Kızılötesi (İnfrared) Işınım ile Haberleşme

Kızılötesi (infrared) ışınım, insan gözünün göremediği dalga boyundadır. Kızılötesi ışınımın dalga boyu 750 nanometre ile 1 mikrometre arasındadır. Görsel 2.45'te ışık tayfı görülmektedir.



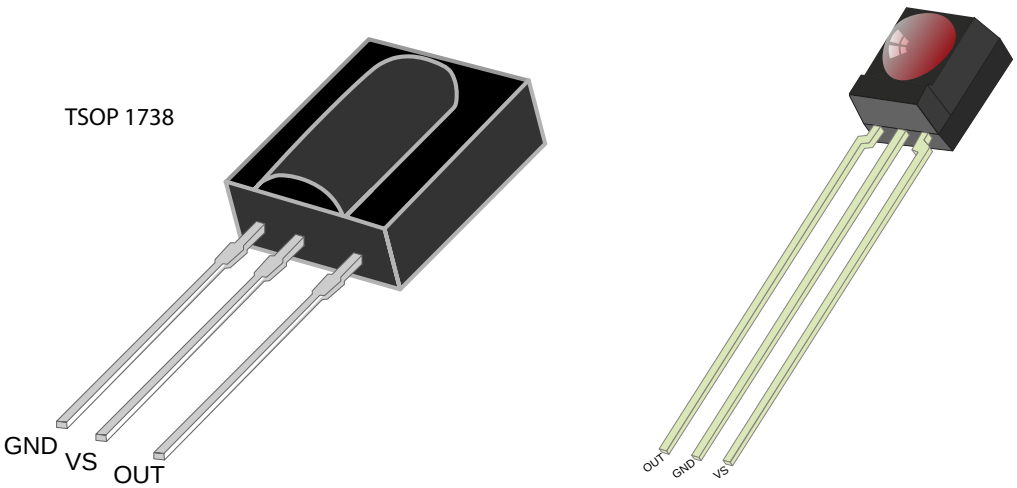
Görsel 2.45: Işık tayfı

Görsel 3.46'da görülen **TSOP1738** 38 KHz IR alıcı modülü üç adet pine sahiptir.

VS (VCC): Besleme gerilimi.

GND: Topraklama pini.

OUT: Çıkış.



Görsel 2.46: IR alıcı pin yapısı

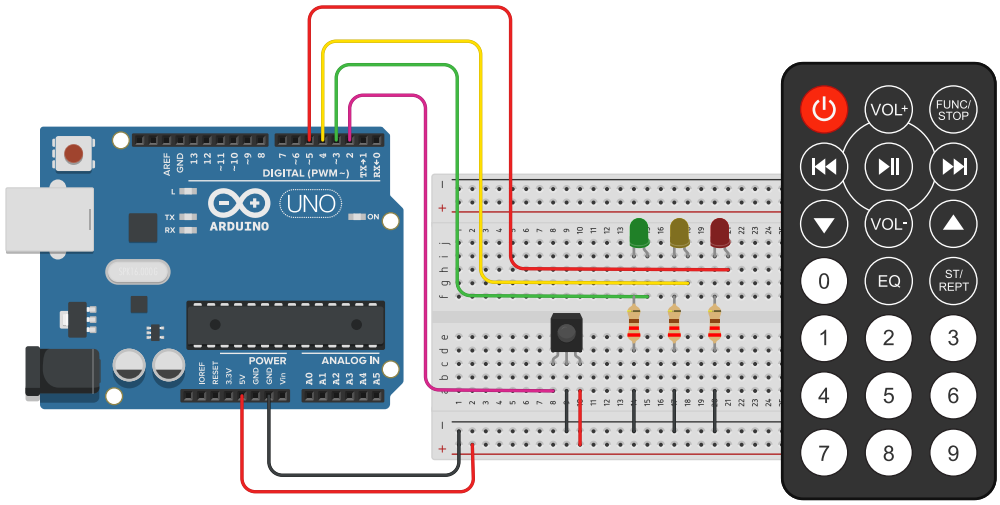
Uygulama Adı: IR ile Uzaktan Kumanda Uygulaması No.: 12

Amaç: IR ile uzaktan kumanda uygulaması yapmak.

Görsel 2.47'de IR ile uzaktan kumanda uygulaması görülmektedir. Burada kullanılan kumandalarda her tuşun farklı bir kodu vardır. Kullanılmak istenen tuşun kodunu öğrenebilmek için menülerden "Taslak → Library Ekle → Kütüphaneleri Yönet" seçeneğinden arama kutusuna "IRremote" yazılarak "IRremote" kütüphanesi bulunur ve kurulur. Menülerden "Taslak → Library Ekle → IRremote" seçilerek programın başına eklenir.

Aşağıdaki program herhangi bir kumandanın basılan tuşlarının **hex**. kodlarını seri ekranda gösterir. Tuş kodları uygulamada kontrol için kullanılmaktadır.

IR alıcının data çıkışı Arduino'nun 2 numaralı pinine bağlıdır. Çıkış olarak ayarlanan pinlere üç ayrı LED veya RGB LED bağlanabilir. Kumandanın 1, 2, 3 numaralı tuşları LED'leri yakıp söndürmektedir. Çıkışta MOSFET, röle vb. güç üniteleri kullanılarak şerit LED, lamba, fan vb. yükler kontrol edilebilir.



Görsel 2.47: IR ile uzaktan kumanda uygulaması

IR ile uzaktan kumanda uygulaması programı aşağıdaki gibidir:

```
#include <IRremote.h>
IRrecv irAlici(2); // irAlici isimli nesne 2. pini giriş olarak kullanır.
decode_results tus; // Kodu çözülmüş sonuçları tus değişkenininde sakla.
#define tus1 0xFF30CF // Burada kullanılacak Hex kodu seri ekranda görünür.
#define tus2 0xFF18E7 #define tus3 0xFF7A85
const byte LED[] = {3, 4, 5};

void setup()
{
  Serial.begin(9600);
  irAlici.enableIRIn(); // Alıcıyı başlat.
  pinMode(LED[0], OUTPUT);
  pinMode(LED[1], OUTPUT);
```

```
  pinMode(LED[2], OUTPUT);
}
void loop() {
  if (irAlici.decode(&tus)) // Tuşa basılmışsa...
  {
  Serial.println(tus.value, HEX); // Tuşun Hex kodunu yazdır.
  if (tus.value == tus1)
   digitalWrite( LED[0], !digitalRead(LED[0])); // Çıkışı tersle.
  if (tus.value == tus2)
   digitalWrite( LED[1], !digitalRead(LED[1])); // Çıkışı tersle.
  if (tus.value == tus3)
   digitalWrite( LED[2], !digitalRead(LED[2])); // Çıkışı tersle.
  irAlici.resume();
  }
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.47'deki devreyi kurunuz. Programı yükleyiniz.
4. Kumandadan 1, 2, 3 numaralı tuşlara basarak LED'leri yakıp söndürünüz.
5. Kumanda tuşlarının kodlarını seri ekrandan görebilirsiniz. Her bir tuş için #define tus1 0xFF 30CF şeklinde satırlar düzenleyerek diğer tuşları da aktif hâle getirebilirsiniz.
6. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. 0. tuşa basıldığında LED'i yakıp söndüren düzenlemeyi gerçekleştiriniz.

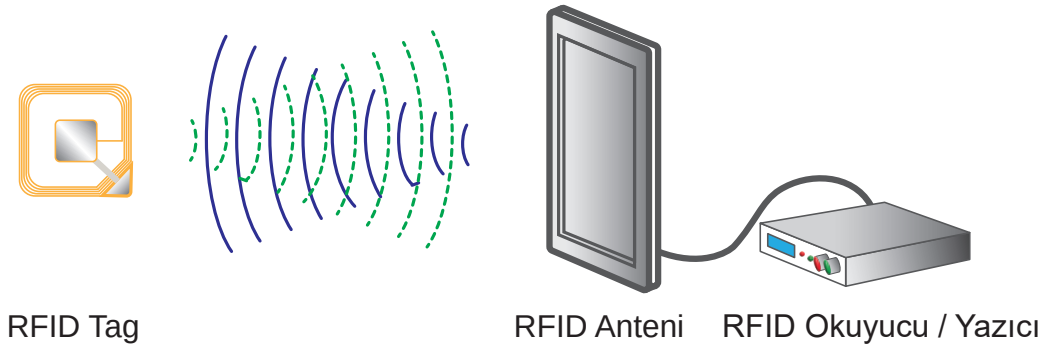
Soru

1. TV kumandaları bu uygulamada kullanılabilir mi? Açıklayınız.

RFID Uygulamaları

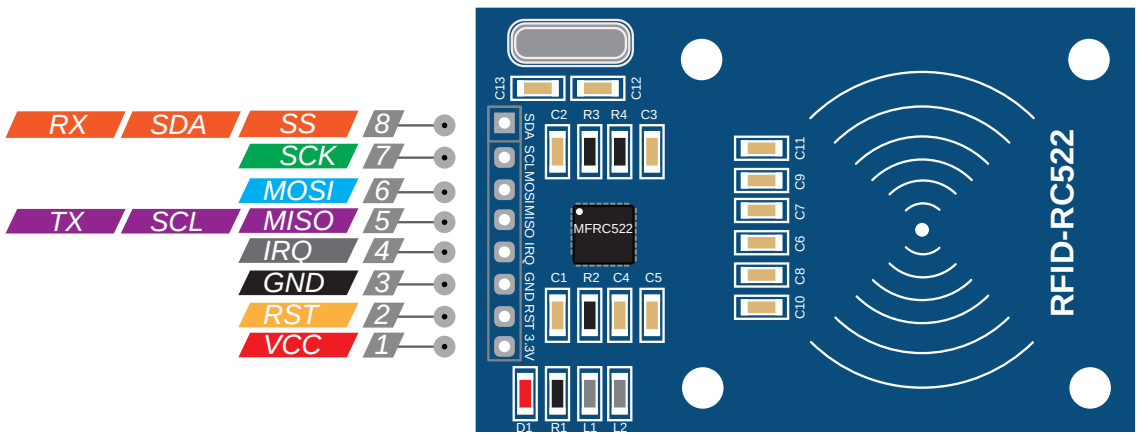
RFID [Radio Frequency IDentification (redyio frikwınsi aydentıfıkeyşın) Radyo Frekans Tanımlama] sistemi, tanımlanacak bir nesneye eklenen bir verici (aktarıcı/etiket) ve alıcı (sorgulayıcı/okuyucu) olmak üzere iki ana bileşenden oluşur.

Alıcı, bir radyo frekansı modülünden ve yüksek frekanslı elektromanyetik alan oluşturan bir antenden meydana gelir. Verici ise genellikle pasif (Pil içermez.) bir cihazdır. Vericinin içinde bilgiyi depolayan ve işleyen bir mikroçip ile sinyali alıp iletmek için bir anten bulunur. Karttaki (verici) kodlanmış bilgilerin okunabilmesi için okuyucunun (alıcı) yakınına yerleştirilir (Okuyucunun doğrudan görüş alanı içinde olması gerekmez.). Okuyucu, elektronların kartın anteninden geçmesini ve ardından çipe güç vermesini sağlayan bir elektromanyetik alan oluşturur. Kartın içindeki güç verilen çip, depolanan bilgiyi başka bir radyo sinyali biçiminde okuyucuya geri göndererek yanıt verir. Görsel 2.48'de RFID alıcı ve verici prensip şeması görülmektedir.



Görsel 2.48: RFID alıcı ve verici prensip şeması

Görsel 2.49'da RC522 RFID alıcı modül ve pin yapısı görülmektedir. RC522 RFID alıcı modülü, RFID etiketleriyle (ISO 14443A standart etiketleri) iletişim kurmak için kullandığı 13,56 MHz'lik bir elektromanyetik alan oluşturmak üzere tasarlanmıştır. Okuyucu, maksimum 10 Mbps veri hızında dört pinli Seri Çevresel Arabirim (SPI) üzerinden bir mikrodenetleyiciyle iletişim kurabilir. Ayrıca I²C ve **UART** protokolleri üzerinden iletişimi destekler.



Görsel 2.49: RFID alıcı modül ve pin yapısı

VCC: 3,3 V besleme. Arduino'nun 3 V çıkışına bağlanır. 5 V pinine bağlamak modüle zarar verir.

RST: Yükselen kenarda modülü sıfırlar.

GND: Topraklama pini.

VCC: 3,3 V besleme. Arduino'nun 3 V çıkışına bağlanır. 5 V pinine bağlamak modüle zarar verir.

RST: Yükselen kenarda modülü sıfırlar.

GND: Topraklama pini.

IRQ: Verici kart yaklaştırıldığında mikrodenetleyiciye uyaran kesme pinidir.

MOSI (Master Out Slave In): RC522 modülüne SPI girişidir.

SCK: SPI için Arduino tarafından sağlanan saat darbelerini kabul eder.

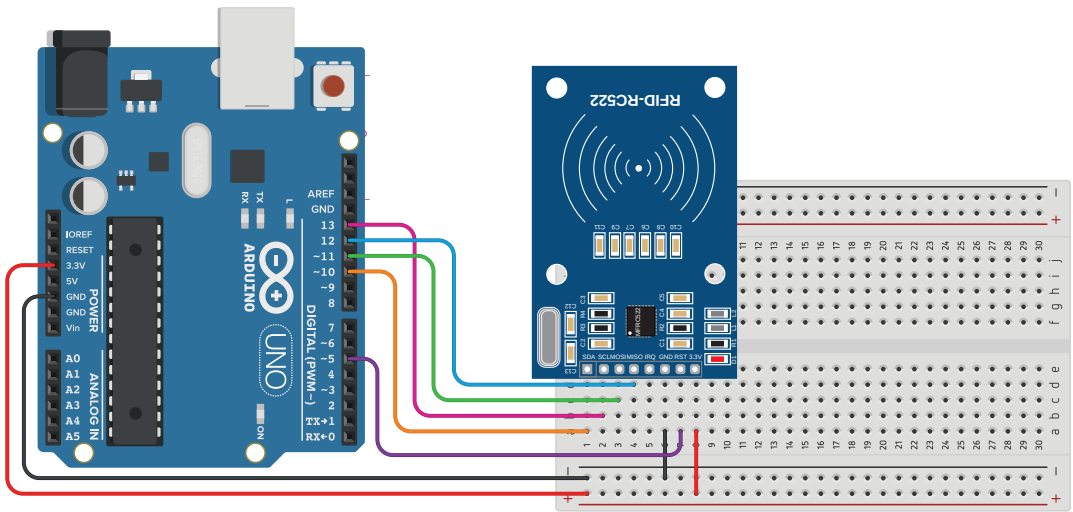
SS / SDA / RX: SPI arayüzü etkinleştirildiğinde sinyal girişi görevi, I^2C arayüzü etkinleştirildiğinde seri veri görevi ve **UART** arayüzü etkinleştirildiğinde seri veri girişi görevi görür.

MİSO / SCL/ TX: SPI arayüzü etkinleştirildiğinde MISO (master in slave out) görevi, I^2C arayüzü etkinleştirildiğinde SCL (serial clock) görevi ve **UART** arayüzü etkinleştirildiğinde seri veri çıkışı görevi görür.

Uygulama Adı: RFID Uygulaması — No.: 13

Amaç: RFID uygulaması yapmak.

Görsel 2.50'deki uygulamada RFID okuyucuya geçerli bir kart okutulduğunda seri ekranda "**geçerli kart**" yazar. Geçersiz kart okutulduğunda kartın ID'sini verir. ID'si programa dâhil edilen kartlar, geçerli kart olur. Kart geçerliyken çalışan kod blokuna istenen kodlar yazılarak istenen kontroller sağlanır.



Görsel 2.50: RFID uygulaması

RFID uygulaması programı aşağıdaki gibidir:

```
#include <SPI.h> // Dahili SPI kütüphanesi
#include <MFRC522.h>  // v1.4.9
#define reset 9 // Reset pini
#define ss 10  // SS pin
MFRC522 kartOkuyucu(ss, reset);  // Kart okuyucu isimli nesne oluşturuldu.
byte i, j, k;
byte kartSayisi = 2; //Kart sayısı doğru girilmelidir.
byte kartlar[][4] = {
  {0xA6, 0x06, 0x1E, 0xF1}, // ilk kartın dört byte'lık ID'si HEX olarak tanımlandı.
  {0xA6, 0x06, 0x1E, 0xF2, // ikinci kart...
};

void setup() {
  Serial.begin(9600);                    //Bilgisayarın seri ekranıyla UART iletişimi.
  SPI.begin();                           // MFRC522 kart okuyucusuyla SPI iletişimi.
  kartOkuyucu.PCD_Init();                // MFRC522'yi başlat
 Serial.println("Lütfen kartı okutunuz.");
}
void loop() {
```

```
  if ( ! kartOkuyucu.PICC_IsNewCardPresent())  //Bir kart okunana kadar bekle.
  return; //Kart yoksa başa dön.
  if ( ! kartOkuyucu.PICC_ReadCardSerial())  // Kartı oku.
  return;
  if (kartKontrol()) { // Kart geçerliyse yapılacaklar...
  Serial.println("Kart geçerli.");
  }
  else { // Kart geçerli değilse yapılacaklar...
  for (i = 0; i < 4; i++) {
   Serial.print(kartOkuyucu.uid.uidByte[i], HEX); //Geçersiz kartın ID'sini ekrana yazdır.
   Serial.print(" ");
  }
  Serial.println(" numaralı kartınız sisteme tanımlı değil!");
  }
  delay(500);
  Serial.println("\nLütfen kartı okutunuz.");
}

boolean kartKontrol() { // Kart geçerliye 1, değilse 0 döndürür.
  k = 0;
  for (j = 0; j < kartSayisi; j++) { // Kayıtlı tüm kartlara bak.
    for (i = 0; i < 4; i++) { // Her kartın ID'si dört byte.
  if (kartlar[j][i] == kartOkuyucu.uid.uidByte[i]) {//Okunan kartı kayıtlı olanlarla karşılaştır.
        k++;
        if (k == 4) { // Tüm byte'lar doğrulandıysa...
          return 1; // 1 döndür.
        }
      }
    }
  }
  return 0; // Okunan kart kayıtlı kartlarla eşleşmediyse 0 döndür.
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Görsel 2.50'deki devreyi kurunuz. Programı yükleyiniz.
3. Seri port ekranını açınız.
4. Kartı okutunuz. Kartın ID'si **hex.** formatında ekranda yazacaktır.
5. Kartın **hex.** kodlarını programdakiyle değiştiriniz.
6. Geçerli kartı okutarak seri ekranı gözlemleyiniz.

DİKKAT D'deki **byte**ları yazdırırken 16'dan küçük sayıları ekrana tek haneli yazar. Örneğin **byte**ın değeri 06 ise 6 şeklinde yazar. ID her **byte** arasına boşluk bırakılmadan yazdırılırsa ID yanlış tanımlanabilir.

SIRA SİZDE

1. Geçerli kart okutulunca servo motorla kapı açan uygulamayı gerçekleştiriniz.

```
#include <SPI.h> // Dahili SPI kütüphanesi
#include <MFRC522.h>  // v1.4.9
#include <Servo.h> // Dahili Servo kütüphanesi v1.1.8

Servo servo; // Servo isimli nesne oluşturuldu.

#define reset 9 // Reset pini
#define ss 10  // SS pin

MFRC522 kartOkuyucu(ss, reset);  // Kart okuyucu isimli nesne oluşturuldu.

byte i, j, k;
byte kartSayisi = 2; //Kart sayısı doğru girilmelidir.
byte kartlar[][4] = {
  {0xA6, 0x06, 0x1E, 0xF1}, // İlk kartın dört byte'lık ID'si HEX olarak tanımlandı.
  {0xA6, 0x06, 0x1E, 0xF2}, // İkinci kart...
};

void setup() {
  servo.attach(9); // Servo motor 9 numaralı PWM çıkışına bağlı.
  Serial.begin(9600);                    //Bilgisayarın seri ekranıyla UART iletişimi.
  SPI.begin();                           // MFRC522 kart okuyucusuyla SPI iletişimi.
  kartOkuyucu.PCD_Init();                // MFRC522'yi başlat
  Serial.println("Lütfen kartı okutunuz.");
}

void loop() {
  if ( ! kartOkuyucu.PICC_IsNewCardPresent())  //Bir kart okunana kadar bekle.
    return;

  if ( ! kartOkuyucu.PICC_ReadCardSerial())  // Kartı oku.
    return;

  if (kartKontrol()) { // Kart geçerliyse yapılacaklar...
    Serial.println("Kart geçerli.");
    kapiAc(); // kapiAc fonksiyonunu çağır.
  }
  else { // Kart geçerli değilse yapılacaklar...
    for (i = 0; i < 4; i++) {

      Serial.print(kartOkuyucu.uid.uidByte[i], HEX); //Geçersiz kartın ID'sini ekrana yazdır.
```

```
    Serial.print(" ");
    }
    Serial.println(" numaralı kartınız sisteme tanımlı değil!");
  }
  delay(500);
  Serial.println("\nLütfen kartı okutunuz.");
}

boolean kartKontrol() { // Kart geçerliye 1, değilse 0 döndürür.
  k = 0;
  for (j = 0; j < kartSayisi; j++) { // Kayıtlı tüm kartlara bak.
    for (i = 0; i < 4; i++) { // Her kartın ID'si dört byte.
      if (kartlar[j][i] == kartOkuyucu.uid.uidByte[i]) {//Okunan kartı kayıtlı olanlarla karşı-
laştır.
        k++;
        if (k == 4) { // Tüm byte'lar doğrulandıysa...
          return 1; // 1 döndür.
        }
      }
    }
  }
  return 0; // Okunan kart kayıtlı kartlarla eşleşmediyse 0 döndür.
}

void kapiAc() {
  servo.write(90); //Servo motoru 90 dereceye getirir.
  delay(3000);
  servo.write(0); //Servo motoru 0 dereceye getirir.
  delay(1000);
}
```

2. Kart bilgilerini EEPROM'da tutan uygulamayı yapınız.

NOT Bu uygulamada ilk okutulan kart, hafızaya yönetici kartı olarak tanımlanır. Kart ekleme ve çıkarma işlemleri için önce yönetici kartı okutulur. Yönetici kartından sonra okutulan kart hafızada kayıtlıysa kart silinir. Hafızada kayıtlı değilse kart hafızaya eklenir.

```
#include <EEPROM.h>
#include <SPI.h> // Dahili SPI kütüphanesi
#include <MFRC522.h>  // v1.4.9

#define reset 9 // Reset pini
#define ss 10  // SS pin

MFRC522 kartOkuyucu(ss, reset);  // Kart okuyucu isimli nesne oluşturuldu.
```

```
byte i, j, k; /*j değişkeni kayıtlı kartların dört byte›lık ID›sinin ilk byte numarasını tutmaktadır. */
byte kartSayisi; //EEPROM hafızanın 0. baytında tutulacak.

void setup() {
  Serial.begin(9600);                    //Bilgisayarın seri ekranıyla UART iletişimi.
  SPI.begin();                           // MFRC522 kart okuyucusuyla SPI iletişimi.
  kartOkuyucu.PCD_Init();                // MFRC522'yi başlat.

  kartSayisi = EEPROM.read(0); // Cihaz yeniden başladığında kart sayısını EEPROM'dan al.

  if (!yoneticiKartKontrol()) { // Yönetici kartı sisteme tanımlı değilse...
    Serial.println("Lütfen yönetici olacak kartı okutunuz.");
    kartOku();
    for (i = 0; i < 4; i++) {
      EEPROM.write(1 + i, kartOkuyucu.uid.uidByte[i]);
    } // Yönetici kart ID'si EEPROM hafızanın 1. - 4. baytlarında tutulacak (İlk 5 bayt doldu.).
    Serial.println("Yönetici kartı hafızaya kaydedildi.");
  }
}
void loop() {
  kartOku();
  if (kartKontrol()) { // Kart geçerliyse yapılacaklar...
    Serial.println("Kart geçerli.");
  }
  else { // Kart geçerli değilse yapılacaklar...
    for (i = 0; i < 4; i++) {
      Serial.print(kartOkuyucu.uid.uidByte[i], HEX); //Geçersiz kartın ID'sini ekrana yazdır.
      Serial.print(" ");
    }
    Serial.println(" numaralı kartınız sisteme tanımlı değil!");
  }
  delay(500);
  Serial.println("\n Lütfen kartı okutunuz.");
}
void kartOku() {
  Serial.println("Lütfen kartı okutunuz.");
  if ( ! kartOkuyucu.PICC_IsNewCardPresent())  //Bir kart okunana kadar bekle.
    return;
  if ( ! kartOkuyucu.PICC_ReadCardSerial())  // Kartı oku.
    return;
}
boolean kartKontrol() { // Kart geçerliye 1, değilse 0 döndürür.
```

```
  if (yoneticiKartKontrol()) // Yönetici kartı okutulmuşsa...
    yonetimIslemleri(); // Kart ekleme ve silme işlemlerine geç.
  else { // Yönetici kartı değilse kart kontrol işlemi yap.
    k = 0;
    for (j = 5; j < EEPROM.length() / 4; j = +4) {
      /* Kayıtlı tüm kartlara bak. UNO hafızasına 1024/4=256 kart tanımlanabilir.
        j değişkeni kayıtlı kartların dört byte›lık ID›sinin ilk byte numarasını tutmaktadır.*/
      for (i = 0; i < 4; i++) { // Her kartın ID'si dört byte.
        if (EEPROM.read(j + i)  == kartOkuyucu.uid.uidByte[i]) { //Okunan kartı kayıtlı olanlar-
la karşılaştır. (5. byte'tan sonra)
          k++;
          if (k == 4) { // Tüm byte'lar doğrulandıysa...
            return 1; // 1 döndür.
          }
        }
      }
    }
    return 0; // Okunan kart kayıtlı kartlarla eşleşmediyse 0 döndür.
  }
}

boolean yoneticiKartKontrol() { // Yönetici kartıysa 1, değilse 0 döndürür.
  k = 0;
  for (i = 1; i < 5; i++) { /* 1. - 4. baytları kontrol et. (0. byteta kart sayısı tutulmaktadır.) */
    if (EEPROM.read(i) == kartOkuyucu.uid.uidByte[i]) { //Okunan kart yönetici kartı ise...
      k++;
      if (k == 4) { // Yönetici kartının tüm byteları doğrulandıysa...
        return 1; // 1 döndür.
      }
   }
  }
  return 0; // Okunan kart yönetici kartı değilse 0 döndür.
}

void yonetimIslemleri() {
  Serial.println("Lütfen eklemek veya silmek istediğiniz kartı okutunuz.");
  kartOku();
  if (kartKontrol)//Kart kayıtlıysa...
    kartSil(); // Kartı sil.
  else //Kart kayıtlı değilse...
    kartEkle(); //Kartı ekle.
}

void kartSil() {
```

```
  for (i = j; i < EEPROM.length() - j - 1; i++)
    EEPROM.write(i, EEPROM.read(i + 4));/*Bulunan kart ID'sinin ilk byte numarasından (j) başla-
yarak tüm hafızayı dört byte sola kaydır.*/
  Serial.println("Okutulan kart hafızadan silindi.");
}

void kartEkle() {
  for (i = 0; i < 4; i++)
    EEPROM.write(kartSayisi * 4 + 1 + i, kartOkuyucu.uid.uidByte[i]); /* Yeni kartı sıradaki
dört bytea yaz. */
  Serial.println("Yeni kart hafızaya kaydedildi.");
}
```

3. Geçerli kart okutulunca step motoru çalıştıran fonksiyonu yazınız.

```
void stepMotor() { // Step motor Yaklaşık 90° döner.
  int x, adimSayisi = 500, sure = 2;

  for (x = 0; x < adimSayisi / 4; x++) {
    for (byte i = 6; i <= 9; i++) { //Sağa dön.
      digitalWrite(i, 1);
      delay(sure);
      digitalWrite(i, 0);
    }
  }
  delay(1000); // Kapı açık kalma süresi.
  for (x = 0; x < adimSayisi / 4; x++) {
    for (byte i = 9; i >= 6; i--) { //Sola dön.
      digitalWrite(i, 1);
      delay(sure);
      digitalWrite(i, 0);
    }
  }
}
```

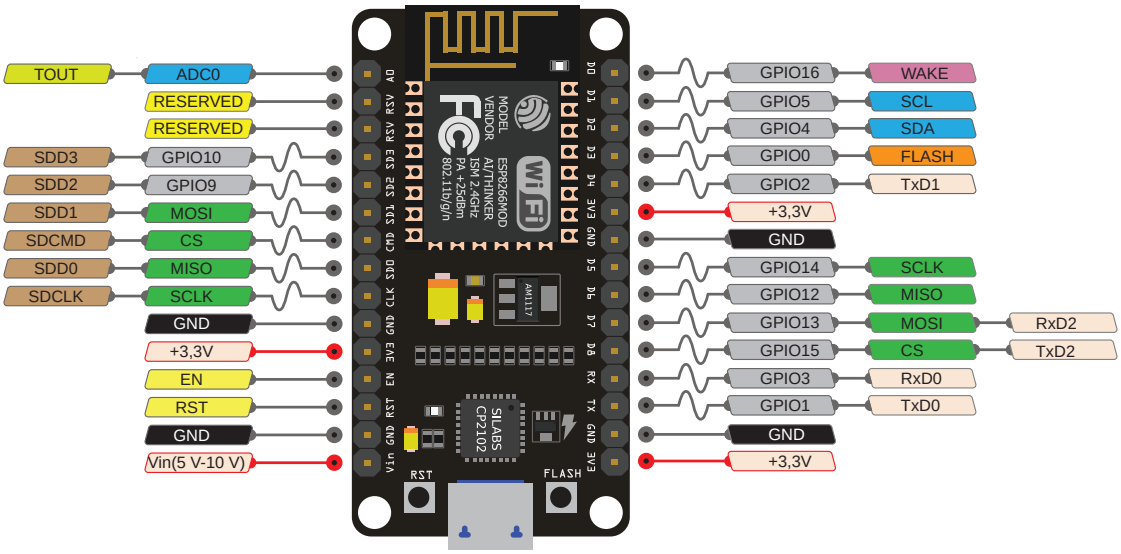
Sorular

1. Kart (verici) içindeki çip enerjisini nereden almaktadır? Belirtiniz.

2. Kart ID'leri string veri tipine dönüştürülüp kontrol edilebilir miydi? Açıklayınız.

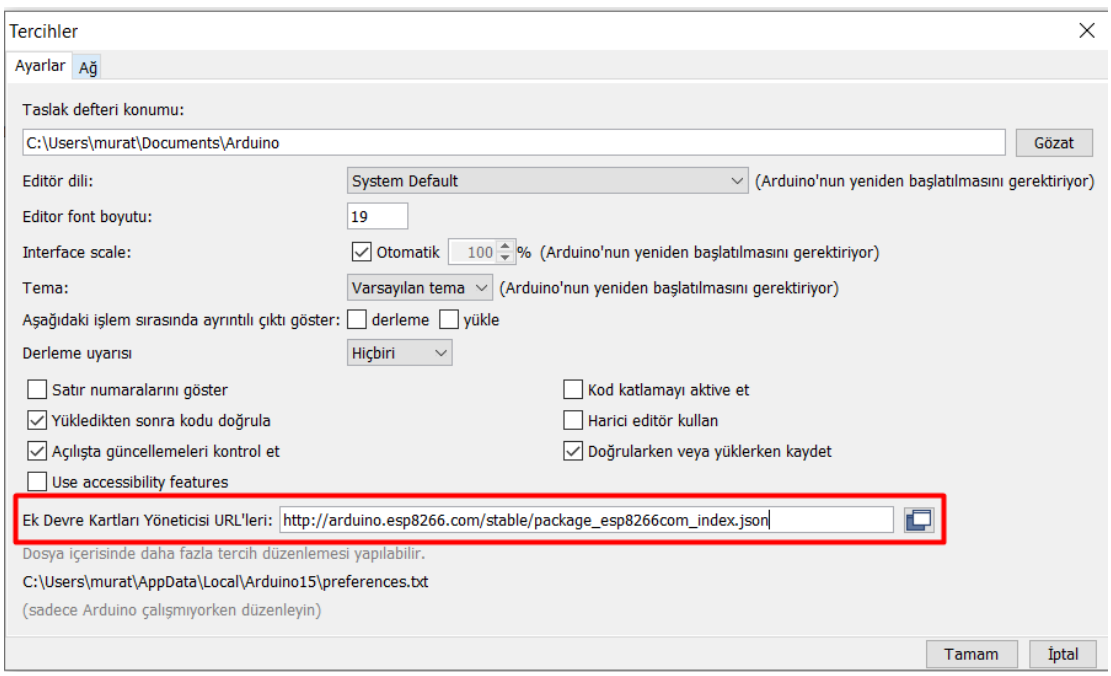
NodeMCU kartını Arduino IDE'ye tanıtma

Görsel 2.51'deki NodeMCU, ESP8266 **Wi-Fi** modülün uygulama kartı şeklidir. Bu kartın Arduino kartından farkı, maliyetinin düşük olması ve **Wi-Fi** modül içerdiğinden, IoT (Internet of Things - Nesnelerin İnterneti) kartı olarak kullanılmasıdır. ESP8266 mikrodenetleyicisini kullanan bu kartın işlemci frekansı 80 MHz, flash hafızası 4 MB ve RAM'i 80 KB'tır.



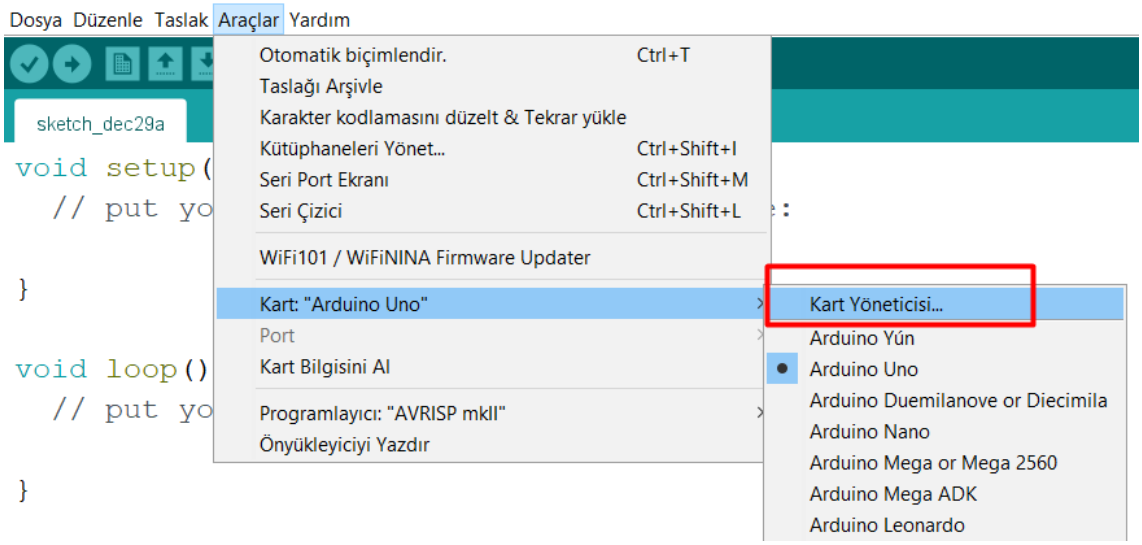
Görsel 2.51: NodeMCU pin yapısı

NodeMCU kartını Arduino IDE'ye tanıtmak için "Dosya → Tercihler" penceresindeki "Ek devre Kartları Yöneticisi URL'leri:" kutusuna http://arduino.esp8266.com/stable/package_esp8266com_index.json adresi yazılarak "Tamam" butonuna tıklanır (Görsel 2.52). Farklı kartları eklemek için (ESP 32 vb.) ilgili kartların bağlantı adresleri virgülle ayrılarak aynı kutuya girilir.



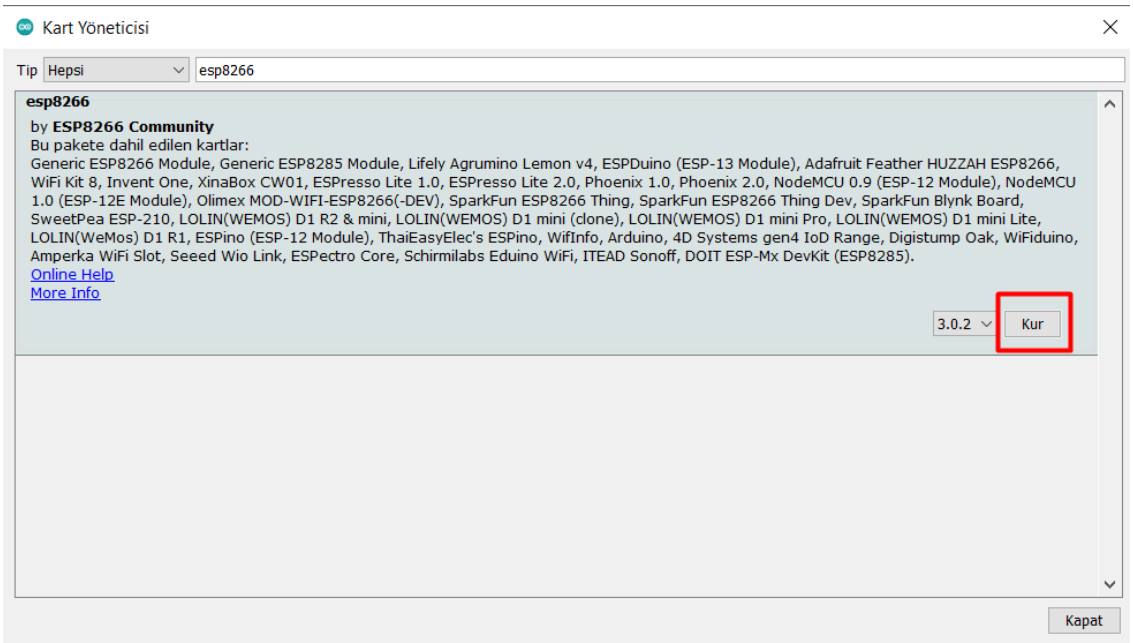
Görsel 2.52: Arduino IDE için ESP8266 tüm kartlar

"Araçlar → Kart: → Kart Yöneticisi" seçilerek kart yöneticisi penceresi açılır (Görsel 2.53).



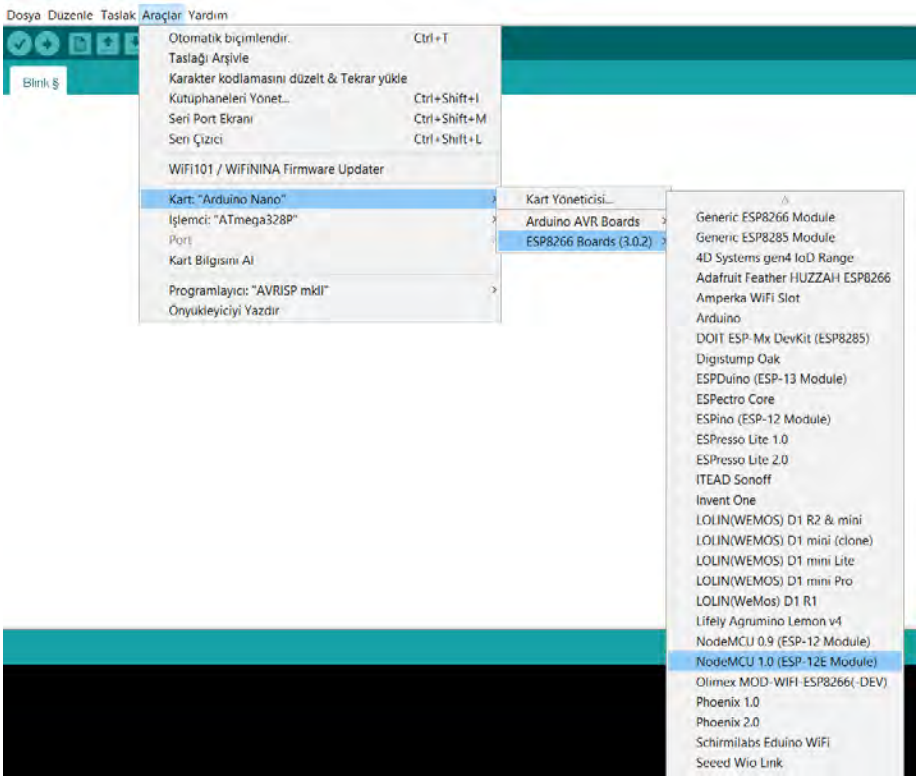
Görsel 2.53: Kart yöneticisi

Arama kutusuna "esp 8266" yazılır. "Kur" düğmesine tıklanarak versiyon 3.0.2 kurulur (Görsel 2.54).



Görsel 2.54: ESP8266 kartını Arduino IDE'ye tanıtma

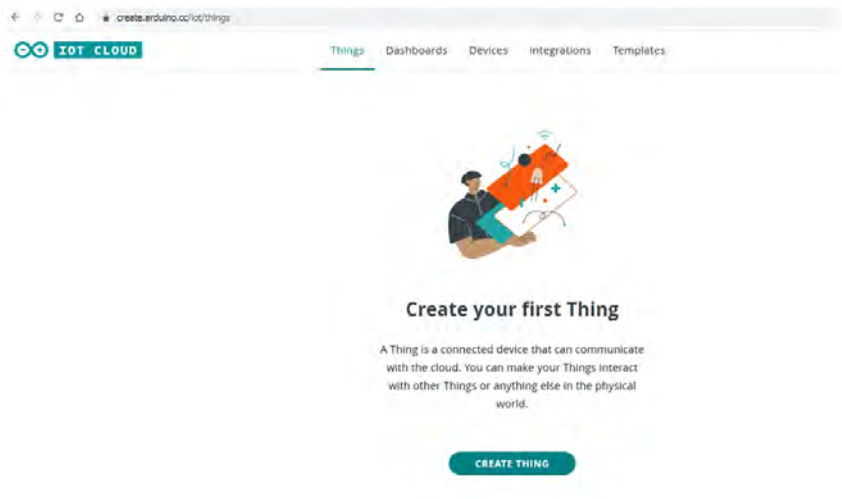
NodeMCU kartını seçmek için "Araçlar → Kart:→ ESP8266 Boards (3.0.2) → NodeMCU 1.0 (ESP-12E Module)" seçilir (Görsel 2.55). İlgili COM port seçilerek yazılan program karta yüklenir.



Görsel 2.55: NodeMCU ESP12E kartı seçimi

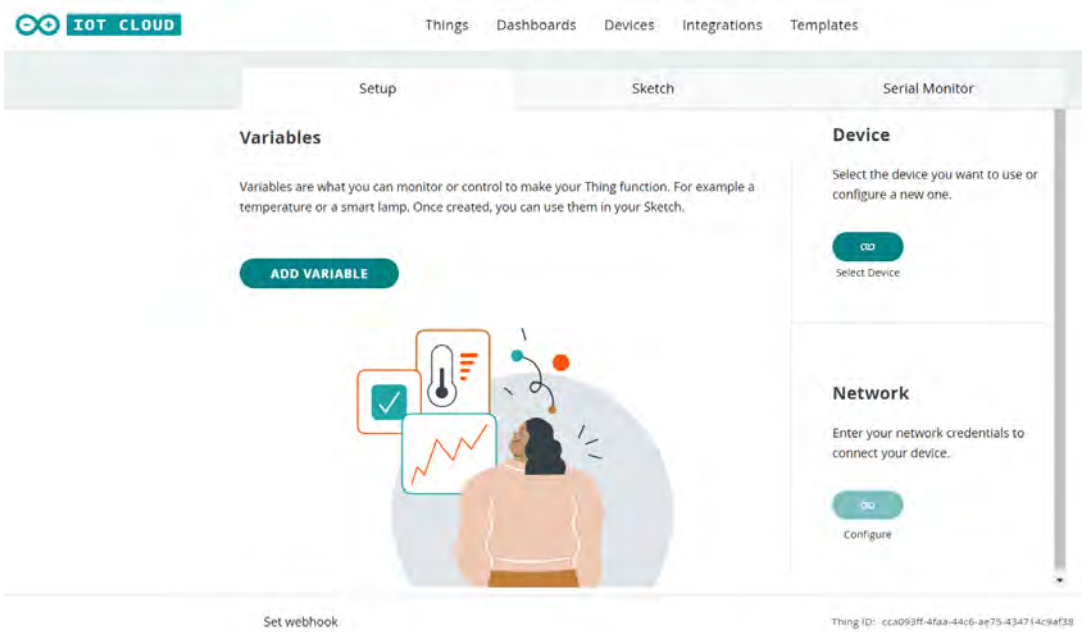
Nesnelerin İnterneti

Arduino IoT Cloud, Arduino tarafından geliştirilen IoT (internet of things) projeleri için tercih edilen bir bulut sistemidir. https://create.arduino.cc/iot adresinden ücretsiz olarak oluşturulabilen Arduino hesabıyla (Google vb. hesaplarla da giriş yapılabilir.) giriş yapılır. Görsel 2.56'daki "CREATE THING" butonuna basılarak yeni internet nesnesi oluşturulur.



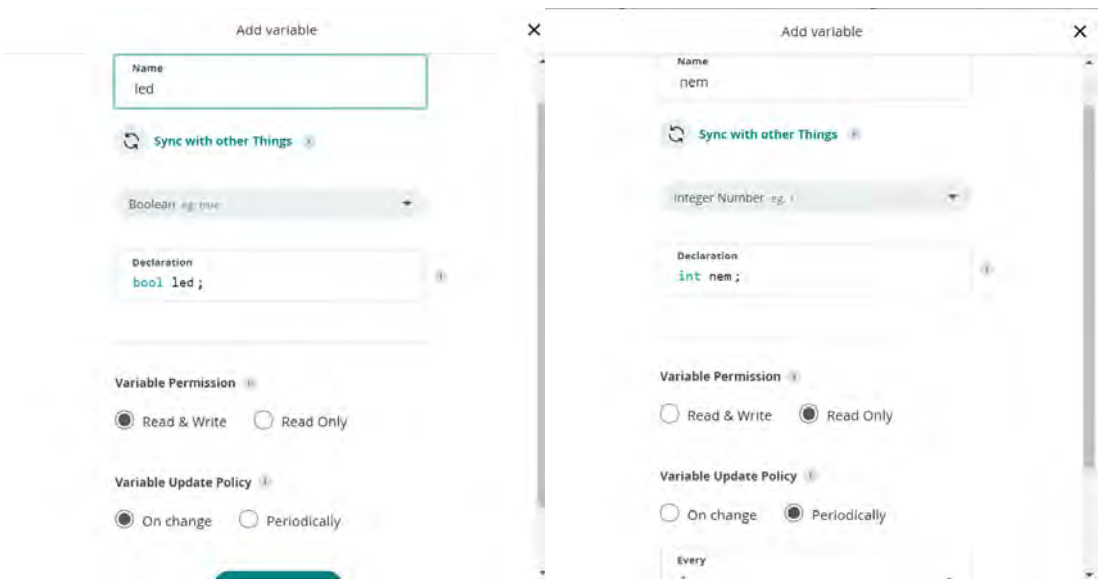
Görsel 2.56: Yeni internet nesnesi oluşturma

Görsel 2.57'de görülen ekrandan değişkenlerin, kullanılan mikrodenetleyici cihaz ve kablosuz ağın ayarları yapılır. "ADD VARIABLE" düğmesine tıklanarak değişkenler oluşturulur.



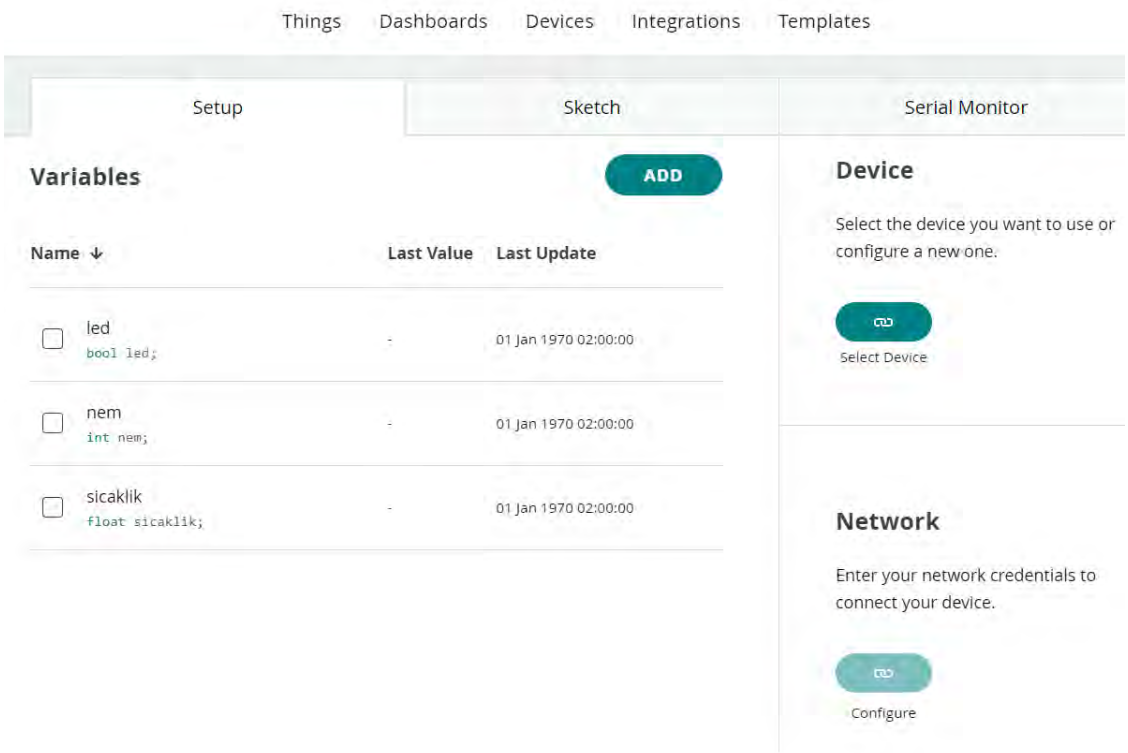
Görsel 2.57: Değişken ekleme

Görsel 2.58'de "led" isminde bool veri tipinde değişken oluşturulmuştur. Burada kullanılan değişken farklı bir nesnedeki aynı tip değişkenle senkronize de edilebilir. Sadece buluta veri gönderilecekse "Read Only", buluttan da mikrodenetleyiciye bilgi gidecekse "Read & Write" izni seçilir. Mikrodenetleyicide değişkenin içeriği değiştiğinde buluttaki bilginin güncellenmesi için "On change", değişkenin içeriği değişmese dahi seçilen süre sonunda buluttaki bilgilerin güncellenmesi için "Periodically" seçilir. "ADD VARIABLE" düğmesi tıklanarak değişken oluşturulur. "nem" ve "sicaklik" değişkenleri de benzer şekilde oluşturulur.

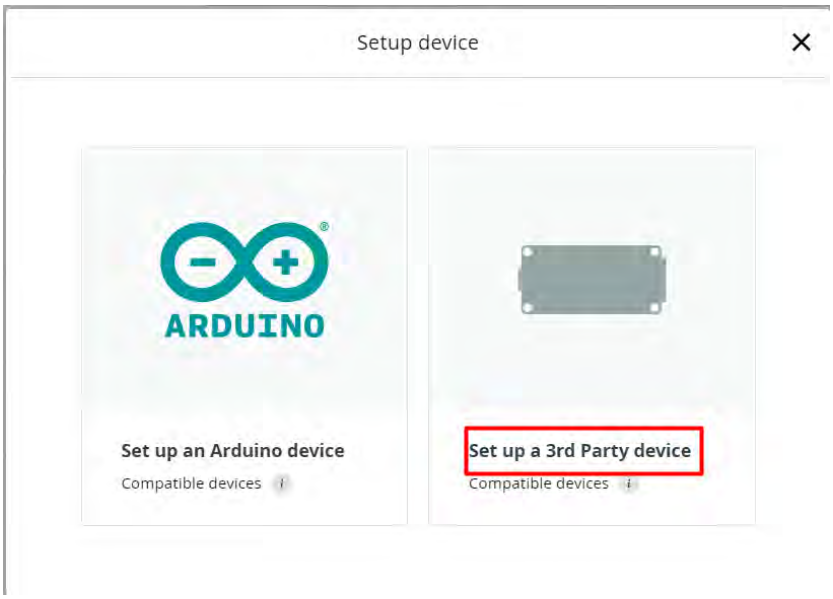


Görsel 2.58: Değişken özellikleri ayarlama

Görsel 2.59'daki "Select Devices" düğmesi tıklanarak Görsel 2.60'tan "Set up 3rd party devices" seçilir.

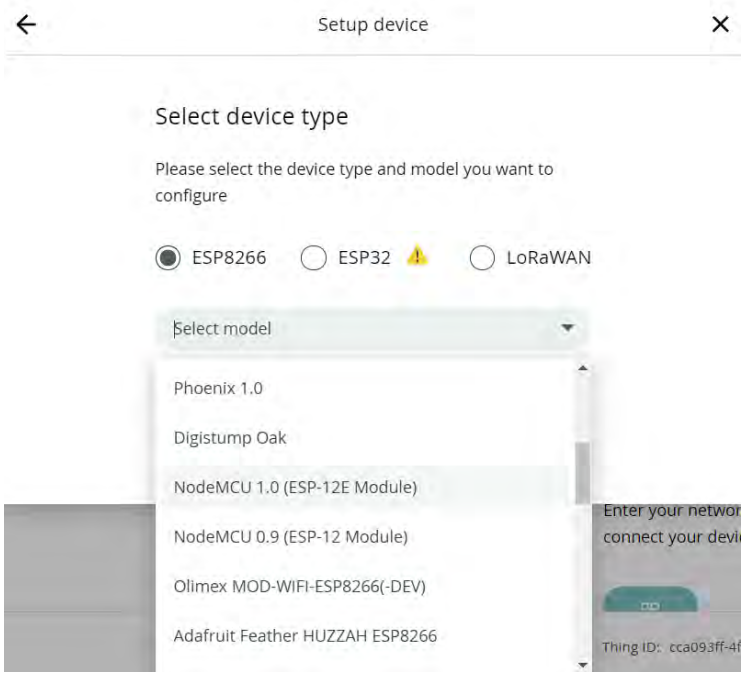


Görsel 2.59: Cihaz seçme



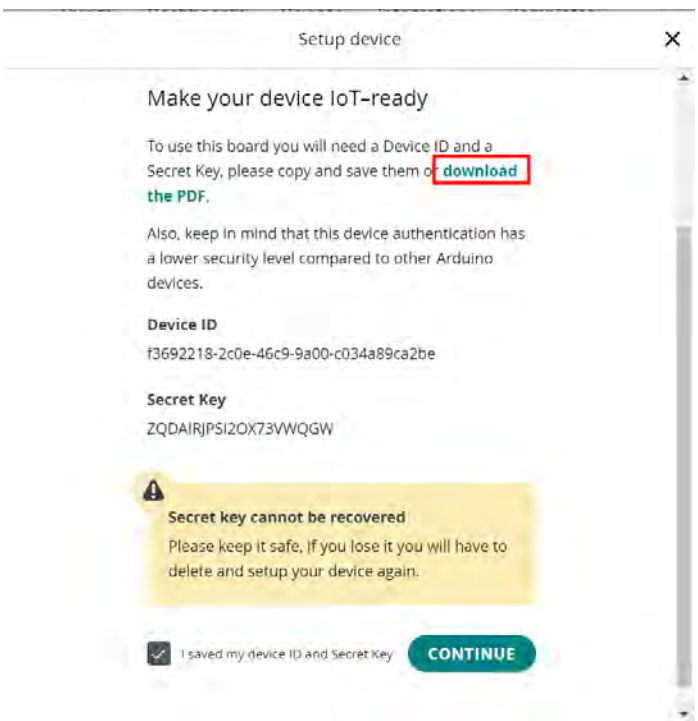
Görsel 2.60: Cihaz ayarları

Görsel 2.61'de görüldüğü gibi ESP8266 seçilerek listeden "NodeMCU 1.0 (ESP-12E Module)" tıklanır. Cihaza bir isim verilerek "next" düğmesine basılır.



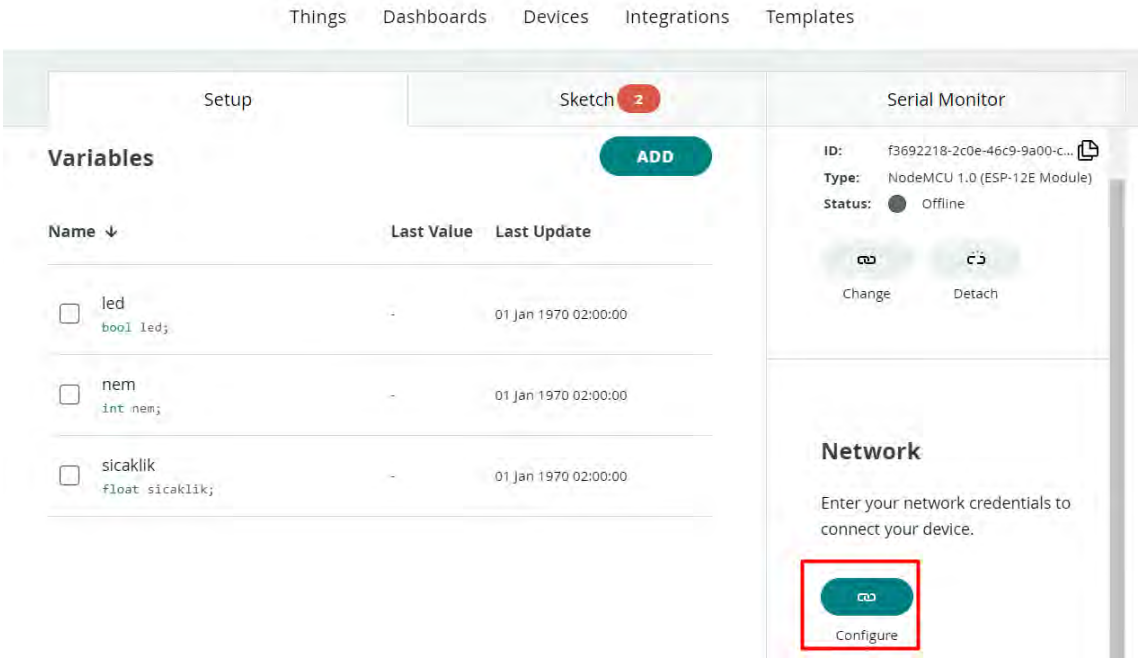
Görsel 2.61: Cihaz tipi seçme

Görsel 2.62'de "download" bağlantısı tıklanarak Device ID ve Secret Key PDF olarak kaydedilir. CONTINUE düğmesiyle cihaz ekleme işlemleri tamamlanır.



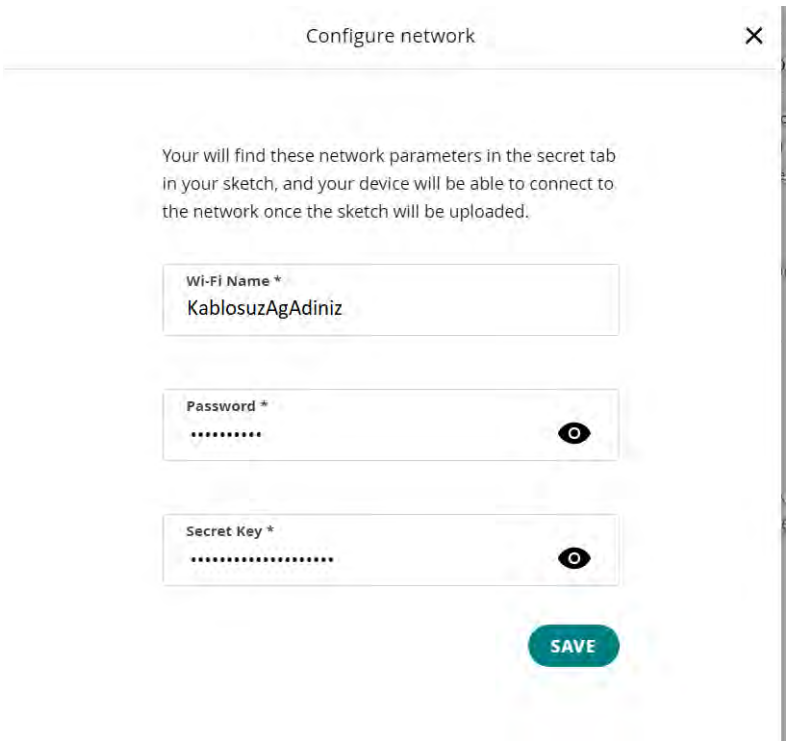
Görsel 2.62: Özel anahtar oluşturma

Wi-Fi ayarlarının yapılması için Görsel 2.63'te görüldüğü gibi Network kısmından Configure düğmesine tıklanır.



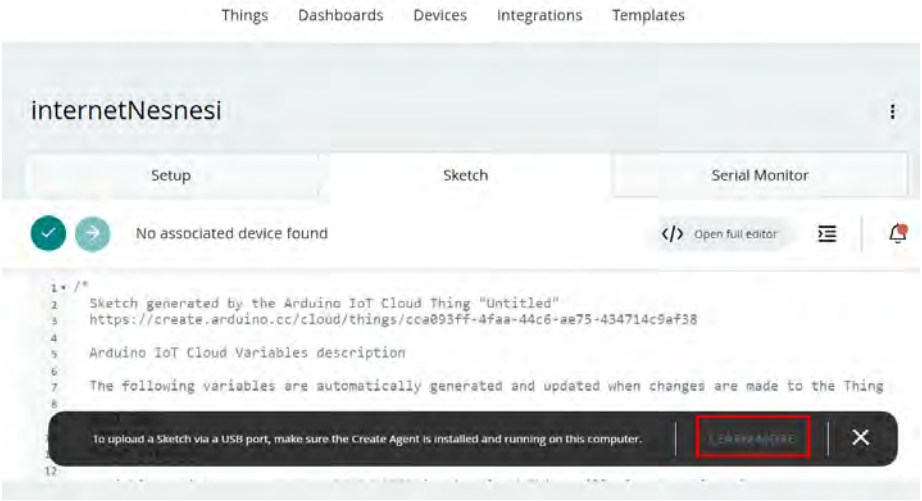
Görsel 2.63: Kablosuz ağ ayarları

Cihazın bağlantı kuracağı kablosuz internetin kullanıcı adı, şifresi ve Görsel 2.62'deki Secret Key girilerek SAVE düğmesi tıklanır (Görsel 2.64).



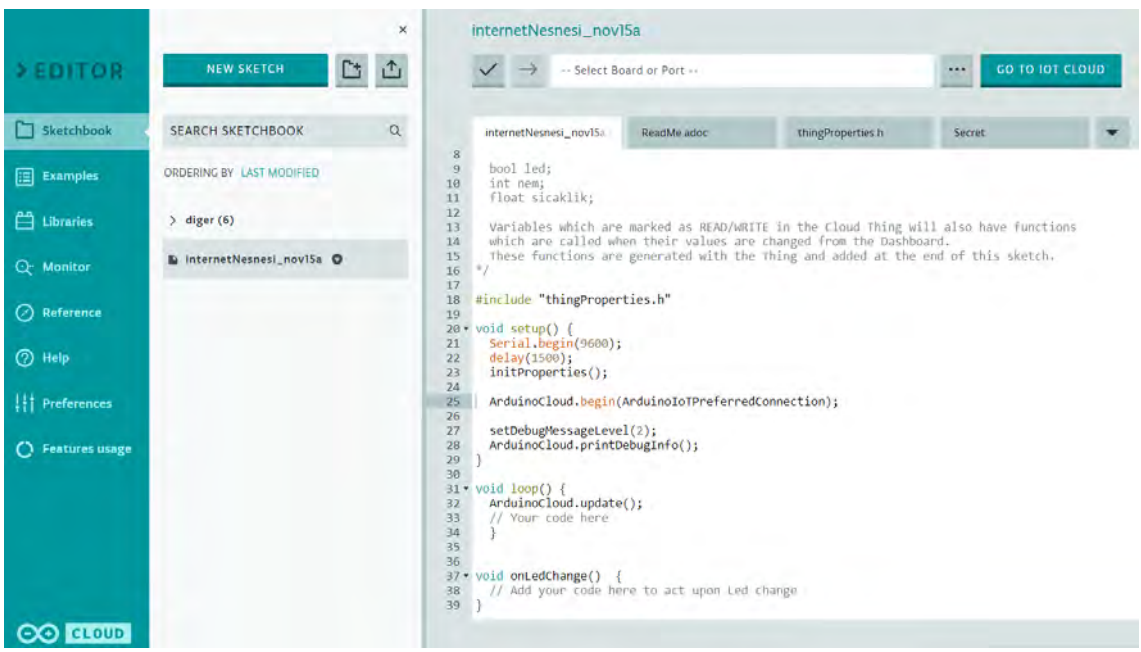
Görsel 2.64: Kablosuz ağ bilgileri

Görsel 2.65'te görüldüğü gibi Sketch sekmesine tıklanır. Cihazı Arduino web editörü üzerinden programlayabilmek için Create Agent eklentisi yüklü olmalıdır. LEARN MORE düğmesine tıklanarak açılan pencereden download bağlantısı tıklanır, Create Agent eklentisi bilgisayara indilir ve kurulur. Create Agent eklentisi bilgisayarda arka planda sürekli çalışır. Open full editör düğmesi tıklanarak Arduino web editörü açılır.



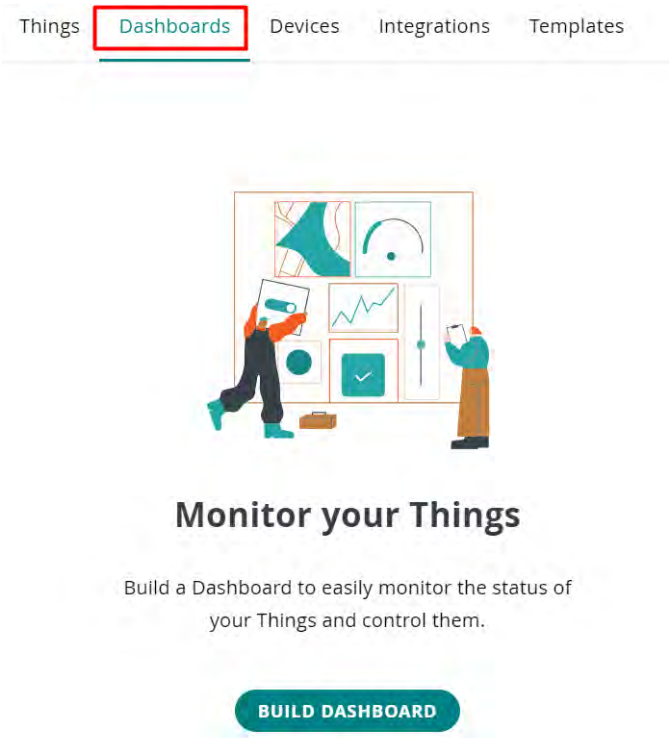
Görsel 2.65: Arduino Create Agent eklentisi yükleme

Görsel 2.66'da Arduino web editörde internet nesnesine ait kodlar hazır gelmektedir. Kullanılacak değişkenler önceden tanımlandığından (thingProperties.h sekmesinde değişkenlerin tanımlandığı görülebilir.) tekrar tanımlamaya gerek yoktur. Değişkenler kullanılarak program loop() fonksiyonu içine yazılır. LED'i yakma butonu read/write olarak tanımlandığından (buluttan cihaza giden bilgi olduğundan) LED'i yakma butonuna ait komutlar void onLedChange() fonksiyonu içine yazılır.



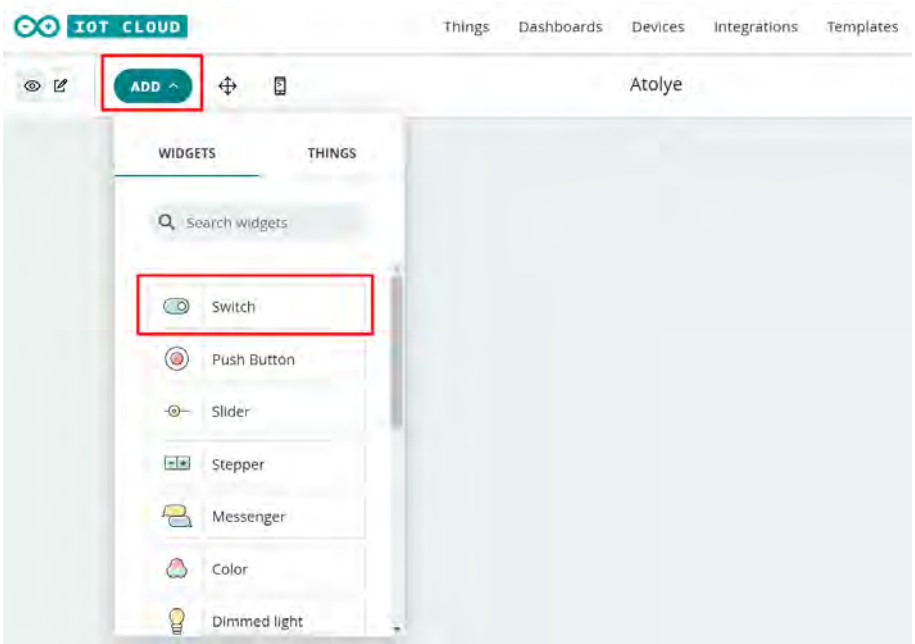
Görsel 2.66: Arduino web editörü

Görsel 2.67'de gösterge panelini oluşturmak için Dashboard sekmesinden "BUILD DASHBOARD" düğmesine tıklanır.



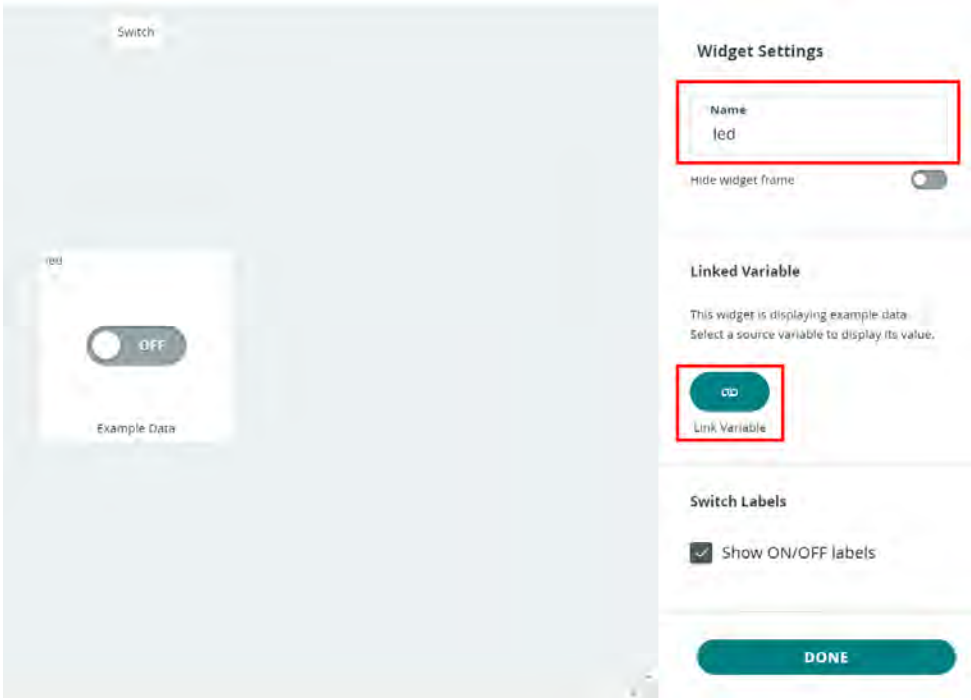
Görsel 2.67: Gösterge paneli oluşturmak

Görsel 2.68'de "ADD" düğmesine tıklanarak switch aracı (widget) seçilir.



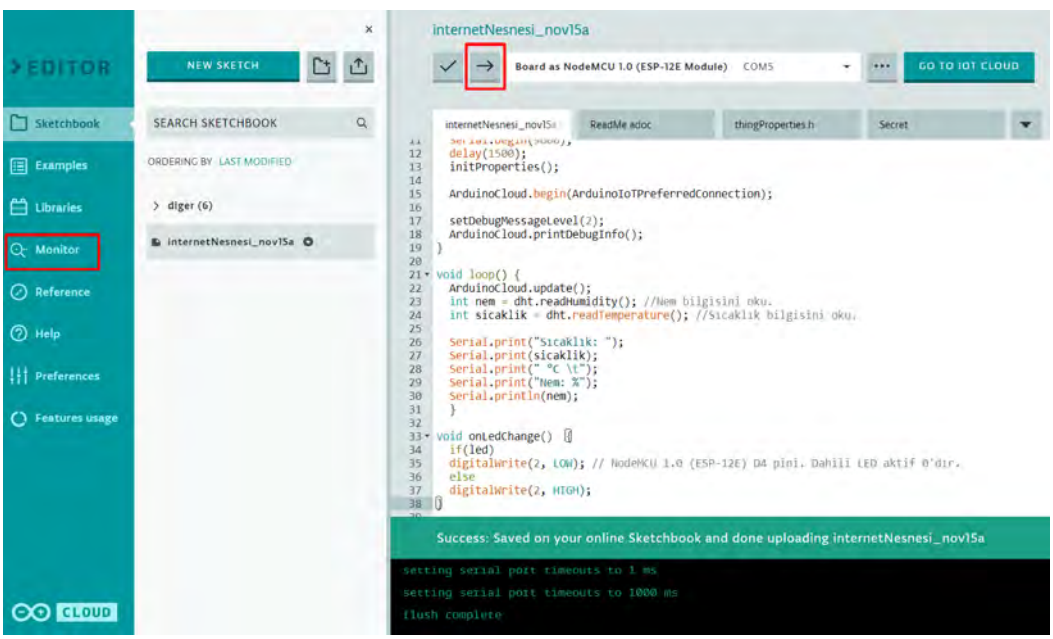
Görsel 2.68: Aracı (widget) seçimi

Görsel 2.69'da görüldüğü gibi switch aracına isim verilebilir. Link Variable düğmesine tıklanarak açılan pencereden önceden tanımlı "led" değişkeni seçilir. "DONE" düğmesiyle işlem tamamlanır. Sıcaklık için **gauge**, nem için **percantage** veya **chart** araçları da benzer şekilde gösterge paneline eklenir.



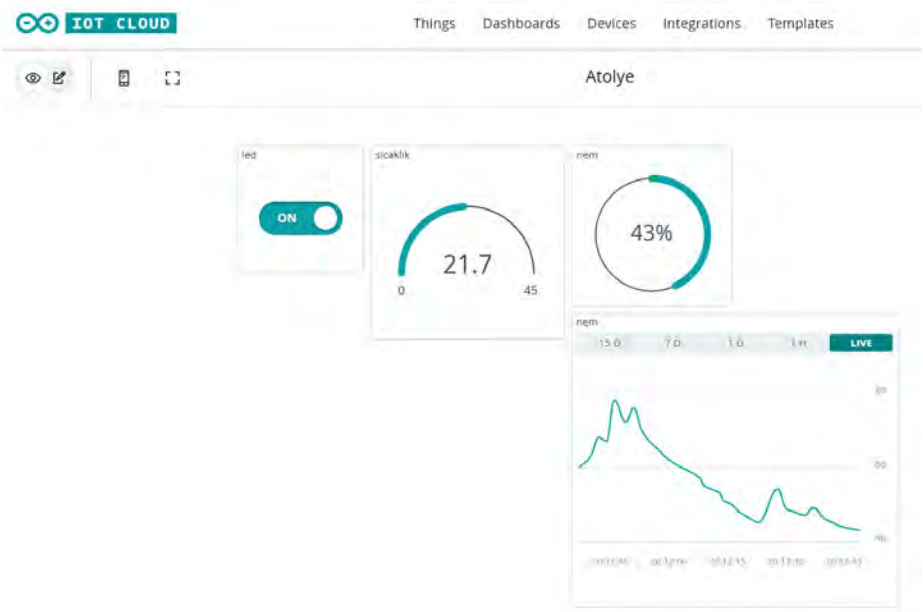
Görsel 2.69: Switch aracına isim verme

Görsel 2.70'te görüldüğü gibi yükleme düğmesiyle program cihaza yüklenir. Monitör kısmından bağlantı çıktıları, sıcaklık ve nem değerleri de ayrıca gözlemlenebilir.



Görsel 2.70: Programı cihaza yükleme

Görsel 2.71'de görüldüğü gibi Dasboards sekmesinden nem ve sıcaklık bilgileri gözlemlenir ve "led" isimli switch aracıyla NodeMCU üzerindeki dâhilî LED yakılıp söndürülür.



Görsel 2.71: Gösterge panelini kullanma

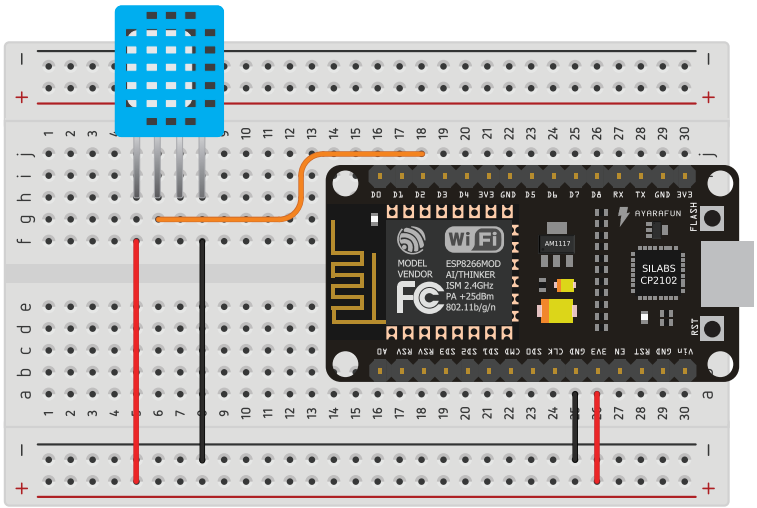
Görsel 2.72'de verildiği gibi gösterge paneline akıllı telefondan https://create.arduino.cc/iot/dashboards/ adresi üzerinden ulaşılarak da aynı işlemler gerçekleştirilir.

Görsel 2.72: Gösterge paneline akıllı telefondan erişim

Uygulama Adı: Wi-Fi Uygulaması — No.: 14

Amaç: Wi-Fi uygulaması yapmak.

Bu uygulamada NodeMCU 1.0 (ESP-12E) kullanılarak Arduino IoT Cloud üzerinden başka bir konumdaki LED'i yakıp söndürme ve ortamın nem ve sıcaklık bilgisini bilgisayar veya akıllı telefon üzerinden görüntüleme yapılmaktadır (Görsel 2.73).



Görsel 2.73: Gösterge paneline akıllı telefondan erişim

Wi-Fi uygulaması programı aşağıdaki gibidir:

```
#include "thingProperties.h"
#include "DHT.h" //Adafruit DHT Sensor Library v1.2.3

#define DHTpin 4     // Sensörün bağlandığı pin. NodeMCU 1.0 (ESP-12E) D2 pini.
#define DHTtipi DHT11   // Sensör tipi: DHT11, DHT21, DHT22.

DHT dht(DHTpin, DHTtipi); //dht isimli nesne oluşturuldu.

void setup() {
  pinMode(2, OUTPUT);     //  GPIO2 pini D4 pini çıkış. (Dahili LED.)
  pinMode(4, INPUT);     //  GPIO4 pini D2 pini giriş.
  dht.begin();
  Serial.begin(9600);
  delay(1500);
  initProperties();

  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}
```

```
void loop() {
  ArduinoCloud.update();
  nem = dht.readHumidity(); //Nem bilgisini oku.
  sicaklik = dht.readTemperature(); //Sıcaklık bilgisini oku.
  Serial.print("Sıcaklık: ");
  Serial.print(sicaklik);
  Serial.print(" °C \t");
  Serial.print("Nem: %");
  Serial.println(nem);
}
void onLedChange()  {
  if (led)
  digitalWrite(2, LOW); // NodeMCU 1.0 (ESP-12E) D4 pini. Dahili LED aktif 0'dır.
  else
  digitalWrite(2, HIGH);
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.73'teki devreyi kurunuz.
4. https://create.arduino.cc/iot adresinden internet nesnesini oluşturunuz.
5. Arduino web editöründen (https://create.arduino.cc/editor/) nodeMCU kartını programlayınız.
6. https://create.arduino.cc/iot/dashboards/ adresinden göstergeleri izleyerek LED'i internet üzerinden kontrol ediniz.
7. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Röle modülü kullanarak 220 V lamba veya motor kontrol eden uygulamayı gerçekleştiriniz.

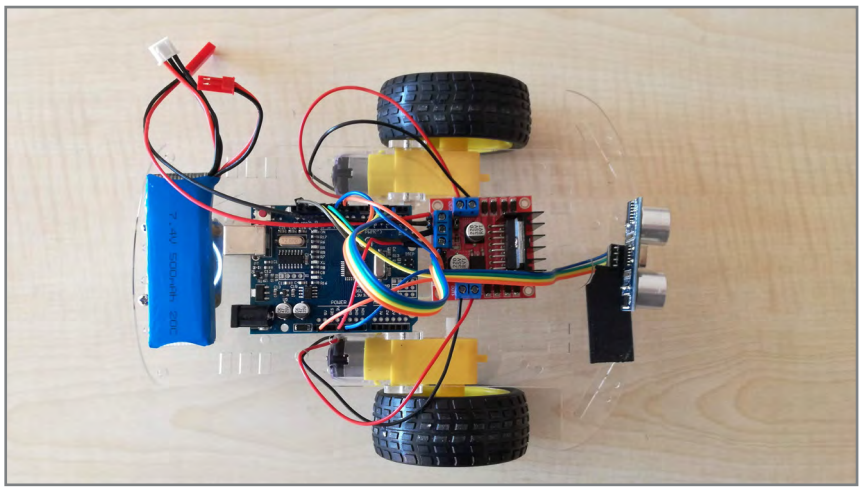
Sorular

1. Yapılan bu uygulamada Arduino web editöre neden ihtiyaç duyulmaktadır? Belirtiniz.
2. Arduino IOT uygulaması için google firebase ve php web server kullanımını araştırınız.

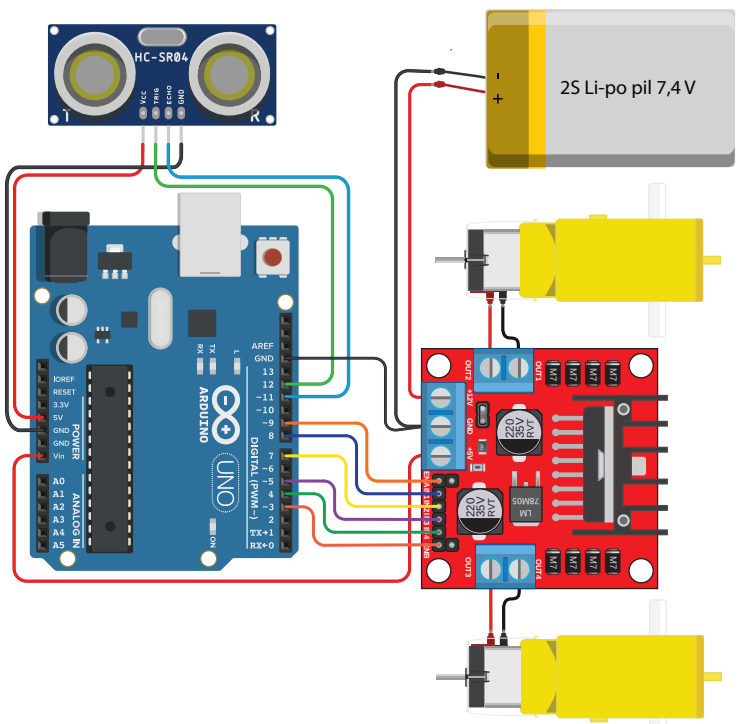
Uygulama Adı: Engelden Kaçan Robot Uygulaması No.: 15

Amaç: Engelden kaçan robot uygulaması yapmak.

Engelden kaçan robotta engelleri algılamak için ultrasonik sensör kullanılmaktadır (Görsel 2.74). Ultrasonik sensörden gelen bilgilerle mesafe hesaplanır. Mesafe 30 cm'nin altına indiğinde motor tarafından sürücüye sol tekerleği geriye döndürme bilgisi gönderilmektedir. 30 cm üzerindeki tüm mesafe ölçümlerinde tekerlekler ileri yönde döndürülecek şekilde motor sürücüye bilgi gönderilmektedir. Taslaktaki "hiz=255;" değeri düz gitme ve dönüş için farklı değerlere ayarlanarak robot yavaşlatılabilir. Görsel 2.75'te engelden kaçan robot uygulama devresi görülmektedir.



Görsel 2.74: Engelden kaçan robot



Görsel 2.75: Engelden kaçan robot uygulaması

Engelden kaçan robot uygulaması programı aşağıdaki gibidir:

```
const byte IN1 = 8, IN2 = 7, IN3 = 5, IN4 = 4; // Motor sürücü yön pinleri
const byte enA = 9, enB = 3; // Motor sürücü PWM pinleri
const byte echo = 11, trig = 12; // Ultrasonik sensör pinleri

byte hiz = 255; // PWM.
unsigned long sure;
int uzaklik;

void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
}

void loop() {
  digitalWrite(trig, 0); // Darbe sinyali gönder.
  delayMicroseconds(10);
  digitalWrite(trig, 1);
  delayMicroseconds(10);

  sure = pulseInLong(echo, 1); // Gelen sinyalin HIGH'da kalma süresini oku.
  uzaklik = (0.0343 * sure) / 2; // Mesafeyi hesapla.

  if (uzaklik < 30) { // Uzaklık 30 cm'den yakınsa sola dön.
    digitalWrite(IN1, 0);
    digitalWrite(IN2, 1);
    analogWrite(enA, hiz);
    analogWrite(enB, hiz);
  }
  else { // Uzaklık 30 cm'den yakın değilse düz git.
    digitalWrite(IN1, 1);
    digitalWrite(IN2, 0);
    digitalWrite(IN3, 1);
    digitalWrite(IN4, 0);
    analogWrite(enA, hiz);
    analogWrite(enB, hiz);
  }
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.75'teki devreyi kurunuz. Programı yükleyiniz.
4. Robotu zemine koyarak robotun çalışmasını gözlemleyiniz.
5. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Motor sürücünün yön girişlerinde (IN1, IN2, IN3, IN4) analogWrite() fonksiyonunu kullanarak devreyi yeniden düzenleyiniz (ENA ve ENB **jumperl**arı takılı kalır.).

```
const byte IN1 = 10, IN2 = 9, IN3 = 6, IN4 = 5; // PWM pinleri. (enA ve enB'de jumper takılı).
const byte echo = 11, trig = 12; // Ultrasonik sensör pinleri

byte hiz = 70; // PWM.
unsigned long sure;
int uzaklik;

void setup() {
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
}

void loop() {
  digitalWrite(trig, 0); // Darbe sinyali gönder.
  delayMicroseconds(10);
  digitalWrite(trig, 1);
  delayMicroseconds(10);

  sure = pulseInLong(echo, 1); // Gelen sinyalin HIGH'da kalma süresini oku.
  uzaklik = (0.0343 * sure) / 2; // Mesafeyi hesapla.

  if (uzaklik < 30) { // Uzaklık 30 cm'den yakınsa sola dön.
    analogWrite(IN1, 0);
    analogWrite(IN2, hiz);
    analogWrite(IN3, hiz);
    analogWrite(IN4, 0);
  }
  else { // Uzaklık 30 cm'den yakın değilse düz git.
    analogWrite(IN1, hiz);
    analogWrite(IN2, 0);
    analogWrite(IN3, hiz);
    analogWrite(IN4, 0);
```

```
  }
}
```

2. Üç adet ultrasonik sensörü ön tarafa 15° lik açılarla yerleştirerek uygulamayı gerçekleştiriniz.

```
const byte IN1 = 10, IN2 = 9, IN3 = 6, IN4 = 5; // PWM pinleri. (enA ve enB'de jumper takılı).
const byte echoSag = 3, trigSag = 4; // Sağ ultrasonik sensör pinleri
const byte echoOrta = 7, trigOrta = 8; // Orta ultrasonik sensör pinleri
const byte echoSol = 12, trigSol = 13; // Sol ultrasonik sensör pinleri

byte hiz = 100; // PWM.
unsigned long sure;
int uzaklik;

void setup() {
  pinMode(trigSag, OUTPUT);
  pinMode(echoSag, INPUT);
  pinMode(trigOrta, OUTPUT);
  pinMode(echoOrta, INPUT);
  pinMode(trigSol, OUTPUT);
  pinMode(echoSol, INPUT);
}

void loop() {

  byte uzaklikSag = mesafe(trigSag, echoSag);
  byte uzaklikOrta = mesafe(trigOrta, echoOrta);
  byte uzaklikSol = mesafe(trigSol, echoSol);

  if (uzaklikSag < 50)
    sag();

  if (uzaklikSol < 50)
    sol();


  if (uzaklikOrta < 30) { // Uzaklık 30 cm'den yakınsa dur, geri git, sola dön.
    dur();
    delay(1000);
    geri();
    delay(300);
    sol();
    delay(400);
  }
  else  ileri(); // Uzaklık 30 cm'den yakın değilse düz git.
}
byte mesafe(byte trig, byte echo) {
```

```
  digitalWrite(trig, 0); // Darbe sinyali gönder.
  delayMicroseconds(10);
  digitalWrite(trig, 1);
  delayMicroseconds(10);

  sure = pulseInLong(echo, 1); // Gelen sinyalin HIGH'da kalma süresini oku.
 uzaklik = (0.0343 * sure) / 2; // Mesafeyi hesapla.
  return uzaklik;
}
void ileri() {
  analogWrite(IN1, hiz);
  analogWrite(IN2, 0);
  analogWrite(IN3, hiz);
  analogWrite(IN4, 0);
}
void geri() {
  analogWrite(IN1, 0);
  analogWrite(IN2, hiz);
  analogWrite(IN3, 0);
  analogWrite(IN4, hiz);
}
void sag() {
  analogWrite(IN1, hiz);
  analogWrite(IN2, 0);
  analogWrite(IN3, 0);
  analogWrite(IN4, 0);
}
void sol() {
  analogWrite(IN1, 0);
  analogWrite(IN2, 0);
  analogWrite(IN3, hiz);
  analogWrite(IN4, 0);
}
void dur() {
  analogWrite(IN1,  0);
  analogWrite(IN2, 0);
  analogWrite(IN3, 0);
  analogWrite(IN4, 0);
}
```

Sorular

1. İleri ve dönüş hızları farklı ayarlanabilir mi? Belirtiniz.
2. Motor sürücünün yön girişlerinde analogWrite() fonskiyonu kullanıldığında IN1, IN2, IN3, IN4 uçlarını çıkış olarak ayarlamak gerekir mi? Nedenini açıklayınız.

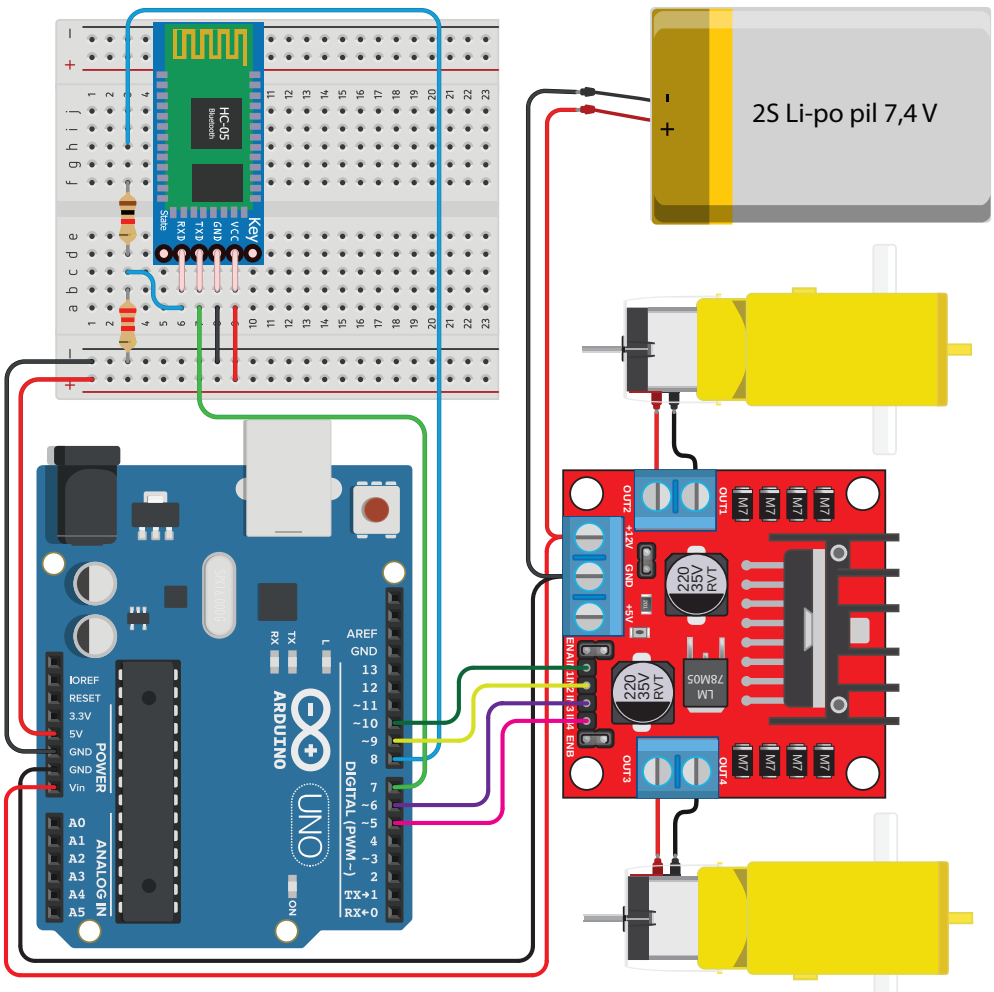
Uygulama Adı: Bluetooth Kontrollü Araç Uygulaması No.: 16

Amaç: Bluetooth kontrollü araç uygulaması yapmak.

Görsel 2.76'da bluetooth modülün RX ve TX pinlerinin SoftwareSerial.h kütüphanesi kullanılarak 8 ve 7 numaralı pinlere bağlanması gösterilmiştir. Modülün TX ucu Arduino'nun RX (8. pin) ucuna, modülün RX ucu Arduino'nun TX (7. pin) ucuna bağlanmalıdır. Ancak modülün TX ucundan çıkan 3,3 V Arduino RX ucunda lojik 1 olarak kabul edilirken Arduino'nun TX ucundan çıkan 5 V modülün RX ucuna doğrudan uygulandığında bozulmasına neden olacaktır. Bu nedenle Arduino'nun TX ucundan çıkan 5 V'u 3,3 V'a düşürmek için gerilim bölücü dirençler kullanılmalıdır.

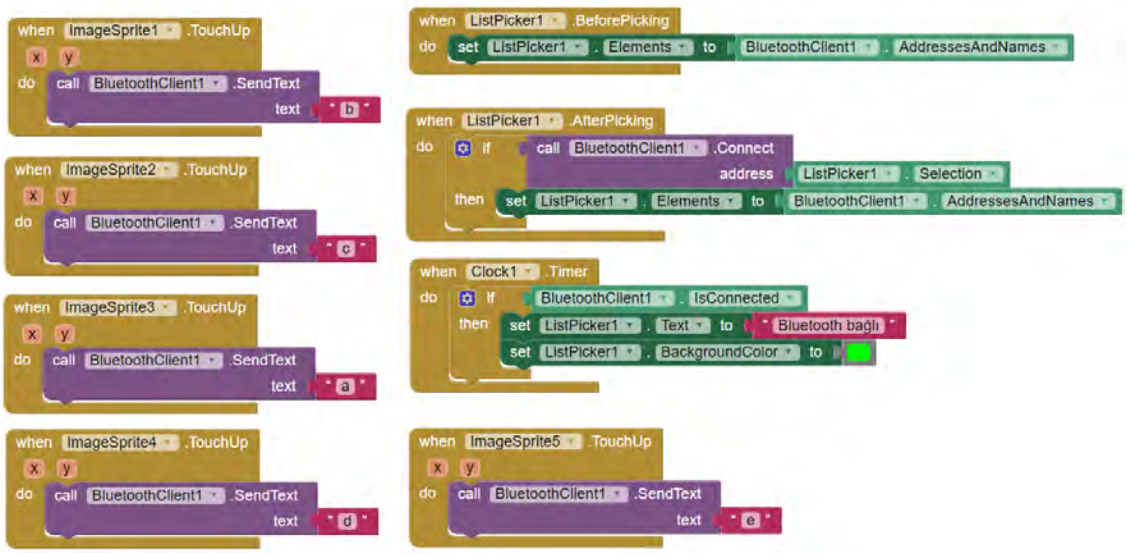
Motor sürücüde ENA ve ENB uçlarındaki **jumper** takılı bırakılarak diğer dört girişten analogWrite fonksiyonuyla PWM verilmiştir.

Devreye enerji verildiğinde bluetooth modülle haberleşmek için Android telefondan bluetooth bağlantısı açılarak hc-05 cihazı bulunur. Varsayılan şifre "1234" veya "0000"dır. Android telefondan veri gönderebilmek için **play store**dan "Arduino bluetooth controller" benzeri uygulamalar kullanılır (Görsel 2.77) veya MIT app invertor ile akıllı telefon uygulaması yapılır (Görsel 2.78). Telefondan gönderilen karakterlere göre aracın yönü belirlenir.



Görsel 2.76: Bluetooth kontrollü araç uygulaması

Görsel 2.77: Bluetooth araç Android uygulaması



Görsel 2.78: Akıllı telefon uygulaması MIT app invertor kodları

Bluetooth kontrollü araç uygulaması programı aşağıdaki gibidir:

```
#include <SoftwareSerial.h> // Arduino IDE ile gelir. Kurulum gerektirmez.
SoftwareSerial bluetooth(7, 8); // RX, TX (Arduino tarafında).
const byte IN1 = 10, IN2 = 9, IN3 = 6, IN4 = 5; // PWM pinleri. (enA ve enB'de jumper takılı).
char karakter;
byte hiz = 255; // Başlangıç hızı.

void setup() {
  bluetooth.begin(9600); // Bluetooth iletişimi başlat.
}
void loop() {
  if (bluetooth.available()) { // Veri geliyorsa...
    karakter = bluetooth.read(); // Gelen karakteri karakter değişkenine ata.
    if (karakter == 'a') { // İleri.
      analogWrite(IN1, hiz);
      analogWrite(IN2, 0);
```

```
      analogWrite(IN3, hiz);
      analogWrite(IN4, 0);
    } else if (karakter == 'b') { // Geri.
      analogWrite(IN1, 0);
      analogWrite(IN2, hiz);
      analogWrite(IN3, 0);
      analogWrite(IN4, hiz);
    } else if (karakter == 'c') { // Sol.
      analogWrite(IN1, hiz);
      analogWrite(IN2, 0);
      analogWrite(IN3, 0);
      analogWrite(IN4, hiz);
    } else if (karakter == 'd') { // Sağ.
      analogWrite(IN1, 0);
      analogWrite(IN2, hiz);
      analogWrite(IN3, hiz);
      analogWrite(IN4, 0);
    } else if (karakter == 'e') { // Dur.
      analogWrite(IN1, 0);
      analogWrite(IN2, 0);
      analogWrite(IN3, 0);
      analogWrite(IN4, 0);
    }
  }
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.76'daki devreyi kurunuz. Programı yükleyiniz.
4. Telefondan bluetooth açarak modülle eşleştiriniz.
5. Telefondan uygulamayla a, b, c, d, e karakterleri göndererek aracı yönlendiriniz.
6. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

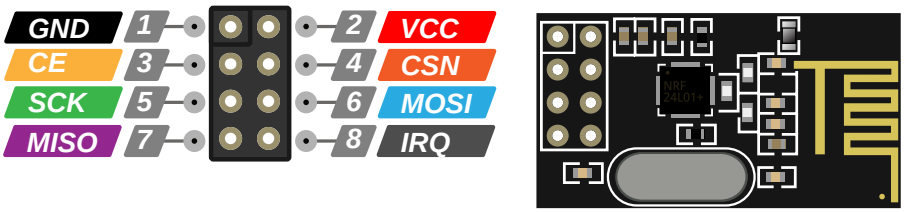
1. Bluetooth araçta, telefondan **"k"** harfi gönderildiğinde korna çalan (**buzzer**), **"f"** harfi gönderildiğinde farları yakan (LED) uygulamayı gerçekleştiriniz.

Soru

1. **UART** iletişim için 0 ve 1 numaralı pinler neden tercih edilmemiştir? Açıklayınız.

Kablosuz Haberleşme

Görsel 2.79'da verilen nRF24L01+ alıcı-verici modülü, 2,4 GHz ISM frekans bandında çalışmak üzere tasarlanmıştır. Modül, veri iletimi için GFSK modülasyonunu kullanır. Veri aktarım hızı 250 kbps, 1 Mbps ve 2 Mbps olabilir. nRF24L01+ alıcı-verici modülü, maksimum 10 Mbps veri hızıyla dört pinli Seri Çevresel Arabirim (SPI) üzerinden iletişim kurar. Antensiz modülün mesafesi 100 m iken antenli modül 1000 m mesafeden iletişim kurabilir.



Görsel 2.79: nRF24L01+ pin yapısı

VCC: Topraklama pini.

GND: 3,3 V besleme. Bu modüle uygun 5 V adaptör kullanıldığında 5 V verilebilir. Adaptör kullanılmadan 5 V verilirse modül bozulur.

CE (Chip Enable): Lojik 1 yapıldığında nRF24L01, o anda hangi modda olduğuna bağlı olarak ya gönderir ya da alır.

CSN (Chip Select Not): Lojik 0 yapıldığında nRF24L01 veri için SPI portunu dinlemeye başlar.

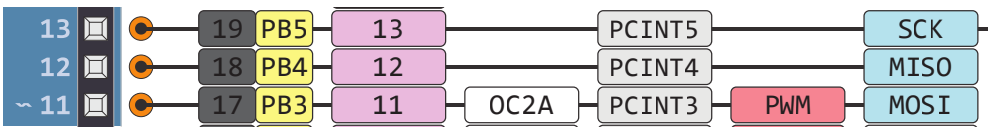
SCK (Serial Clock): SPI bus **master** tarafından sağlanan saat darbelerini kabul eder.

MOSI (Master Out Slave In): nRF24L01'e SPI girişidir.

MISO (Master In Slave Out): nRF24L01'den SPI çıkışıdır.

IRQ: İşlenecek yeni veriler olduğunda **master** cihazı uyaran bir kesme pinidir.

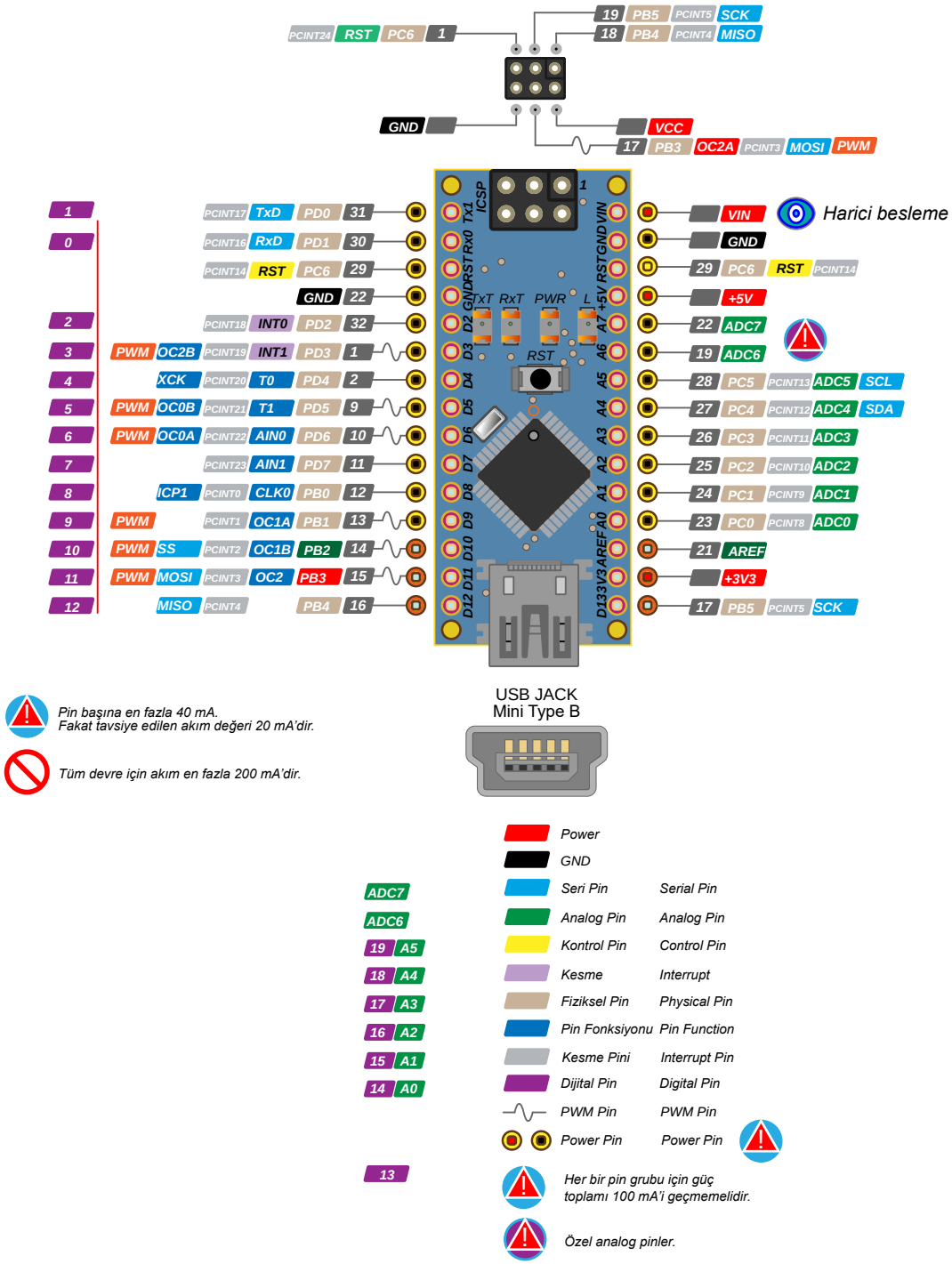
Arduino Uno ve Nano'da SPI haberleşme pinleri, 11-MOSI, 12-MISO ve 13-SCK pinleridir (Görsel 2.80).



Görsel 2.80: Arduino Uno ve Nano'da SPI haberleşme pin numaraları

Arduino Nano pin yapısı

Verici uygulamalarında küçük boyutundan dolayı Arduino Nano tercih edilmiştir. Görsel 2.81'de Arduino Nano pin yapısı verilmiştir.

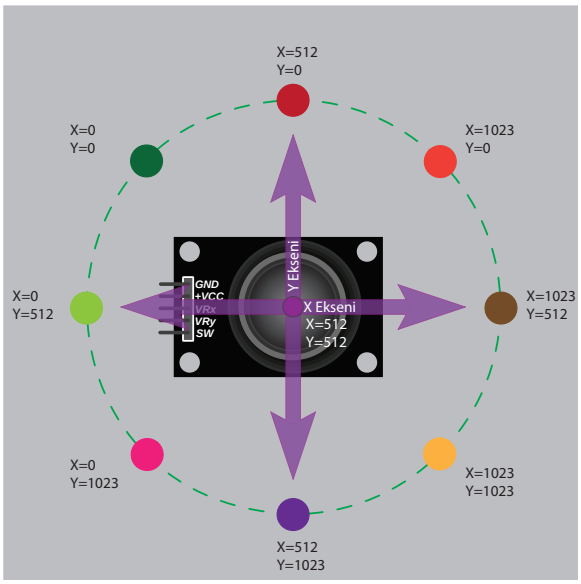


Görsel 2.81: Arduino Nano pin yapısı

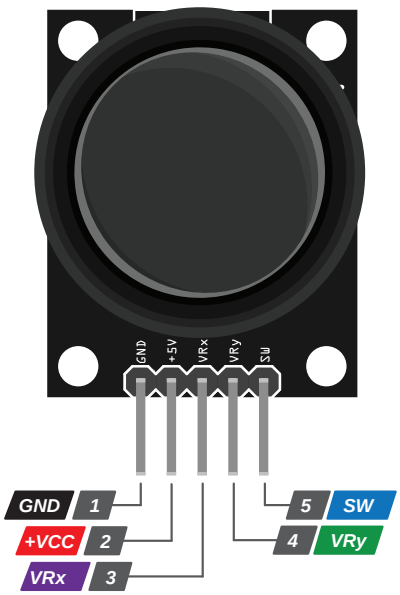
Uygulama Adı: Uzaktan Kontrollü Araba Uygulaması No.: 17

Amaç: Uzaktan kontrollü araba uygulaması yapmak.

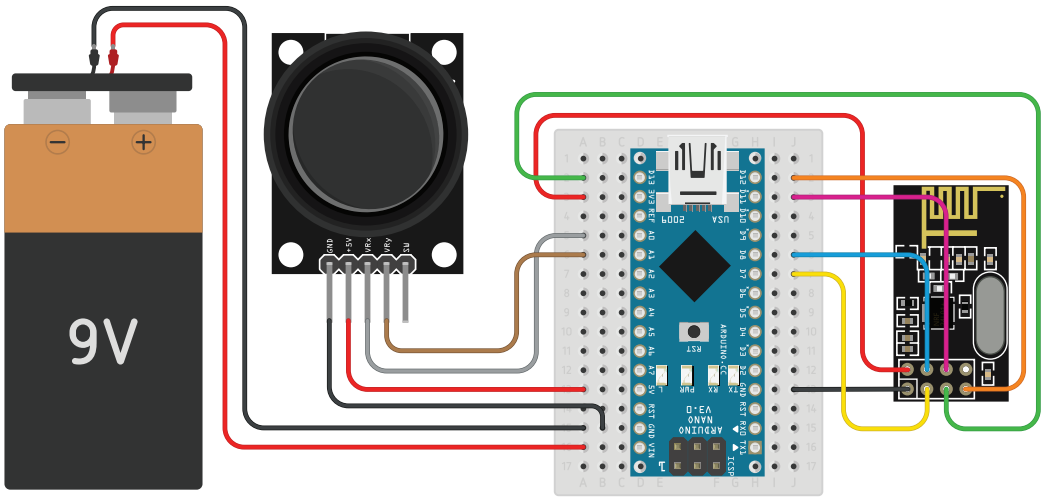
Görsel 2.82'de **joystick** modülünün 10 bitlik ADC'de aldığı değerler verilmiştir. Görsel 2.83'te **joystick** modülünün pin yapısı görülmektedir. Görsel 2.84'teki verici ve Görsel 2.85'teki alıcı devrelerinde birer adet nRF24L01+ modül kullanılarak kablosuz iletişim uygulaması yapılmıştır. nRF24L01+ modülü Arduino ile haberleşirken SPI protokolünü kullanır. **Joystick**le x ve y ekseninde 0-1023 arasında okunan değerler map() foksiyonuyla -255 ile 255 arası değerlere dönüştürülerek alıcıya gönderilir. Alıcıda negatif değerler "-" ile çarpılıp pozitif değer PWM olarak geri ve sol dönüşe uygulanır.



Görsel 2.82: Joystickle x ve y ekseninden okunan değerler



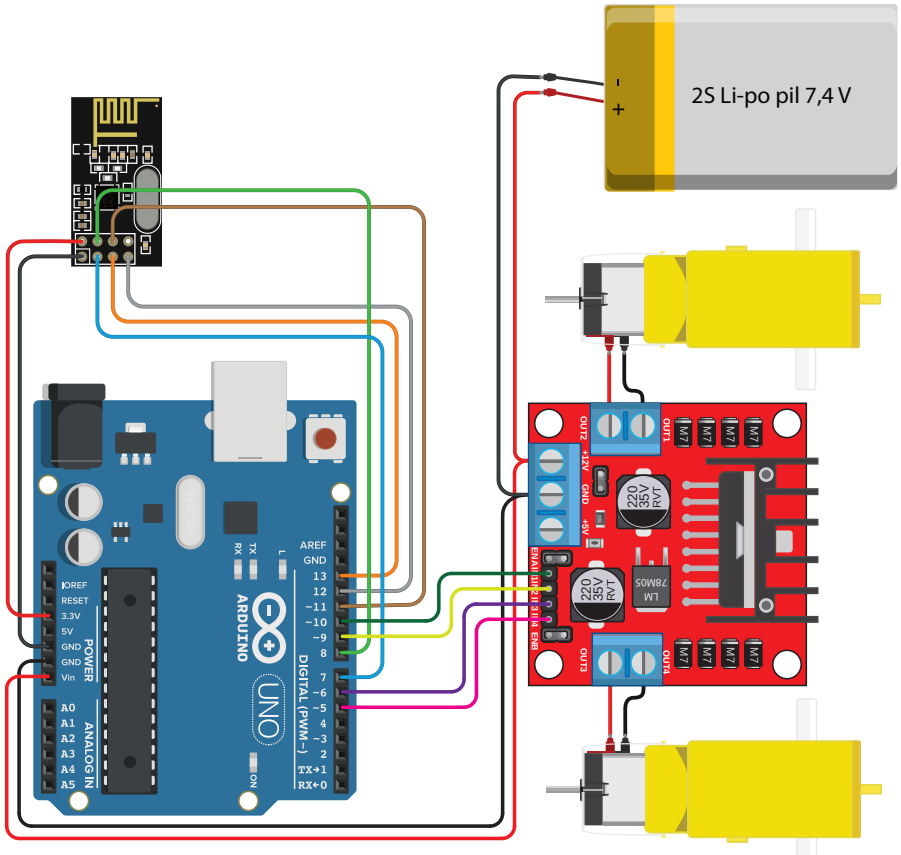
Görsel 2.83: Joystick modülü pin yapısı



Görsel 2.84: Verici uygulaması

Uzaktan kontrollü araba verici uygulaması programı aşağıdaki gibidir:

```
#include <SPI.h> // nRF24L01+ ile SPI iletişim.
#include <RF24.h> // nRF24L01+ için RF24 kütüphanesi v1.4.1

int x, y; // x ve y düzlemlerinin değişkenleri.

RF24 radio(7, 8); // radio isimli nesne oluştur. CE ve CSN pinleri.
int veri[2]; // X ve y düzlemi için dizi tanımlama.

void setup() {
  radio.begin(); // nRF24L01+'yı başlat.
  radio.openWritingPipe(1234); // İletişim anahtarı 1234.
}

void loop() {
  x = analogRead(A0); // joystick x ekseni bilgisini al.
  y = analogRead(A1); // joystick y ekseni bilgisini al.

  veri[0] = map(x, 0, 1023, -255, 255 ); //X düzleminin verisi (Geri-ileri).
  veri[1] = map(y, 0, 1023, -255, 255); //Y düzeleminin verisi (Sol-sağ).

  radio.write(veri, sizeof(veri)); // Veri değişkenindeki bilgiyi nRF24L01+'ya gönder.
}
```



Görsel 2.85: Alıcı uygulaması

Uzaktan kontrollü araba alıcı uygulaması programı aşağıdaki gibidir:

```
#include <SPI.h> // nRF24L01+ ile SPI iletişim.
#include <RF24.h> //  v1.4.2

RF24 radio(7, 8); // radio isimli nesne oluştur. CE ve CSN pinleri.

int veri[2];// X ve Y düzlemi için dizi tanımlama

const byte IN1 = 10, IN2 = 9, IN3 = 6, IN4 = 5; // PWM pinleri. (enA ve enB'de jumper takılı).
// IN1 ve IN2 sol tekerlek, IN3 ve IN4 sağ tekerlek.
void setup() {

  Serial.begin(9600); // Hataları seri monitörde görebilmek için seri iletişimi başlat.
  if (!radio.begin()) { // nRF24L01+'yı başlat.

    Serial.println("nRF24L01+ bağlantıları yanlış veya nRF24L01+ bozulmuş olabilir.");

//nRF24L01+ modülü başlamıyorsa uyarı ver.
    while (1) {}

  }
```

```
 radio.openReadingPipe(0, 1234); // İletişim anahtarı 1234.
  radio.startListening(); // Alıcı moda geç.
}

void loop() {
  if (radio.available()) { // Veri geliyorsa...
    radio.read(&veri, sizeof(veri)); // Gelen veriyi veri değişkenine yükle.

    if (veri[0] > 50) { //İleri.
      analogWrite(IN1, veri[0]);
      analogWrite(IN2, 0);
      analogWrite(IN3, veri[0]);
      analogWrite(IN4, 0);
    }
    if (veri[0] < -50) { //Geri.
      analogWrite(IN1, 0);
      analogWrite(IN2, -veri[0]);
      analogWrite(IN3, 0);
      analogWrite(IN4, -veri[0]);
    }
    if (veri[1] > 50) { // Sağ.
      analogWrite(IN1, veri[1]);
      analogWrite(IN2, 0);
      analogWrite(IN3, 0);
      analogWrite(IN4, veri[1]);
    }
    if (veri[1] < -50) { //Sol.
      analogWrite(IN1, 0);
      analogWrite(IN2, -veri[1]);
      analogWrite(IN3, -veri[1]);
      analogWrite(IN4, 0);
    }
    if (veri[0] > -50 && veri[0] < 50 && veri[1] > -50 && veri[1] < 50) { // Dur.
      analogWrite(IN1, 0);
      analogWrite(IN2, 0);
      analogWrite(IN3, 0);
      analogWrite(IN4, 0);
    }
  }
}
```

İşlem Basamakları

1. Görsel 2.84'teki verici ve Görsel 2.85'teki alıcı devreyi kurunuz. Programları yükleyiniz.
2. Verici devresindeki **joystick**i kullanarak alıcı aracın çalışmasını gözlemleyiniz.

SIRA SİZDE

1. **Joystick** üzerine basınca (sw) araçta korna (**buzzer**) çalan düzenlemeyi gerçekleştiriniz.

Verici kodu

```
#include <SPI.h> // nRF24L01+ ile SPI iletişim.
#include <RF24.h> //  v1.4.2

int x, y; // x ve y düzlemlerinin değişkenleri.

RF24 radio(7, 8); // radio isimli nesne oluştur. CE ve CSN pinleri.
int veri[3]; // x, y düzlemi ve korna (sw) için dizi tanımlama.
const byte sw=2; // sw (buton) pini.

void setup() {
  pinMode(sw, INPUT_PULLUP); // Butona basınca 0.

  Serial.begin(9600); // Hataları seri monitörde görebilmek için seri iletişimi başlat.
  if (!radio.begin()) { // nRF24L01+'yı başlat.

    Serial.println("nRF24L01+ bağlantıları yanlış veya temazsızlık var. nRF24L01+ bozulmuş olabilir."); //nRF24L01+ modülü başlamıyorsa uyarı ver.
    while (1) {}
  }
  radio.openWritingPipe(1234); // İletişim anahtarı 1234.
}

void loop() {
  x = analogRead(A0);
  y = analogRead(A1);
  bool swDurum = digitalRead(sw); // Butonun durumunu oku.

  veri[0] = map(x, 0, 1023, -255, 255 ); //X düzleminin verisi (Geri-ileri).
  veri[1] = map(y, 0, 1023, -255, 255); //Y düzeleminin verisi (Sol-sağ).
  veri[2] = swDurum; //Butona basılma bilgisi. 0 basıldı, 1 basılmadı bilgisi.


  radio.write(veri, sizeof(veri)); // veri değişkenindeki bilgiyi nRF24L01+'ya gönder.
}
```

Alıcı kodu

```
#include <SPI.h> // nRF24L01+ ile SPI iletişim.
#include <RF24.h> //  v1.4.2

RF24 radio(7, 8); // radio isimli nesne oluştur. CE ve CSN pinleri.
```

```
int veri[3];// X, y düzlemi ve sw için dizi tanımlama.

const byte IN1 = 5, IN2 = 6, IN3 = 9, IN4 = 10; // PWM pinleri. (enA ve enB'de jumper takılı).
// IN1 ve IN2 sol tekerlek, IN3 ve IN4 sağ tekerlek.
const byte buzzer = 11; // Buzzer pin.

void setup() {

  Serial.begin(9600); // Hataları seri monitörde görebilmek için seri iletişimi başlat.
  if (!radio.begin()) { // nRF24L01+'yı başlat.

    Serial.println("nRF24L01+ bağlantıları yanlış veya nRF24L01+ bozulmuş olabilir."); //
nRF24L01+ modülü başlamıyorsa uyarı ver.
    while (1) {}
  }
  radio.openReadingPipe(0, 1234); // İletişim anahtarı 1234.
  radio.startListening(); // Alıcı moda geç.
  pinMode(buzzer, OUTPUT);
}

void loop() {
  if (radio.available()) { // Veri geliyorsa...
    radio.read(&veri, sizeof(veri)); // Gelen veriyi veri değişkenine yükle.

    if (veri[0] > 50) { //İleri.
      analogWrite(IN1, veri[0]);
      analogWrite(IN2, 0);
      analogWrite(IN3, veri[0]);
      analogWrite(IN4, 0);
    }
    if (veri[0] < -50) { //Geri.
      analogWrite(IN1, 0);
      analogWrite(IN2, -veri[0]);
      analogWrite(IN3, 0);
      analogWrite(IN4, -veri[0]);
    }
    if (veri[1] > 50) { // Sağ.
      analogWrite(IN1, veri[1]);
      analogWrite(IN2, 0);
      analogWrite(IN3, 0);
      analogWrite(IN4, veri[1]);
 }
    if (veri[1] < -50) { //Sol.
      analogWrite(IN1, 0);
      analogWrite(IN2, -veri[1]);
```

```
      analogWrite(IN3, -veri[1]);
      analogWrite(IN4, 0);
    }
    if (veri[0] > -50 && veri[0] < 50 && veri[1] > -50 && veri[1] < 50) { // Dur.
      analogWrite(IN1, 0);
      analogWrite(IN2, 0);
      analogWrite(IN3, 0);
      analogWrite(IN4, 0);
    }
    if (veri[2] == 0)
      digitalWrite(buzzer, HIGH); // Korna çal.
    else
      digitalWrite(buzzer, LOW); // Korna durdur.
  }
}
```

2. Aracı ileri-sağ, ileri-sol, geri-sağ ve geri-sol yönlere çapraz hareket edebilecek şekilde programı düzenleyiniz.

```
 #include <SPI.h> // nRF24L01+ ile SPI iletişim.
#include <RF24.h> //  v1.4.2

RF24 radio(7, 8); // radio isimli nesne oluştur. CE ve CSN pinleri.

byte solPwm, sagPwm;
int veri[2];// X ve Y düzlemi için dizi tanımlama

const byte IN1 = 5, IN2 = 6, IN3 = 9, IN4 = 10; // PWM pinleri. (enA ve enB'de jumper takılı).
// IN1, IN2 sol tekerlek, IN3, IN4 sağ tekerlek.

void setup() {

  Serial.begin(9600); // Hataları seri monitörde görebilmek için seri iletişimi başlat.
  if (!radio.begin()) { // nRF24L01+'yı başlat.

    Serial.println("nRF24L01+ bağlantıları yanlış veya nRF24L01+ bozulmuş olabilir."); //
nRF24L01+ modülü başlamıyorsa uyarı ver.
    while (1) {}
  }
  radio.openReadingPipe(0, 1234); // İletişim anahtarı 1234.
  radio.startListening(); // Alıcı moda geç.
}

void loop() {
  if (radio.available()) { // Veri geliyorsa...

    radio.read(&veri, sizeof(veri)); // Gelen veriyi veri değişkenine yükle.
    int x = veri[0]; // x'in içeriği -255 ile 255 arasında değişmektedir.
```

```
    int y = veri[1]; // y'in içeriği -255 ile 255 arasında değişmektedir.

    if (y > 0) { // y pozitif ise araç sağa döner.
      solPwm = x; // Sol tekerlek hızını x ekseninden gelen sayı yap.
      sagPwm = x - y / 5; // Sağ tekerlek hızını (pwm) y ekseninden gelen sayının 1/5'i kadar
azalt.
    }
    else { // y negatif ise araç sola döner.
      sagPwm = x;
      solPwm = x + y / 5; // Solda y negatiftir (Sol tekerleğin hızını azalt).
    }

    if (x > 50) { // İleri yönler.
      analogWrite(IN1, solPwm);
      analogWrite(IN2, 0);
      analogWrite(IN3, sagPwm);
      analogWrite(IN4, 0);
    }
    if (x < -50) { // Geri yönler.
      analogWrite(IN1, 0);
      analogWrite(IN2, 255 - solPwm); // Joystick'i geri çektikçe geriye doğru hızlan.
      analogWrite(IN3, 0);
      analogWrite(IN4, 255 - sagPwm);
    }
    if (x > -50 && x < 50 && y > -50 && y < 50) { // Joystick sabitse dur.
      analogWrite(IN1, 0);
      analogWrite(IN2, 0);
      analogWrite(IN3, 0);
      analogWrite(IN4, 0);
    }
  }
}
```

Sorular

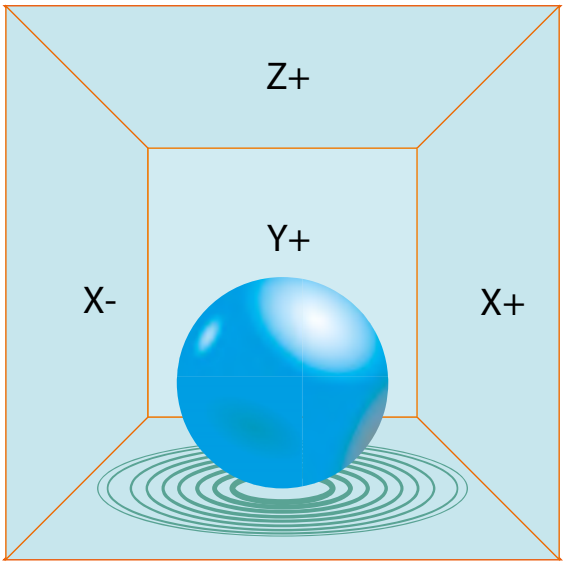
1. **Joystick** modülü ile potansiyometre arasındaki fark nedir? Açıklayınız.
2. veri[0] hangi eksenin (x, y) bilgisini içermektedir? Belirtiniz.
3. **Joystick** sabitken x ve y'nin değeri nedir? Belirtiniz.
4. "if (veri[0] > 50) { //İleri." satırında neden 0 yerine 50 kullanılmıştır? Açıklayınız.
5. veri[0] değişkeni negatif bir sayı içeriyorsa -veri[0] ile abs(veri[0]) aynı sonucu verir mi? Nedeni ile açıklayınız.

İvme ve Eğim Sensörü

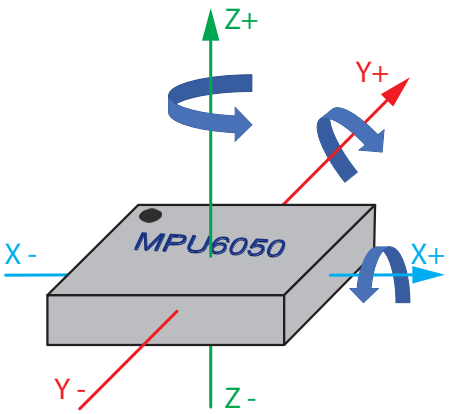
MPU6050, üzerinde üç eksenli gyro (Görsel 2.86) ve üç eksenli açısal ivmeölçer (Görsel 2.87) bulunduran IMU [Inertial Measurement Unit (inırşıl mejırmınt yunıt) Atalet ölçü birimi] sensör kartıdır. IMU sensörleri, denge robotları, uçaklar, cep telefonları, tabletler, uzay araçları, uydular, hava aracı vb. araçlarda konum tespiti, yön bulma, hareket izleme ve uçuş kontrolü için kullanılır. İvmeölçer doğrusal ivmeyi ölçerken jiroskop açısal dönüşü (°/s) ölçer.

MPU6050 sensör kartı üzerinde voltaj regulatörü bulunduğundan 3 V ile 5 V arası bir besleme voltajı ile çalıştırılabilir (Görsel 2.88). İvmeölçer ve gyro çıkışlarının her ikisi de ayrı kanallardan I²C çıkışı vermektedir. Her eksende 16 bitlik bir çözünürlükle çıkış verebilmektedir.

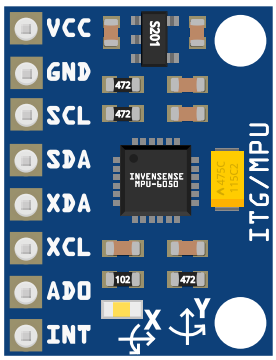
İvmeölçer üzerine düşen statik (yer çekimi) veya dinamik (aniden hızlanma veya durma) ivmeyi ölçmektedir. Görsel 2.86'da çalışma prensibi verilen sensöre 1g'lık bir yer çekimi kuvveti etki etmektedir. Bu kuvvet de yaklaşık olarak 9.8 m/sn.dir. Bu durumda MPU6050 kütüphanesi Z değerini 16384 olarak verir. Sensör ters çevrildiğinde Z değeri -16384 olur. X ve y eksenleri için de aynı durum geçerlidir. Kartın açılı olması durumunda bulunduğu konuma göre X, Y ve Z değerleri -16384 ile 16384 arasında değer alır.



Görsel 2.86: İvmeölçer çalışma prensibi



Görsel 2.87: Hassasiyet eksenlerinin yönü ve dönme polaritesi



Görsel 2.88: MPU6050 pin yapısı

VCC: 5 V – 35 V besleme.

GND: Topraklama pini.

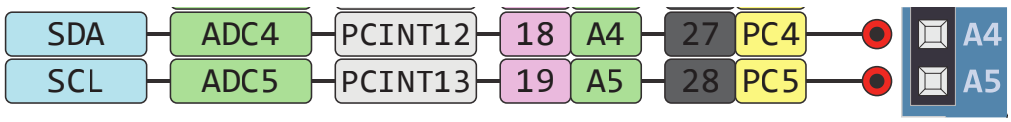
SCL: I2C saat pinidir. **Master** cihaz tarafından sağlanan bir zamanlama sinyalidir.

SDA: I2C veri pinidir. Bu hat hem gönderme hem de alma için kullanılır.

XDA: Haricî I2C veri hattıdır. Haricî I2C veri yolu, haricî sensörleri bağlamak içindir.

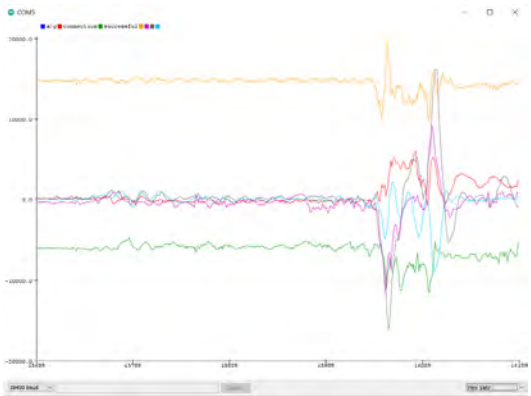
AD0: MPU6050 modülünün dâhilî I2C adresini değiştirmenizi sağlar. Modül başka bir I2C cihazıyla çakışıyorsa veya aynı I2C veri yolunda iki MPU6050 kullanılırsa bu pin kullanılır. AD0 pini boş bırakıldığında varsayılan I2C adresi 0x68 **HEX** olur. 3,3 V'a bağlandığında I2C adresi 0x69 **HEX** olur.

Arduino Uno ve Nano'da I2C haberleşme pinleri, A4 ve A5 analog giriş pinleridir. A4-SDA ve A5-SCL'dir (Görsel 2.89).



Görsel 2.89: Arduino Uno ve Nano'da I2C haberleşme için pin numaraları

Sensörü test etmek için bağlantılar yapıldıktan ve MPU6050 kütüphanesi yüklendikten sonra Arduino IDE menülerinden "Dosya →Örnekler→MPU6050→ MPU6050_raw" örneği açılır. Altı eksenin değeri seri monitörde gözlenir. Değerleri daha rahat izleyebilmek için araçlar menüsünden seri çizici seçilerek Serial.print() fonksiyonuyla yazdırılan değerler grafik olarak gözlemlenebilir (Görsel 2.90).



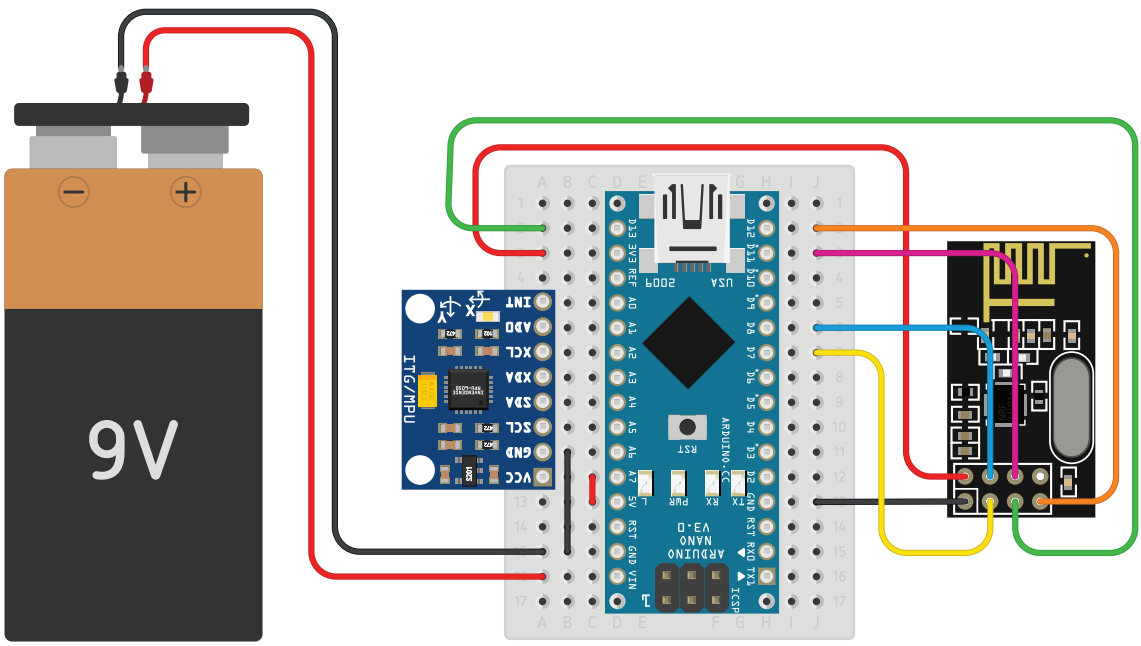
Görsel 2.90: Değerleri seri çizicide gösterme

Uygulama Adı: Elle Uzaktan Kontrollü Araba Uygulaması No.: 18

Amaç: Elle uzaktan kontrollü araba uygulaması yapmak.

Görsel 2.91'de eldiven üzerine yerleştirilmiş verici devresi görülmektedir. Görsel 2.92'deki vericide kullanılan MPU6050 sensörü küçük deney borduna SDA pini A4, SCL pini A5'e gelecek şekilde takılmıştır. Görsel 2.93'teki alıcı devresinde nRF24L01+ modül kullanılarak kablosuz iletişim uygulaması yapılmıştır. nRF24L01+ modülü Arduino ile haberleşirken SPI protokolünü, MPU6050 ivme sensörü I2C protokolünü kullanmaktadır. İvme sensörünün x ve y ekseninde -16384 ile 16384 arasında ürettiği değerler map() foksiyonuyla -255 ile 255 arası değerlere dönüştürülerek alıcıya gönderilir. Alıcıda negatif değerler "-" ile çarpılarak pozitif değer PWM olarak geri ve sağ dönüşe uygulanır.

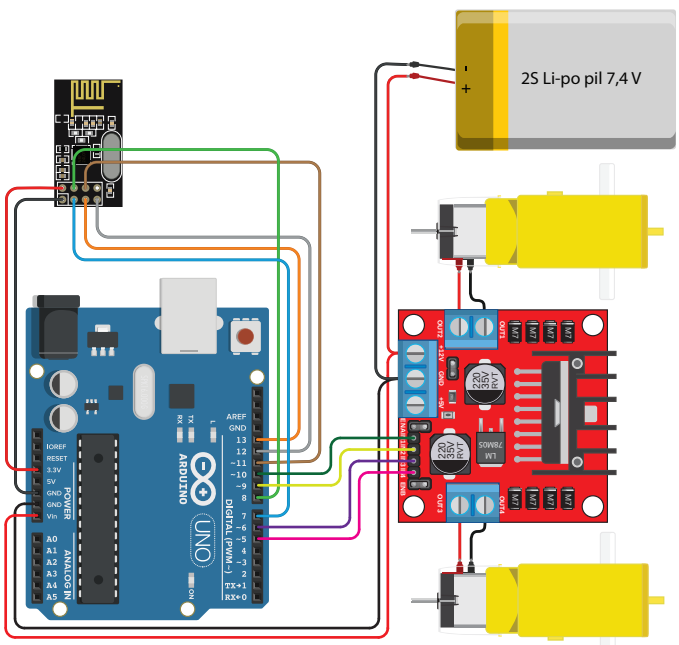
Görsel 2.91: Vericiyi eldivenle kontrol etme



Görsel 2.92: Verici uygulaması

Elle uzaktan kontrollü araba verici uygulaması programı aşağıdaki gibidir:

```
#include <SPI.h> // nRF24L01+ ile SPI iletişim.
#include <RF24.h> // nRF24L01+ kütüphanesi v1.4.1
#include <Wire.h> // Mpu6050 ile I2C iletişim.
#include <MPU6050.h> // Mpu6050 kütüphanesi v0.3.0

MPU6050 ivme_sensor; // ivme_sensor isimli nesne oluştur.
int x, y, z; // x, y, z düzlemlerinin ivme değişkenleri.

RF24 radio(7, 8); // radio isimli nesne oluştur. CE ve CSN pinleri.
int veri[2]; // X ve y düzlemi için dizi tanımlama.

void setup() {
  Wire.begin(); // I2C iletişimi başlat.
  ivme_sensor.initialize(); // Mpu6050'yi başlat.

  radio.begin(); // nRF24L01+'yı başlat.
  radio.openWritingPipe(1234); // İletişim anahtarı 1234.
}

void loop() {
  ivme_sensor.getAcceleration(&x, &y, &z); // Sensörden ivme bilgilerini al.

  veri[0] = map(x, -16384, 16384, -255, 255 ); // X düzleminin verisi (ileri-geri)
  veri[1] = map(y, -16384, 16384, -255, 255); // Y düzeleminin verisi (sağ-sol)

  radio.write(veri, sizeof(veri)); // Veri değişkenindeki bilgiyi nRF24L01+'ya gönder.
}
```



Görsel 2.93: Alıcı uygulaması

Elle uzaktan kontrollü araba alıcı uygulaması programı aşağıdaki gibidir:

```
#include <SPI.h>
#include <RF24.h> // v1.4.1

RF24 radio(7, 8); // radio isimli nesne oluştur. CE ve CSN pinleri.

int veri[2];// X ve Y düzlemi için dizi tanımlama
const byte IN1 = 5, IN2 = 6, IN3 = 9, IN4 = 10; // PWM pinleri. (enA ve enB'de jumper takılı).

void setup() {
  radio.begin(); // NRF'yi başlat
  radio.openReadingPipe(0, 1234); // İletişim anahtarı 1234.
  radio.startListening(); // Alıcı moda geç.
}

void loop() {
  if (radio.available()) { // Veri geliyorsa...
    radio.read(veri, sizeof(veri)); // Gelen veriyi veri değişkenine yükle.

    if (veri[0] > 50) { //İleri.
      analogWrite(IN1, veri[0]);
      analogWrite(IN2, 0);
      analogWrite(IN3, veri[0]);
      analogWrite(IN4, 0);
    }
    if (veri[0] < -50) { //Geri.
      analogWrite(IN1, 0);
      analogWrite(IN2, -veri[0]);
      analogWrite(IN3, 0);
      analogWrite(IN4, -veri[0]);
    }
    if (veri[1] > 50) { //Sol.
      analogWrite(IN1, 0);
      analogWrite(IN2, veri[1]);
      analogWrite(IN3, veri[1]);
      analogWrite(IN4, 0);
    }
    if (veri[1] < -50) { // Sağ.
      analogWrite(IN1, -veri[1]);
      analogWrite(IN2, 0);
      analogWrite(IN3, 0);
      analogWrite(IN4, -veri[1]);
    }
```

```
  if (veri[0] > -50 && veri[0] < 50 && veri[1] > -50 && veri[1] < 50) { // Dur.
   analogWrite(IN1, 0);
   analogWrite(IN2, 0);
   analogWrite(IN3, 0);
   analogWrite(IN4, 0);
  }
  }
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.92'deki verici ve Görsel 2.93'teki alıcı devreyi kurunuz. Programları yükleyiniz.
4. Verici devresine x ve y eksenlerinde eğim vererek alıcı aracın çalışmasını gözlemleyiniz.
5. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Sensör titreştirildiğinde alıcıda korna (**buzzer**) çalan uygulamayı gerçekleştiriniz.

Verici kodu

```
#include <SPI.h> // nRF24L01+ ile SPI iletişim.
#include <RF24.h> // v1.4.2
#include <Wire.h> // Mpu6050 ile I2C iletişim.
#include <MPU6050.h> //Mpu6050 kütüphanesi v0.3.0

MPU6050 ivme_sensor; // ivme_sensor isimli nesne oluştur.
int ax, ay, az, gx, gy, gz; // x, y, z düzlemlerinin ivme değişkenleri.

RF24 radio(7, 8); // radio isimli nesne oluştur. CE ve CSN pinleri.
int veri[3]; // X ve y düzlemi için dizi tanımlama.

void setup() {
  Wire.begin(); // I2C iletişimi başlat.
  ivme_sensor.initialize(); // Mpu6050'yi başlat.


  Serial.begin(9600); // Hataları seri monitörde görebilmek için seri iletişimi başlat.
  if (!radio.begin()) { // nRF24L01+'yı başlat.
    Serial.println("nRF24L01+ bağlantıları yanlış veya temazsızlık var. nRF24L01+ bozulmuş olabilir."); //nRF24L01+ modülü başlamıyorsa uyarı ver.
    while (1) {}
  }
  radio.openWritingPipe(1234); // İletişim anahtarı 1234.
```

```
}

void loop() {

  ivme_sensor.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);  // Sensörden ivme bilgilerini al.

  veri[0] = map(ax, -16384, 16384, -255, 255 ); //X düzleminin verisi (ileri-geri)
  veri[1] = map(ay, -16384, 16384, -255, 255); //Y düzleminin verisi (sağ-sol)
  veri[2] = gx;

  radio.write(&veri, sizeof(veri)); // Veri değişkenindeki bilgiyi nRF24L01+'ya gönder.
}
```

Alıcı kodu

```
#include <SPI.h> // nRF24L01+ ile SPI iletişim.
#include <RF24.h> //  v1.4.2

RF24 radio(7, 8); // radio isimli nesne oluştur. CE ve CSN pinleri.

int veri[3];// X ve Y düzlemi için dizi tanımlama

const byte IN1 = 5, IN2 = 6, IN3 = 9, IN4 = 10; // PWM pinleri. (enA ve enB'de jumper takılı).
const byte buzzer = 4;
void setup() {

  Serial.begin(9600); // Hataları seri monitörde görebilmek için seri iletişimi başlat.
  if (!radio.begin()) { // nRF24L01+'yı başlat.

    Serial.println("nRF24L01+ bağlantıları yanlış veya nRF24L01+ bozulmuş olabilir.");

 //nRF24L01+ modülü başlamıyorsa uyarı ver.
    while (1) {}
  }
  radio.openReadingPipe(0, 1234); // İletişim anahtarı 1234.
  radio.startListening(); // Alıcı moda geç.
  pinMode(buzzer, OUTPUT); // Buzzer pin.
}

void loop() {
  if (radio.available()) { // Veri geliyorsa...
    radio.read(&veri, sizeof(veri)); // Gelen veriyi veri değişkenine yükle.

    if (veri[0] > 50) { //İleri.
      analogWrite(IN1, veri[0]);
      analogWrite(IN2, 0);
      analogWrite(IN3, veri[0]);
      analogWrite(IN4, 0);
    }
    if (veri[0] < -50) { //Geri.
```

```
      analogWrite(IN1, 0);
      analogWrite(IN2, -veri[0]);
      analogWrite(IN3, 0);
      analogWrite(IN4, -veri[0]);
    }
    if (veri[1] > 50) { //Sol.
      analogWrite(IN1, 0);
      analogWrite(IN2, veri[1]);
      analogWrite(IN3, veri[1]);
      analogWrite(IN4, 0);
    }
    if (veri[1] < -50) { // Sağ.
      analogWrite(IN1, -veri[1]);
      analogWrite(IN2, 0);
      analogWrite(IN3, 0);
      analogWrite(IN4, -veri[1]);
    }
    if (veri[0] > -50 && veri[0] < 50 && veri[1] > -50 && veri[1] < 50) { // Dur.
      analogWrite(IN1, 0);
      analogWrite(IN2, 0);
      analogWrite(IN3, 0);
      analogWrite(IN4, 0);
    }
    if (veri[2] > 16000) {
      digitalWrite(buzzer, HIGH); // Korna çal.
      delay(200);
    }
    else
      digitalWrite(buzzer, LOW); // Korna durdur.
  }
}
```

Soru

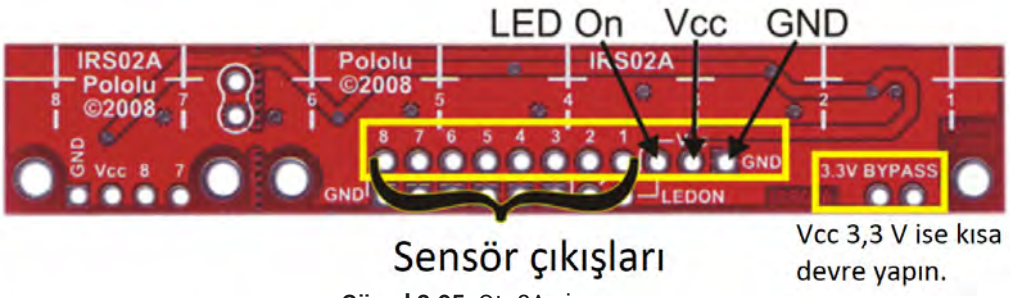
1. MPU6050 ile denge robotu nasıl yapılır? Araştırınız.

Çizgi İzleyen Sensör

Çizgi izleyen robotlarda kullanılan sensörler kızılötesi verici ve kızılötesi alıcı çiftinden oluşur. Vericiden çıkan kızılötesi ışık beyaz yüzeye çarptığında yansır. Alıcı, yansıyan bu ışığı algılar. Siyah zemin kızılötesi ışığı yansıtmayacağı için alıcıda algılama gerçekleşmez. Görsel 2.94'teki QTR8A sensöründe kızılötesi sensör çiftinden sekiz adet bulunmaktadır. Analog çıkış veren bu sensör grubunda beyaz zemine denk gelen sensör 0 V'a yakın değerler verirken siyah zemine denk gelen sensörler 5 V'a (Vcc) yakın değerler verir.

Görsel 2.94: Qtr 8A sensör dizilimi

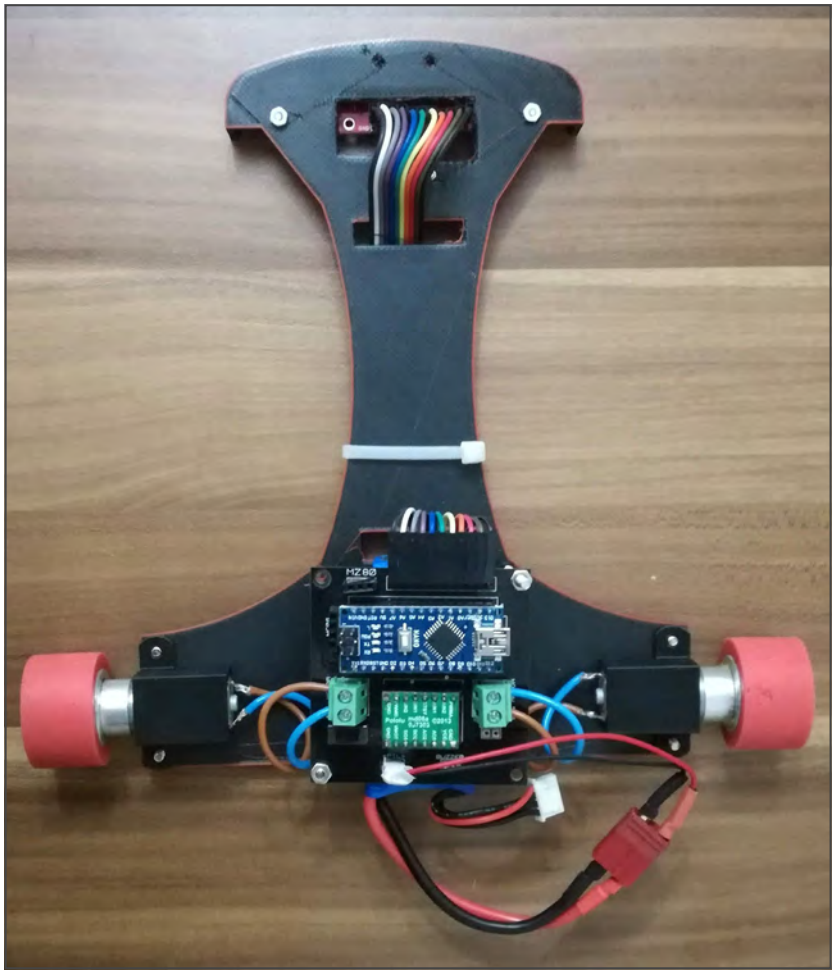
Görsel 2.95'teki sensör kartı 3,3 V-5 V gerilimle çalışmaktadır ve 100 mA akım çekmektedir. En iyi algılama mesafesi 3 mm'dir. Sensörün zeminden uzaklığı 6 mm'yi geçmemelidir. "LED On" pini lojik 1 olduğunda veya boş bırakıldığında (Dâhilî pull-up bağlıdır.) verici LED'ler çalışmaktadır. Lojik 0 olduğunda verici LED'ler kapalıdır.



Görsel 2.95: Qtr 8A pin yapısı

Bu kartla birlikte kullanılan QTRSensors.h kütüphanesi farklı sayıda kızılötesi sensör içeren diğer sensör kartlarında da kullanılabilir.

Çizgi izleyen robotlar çizgiyi takip edecek şekilde programlanır. QTRSensors.h kütüphanesi sensör sayısına göre her bir sensörden gelen değerlerin ağırlıklı ortalamasını alarak çizginin bulunduğu pozisyonu verir. Çizginin pozisyonu hedef değerden farklıysa kapalı çevrim kontrol sistemi komutlarıyla (**pid**) pozisyon değeri hedef değere yaklaşacak şekilde motor pwm değerleri değiştirilerek araç çizgide yol alır. Çizgi izleyen robotta dışarıdan bir müdahale olmadığından (rüzgâr, dalga, arazi yolu) **pid** sisteminde integral kullanılmasına gerek yoktur. **Pid** sisteminde kullanılan katsayılar aracın boyuna ve ağırlığına göre değişmektedir (Görsel 2.96).

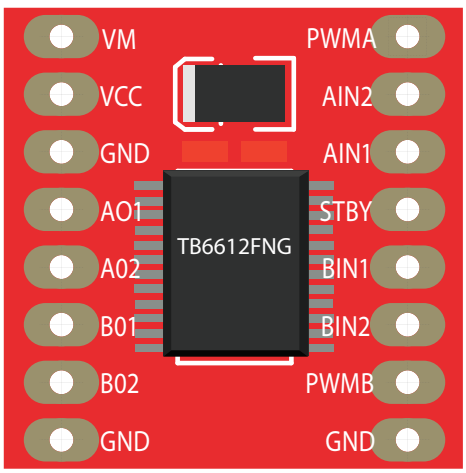


Görsel 2.96: Çizgi izleyen robot

Uygulama Adı: Çizgi İzleyen Robot Uygulaması No.: 19

Amaç: Çizgi izleyen robot uygulaması yapmak.

Görsel 2.97'de Sparkfun TB6612fng motor sürücü kartı pin yapısı verilmiştir. Görsel 2.98'deki çizgi izleyen robot devresinde Sparkfun TB6612fng (kırmızı renkli PCB) motor sürücü kartı kullanılmıştır. Pololu TB6612fng entegresini kullanan yeşil renkli PCB kartın boyutları ve pin yerleşim düzeni farklıdır. Çizgi izleyen robot programı L298N motor sürücüsüyle de kullanılabilir.



Görsel 2.97: TB6612fng pin yapısı

VCC: 5 V besleme.

GND: Topraklama pini.

VM: Motor besleme girişi 2,5 V-13,5 V.

STBY: "LOW"=standby. Sürücüyü aktif yapmak için lojik 1 yapılır. 5 V hattına bağlanılarak da kullanılabilir.

AO1, AO2: Birinci motor (sağ) çıkışları. 1,2 A sürekli, 3,2 A anlık.

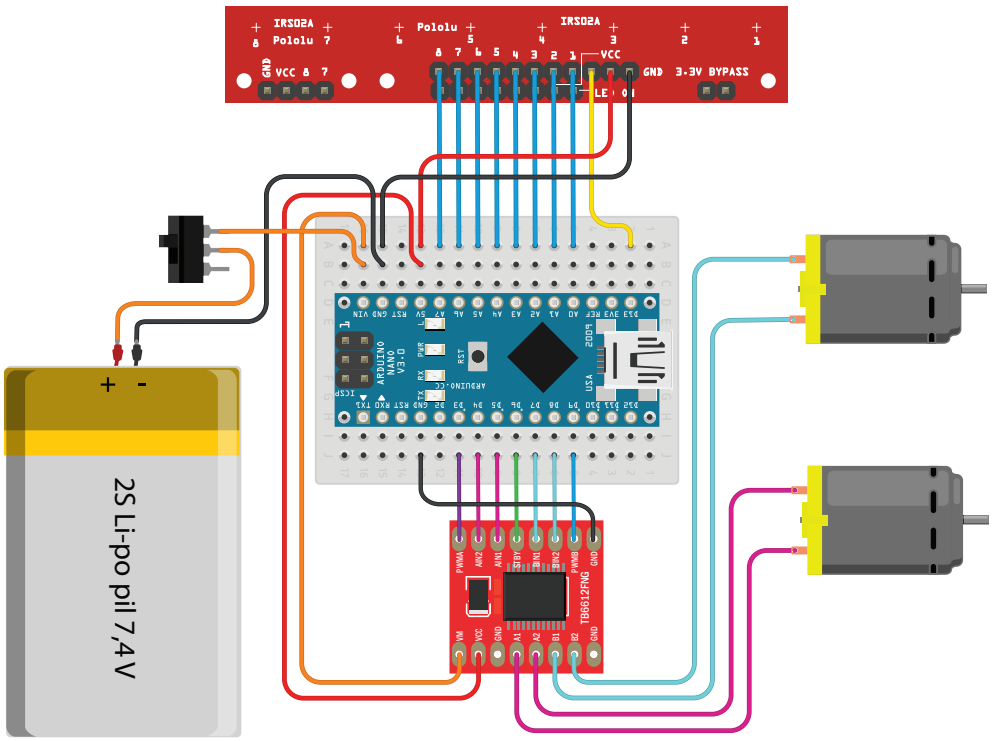
BO1, BO2: İkinci motor (sol) çıkışları. 1,2 A sürekli, 3,2 A anlık.

AIN1, AIN2: Birinci motor (sağ) kontrol girişleri. 200 kΩ dâhilî pull-down.

BIN1, BIN2: İkinci motor (sol) kontrol girişleri. 200 kΩ dâhilî pull-down.

PWMA: Birinci motor (sağ) PWM girişi.

PWMB: İkinci motor (sol) PWM girişi.



Görsel 2.98: Çizgi izleyen robot uygulaması

Çizgi izleyen robot uygulaması programı aşağıdaki gibidir:

```
#include <QTRSensors.h> //Qtr v4.0

#define PWMA 3 // A sağ motor.
#define AIN2 4
#define AIN1 5
#define STBY 6
#define BIN1 7 // B sol motor.
#define BIN2 8
#define PWMB 9

#define sensorSayisi 8
#define sensorOrnekSayisi 4
#define emiterPini 11
#define LED 13

int maxHiz = 70; // Motor pwm ayarı 0 - 255.

int hata = 0, turev = 0;

float KP = 0.03, KD = 0.5; // Oran (KP) ve türev (KD) sabitleri. (Her araca göre ayar yapılmalıdır.)
unsigned int pozisyon = 3500;
```

```
int fark = 0; // Motorlara uygulanan fark.
int sonHata; // Orantılı son değer. (Hatanın türevini hesaplamak için kullanılır.)
int hedef = 3500; // Sensörden gelen 0 - 7000 arası değerin orta noktası.

QTRSensors qtr; // qtr isimli nesne oluşturuldu.
unsigned int sensor[sensorSayisi];

void setup() {

  qtr.setTypeAnalog();  //QTR-8A ayarla. (QTR-8RC için qtr.setTypeRC() fonksiyonu kullanılır.)
  qtr.setSensorPins((const uint8_t[]) {
    A0, A1, A2, A3, A4, A5, A6, A7
  }, 8);
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(BIN1, OUTPUT);
  pinMode(BIN2, OUTPUT);
  pinMode(LED, OUTPUT);
  pinMode(STBY, OUTPUT);

  delay(1000); //Araca enerji verince 1sn bekle
  kalibrasyon(1); // 0 elle, 1 otomatik kafa sallama.
}

void loop() {
  sensorOku();
  pd();
}

void sensorOku() {
  pozisyon = qtr.readLineWhite(sensor); // Beyaz çizginin pozisyonunu oku. (0 - 7000)
  hata = pozisyon - hedef; // Pozisyondan 3500 (hedef) çıkar. Hatayı bul.
  qtr.read(sensor); // Sekiz sensörün ham değelerini oku.
}

void pd() {
  turev = hata - sonHata; // Hatadan bir önceki hatayı çıkar.
  sonHata = hata; // Şimdiki hatayı kaydet.
 fark = ( hata * KP) + ( turev * KD ); // Motorlara uygulanacak farkı hesapla.

  constrain(fark, -maxHiz, maxHiz); // Fark en fazla maxHiz olsun.

  if ( fark < 0 ) // fark negatif ise
```

```
  motor(maxHiz, maxHiz + fark); // Sağ motorun hızını düşür.
  else // fark negatif değilse
    motor(maxHiz - fark, maxHiz); // Sol motorun hızını düşür.
}

void motor(int solMotorPWM, int sagMotorPWM) {
  digitalWrite(STBY, HIGH);

  if ( solMotorPWM >= 0 )  { // İleri.
    digitalWrite(BIN1, HIGH);
    digitalWrite(BIN2, LOW);
  }
  else  { // Negatifse geri döndür.
    digitalWrite(BIN1, LOW);
    digitalWrite(BIN2, HIGH);
    solMotorPWM *= -1;
  }
  analogWrite(PWMB, solMotorPWM);

  if ( sagMotorPWM >= 0 )  { // İleri.
    digitalWrite(AIN1, HIGH);
    digitalWrite(AIN2, LOW);
  }
  else  { // Negatifse geri döndür.
    digitalWrite(AIN1, LOW);
    digitalWrite(AIN2, HIGH);
    sagMotorPWM *= -1;
  }
  analogWrite(PWMA, sagMotorPWM);
}

void kalibrasyon(bool secim) { // 1 otomatik, 0 elle.
  if (secim) { // secim 1 ise otomatik kalibrasyon yap.
    byte hiz = 40; // Aracın kafasını sallama hızı.
    for (byte i = 0; i < 3; i++) {  // Sağa sola üç kez kafa salla.
      while (sensor[7] < 300) {
    motor(hiz, -hiz);
        qtr.calibrate();
        sensorOku();
      }
      while (sensor[7] > 700) {
        motor(hiz, -hiz);
        qtr.calibrate();
        sensorOku();
      }
```

```
      while (sensor[0] < 300) {
        motor(-hiz, hiz);
        qtr.calibrate();
        sensorOku();
      }
      while (sensor[0] > 700) {
        motor(-hiz, hiz);
        qtr.calibrate();
        sensorOku();
      }
      while (sensor[3] > 500)  { // Ortada dur.
        motor(hiz, -hiz);
        qtr.calibrate();
        sensorOku();
      }
    }
  } else { // Seçim 0 ise elle kalibrasyon yap.

    for ( byte i = 0; i < 70; i++)  { // Dahili LED yanıp söndüğü sürece (3 sn) elle kalibrasyon yap.
      digitalWrite(LED, HIGH); delay(20);
      qtr.calibrate();
      digitalWrite(LED, LOW); delay(20);
      qtr.calibrate();
    }
  }
  motor(0, 0);
  delay(2000); // Kalbirasyondan sonra 3 sn bekle.
}
```


İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.98'deki devreyi araç üzerine kurunuz. Programı yükleyiniz.
4. Siyah zemin üzerine beyaz çizgiye aracı yerleştiriniz.
5. Araca güç verdikten sonra sekiz sensör siyah ve beyaz renkleri görecek şekilde üç saniye (dâhilî LED'in yanıp sönmesi bitene kadar) sensörü sola ve sağa gezdiriniz.
6. Araç çizgiden çıkıyorsa veya zikzak yapıyorsa KP ve KD değerlerini değiştiriniz.
7. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Herhangi bir optik sensör kullanarak (QTR1C olabilir.) tekerlek üzerine işaretlenen beyaz veya siyah alanı (1, 2, 4 veya 8 nokta olabilir.) sayarak aracın aldığı yolu bulunuz.

```
// Aracın aldığı yolu cm cinsinden float veri tipinde döndüren fonksiyon
#define enkoderPini 2
bool yuzeyKontrol = 0;
int noktaSayisi = 0; // Tekerleğin üzerindeki yansıtıcı (beyaz) nokta sayısı.
float r = 1.55; // Tekerleğin yarıçapı (cm).

float enkoder() {
  if (digitalRead(enkoderPini) && yuzeyKontrol) { /* Optik sensör beyaz alanı gördüyse ve yuzey-
Kontrol değişkeni kurulu (1) ise.. */
    noktaSayisi++; // Sayacı bir arttır.
    yuzeyKontrol = beyaz; /* yuzeyKontrol bitini sıfırla (Beyaz alandan çıkmadığı sürece sayacı
arttırma-bu if yapısına girme). */
  }
  if (digitalRead(enkoderPini) == 0) // Optik sensör beyaz alandan çıkarsa...
    yuzeyKontrol = siyah; /* Sayım için yuzeyKontrol bitini kur (Sensör beyaz alandan çıktı). */

  float mesafe = 2 * 3.14 * r * noktaSayisi / 4; // cm cinsinden gidilen yol.
  /* 2*π*r (tekerleğin çevresi) * noktaSayisi/4 (tekerleğin üzerindeki yansıtıcı (beyaz) nokta
sayısı.) */
  return mesafe; // mesafe bilgisini döndür.
}
```

2. Yoldaki dik çizgilerin başlangıç noktasına olan uzaklıklarını EEPROM hafızaya kaydeden fonsiyonları yazınız.

```
void setup() { // setup fonksiyonuna eklenecek kodlar.
  Serial.begin(9600);
  // eepromReset(); Serial.println("Hafıza sıfırlanıyor. Lütfen bekleyiniz..."); // İhtiyaç du-
yulduğunda kullanılır.
  eepromOku();
}
bool dikCizgide = 0;
void sensorOku() { // Dik çizgiyi algılayan kodlar eklenir.

  pozisyon = qtr.readLineWhite(sensor); // Beyaz çizginin pozisyonunu oku. (0 - 7000)
  hata = pozisyon - hedef; // Pozisyondan 3500 (hedef) çıkar. Hatayı bul.
qtr.read(sensor); // Sekiz sensörün ham değerlerini oku.

  if (sensor[7] < 300 && sensor[6] < 300 && sensor[5] < 300 && sensor[4] < 300 && sensor[3] <
300 && sensor[2] < 300 && sensor[1] < 300 && sensor[0] < 300) // Dik çizgi varsa...
    dikCizgide = 1; // Dik çizginin içinde.
  else if (sensor[7] > 700 && sensor[0] > 700)
    dikCizgide = 0; // Dik çizgi geçildi.
```

```
}

// Dik çizgi sayısını döndüren fonksiyon. Ayrıca dik çizgilerin bulunduğu mesafeleri eeprom'a yazar.
int dikCizgiSayisi = 0;
bool dikCizgiKontrol = 1;


int dikCizgiSay() {

  if (dikCizgide && dikCizgiKontrol) { /* Optik sensörler beyaz alanı gördüyse ve yuzeyKontrol
değişkeni kurulu (1) ise.. */
    dikCizgiSayisi++; // Sayacı bir arttır.

    dikCizgiKontrol = beyaz; /* yuzeyKontrol bitini sıfırla (Beyaz alandan çıkmadığı sürece sa-
yacı arttırma-bu if yapısına girme). */
    EEPROM.write(dikCizgiSayisi - 1, enkoder()); /* Dik çizgileri  EEPROM hafızaya yaz. */
  }


  if (dikCizgide == 0) // Optik sensörler beyaz alandan çıkarsa...

    dikCizgiKontrol = siyah; /* Sayım için yuzeyKontrol bitini kur (Sensörler beyaz alandan çıktı). */
  return dikCizgiSayisi;
}
// enkoder değerlerini hafızadan okuma fonksiyonu
byte deger;


void eepromOku() {
  int hafizaBoyutu = EEPROM.length(); // Kartın hafıza boyutunu al.
  for (int adres = 0; adres < hafizaBoyutu; adres++) {
    deger = EEPROM.read(adres);

    if (deger) {
      Serial.print(adres + 1);
      Serial.print(". çizgi ");
      Serial.print(deger);
      Serial.println(" cm'dedir.");

  delay(1);
    }
  }
}
void eepromReset() {
  int hafizaBoyutu = EEPROM.length(); // Kartın hafıza boyutunu al.
  for (int adres = 0; adres < hafizaBoyutu; adres++) {
    EEPROM.write(adres, 0); // Tüm hafızayı sıfırla.
  }
}
```

3. Düzlüklerde aracı hızlandıran fonksiyonu yazınız.

```
// Girilen aralıklarda aracı hızlandıran fonksiyon.


void hizYap(float baslangic, float bitis, int virajMesafesi) { /* başlangıç noktası (cm), bitiş
noktası (cm), dönüşten önceki yavaşlama mesafesi (cm) */
  if (enkoder() > baslangic) {
    while (enkoder() < bitis) {
      sensorOku();
      pd();
      enkoder();
      if (enkoder() < bitis - virajMesafesi) {
        if (maxHiz < 255) { // Adım adım hızlan (en fazla 255).
          maxHiz++; // Hızlanma adımı.
          //delay(1); // Hızlanma gecikmesi.
        }
      } else maxHiz = 70; // Viraja girme hızı (en düşük hız).
    }
  }
}
```

4. Aracın önüne optik sensör (MZ80 vb.) ekleyerek sensörü engelde duracak şekilde düzenleyiniz.

5. Aracı L298N motor sürücüsüne göre düzenleyiniz.

```
// L298N motor sürücü pin tanımlamaları.
#define ENA 9 // A sağ motor.
#define IN1 8
#define IN2 7
#define IN3 5 // B sol motor.
#define IN4 4
#define ENB 3
```

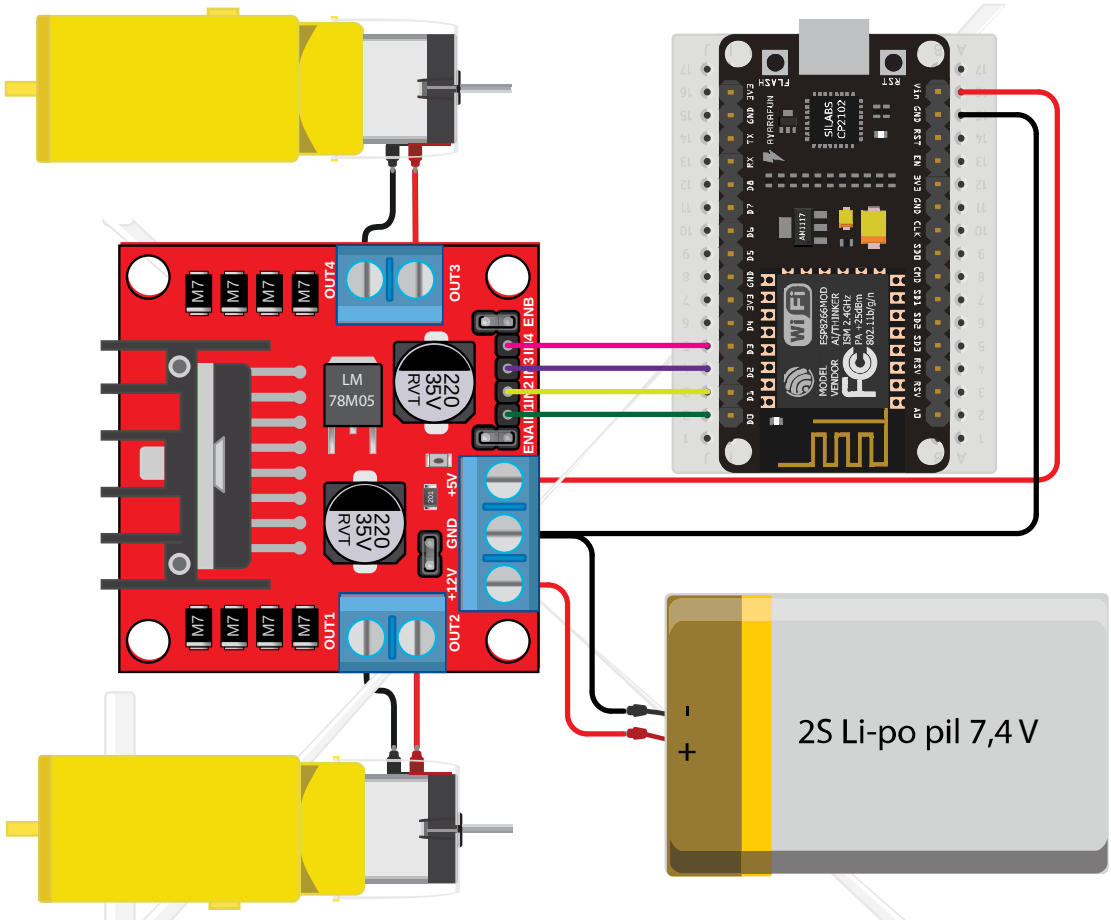
Sorular

1. Araç keskin dönüşler yapabilmekte midir? Neden?
2. **if (fark < 0) // fark negatif ise motor(maxHiz, maxHiz + fark); // Sağ motorun hızını düşür. else // fark negatif değilse motor(maxHiz - fark, maxHiz); // Sol motorun hızını düşür.** komut grubu yerine **motor(maxHiz - fark, maxHiz + fark);** komutu kullanılırsa aracın gidişinde nasıl bir değişim olur?
3. Aracın pozisyon bilgisi eş zamanlı olarak bluetooth üzerinden cep telefonuna aktarılabilir mi? Araştırınız.

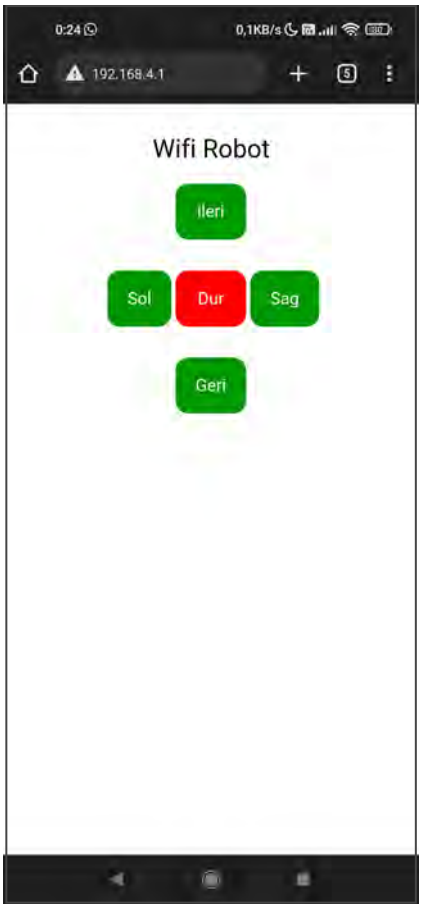
Uygulama Adı: Wi-Fi Kontrollü Araba Uygulaması No.: 20

Amaç: Wi-Fi kontrollü araba uygulaması yapmak.

Görsel 2.99'da Wi-Fi kontrollü araba uygulamasında NodeMCU acces point (AP) modunda çalışmaktadır. NodeMCU'ya bağlanmak için kablosuz ağlardan "Robot" seçilir. Şifre "12345678" girilir. Bilgisayar veya akıllı telefondan tarayıcı (Chrome vb.) açılarak adres satırına 192.168.4.1 girilerek NodeMCU içindeki web server sayfasından araç kontrol edilir (Görsel 2.100).



Görsel 2.99: Wi-Fi kontrollü araba uygulaması



Görsel 2.100: Tarayıcıdan NodeMCU'ya bağlanma

Wi-Fi kontrollü araba programı aşağıdaki gibidir:

```
/*  NodeMCU Access Point (AP) modunda çalışmaktadır.
    Kablosuz ağlardan «Robot»a bağlanın. Şifre: 12345678
    Tarayıcıdan 192.168.4.1 adresine girin.
*/
#include <ESP8266WiFi.h> // NodeMCU kurulumuyla gelir.
#include <WiFiClient.h> // NodeMCU kurulumuyla gelir.
#include <ESP8266WebServer.h> // NodeMCU kurulumuyla gelir.
#include <WebSocketsServer.h> // WebSockets v2.3.6
#include <Hash.h> // NodeMCU kurulumuyla gelir.

const char* ssid = “Robot”;
const char* password = “12345678”; // Şifre en az sekiz haneli olmalı.

const byte IN1 = D0, IN2 = D1, IN3 = D2, IN4 = D3; // PWM pinleri. (enA ve enB’de jumper takılı).

int hiz = 150;
```

```
char karakter;

static const char PROGMEM INDEX_HTML[] = R"rawliteral(
<!DOCTYPE html>
<html>
<head>
<meta name = "viewport" content = "width = device-width, initial-scale = 1.0, maximum-scale = 1.0, user-scalable=0">
<title>Wifi Robot</title>
<style>
"body { background-color: #808080; font-family: Arial, Helvetica, Sans-Serif; Color: #000000; }"
#JD {
  text-align: center;
}
#JD {
  text-align: center;
  font-family: "Lucida Sans Unicode", "Lucida Grande", sans-serif;
  font-size: 24px;
}
.foot {
  text-align: center;
  font-family: "Comic Sans MS", cursive;
  font-size: 30px;
  color: #F00;
}
.button {
    border: none;
    color: white;
    padding: 17px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    cursor: pointer;
    border-radius: 12px;
  width: 100%;
}
.red {background-color: #F00;}
.green {background-color: #090;}
.yellow {background-color:#F90;}
.blue {background-color:#03C;}
</style>
<script>
```

```
var websock;
function start() {
  websock = new WebSocket('ws://' + window.location.hostname + ':81/');
  websock.onopen = function(evt) { console.log('websock open'); };
  websock.onclose = function(evt) { console.log('websock close'); };
  websock.onerror = function(evt) { console.log(evt); };
  websock.onmessage = function(evt) {
    console.log(evt);
    var e = document.getElementById('ledstatus');
    if (evt.karakter === 'ledon') {
      e.style.color = 'red';
    }
    else if (evt.karakter === 'ledoff') {
      e.style.color = 'black';
    }
    else {
      console.log('unknown event');
    }
  };
}
function buttonclick(e) {
  websock.send(e.id);
}
</script>
</head>
<body onload="javascript:start();">
 
<table width="100%" border="0">
  <tr>
    <td bgcolor="#FFFFFF" id="JD">Wifi Robot </td>
  </tr>
</table>
<table width="100" height="249" border="0" align="center">
    <td> </td>
    <td align="center" valign="middle"><form name="form1" method="post" action="">
      <label>
        <button id="a"  type="button" onclick="buttonclick(this);" class="button green">ileri</button>
      </label>
    </form></td>
    <td> </td>
  </tr>
    <td align="center" valign="middle"><form name="form1" method="post" action="">
      <label>
        <button id="c"  type="button" onclick="buttonclick(this);" class="button green">Sol  </
button>
```

```
var websock;
function start() {
  websock = new WebSocket('ws://' + window.location.hostname + ':81/');
  websock.onopen = function(evt) { console.log('websock open'); };
  websock.onclose = function(evt) { console.log('websock close'); };
  websock.onerror = function(evt) { console.log(evt); };
  websock.onmessage = function(evt) {
    console.log(evt);
    var e = document.getElementById('ledstatus');
    if (evt.karakter === 'ledon') {
      e.style.color = 'red';
    }
    else if (evt.karakter === 'ledoff') {
      e.style.color = 'black';
    }
    else {
      console.log('unknown event');
    }
  };
}
function buttonclick(e) {
  websock.send(e.id);
}
</script>
</head>
<body onload="javascript:start();">
 
<table width="100%" border="0">
  <tr>
    <td bgcolor="#FFFFFF" id="JD">Wifi Robot </td>
  </tr>
</table>
<table width="100" height="249" border="0" align="center">
    <td> </td>
    <td align="center" valign="middle"><form name="form1" method="post" action="">
      <label>

        <button id="a"  type="button" onclick="buttonclick(this);" class="button green">ileri</button>
      </label>
    </form></td>
    <td> </td>
  </tr>
    <td align="center" valign="middle"><form name="form1" method="post" action="">
      <label>
        <button id="c"  type="button" onclick="buttonclick(this);" class="button green">Sol  </
button>
```

```
      </label>
    </form></td>
  <td align=”center” valign=”middle”><form name=”form1” method=”post” action=””>
      <label>
        <button id=”e”  type=”button” onclick=”buttonclick(this);” class=”button red”>Dur  </
button>
      </label>
    </form></td>
    <td align=”center” valign=”middle”><form name=”form1” method=”post” action=””>
      <label>
        <button id=”d”  type=”button” onclick=”buttonclick(this);” class=”button green”>Sag  </
button>
      </label>
    </form></td>
  </tr>
   <td> </td>
    <td align=”center” valign=”middle”><form name=”form1” method=”post” action=””>
      <label>
        <button id=”b”  type=”button” onclick=”buttonclick(this);” class=”button green”>Geri </
button>
      </label>
    </form></td>
    <td> </td>
  </tr>
</table>
</body>
</html>
)rawliteral”;

WebSocketsServer webSocket = WebSocketsServer(81);
ESP8266WebServer server(80);

void setup() {
 Serial.begin(9600);

 pinMode(IN1, OUTPUT);
 pinMode(IN2, OUTPUT);
 pinMode(IN3, OUTPUT);
 pinMode(IN4, OUTPUT);

  WiFi.mode(WIFI_AP);           // Access point modunda.
 WiFi.softAP(ssid, password);  // Kablosuz ağı başlat.
```

```
  IPAddress IP = WiFi.softAPIP();
 server.on("/", [](){
  server.send(200, "text/html", INDEX_HTML);
  });

  server.begin();
  webSocket.begin();
  webSocket.onEvent(webSocketFonksiyonu);
}

void loop() {

if (karakter == 'a') { // ileri.
      analogWrite(IN1, hiz);
      analogWrite(IN2, 0);
      analogWrite(IN3, hiz);
      analogWrite(IN4, 0);
    } else if (karakter == 'b') { // Geri.
      analogWrite(IN1, 0);
      analogWrite(IN2, hiz);
      analogWrite(IN3, 0);
      analogWrite(IN4, hiz);
    } else if (karakter == 'c') { // Sol.
      analogWrite(IN1, hiz);
      analogWrite(IN2, 0);
      analogWrite(IN3, 0);
      analogWrite(IN4, hiz);
    } else if (karakter == 'd') { // Sağ.
      analogWrite(IN1, 0);
      analogWrite(IN2, hiz);
      analogWrite(IN3, hiz);
      analogWrite(IN4, 0);
    } else if (karakter == 'e') { // Dur.
      analogWrite(IN1, 0);
      analogWrite(IN2, 0);
      analogWrite(IN3, 0);
      analogWrite(IN4, 0);
    }

   webSocket.loop();
   server.handleClient();
}

void webSocketFonksiyonu(byte sayi, WStype_t type,byte * veri, size_t length){
  switch(type) {
```

```
    case WStype_DISCONNECTED:
      break;
    case WStype_CONNECTED:
      {IPAddress ip = webSocket.remoteIP(sayi);}
      break;
    case WStype_TEXT:
      karakter = veri[0];
      Serial.println(karakter);
      webSocket.broadcastTXT(veri, length);
      break;
    case WStype_BIN:
      hexdump(veri, length);
      webSocket.sendBIN(sayi, veri, length);
      break;
      default:
      break;
  }
}
```

İşlem Basamakları

1. İş sağlığı ve güvenliği tedbirlerine uyarak araç gerecinizi hazırlayınız.
2. Öğretmeninizden teslim aldığınız malzemelerin gerekli kontrollerini yapınız.
3. Görsel 2.99'daki devreyi araç üzerine kurunuz. Programı yükleyiniz.
4. Bilgisayar veya akıllı telefondan "Robot" isimli kablosuz ağa "12345678" şifresiyle bağlantı gerçekleştiriniz.
5. Tarayıcı adres çubuğuna "192.168.4.1" yazınız.
6. Web sayfası üzerindeki düğmelerle aracı yönlendiriniz.
7. Öğretmeninizden teslim aldığınız malzemeleri gerekli kontrollerini yaparak iade ediniz.

SIRA SİZDE

1. Araç korna çalacak şekilde donanım ve yazılımı düzenleyiniz.

Soru

1. NodeMCU üzerinde çalışan **web server**a kaç adet istemci bağlanabilir? Araştırınız.

NOT DEFTERİ

KAYNAKÇA

Adafruit Industries. (2022). Adafruit Optical Fingerprint Sensor. New York.

InvenSense Inc. (2013). MPU-6000 and MPU-6050 Product Specification. U.S.A.

KÜÇÜKKOÇ, İ. (2020). Algoritma Ve Programlamaya Giriş. Balıkesir.

Microchip Technology Inc. (2018). ATmega48A/PA/88A/PA/168A/PA/328/P.

Nordic Semiconductor. (2008). nRF24L01+ Single Chip 2.4GHz Transceiver Preliminary Product Specification. Trondheim.

Pololu Corporation. (2014). QTR-8A and QTR-8RC Reflectance Sensor Array User's Guide.

T.C. Millî Eğitim Bakanlığı. (2012). Mikro İşlemci Ve Mikrodenetleyiciler. Ankara.

GENEL AĞ KAYNAKÇASI

- https://www.arduino.cc/ (27.07.2021 saat 16.20)
- https://lastminuteengineers.com/ (24.04.2022 saat 22.14)
- https://dronebotworkshop.com/ (12.05.2022 saat 20.48)
- https://randomnerdtutorials.com/ (14.06.2022 saat 20.10)
- https://app.code2flow.com/ (21.06.2022 saat 10.35)
- https://learn.adafruit.com/ (15.06.2022 saat 18.12)
- https://maker.robotistan.com/ (04.06.2022 saat 10.32)
- http://www.temrinler.com/ (12.08.2021 saat 12.10)