Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA210-VET

Küçük Ölçekli Mesleki Eğitim ve Öğretim Ortaklık Projesi

Proje adı: "Mesleki Eğitimde Arduinoları Kullanma"

Proje kısaltması: "UsingARDinVET"

Proje No: "2023-1-RO01-KA210-VET-000156616"

***** *UsingARDinVET KILAVUZU* *****

Co-funded by the
Erasmus+ Programme
of the European Union

"Bu proje Erasmus+ Programı kapsamında Avrupa Komisyonu tarafından desteklenmektedir. Ancak burada yer alan görüşlerden Avrupa Komisyonu sorumlu tutulamaz."

“This project is Funded by the Erasmus+ Program of the European Union. However, European Commission cannot be held responsible for any use which may be made of the information contained there in”

Co-funded by the
Erasmus+ Programme
of the European Union

KOORDİNATÖR:

LICEUL TEHNOLOGIC ELENA CARAGIANI (Tecuci, Romania)

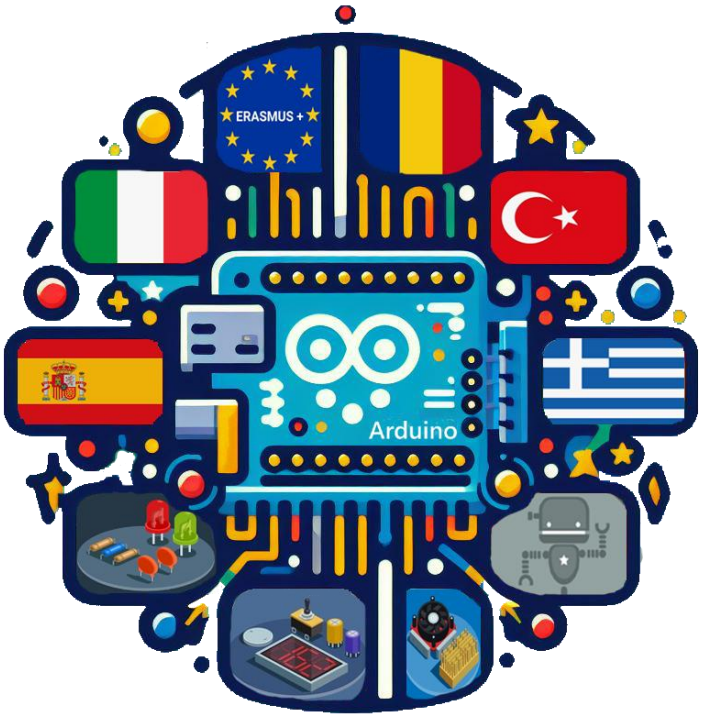
ORTAKLAR:

Ahi Evran Mesleki Eğitim Merkezi (Ankara, Türkiye)

2 EK Peiraia (Piraeus / Yunanistan)

Instituto De Educaiión Secundaria "María Molıner" (SEGOVIA, İspanya)

IPIAS G. GIORGI (Potenza, İtalya)



Co-funded by the Erasmus+ Programme of the European Union

Proje Özeti

“Mesleki Eğitimde Arduinoları Kullanma”

Amaçlar:
Bir şeyi öğrenmenin en iyi yolu onu yaparak ve deneyimleyerek öğrenmektir. Mesleki eğitimdeki en önemli eğitim materyalleri, mesleki eğitimdeki deney setleridir.
Mesleki eğitim okullarının müfredatları incelendiğinde, bu setlerle Arduino eğitimi vermenin zor olduğu görülmektedir. Bu sorunları aşmak, öğrencilerimize Arduino dersinde daha verimli bir ortam ve deney setleri sağlamak ve öğrenmenin daha kalıcı olmasını sağlamak için bu projeyi hazırladık.

Uygulamalar:
*5 Ulusötesi toplantı; 2 yıllık proje süresince 5 TPM düzenlenecektir. Bu toplantıların katılımcıları ortakların proje ekipleridir.
*"Projemiz mesleki eğitim okulları, elektronik, BİT endüstrileri, işgücü piyasalarıyla buluşuyor" çalıştayı ile proje sonuçları, ürünleri çalıştay katılımcılarına sunulacaktır.
-Proje ekibi ve araçlarının oluşturulması.
-Proje Web Sitesi.
-Eğitim Kitleri ve Seti.
-UsingARDinVET Rehberi.
-Eğitim Videoları.
-Proje DVD'si.
-5 Haber Mektubu.
-Erasmus ağaçları dikimi.

Sonuçlar:
-Öğrencilerin “Mesleki Eğitimde Arduino'ları Öğretme, Öğrenme, Kullanma” algısını değiştirmek.
-Arduino derslerindeki devamsızlık seviyesini azaltmak.
-Öğretmenlerin Arduinolar hakkında yenilikçi metodolojileri öğrenmesini sağlamak.
-Öğrencilerine daha iyi eğitim hizmetleri sunmak.
-Olumlu bir okul iklimi yaratmak ve öğrenme ortamını iyileştirmek.
-Okullarda Arduino atölyelerini ve derslerini daha iyi bir şekilde iyileştirmek.
-Mezunların istihdamını artırmak.
-Kültürlerarası diyalogların geliştirilmesi

İÇİNDEKİLER

Co-funded by the
Erasmus+ Programme
of the European Union

UsingARDinVET

PROJE MODÜLLERİ ve KİTLERİ

Co-funded by the
Erasmus+ Programme
of the European Union

Erasmus+ KA210-VET

Küçük Ölçekli Mesleki Eğitim ve Öğretim Ortaklık Projesi

Proje adı: "Meseleki Eğitimde Arduinoları Kullanma"

Proje kısaltması: "UsingARDinVET"

Proje No: "2023-1-RO01-KA210-VET-000156616"

ARDUINOLARA GİRİŞ

(ARDUINO'NUN TEMELLERİ)

ARDUNIO'YA GİRİŞ

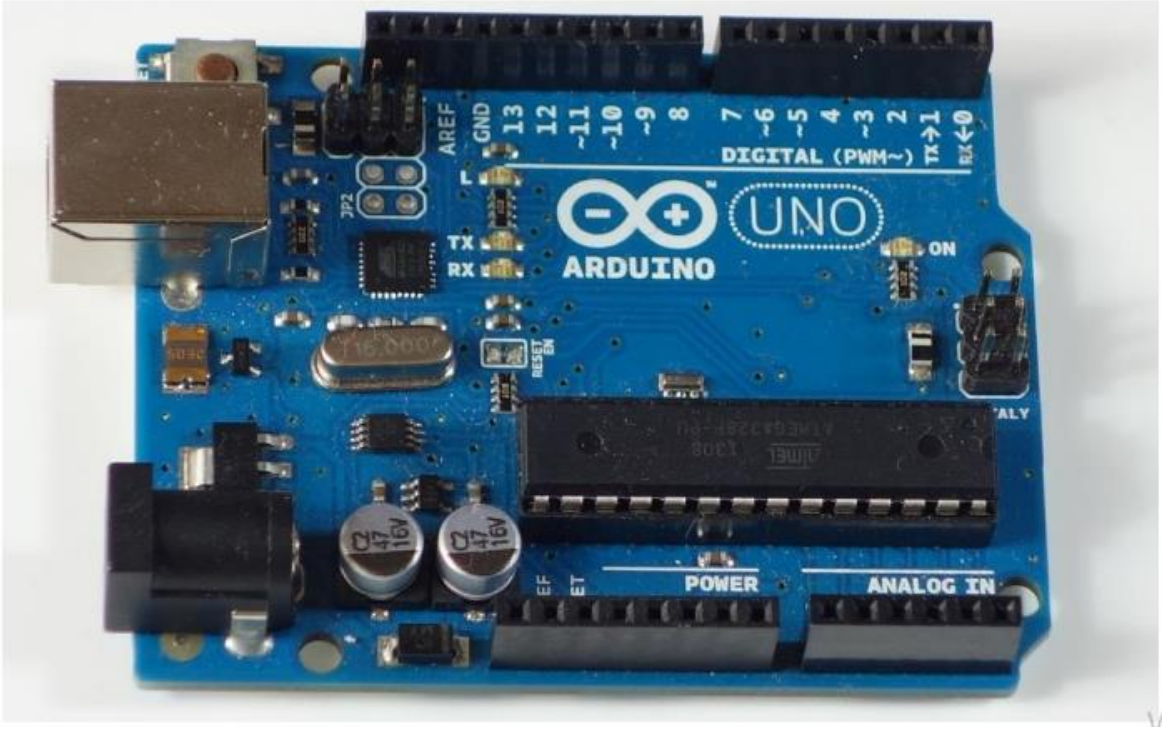
Arduino, donanım ve yazılıma dayalı, kullanımı kolay bir prototip platformudur (açık kaynak). Programlanabilen bir devre kartı (mikrodenetleyici olarak adlandırılır) ve bilgisayar kodunu fiziksel karta yazmak ve yüklemek için kullanılan Arduino IDE (Entegre Geliştirme Ortamı) adlı hazır bir yazılımdan oluşur.

Diğer bir deyişle Arduino, çevrenizdeki dünyadan bilgi okumak ve dış dünyaya komutlar göndermek için programlayabileceğiniz küçük bir bilgisayardır. İstediğinizi yapmak için Arduino'ya çeşitli cihaz ve bileşenler bağlayabilirsiniz. Onunla yapabileceklerinizin sınırı yoktur ve hayal gücünüzü kullanarak harika projeler gerçekleştirebilirsiniz.

Basit bir ifadeyle Arduino, çeşitli girdi ve çıktı biçimleriyle etkileşim kurmak için talimatlarınızla programlanabilen küçük bir bilgisayar sistemidir. Mevcut Arduino Uno modeli ortalama insan eline kıyasla oldukça küçüktür.

Ardunio Nedir?

Arduino kartı aşağıda gösterildiği şekildedir.

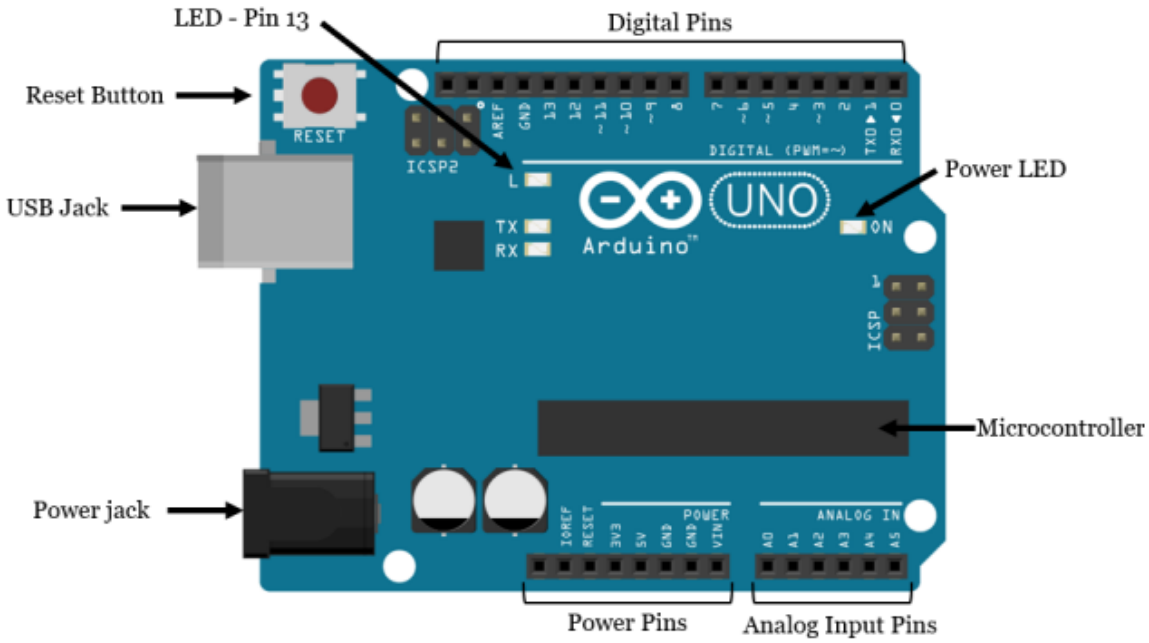


Temel olarak elektrik devrelerine bağlayabileceğiniz, beyni (mikrodenetleyici olarak da bilinir) olan küçük bir geliştirme kartıdır. Bu, girdilerin okunmasını – dışarıdan veri okunmasını – ve

çıktıların kontrol edilmesini – dışarıya bir komut gönderilmesini kolaylaştırır. Bu kartın beyni (Arduino Uno), Arduino'nuza ne yapacağını söyleyecek programlarınızı saklayabileceğiniz bir ATmega328p yongasıdır.

Arduino Uno Kartı Keşfetmek

Aşağıda etiketlenmiş bir Arduino kartı görebilirsiniz. Her parçanın ne yaptığını görelim.



• **Mikrodenetleyici:** ATmega328p, Arduino'nun beynidir. Arduino kartındaki her şey bu mikrodenetleyiciyi desteklemek içindir. Arduino'ya ne yapacağını söylemek için programlarınızı burada saklarsınız.

• **Dijital pinler:** Arduino'nun 0'dan 13'e kadar etiketlenmiş, giriş veya çıkış görevi görebilen 14 dijital pini vardır. Giriş olarak ayarlandığında bu pinler voltaj okuyabilir. Yalnızca iki durumu okuyabilirler: YÜKSEK veya DÜŞÜK. Çıkış olarak ayarlandığında bu pinler voltaj uygulayabilir. Yalnızca 5V (YÜKSEK) veya 0V (DÜŞÜK) uygulayabilirler.

• **PWM pinleri:** Bunlar ~ ile işaretlenmiş dijital pinlerdir (11, 10, 9, 6, 5 ve 3 numaralı pinler). PWM, "darbe genişliği modülasyonu" anlamına gelir ve dijital pinlerin değişen miktarlarda "sahte" voltaj çıkışına izin verir. PWM hakkında daha sonra daha fazlasını öğreneceksiniz.

• **TX ve RX pinleri:** T, "göndermek" ve R ise "almak" anlamına gelir. Arduino, seri bağlantı aracılığıyla diğer elektroniklerle iletişim kurmak için bu pinleri kullanır. Arduino yeni kod

yüklerken bilgisayarınızla iletişim kurmak için de bu pinleri kullanır. Gerekmedikçe bu pinleri seri iletişim dışındaki diğer görevler için kullanmaktan kaçının.

• **Dijital pin 13'e bağlı LED:** Arduino senaryolarının kolay hata ayıklaması için kullanışlıdır.

• **TX ve RX LED'leri:** Bilgisayar ve Arduino arasında bilgi alışverişi olduğunda bu ledler yanıp söner.

• **Analog pinler:** Analog pinler A0'dan A5'e kadar etiketlenir ve genellikle analog sensörleri okumak için kullanılır. 0 ile 5V arasında farklı voltaj miktarlarını okuyabilirler. Ayrıca dijital pinler gibi dijital çıkış/giriş pinleri olarak da kullanılabilirler.

• **Güç pinleri:** Arduino bu pinler üzerinden 3.3V veya 5V sağlar. Çoğu bileşenin çalışması için 3.3V veya 5V gerektiğinden bu gerçekten yararlıdır. "GND" olarak etiketlenen pinler topraklama pinleridir.

• **Reset butonu:** Bu butona bastığınızda Arduino'nuzda çalışmakta olan program yeniden başlar. Ayrıca, sıfırlama düğmesi görevi gören güç pinlerinin yanında bir Reset pini vardır. O pine küçük bir voltaj uyguladığınızda Arduino'yu sıfırlayacaktır.

• **Güç AÇIK LED'i:** Arduino'ya güç verildiğinden yanacaktır.

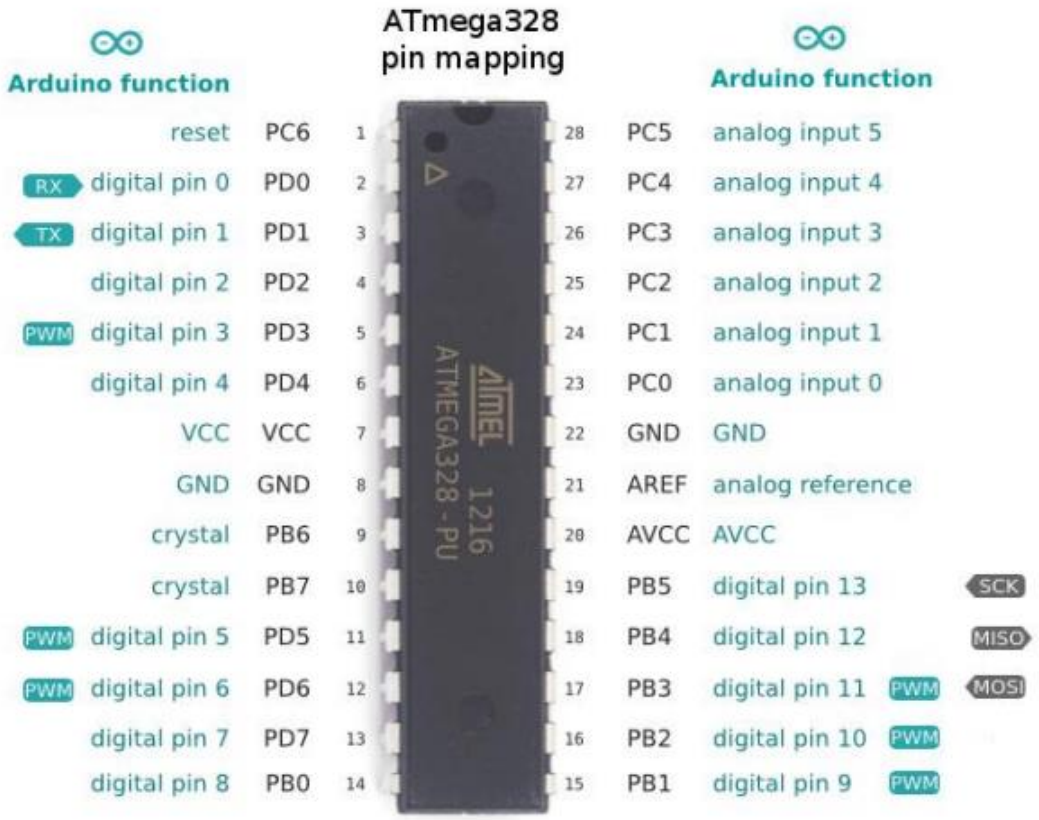
• **USB jakı:** Bilgisayarınızdan Arduino kartınıza program yüklemek için bir erkek USB A - erkek USB B kablosuna (aşağıdaki şekilde gösterilmiştir) ihtiyacınız vardır. Bu kablo aynı zamanda Arduino'nuza da güç sağlar.

• **Güç jakı:** Arduino'yu güç jakı üzerinden çalıştırabilirsiniz. Önerilen giriş voltajı 7V ila 12V'dir. Arduino'nuzu güçlendirmenin birkaç yolu vardır: örneğin; şarj edilebilir piller, tek kullanımlık piller, adaptörler, güneş paneli.

Arduino Özellikleri

Arduino Uno; Atmel Atmega 328P mikrodenetleyiciye sahiptir. Ayrıca USB bağlantı girişi, power jack girişi, reset butonu bulunmaktadır. Arduino bir mikrodenetleyicide olması gereken herşeye sahiptir.

Mikrodenetleyici	Atmega328P
Çalışma voltajı	5V
Giriş voltajı(önerilir)	7-12V
Giriş voltajı(sınır)	6-20V
Dijital giriş / çıkış pini	14
PWM giriş / çıkış pini	6
Analog giriş pini	6
Giriş / çıkış pini başına DC akımı	20mA
3.3V için DC akım	50mA
Flash bellek	32 KB
Sram	2KB
EEPROM	1 KB
Saat hızı	16 MHz
Uzunluk	68.6 mm
Genişlik	53.4 mm
Ağırlık	25 g
Şekil: Arduino Uno Özellikleri	



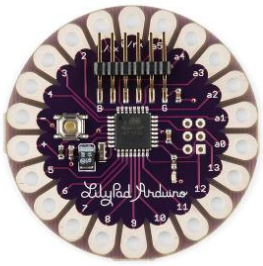
Şekil: Atmega 328P Pinleri

DİĞER ARDUNIO TÜRLERİ

ARDUINO MEGA

Üzerinde Atmega 2560 mikrodenetleyici bulunmaktadır. 54 dijital giriş-çıkış pinine, 16 analog girişe, 4 donanım seri portuna ve 16 mhz kristal osilatöre sahiptir. Hem USB hem de DC adaptör tarafından desteklenmektedir. Genel olarak Arduino UNO ile aynı özelliklere sahip olan kart, daha fazla pine sahip olduğu için daha büyük projelerde tercih edilmektedir.	

ARDUINO LILYPAD

Lilypad elbiselere ve kumaşlara dikilmek üzere tasarlanmıştır. Bu sayede giyilebilir olarak tasarlanabilecek ilginç projelerde kullanılabilir. Üzerinde Atmega 168V mikrodenetleyici bulunmaktadır.	

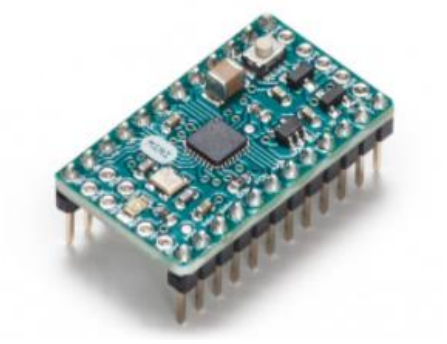
ARDUINO ETHERNET

İnternet bağlantılı projeler yapmak için bir Ethernet yongası ve bir Ethernet bağlantı noktasına sahiptir. Mikrodenetleyici olarak Atmega 328 modelinin bulunduğu kart üzerinde ayrıca bir SD-Kart yuvası bulunuyor.	

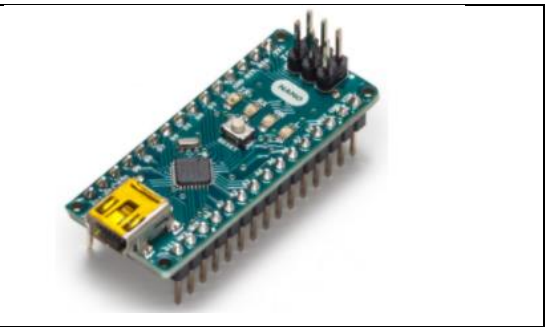
ARDUINO BLUETOOTH

Arduino BT üzerinde Bluetooth protokolü ile haberleşen uygulamalar yapmak için ideal bir Bluetooth modülü bulunmaktadır. Bu modül, Arduino'yu Bluetooth üzerinden programlamak için de kullanılabilir.	

ARDUINO MİNİ

Bir breadboard üzerinde çalıştırılmak veya başka bir tasarıma entegre edilmek üzere tasarlanmış bir Arduino modelidir. Üzerinde Atmega 168 veya Atmega 328 model mikrodenetleyici bulunmaktadır. Küçük boyutun özellikle önemli olduğu uygulamalar için idealdir.	

ARDUINO NANO

Atmega 328 veya Atmega 168 mikrodenetleyici, voltaj regülatörü, seriden USB'ye dönüştürücü chip, DC voltaj giriş portu ve mini USB portuna sahiptir.	

ARDUINO LEONARDO

Arduino Leonardo üzerinde bulunan Atmega 32u4 mikrodenetleyici içeren ve USB bağlantısı için ek bir chip gerektirmeyen Arduino kartlarından biridir. 20 adet dijital giriş/çıkış ve 12 adet analog giriş ile kart üzerindeki mikrodenetleyicinin yüzeye montaj kapağı bulunmaktadır. Leonardo, USB bağlantı yetenekleri sayesinde bilgisayara fare veya klavye olarak bağlanabilir.	

ARDUINO ESPLORA

Esplora, diğerlerinden farklı olarak çeşitli sensörler içeren bir Arduino kartıdır. Kart üzerinde bulunan sensörler sayesinde başka ilavelere ve aşırı elektronik bilgisine ihtiyaç duymadan birçok uygulamayı gerçekleştirmek mümkündür. Esplora, sürgülü potansiyometre, ışık ve ses sensörü, sıcaklık sensörü, ses üreteci, 2 eksenli mini analog joystick, 3 renkli LED ve ivme ölçer ile donatılmıştır. Esplora ayrıca Leonarda gibi Atmega 32U4 AVR mikrodenetleyici ile donatılmıştır. Mikro USB bağlantısı ile bilgisayara bağlanıldığında fare veya klavye görevi görebilen uygulamalar geliştirilebilir.	

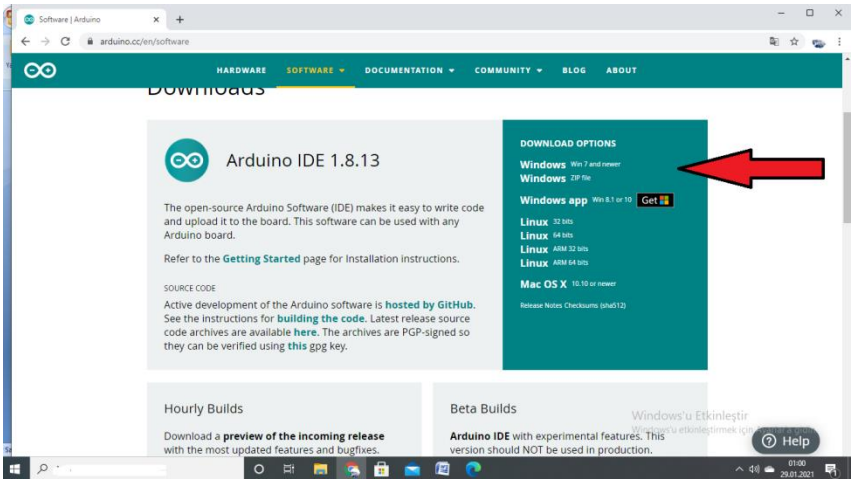
Arduino IDE'yi İndirmek

Arduino IDE (Entegre Geliştirme Ortamı), Arduino'ya ne yapacağını söyleyen programlarınızı geliştirdiğiniz yerdir.

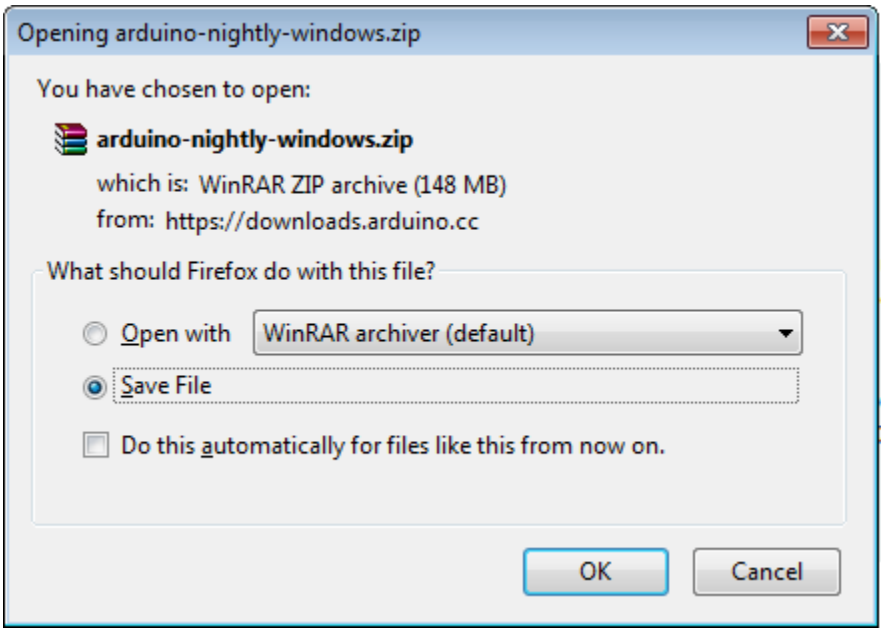
Windows için Arduino IDE'yi kurmak için talimatları izlemeliyiz.

Arduino IDE'yi kullanarak USB üzerinden ana çip olan ATmega328p'ye yeni programlar yükleyebilirsiniz.

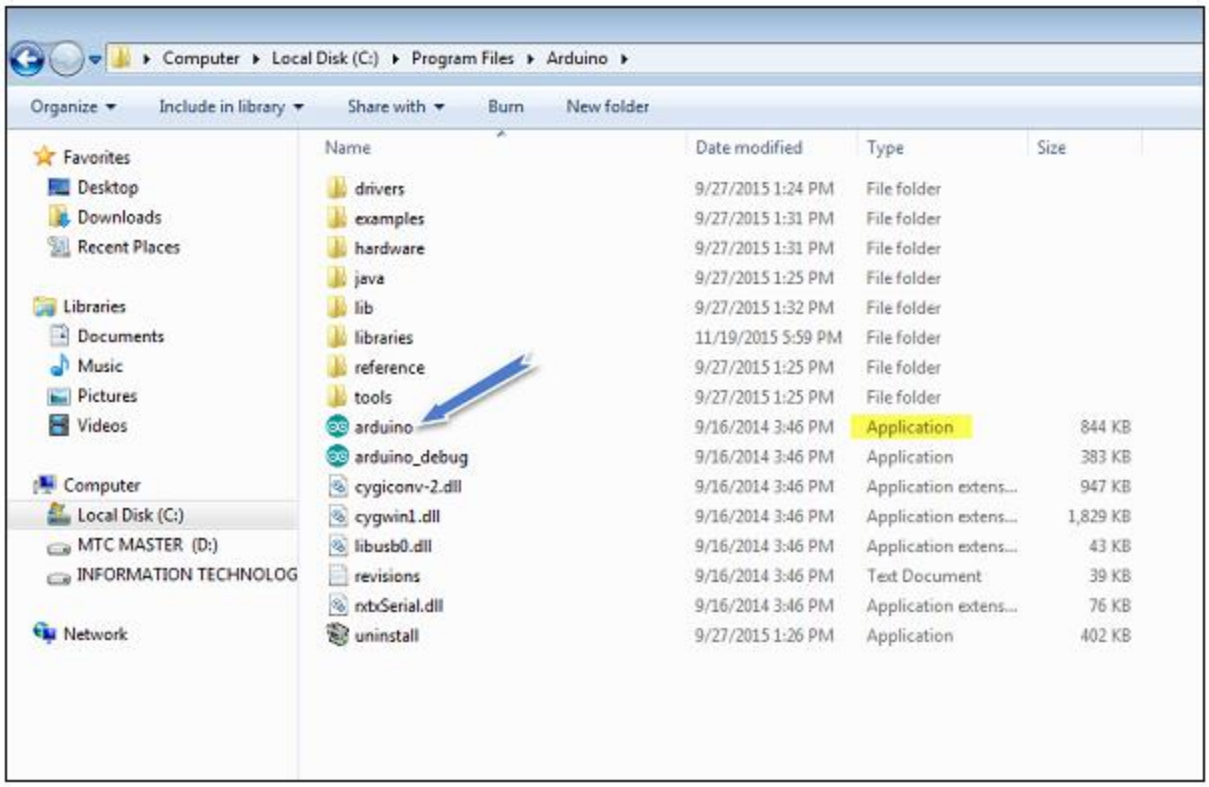
Arduino IDE'yi indirmek için lütfen aşağıdaki bağlantıya tıklayın:
https://www.arduino.cc/en/Main/Software



Kullanmakta olduğunuz İşletim Sistemini seçin ve indirin. Arduino IDE yazılımımız indirildikten sonra klasörü açmamız gerekiyor.



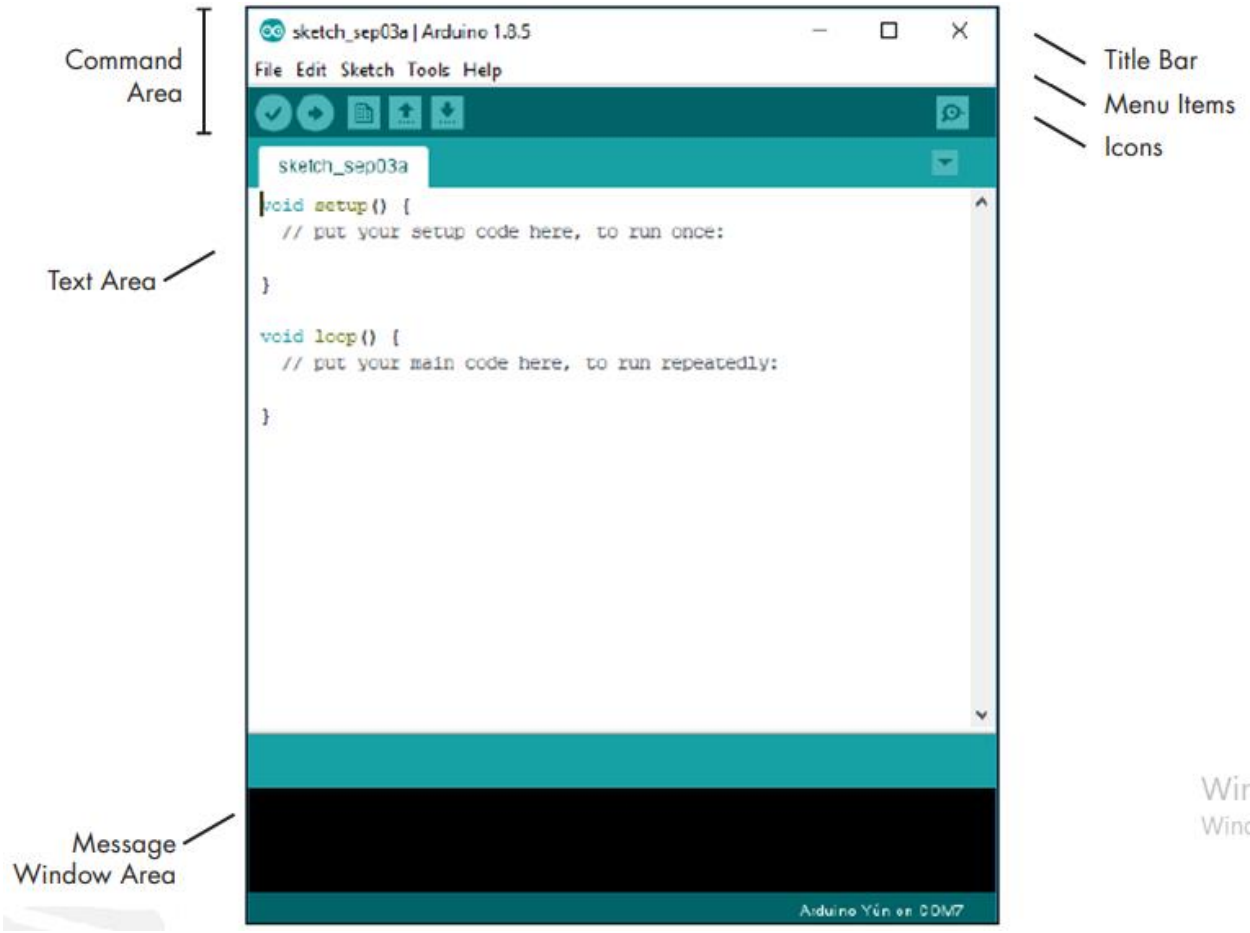
Klasörün içinde, sonsuzluk etiketli (application.exe) uygulama simgesini bulabiliriz. IDE'yi başlatmak için simgeye çift tıklayın. Ardından, Arduino IDE'yi kurmak için kurulum sihirbazını takip etmeniz yeterlidir.



Program Yazmak için Arduino IDE Penceresi

Arduino IDE'yi ilk açtığınızda, aşağıdaki şekle benzer bir şey görmelisiniz.

Aşağıdaki şekilde gösterildiği gibi, Arduino IDE basit bir kelime işlemciye benzer. IDE üç ana alana bölünmüştür: komut alanı, metin alanı ve mesaj penceresi alanı.



Menu Öğeleri

Herhangi bir kelime işlemci veya metin düzenleyicide olduğu gibi, seçeneklerini görüntülemek için menü öğelerinden birine tıklayabilirsiniz.
File(dosya): Senaryoları kaydetme, yükleme ve yazdırma seçeneklerini içerir;
Edit(düzen): Herhangi bir kelime işlemcide ortak olan normal kopyalama, yapıştırma ve arama işlevlerini içerir
Sketch(çizim): Bir panoya yüklemeden önce çiziminizi doğrulama işlevini ve bazı çizim klasörünü ve içe aktarma seçeneklerini içerir.
Tools(araçlar): Arduino kart tipini ve USB bağlantı noktasını seçme komutlarının yanı sıra çeşitli işlevleri içerir.
Help(yardım): Çeşitli ilgi alanlarına ve IDE sürümüne bağlantılar içerir.

Sketch Nedir?

Arduino projesi, belirli bir görevi gerçekleştirmek için oluşturduğunuz bir dizi talimattır; başka bir deyişle, eskiz bir programdır.
Sketch, Arduino'nun gerçekleştirmesi için bir dizi talimattan başka bir şey değildir. Arduino IDE kullanılarak oluşturulan eskizler .pde dosyaları olarak kaydedilir. Bir program oluşturmak için üç ana bölümün olması gerekir: Değişken tanımlamaları, Kurulum işlevi ve ana Döngü işlevi.

Arduino IDE Araç Çubuğu Butonları

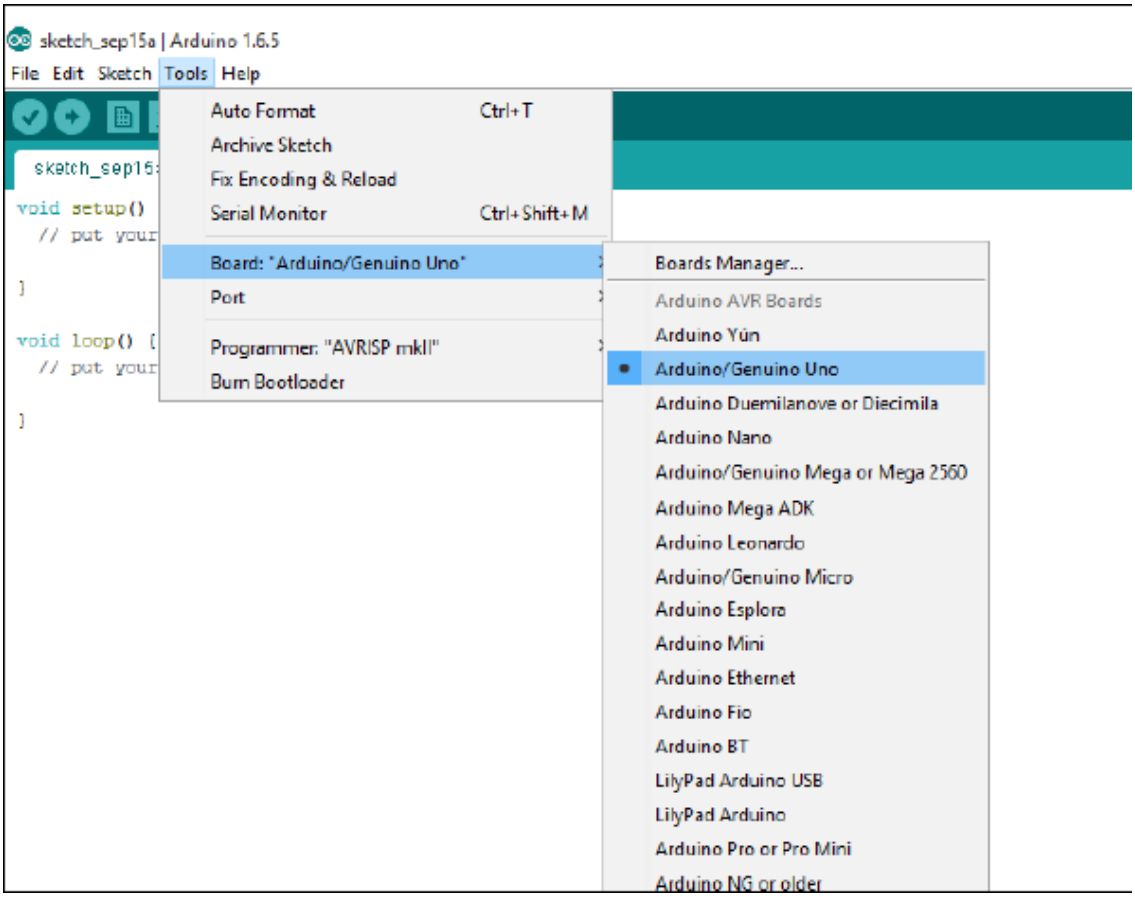
Menü araç çubuğunun altında altı simge bulunur. Adını görüntülemek için fareyi her simgenin üzerine getirin. Simgeler, soldan sağa aşağıdaki gibidir:

	**Doğrula (Derle) :** Arduino taslağının geçerli olduğunu ve herhangi bir programlama hatası içermediğini kontrol etmek için buna tıklayın.
	**Yeni(New):** Yeni bir pencerede yeni bir boş Sketch açmak için buna tıklayın.
	**Aç(Open):** Kaydedilmiş bir programı açmak için buna tıklayın.
	**Kaydet(Save):** Açık programı kaydetmek için buna tıklayın. Sketch daha önce kaydedilmemişse bir isim vermeni istenir.
	**Yükleme(Upload):** Onaylamak ve ardından programınızı Arduino panosuna yüklemek için buna tıklayın.
	**Seri Monitör:** Arduino'nuz ve IDE arasında veri göndermek ve almak için yeni bir pencere açmak için buna tıklayın.

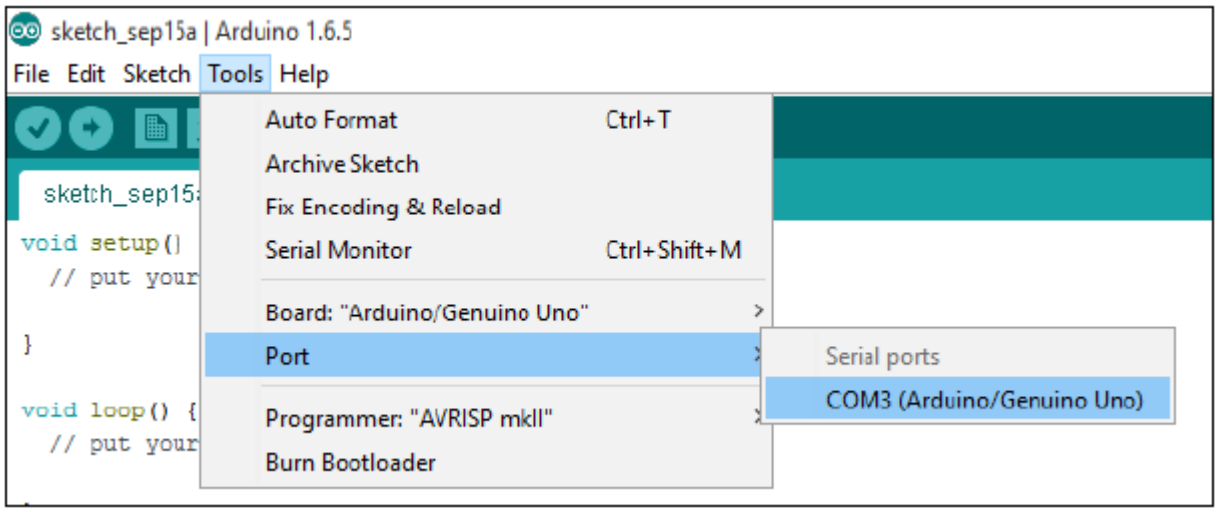
Ardunio'yu Bağlama

Arduino UNO'nuzu USB üzerinden bilgisayarınıza bağlayın.
Arduino'nuzu bir USB kablosu ile bağladıktan sonra Arduino IDE'nin doğru kartı seçtiğinden emin olmanız gerekir.

Arduino Uno kullanıyoruz, bu yüzden **Tools → Board→Arduino/Genuino Uno** seçili olmalıdır.



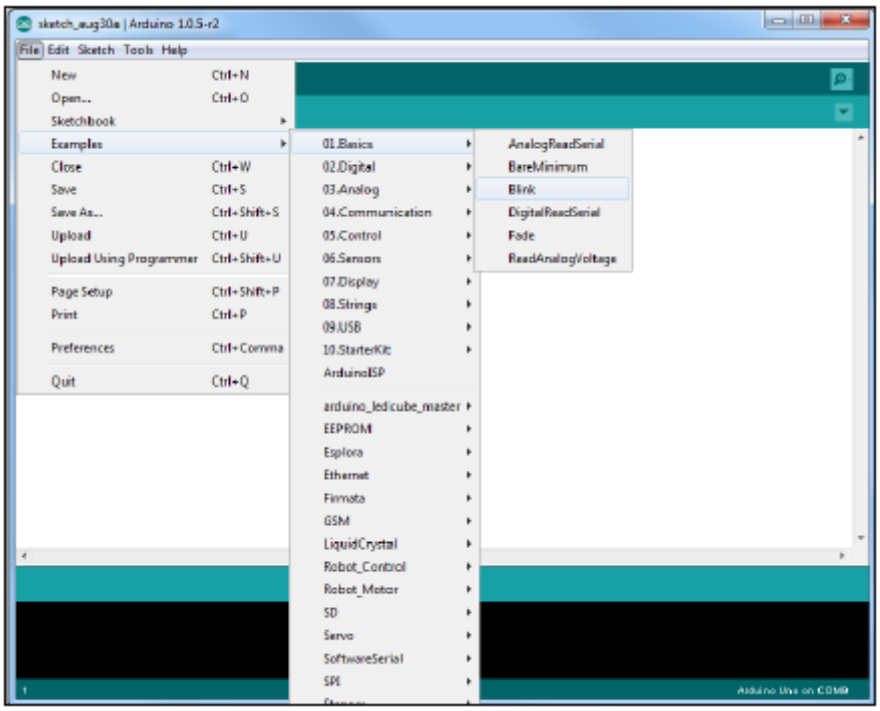
Port'a gidin ve doğru portu seçin. Ardından Arduino'nuzun bağlı olduğu seri portu seçmelisiniz. **Tools→Port**'a gidin ve doğru portu seçin.



Arduino Sketch Yükleme

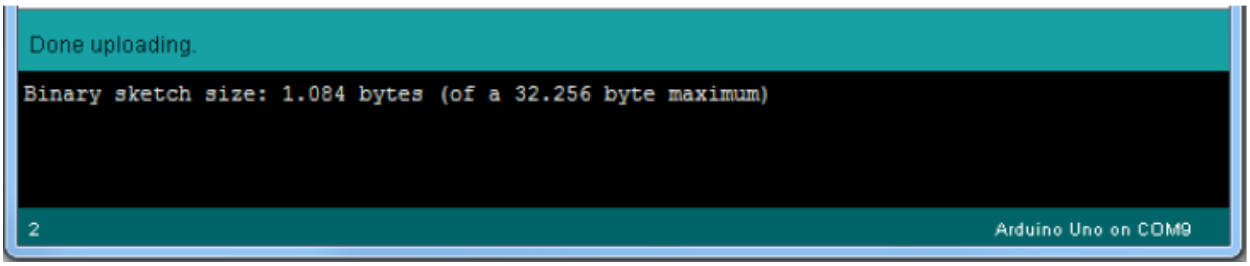
Arduino kartınıza nasıl kod yükleyeceğinizi göstermek için size basit bir örnek. Bu en temel örneklerden biridir. LED yakıp söndürür.

1. Arduino IDE'nizi açın.
2. **File→Examples→ 01.Basics→Blink e gidin**



Varsayılan olarak, Arduino IDE, Arduino UNO için önceden yapılandırılmış olarak gelir. Yükle düğmesini tıklayın ve birkaç saniye bekleyin.

Birkaç saniye sonra, **Yükleme tamamlandı (Done uploading)** mesajını görmelisiniz.



Bu kod, Arduino UNO'nuzdaki yerleşik LED'i yakıp söndürür (kırmızı renkle vurgulanır). Küçük LED'in bir saniye boyunca yandığını ve bir saniye söndüğünü görmelisiniz.

Çıkışı Kontrol Edin ve Girişi Okuyun

Bir Arduino kartı, dijital pinler, analog pinler ve PWM pinleri içerir.

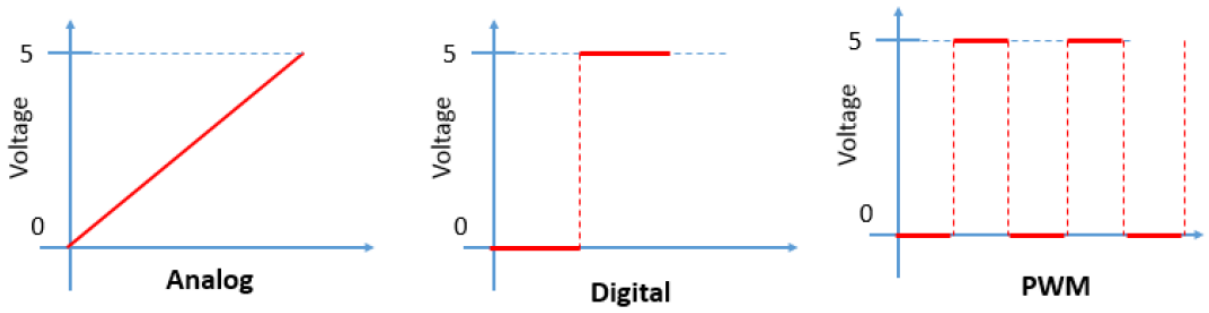
Dijital, analog ve PWM arasındaki fark

Dijital pinlerde, açık veya kapalı olmak üzere sadece iki olası durumunuz vardır. Bunlar ayrıca Yüksek veya Düşük, 1 veya 0 ve 5V veya 0V olarak da adlandırılabilir.

Örneğin, bir LED yanıyorsa durumu Yüksek veya 1 veya 5V'dir. Kapalıysa, Düşük veya 0 veya 0V değerine sahip olursunuz.

Analog pinlerde, 0 ile 1023 arasında sınırsız olası durumunuz vardır. Bu, sensör değerlerini okumanızı sağlar. Örneğin, ışık sensörü ile çok karanlıksa 1023, çok parlaksa 0 okursunuz Karanlık ile çok parlak arasında bir parlaklık varsa 0 ile 1023 arasında bir değer okursunuz.

PWM pinleri dijital pinlerdir, yani 0 veya 5V çıkış verirler. Ancak bu pinler “Pulse Width Modulation” (PWM) yapabildikleri için 0 ile 5V arasında “sahte” ara gerilim değerleri verebilirler. PWM, Arduino'nun çıkış voltajını titreştirerek değişen güç seviyelerini "simüle etmeye" izin verir.



Bir çıkışı kontrol etme

Dijital bir çıkışı kontrol etmek için digitalWrite() işlevini ve yazdığınız parantezler arasında, kontrol etmek istediğiniz pini ve ardından YÜKSEK veya DÜŞÜK kullanın.

Bir PWM pinini kontrol etmek için analogWrite() işlevini kullanırsınız ve parantezler arasına kontrol etmek istediğiniz pini ve 0 ile 255 arasında bir sayı yazarsınız.

Giriş okuma

Bir analog girişi okumak için analogRead() işlevini ve bir dijital giriş için digitalRead() işlevini kullanırsınız.

Not: Arduino öğrenmenin en iyi yolu pratik yapmaktır. Bu yüzden birçok proje yapın ve bir şeyler geliştirmeye başlayın.

Co-funded by the
Erasmus+ Programme
of the European Union

Erasmus+ KA210-VET

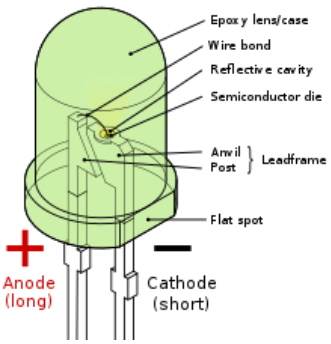
Küçük Ölçekli Mesleki Eğitim ve Öğretim Ortaklık Projesi

Proje adı: "Mesleki Eğitimde Arduinoları Kullanma"

Proje kısaltması: "UsingARDinVET"

Proje No: "2023-1-RO01-KA210-VET-000156616"

Arduino Giriş/Çıkış Modülü ve Kiti



Proje Planı

İlk projelerimize başlarken, yeni bir fikir bulduktan hemen sonra taslağınızı yazmak isteyebilirsiniz. Ancak yazmaya başlamadan önce birkaç temel hazırlık adımı sıralanmıştır. Sonuçta, Arduino kartınız bir zihin okuyucu değildir; kesin talimatlara ihtiyaç duyar ve bu talimatlar Arduino tarafından yürütülsede, küçük bir ayrıntıyı bile gözden kaçırırsanız sonuçlar beklediğiniz gibi olmayabilir.

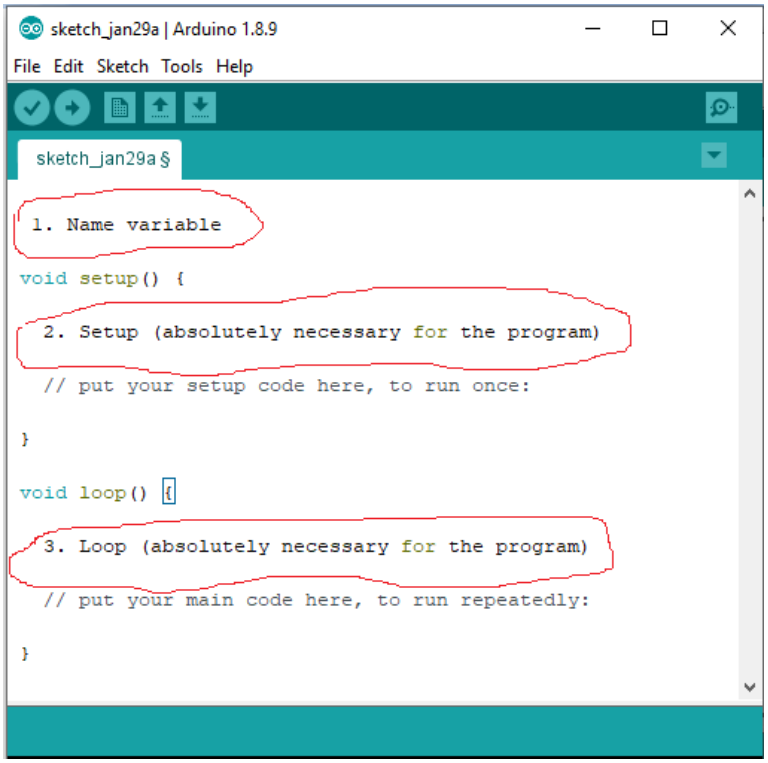
İster basitçe yanıp sönen bir proje, ister otomatikleştirilmiş bir demiryolu sinyali modeli oluşturuyor olun, başarının temeli ayrıntılı bir plandır. Arduino projelerinizi tasarlarken şu temel adımları izleyin:

1. Hedefinizi tanımlayın. Neye ulaşmak istediğinizi belirleyin.
2. Algoritmanızı yazın. Algoritma, projenizi nasıl gerçekleştireceğinizi açıklayan bir dizi talimattır. Algoritmanız, projenizin amacına ulaşmanız için gerekli adımları listeleyecektir.
3. Donanımınızı seçin. Arduino'ya nasıl bağlanacağını belirleyin.
4. Eskizinizi yazın. Arduino'ya ne yapacağını söyleyen ilk programınızı oluşturun.
5. Kabloyu bağlayın. Donanımınızı, devrenizi ve diğer öğelerinizi Arduino kartına bağlayın.
6. Test edin ve hata ayıklayın. Çalışıyor mu? Bu aşamada, ister çizimde, ister donanımda veya algoritmada olsun, hataları tanımlar ve nedenlerini bulursunuz.

Projenizi planlamak için ne kadar çok zaman harcarsanız, test ve hata ayıklama aşaması daha kolay olur.

Bir projenin temel yapısı

Arduino programı “sketch” olarak adlandırılır. Bir Sketch üç bölüme ayrılabilir.



1. Name variable(Değişken tanımlama-zorunlu değildir):
İlk bölümde programın elemanları isimlendirilmiştir. Bu kısım zorunlu değildir.

2. Setup(Kurulum-program için kesinlikle gereklidir):
Setup'daki kodlar sadece bir kez çalıştırılır.
Giriş ve çıkış pinleri belirlenmelidir.
Çıkış pinleri: Pin bir voltaj vermelidir. Örneğin: LED 'in yanması
Giriş pinleri: Kart bir voltaj okumalıdır. Örneğin: Bir anahtarın harekete geçirilmesi

3. Loop(Döngü-program için kesinlikle gereklidir):
Bu döngü kısmı kart tarafından sürekli olarak tekrarlanacaktır. Proje çalışır, sona gelince başa döner ve böyle devam eder.

Diğer Sözdizimi Kuralları

Arduino programları yazarken bu kurallara dikkat etmek gerekir. Aksi takdirde programımız başarısız olacaktır.

; (Noktalı virgül):

; (Noktalı virgül) bir ifadeyi bitirmek için kullanılır. Bir satırı noktalı virgülle bitirmeyi unutmak derleyici hatasına neden olur.

Örnek: int a=13;

{ } (Küme Parantezi):

Küme parantezleri Arduino programlama dilinin önemli bir parçasıdır. Birkaç farklı yapıda kullanılırlar ve bu bazen yeni başlayanlar için kafa karıştırıcı olabilir. Bir açılış küme parantezi "{" her zaman bir kapanış küme ayracı "}" izlemelidir.

Küme parantezinin ana kullanımları: Fonksiyonlar, Döngüler, Koşullu ifadeler

Örnek:

```
 void myfunction(datatype argument)
{           statements(s)

}
```

// (Tek satırlı yorum) ve /* */ (Çok satırlı yorum):

Bunlar, programın çalışma şekli hakkında kendinizi veya başkalarını bilgilendirmek için kullanılan programdaki satırlardır. Derleyici tarafından yok sayılırlar ve işlemciye aktarılmazlar, bu nedenle Atmega çipinde herhangi bir yer kaplamazlar. Yorumların tek amacı, programınızın nasıl çalıştığını anlamanıza (veya hatırlamanıza) yardımcı olmak veya programınızın nasıl çalıştığını başkalarına bildirmektir. Bir satırı yorum olarak işaretlemenin iki farklı yolu vardır:

Örnek:

x = 5; // Bu tek satırlık bir yorumdur. Eğik çizgiden sonraki her şey bir yorumdur

x = 5; **/*** Bu çok satırlı bir yorumdur ……

Bu, çok satırlı bir yorumun sonu. */

#define:

#define program derlenmeden önce programcının sabit bir değere bir isim vermesini sağlar. Arduino'da tanımlı sabitler çip üzerinde herhangi bir program hafızasında yer kaplamaz. Genel olarak, sabitleri tanımlamak için const anahtar sözcüğü tercih edilir ve bunun yerine #define kullanılmalıdır.

Örnek:

#define ledPin 3 // Derleyici, derleme zamanında ledPin'in değerini 3 değeriyle değiştirecektir.

#include:

#include projenize dış kitaplıkları dâhil etmek için kullanılır. Bu, programcıya geniş bir standart C kitaplığı grubuna (önceden oluşturulmuş işlev grupları) ve ayrıca özellikle Arduino için yazılmış kitaplıklara erişim sağlar.

Örnek:

#include <servo.h>

Ardunio – Veri Türleri

Veri türleri, değişkenleri veya farklı türlerdeki işlevleri bildirmek için kullanılır. Bir değişkenin türü, depolamada ne kadar yer kapladığını ve depolanan bit modelinin nasıl yorumlanacağını belirler.

Herhangi bir bilgiyi bellekte saklamak için kullanılan ve program akışı sırasında değerini değiştirebilen ifadelere değişkenler denir. Değişkenler sayılar, karakterler veya mantıksal ifadeler olabilir. Değişken tipine göre uygun veri tipi seçilmelidir. Değişkeni bellekte tanımlamada kullanılan veri tipine göre belirli bir alan tahsis edilir.

Aşağıdaki tablo Arduino programlama sırasında kullanacağınız tüm veri tiplerini sunmaktadır.

Veri tipi	Değeri
boolean	Doğru(True) veya yanlış(False)
char	-128 to 127
byte	0 to 255
unsigned char	0 to 255
int	-32,768 to 32,767
unsigned int	0 to 65,535
word	İşaretsiz int
long (or long int)	-2,147,483,648 to 2,147,483,647
unsigned long	0 to 4,294,967,295
float	-3.4028235E+38 to 3.4028235E+38
double	Float benzeri

Arduino - Değişkenler ve Sabitler

Değişken tanımlanırken değişkenin adı, değişkenin değeri ve değişkene uygun veri türü belirlenmelidir.

Tanım aşağıdaki örnekte görüldüğü gibi yapılmıştır:

int LED =12;

int=veri tipi **LED**= Değişken adı **12**=Değişken değeri

Arduino'nun kullandığı değişkenler, kapsam adı verilen bir özelliğe sahiptir. Kapsam, programın bir bölgesidir ve değişkenlerin bildirilebileceği üç yer vardır. Bunlar:

- Yerel değişkenler olarak adlandırılan bir fonksiyon veya blok içinde.
- Biçimsel parametreler olarak adlandırılan fonksiyon parametrelerinin tanımında.
- Global değişkenler olarak adlandırılan tüm fonksiyonların dışında.

Sabit, değişkenin davranışını değiştiren ve bir değişkeni *"salt okunur"* yapan bir değişken niteleyicidir. Bu, değişkenin kendi türündeki herhangi bir değişken gibi kullanılabileceği, ancak değerinin değiştirilemeyeceği anlamına gelir. Bir const değişkenine değer atamaya çalışırsanız derleyici hatası alırsınız.

const anahtar sözcüğüyle tanımlanan sabitler, diğer değişkenleri yöneten değişken kapsamı kurallarına uyar. Bu ve #define kullanmanın tuzakları, const anahtar sözcüğünü sabitleri tanımlamak için üstün bir yöntem yapar ve #define kullanımına göre tercih edilir.

Örnek:
const float pi = 3.14;

Not: #define veya const: Sayısal veya dize sabitleri oluşturmak için const veya #define kullanabilirsiniz. Diziler için const kullanmanız gerekecektir. Genel olarak sabitleri tanımlamak için #define yerine const tercih edilir.

Arduino – Operatörler

Operatör, derleyiciye belirli matematiksel veya mantıksal işlevleri gerçekleştirmesini söyleyen bir semboldür. Arduino'da aşağıdaki operatör türleri kullanılabilir.

Matematiksel Operatörler:

A değişkeninin 10'u ve B değişkeninin 20'yi tuttuğunu varsayalım, o zaman

Operatör adı	Operatör sembolü	Örnek
atama operatörü	=	A = B
toplama	+	A + B sonuç 30
çıkartma	-	A - B sonuç -10
çarpma	*	A * B sonuç 200
bölme	/	B / A sonuç 2

mod	%	B % A sonuç 0

Karşılaştırma Operatörleri:

A değişkeninin 10'u ve B değişkeninin 20'yi tuttuğunu varsayalım, o zaman

Operatör adı	Operatör sembolü	Örnek
eşit eşit	==	(A == B) yanlış (false)
eşit değil	!=	(A != B) doğru(true)
küçüktür	<	(A < B) doğru(true)
büyüktür	>	(A > B) yanlış (false)
küçük eşit	<=	(A <= B) doğru(true)
Büyük eşit	>=	(A >= B) yanlış (false)

Mantıksal Operatörler

A değişkeninin 10'u ve B değişkeninin 20'yi tuttuğunu varsayalım, o zaman

Operatör adı	Operatör sembolü	Örnek
and	&&	(A && B) doğru(true)
or	||	(A || B) doğru(true)
not	!	!(A && B) yanlış (false)

Bitsel Operatörler

A değişkeninin 60'ı ve B değişkeninin 13'yi tuttuğunu varsayalım, o zaman

Operatör adı	Operatör sembolü	Örnek
and	&	(A & B) sonuç 12 (0000 1100)
or	|	(A | B) sonuç 61 (0011 1101)
xor	^	(A ^ B) sonuç 49 (0011 0001)
not	~	(~A) sonuç -60 (1100 0011)
shift left	<<	A << 2 sonuç 240 (1111 0000)
shift right	>>	A >> 2 sonuç15 (0000 1111)

Arduino - I/O Fonksiyonları (Komutlar)

Fonksiyonlar, programların bireysel görevleri gerçekleştirecek şekilde yapılandırılmasına izin verir. Bir programda aynı eylemi birden çok kez gerçekleştirmeniz gerektiğinde fonksiyonlar kullanılır. Aşağıdaki devrelerde ve Arduino programlarında gerçek program örneklerini gösterdiğimizde daha anlaşılır olacaktır. Çeşitli komut işlevlerini açıklamaya yardımcı olmak için bunları ayrı komutlara ayırdık.

Arduino kartındaki pinler giriş veya çıkış olarak yapılandırılabilir. Bu modlarda pinlerin işleyişini anlatacağız. Arduino analog pinlerinin çoğunluğunun dijital pinlerle tamamen aynı şekilde yapılandırılabileceğini ve kullanılabileceğini belirtmek önemlidir.

pinMode() Fonksiyonu:

pinMode() fonksiyonu, belirli bir pini giriş veya çıkış olarak davranacak şekilde yapılandırmak için kullanılır. INPUT_PULLUP modu ile dahili pull-up dirençlerini etkinleştirmek mümkündür.

Kullanımı: pinMode(pin, mod)

```
Void setup () {
pinMode (pin , mod);
        }
```

Parametreler:
pin: Arduino pin numarası.
mod: INPUT, OUTPUT veya INPUT_PULLUP.

Örnekler:

```
pinMode(13, OUTPUT);      // dijital pin 13'ü çıkış olarak ayarlar
pinMode(5, INPUT);        // dijital pin 5'i giriş olarak ayarlar
```

digitalWrite() Fonksiyonu:

digitalWrite() fonksiyonu, bir dijital pine YÜKSEK veya DÜŞÜK bir değer yazmak için kullanılır. Pin, pinMode() ile bir ÇIKIŞ olarak yapılandırılmışsa, voltajı ilgili değere ayarlanacaktır. YÜKSEK için 5V, DÜŞÜK için 0V (toprak). Pin bir GİRİŞ olarak yapılandırılmışsa, digitalWrite() giriş pinindeki dahili çekmeyi etkinleştirir (YÜKSEK) veya devre dışı bırakır (DÜŞÜK). Dahili pull-up direncini etkinleştirmek için pinMode() öğesini INPUT_PULLUP olarak ayarlamanız önerilir.

pinMode() öğesini OUTPUT olarak ayarlamazsanız ve bir LED'i bir pine bağlamazsanız, digitalWrite(HIGH) çağrılırken LED loş görünebilir. pinMode() açıkça ayarlanmadan, digitalWrite() büyük bir akım sınırlayıcı direnç gibi davranan dahili pull-up direncini etkinleştirmiş olacaktır.

Kullanımı:

```
digitalWrite (pin ,değer);
```

pin: Modunu ayarlamak istediğiniz Ardunio pin numarası.

değer: HIGH (1) veya LOW(0).

Örnek:

```
digitalWrite(LED, HIGH);        // Led yanar

digitalWrite(LED, LOW);         // Led söner
```

digitalRead()Fonksiyonu:

digitalRead() fonksiyonu, değeri YÜKSEK veya DÜŞÜK olarak belirtilen bir dijital pinden okur.

Kullanımı:

digitalRead(pin);

pin: Okumak istediğiniz Arduino pin numarası.

Örnekler:

değer = digitalRead(inPin); // giriş pinini oku

Not: Analog giriş olarak adlandırılan A0, A1, vb., pinleri dijital pinler olarak kullanılabilir.

delay() Fonksiyonu:

delay() fonksiyonu, programı parametre olarak belirtilen süre boyunca (milisaniye cinsinden) duraklatır. (Bir saniyede 1000 milisaniye vardır.)

Kullanımı: delay(ms);

ms: duraklatılacak milisaniye sayısı. İzin verilen veri türleri: işaretsiz long

Örnekler:

```
delay(1000);                    // 1 saniye bekletir
delay(2000);                    // 2 saniye bekletir
```

analogWrite() Fonksiyonu:

analogWrite() fonksiyonu, bir pine bir analog değer (PWM dalgası) yazar. Bir LED'i değişen parlaklıklarda yakmak veya bir motoru çeşitli hızlarda sürmek için kullanılabilir. analogWrite() çağrısından sonra pin, aynı pin üzerinde bir sonraki analogWrite() çağrısına (veya digitalRead() veya digitalWrite() çağrısına) kadar belirtilen görev döngüsünün sabit bir dikdörtgen dalgasını üretecektir.

Örnekler:

Potansiyometreden okunan değerle orantılı olarak çıkışı LED'e ayarlar.

```
değer = analogRead(analogPin);  // giriş pinini oku
analogWrite(ledPin, değer /4);  // analogRead değerleri 0'dan 1023'e, analogWrite değerleri 0'dan 255'e
```

analogRead( ) Fonksiyonu:

Arduino, pinlerinden birine voltaj uygulanıp uygulanmadığını tespit edip digitalRead() fonksiyonu ile raporlayabilmektedir. (Bir nesnenin varlığını algılayan) bir açma/kapama sensörü ile değeri sürekli değişen bir analog sensör arasında fark vardır. Bu tip sensörü okumak için farklı bir pin tipine ihtiyacımız var.

Arduino kartının sağ alt kısmında "Analog In" olarak işaretlenmiş altı pin göreceksiniz. Bu özel pinler sadece kendilerine uygulanan bir voltajın olup olmadığını değil, değerini de söyler. AnalogRead() fonksiyonunu kullanarak pinlerden birine uygulanan voltajı okuyabiliriz.

Bu işlev, 0 ile 5 volt arasındaki voltajları temsil eden 0 ile 1023 arasında bir sayı döndürür. Örneğin, 0 numaralı pine uygulanan 2,5 V'luk bir voltaj varsa, analogRead(0) 512 değerini döndürür.

Kullanımı: analogRead(pin);

pin: 0'dan 5'e kadar okunacak analog giriş pininin numarası.

Örnek:

```
değer= analogRead(analogPin);      // giriş pinini oku
Serial.println(değer);             // değeri Seri monitöre yaz
```

If Komutu:

if deyimi bir koşulu kontrol eder ve koşul "doğru" ise aşağıdaki deyimi veya deyim kümesini yürütür.

Kullanımı:

```
if (koşul) {
        // koşul doğru olduğunda yürütülecek işlemler
}
```

koşul: bir boolean ifadesi (yani, doğru veya yanlış olabilir).

Örnekler: Bir if ifadesinden sonra parantezler atlanabilir. Bu yapılırsa, sonraki satır (noktalı virgülle tanımlanır) tek koşullu ifade olur.

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

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

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

if (x > 120) {
  digitalWrite(LEDpin1, HIGH);
  digitalWrite(LEDpin2, HIGH);
}
            // hepsi doğru
```

if-else Komutu:

if…else, birden çok koşulun gruplandırılmasına izin vererek, kod akışı üzerinde temel if ifadesinden daha fazla kontrol sağlar. If deyimindeki koşul yanlış(false) ile sonuçlanırsa, else yan tümcesi (eğer varsa) yürütülür. Else, aynı anda birden fazla, birbirini dışlayan testin çalıştırılabilmesi için bir başka if koşuluna geçebilir.

Doğru koşulla karşılaşılıncaya kadar her koşul bir sonrakine geçecektir. Bir koşul doğru olduğunda ilgili kod bloğu çalıştırılır ve program daha sonra tüm if/else yapısını izleyen satıra atlar. Hiçbir koşul doğru değilse, varsa varsayılan else bloğu yürütülür.

Kullanımı:

```
if (koşul DOĞRU ise) {
  // A işini yap
}
else
  //eğer YANLIŞ ise,  B işini yap
}
```

```
if (koşul1) {
  // A işini yap
}
else if (koşul2) {
  // B işini yap
}
else {
  // C işini yap
}
```

Örnekler: Aşağıda, sıcaklık sensörü sistemi için yazılmış koddan bir alıntı bulunmaktadır.

```
if (sıcaklık >= 70) {
  // Tehlike! Sistemi kapatın.

}
```

```
else if (sıcaklık >= 60) { // 60 <= sıcaklık < 70
  // Uyarı! Kullanıcının dikkati gereklidir.

}
else { // sıcaklık< 60
  // Güvenli! Varsayılan işlemleri yap
}
```

for Komutu:

For ifadesi, küme parantezleri içine alınmış bir ifade bloğunu tekrarlamak için kullanılır. Bir artış sayacı genellikle döngüyü artırmak ve sonlandırmak için kullanılır. for ifadesi, herhangi bir tekrarlayan işlem için kullanışlıdır ve genellikle veri/pin koleksiyonları üzerinde çalışmak için dizilerle birlikte kullanılır.

Kullanımı:

```
for (başlangıç değeri; koşul; artış miktarı) {
        // yapılacaklar;
}
```

Parametreler:

başlangıç: önce ve tam olarak bir kez olur.

koşul: Döngü boyunca her seferinde koşul test edilir; doğruysa, deyim bloğu ve artış yürütülürse, koşul yeniden test edilir. Koşul yanlış olduğunda döngü sona erer.

artış: koşul doğru olduğunda döngü boyunca her seferinde yürütülür.

Örnekler:

```
for (int i = 0; i <= 255; i++)          {
   analogWrite(PWMpin, i);              }

for (int x = 2; x < 100; x = x * 1.5) {
      println(x);                      }

for (int i = 0; i > -1; i = i + x)   {
  analogWrite(PWMpin, i);        }
```

switch...case Komutu:

If ifadelerinde olduğu gibi, switch/case, programcıların çeşitli koşullarda yürütülmesi gereken farklı kodlar belirlemesine izin vererek programların akışını kontrol eder. Özellikle, bir switch ifadesi, bir değişkenin değerini case ifadelerinde belirtilen değerlerle karşılaştırır. Değeri değişkeninkiyle eşleşen bir case ifadesi bulunduğunda, o case ifadesindeki kod çalıştırılır.

break anahtar sözcüğü, switch deyiminden çıkar ve genellikle her durumun sonunda kullanılır. Bir break ifadesi olmadan, switch ifadesi bir break veya switch ifadesinin sonuna ulaşılana kadar aşağıdaki ifadeleri yürütmeye devam eder.

Kullanımı:

```
switch (var) {
  case sabit1:
    // yapılacaklar
    break;
  case sabit2:
    // yapılacaklar
    break;
  default:
    // yapılacaklar
    break;
}
```

Örnek Kod:

```
switch (değişken) {
  case 1:
    // değişken 1'e eşit olduğunda
yapılacaklar
    break;
  case 2:
    // değişken 2'e eşit olduğunda yapılacaklar
    break;
  default:
    // başka hiçbir şey eşleşmiyorsa,
varsayılanı yapın
    // varsayılan isteğe bağlıdır
    break;
}
```

var: değeri çeşitli durumlarla karşılaştırılacak bir değişken. İzin verilen veri türleri: int, char.

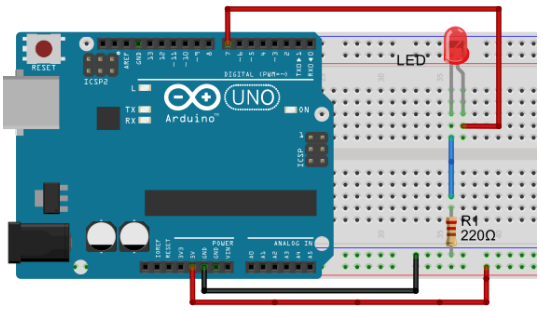
sabit1,sabit2: sabitler. İzin verilen veri türleri: int, char.

Not:

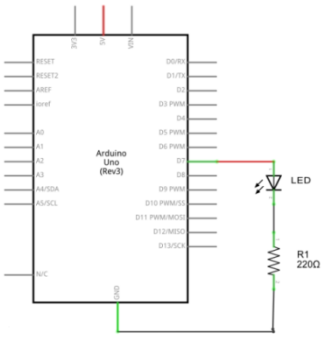
Arduino devreleri iki şekilde gösterilmiştir;

1-) Breadboard görünümü

2-) Şematik görünüm



1-) Breadboard görünümü



2-) Şematik görünüm

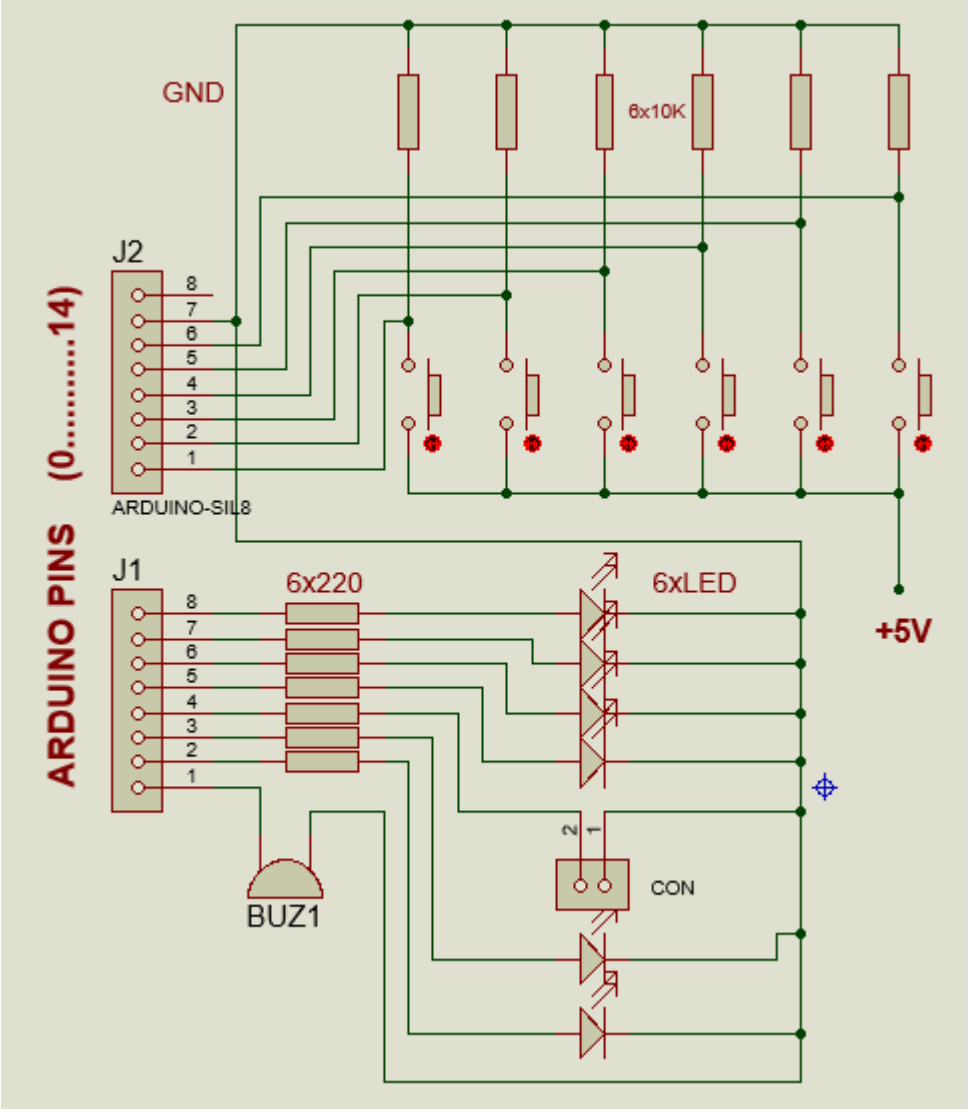
Daha çok kullanılan Breadboard görünümünü kullanacağız.

Yeter konuşma! Hadi bir şeyler yapalım!

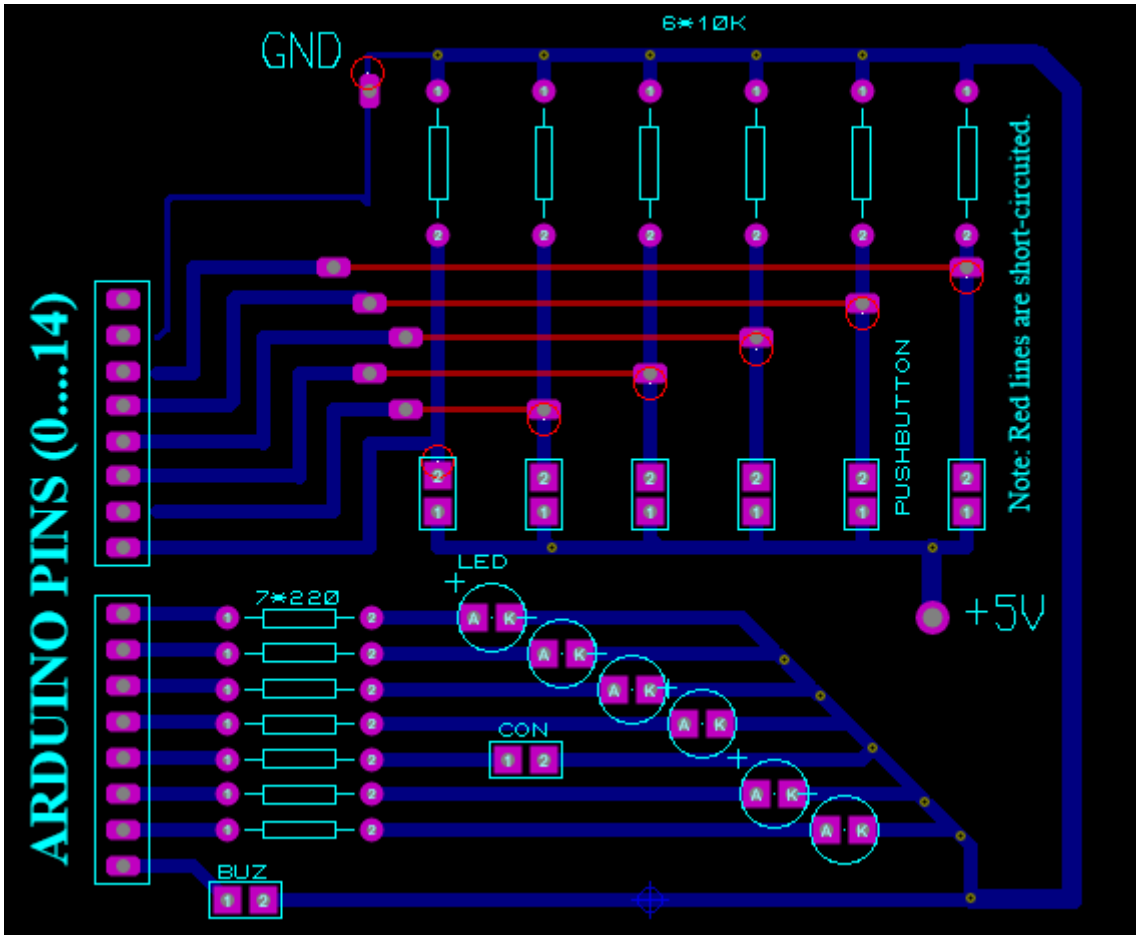
Arduino Buton Devreleri ve LED Modül Kiti

Arduino üzerindeki pinlerin çoğu giriş veya çıkış olarak konfigüre edilebilir. Buton ve LED Modül Kiti, öğrencilerin Arduino'ların I/O sistemlerini öğrenmelerini sağlamak için UsingARDinVET 'in ilk eğitim Kit'idir. Bu nedenle I/O Arduino eğitim KIT olarak adlandırılabilir. Bu eğitim kitinde aşağıda görüldüğü gibi 6 adet buton giriş olarak Arduino'ya bağlanmıştır. Ve bir buzzer, 2 pinli konnektör ve 6 LED çıkış olarak bağlanır.

Öğrenciler bu eğitim setini Arduino'nun I/O sistemlerini öğrenmek için kullanabilecekleri için bu eğitim seti Arduino devresini daha kolay test etmelerini sağlayabilir. Ayrıca Arduino devrelerini ve deneylerini test etmek için bir devre tahtası kullanabilirler.



Şekil: Buton ve LED Modül Kitinin Açık Devresi



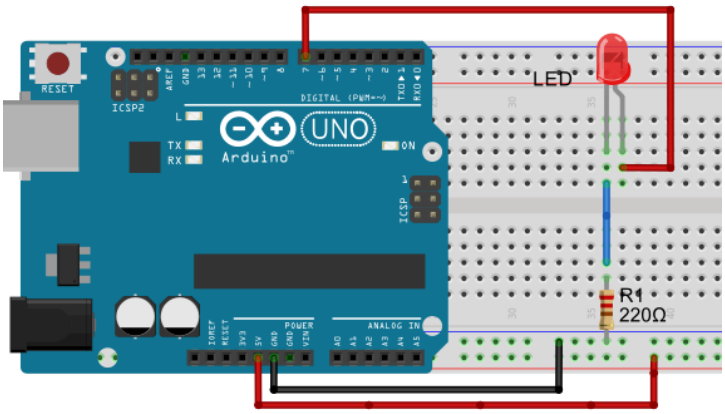
Şekil: Buton ve LED Modül Kitinin PCB Şeması

Örnek Arduio Devreleri ve Programları

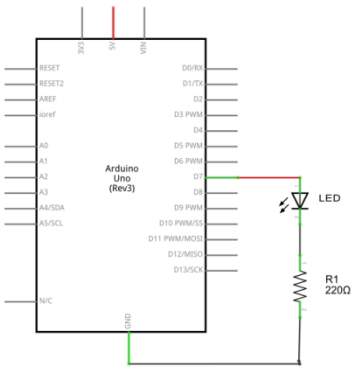
Devre 1:

Devre Adı: Led yakma söndürme .

Devre Açıklaması: Arduino'nun 13. pinine bir LED bağlanmıştır. LED 1 saniye aralıklarla sürekli olarak yanıp söner.



1-) Breadboard görünümü



2-) Şematik görünümü

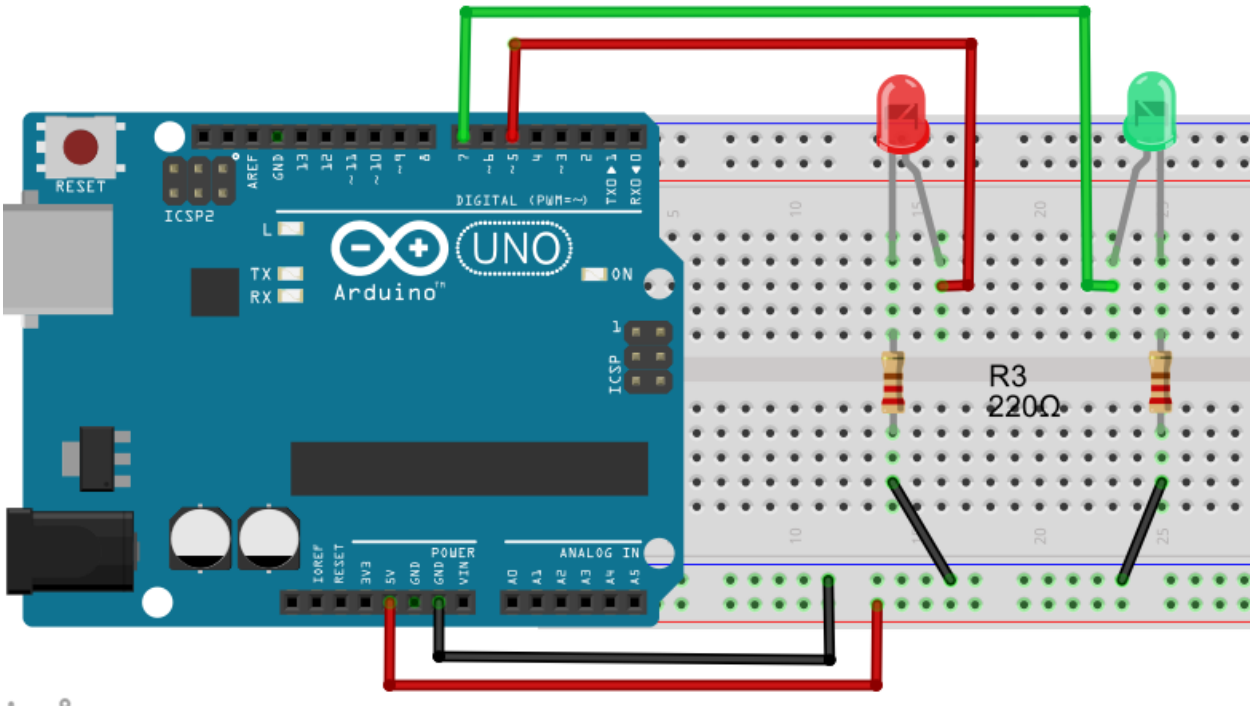
/* Yanıp sönen LED Programı, LED'i açma ve kapatma */

```
int led = 7;                  // led isimli tamsayı değişken tanımlandı, değer verildi
void setup() {                // setup() fonksiyonu sadece bir defa yürütülür
pinMode(led, OUTPUT);         // led PIN'i dijital çıkış olarak ayarlandı
}

void loop() {                 // loop() methodu tekrarlanır
digitalWrite(led, HIGH);      // led'i yakar
delay(1000);                  // programı 1000 milisaniye için durdur
digitalWrite(led, LOW);       // led'i söndürür
delay(1000);                  // programı 1000 milisaniye için durdur
}
```

Devre 2:

Devre Adı: 2 LED'i sırayla yakıp söndürme.

Devre Açıklaması: 2 LED 2 saniye aralıklarla sırayla yanıp söner.

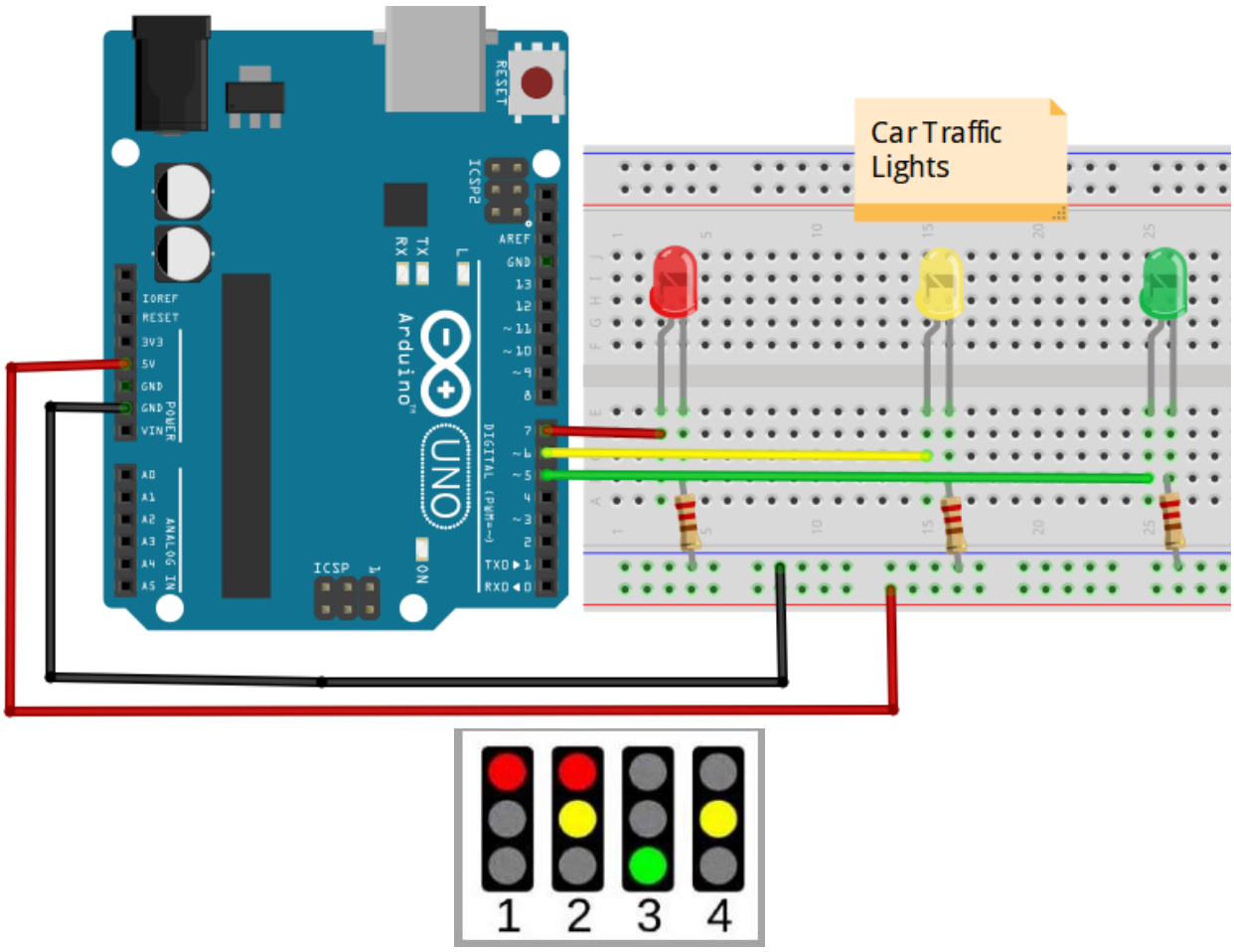



```
/*    Flip Flop     */

int greenLED=5;                    // greenLED isimli değişken tanımlandı, değer verildi
int redLED=7;                      // redLED isimli değişken tanımlandı, değer verildi

void setup()  {
pinMode(greenLED, OUTPUT);         // green LED pin dijital çıkış olarak ayarlandı
pinMode(redLED, OUTPUT);           // red LED pin dijital çıkış olarak ayarlandı
                }

void loop(){
digitalWrite(greenLED,  HIGH);     // green LED'i yak
digitalWrite(redLED,  LOW);        // red LED'i söndür
delay(2000);                       // programı 1000 milisaniye için durdur

digitalWrite(greenLED,  LOW);      // green LED'i söndür
digitalWrite(redLED,  HIGH);       // red LED'i yak
delay(2000);
                }
```

Devre 3:

Devre Adı: Trafik Lambası.

Devre Açıklaması: Bu projede, bir trafik ışığı sistemi kuracağız. Farklı renklerde (yeşil, sarı ve kırmızı) 3 adet led bulunmaktadır.



/* Trafik Lambası */

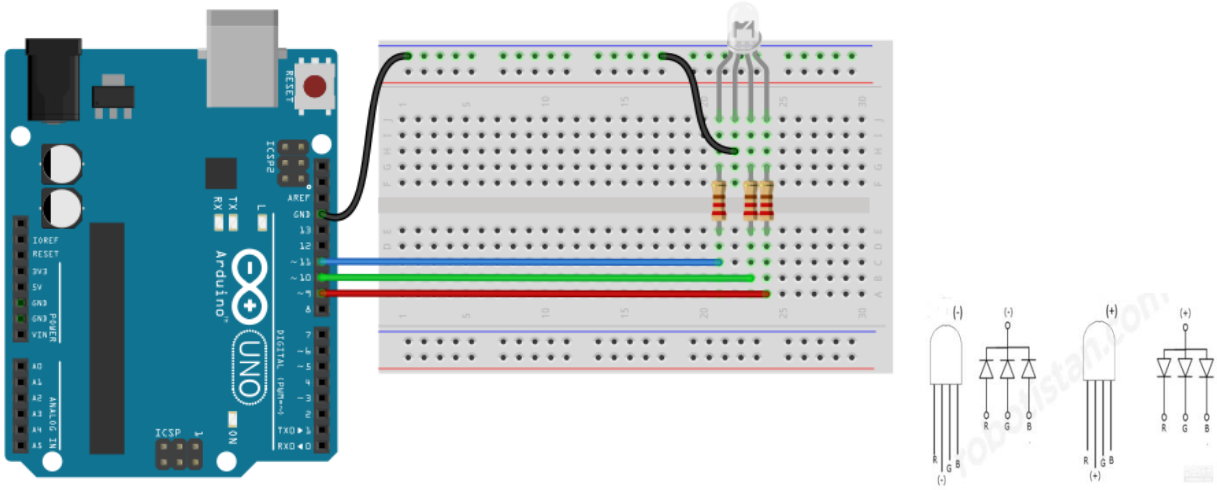
```
int redLED = 7;
int yellowLED =6;
int greenLED = 5;

void setup() {                    // burada pinlerimizi çıkış olarak ayarlıyoruz
```

```
pinMode(redLED, OUTPUT);
pinMode(yellowLED, OUTPUT);
pinMode(greenLED, OUTPUT);
}

void loop() {

digitalWrite(redLED, HIGH);          // redLED'i 9 saniye yak
digitalWrite(yellowLED, LOW);
digitalWrite(greenLED, LOW);
delay(9000);

digitalWrite(redLED, HIGH);          // redLED ve yellowLED'i 2 saniye yak
digitalWrite(yellowLED, HIGH);
digitalWrite(greenLED, LOW);
delay(2000);

digitalWrite(redLED, LOW);           // greenLED'i 9 saniye yak
digitalWrite(yellowLED, LOW);
digitalWrite(greenLED, HIGH);
delay(9000);

digitalWrite(redLED, LOW);           // tekrar, yellowLED'i 2 saniye yak
digitalWrite(yellowLED, HIGH);
digitalWrite(greenLED, LOW);
delay(2000);


/* Döngü yeniden başlıyor */

 }
```

Devre 4:

Devre Adı: RGB LED Uygulaması.

Devre Açıklaması: RGB LED, tek bir kasaya yerleştirilmiş ortak olarak bağlı 3 LED'dir. Üç rengin ışık yoğunluğunu dijital olarak kontrol etmek mümkündür. Ayrıca PWM tekniği kullanılarak istenilen renkler elde edilebilir.



/* RGB LED'in her rengini 1 saniye aralıklarla yakıp söndüreceğiz. Beyaz ışık göstermek istiyorsak tüm ledleri açmamız gerekiyor. */

```
const int BlueLed=11;          // BlueLed'e 11 değerini sabit olarak ata

 const int GreenLed=10;        // GreenLed'e 10 değerini sabit olarak ata

const int RedLed=9;            // RedLed'e 9 değerini sabit olarak ata

        // Ledlerin bağlı olduğu pinleri çıkış olarak atadık.

void setup()  {
pinMode(BlueLed,OUTPUT);
pinMode(GreenLed,OUTPUT);
pinMode(RedLed,OUTPUT);      }

        //  Döngü burada başlıyor

void loop() {
digitalWrite(BlueLed, LOW);         // RedLed açık.
digitalWrite(GreenLed, LOW);
digitalWrite(RedLed, HIGH);
delay(1000);

 digitalWrite(BlueLed,  LOW );     // GreenLed açık.
digitalWrite(GreenLed, HIGH);
digitalWrite(RedLed, LOW );
delay(1000);

 digitalWrite(BlueLed,  HIGH);          // BlueLed açık.
digitalWrite(GreenLed,  LOW);
digitalWrite(RedLed, LOW);
delay(1000);
```

// Tüm ledleri aktif hale getirerek beyaz rengi gösteriyoruz.

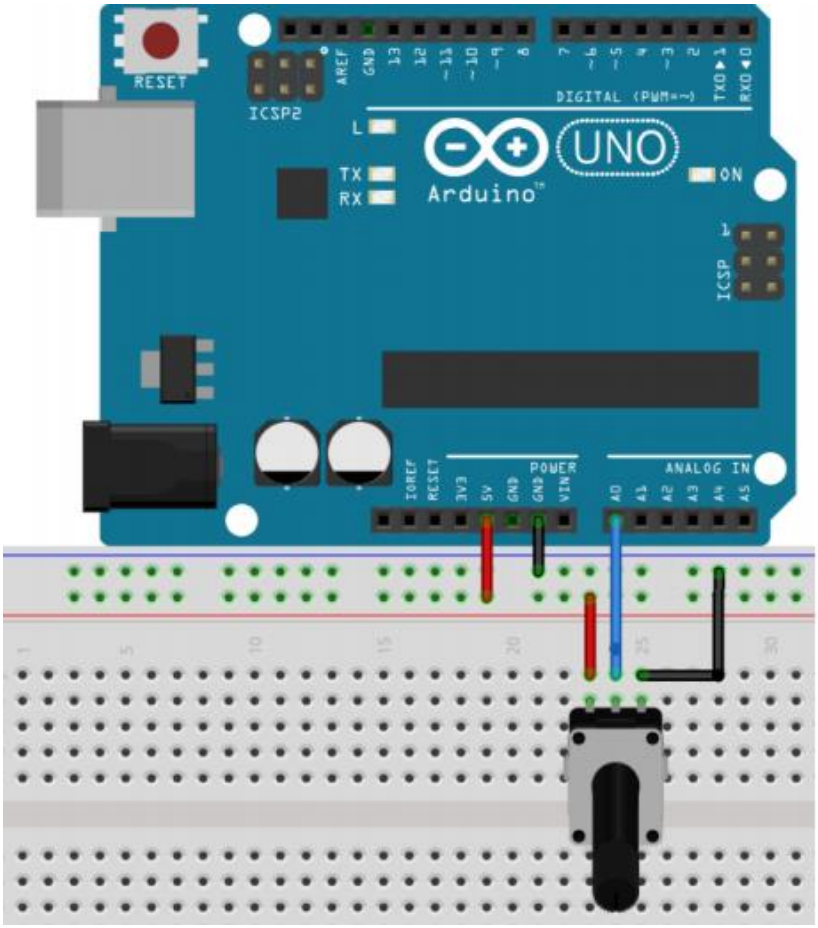
```
digitalWrite(BlueLed,  HIGH);
digitalWrite(GreenLed,  HIGH);
digitalWrite(RedLed,  HIGH);
delay(1000);
}
```

Devre 5:

Devre Adı: Analog Voltaj Okuma, Potansiyometreden Değer Okuma Uygulaması.

Devre Açıklaması: Burada, analog pin-0'da bir analog girişin nasıl okunacağını öğreniyoruz. Giriş, analogRead()'den voltaja dönüştürülür ve Arduino IDE'nin seri monitörüne yazdırılır.

Potansiyometre: Bir potansiyometre (veya pot), basit bir elektro-mekanik dönüştürücüdür. Giriş operatöründen gelen döner veya doğrusal hareketi bir direnç değişikliğine dönüştürür. Bu değişiklik, bir hi-fi sisteminin hacminden, büyük bir konteyner gemisinin yönüne kadar her şeyi kontrol etmek için kullanılabilir (veya kullanılabilir).	



/* ReadAnalogVoltage: Pin 0'daki bir analog girişi okur, onu voltaja dönüştürür ve sonucu seri monitöre yazdırır. Seri çizici kullanılarak grafik gösterimi mevcuttur (Araçlar→Seri Çizici menüsü). Bir potansiyometrenin merkez pinini A0 pinine ve dış pinleri +5V'a ve toprağa takın. */

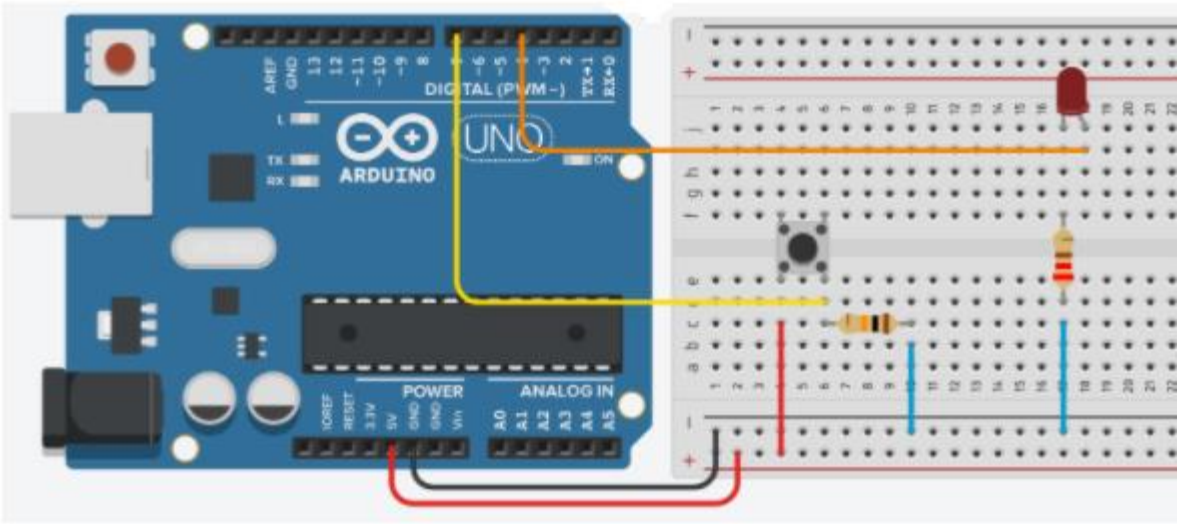
```
        // Reset'e bastığınızda kurulum rutini bir kez çalışır:
void setup()        {
  Serial.begin(9600);          // saniyede 9600 bitte seri iletişimi başlat
}
        void loop() {          // döngü rutini sonsuza kadar tekrar tekrar çalışır:
```

```
    int sensorValue = analogRead(A0);        //analog pin 0'daki girişi okuyun
    Serial.println(sensorValue);        // okuduğunuz değeri Seri monitöre yazdırın
    delay(1000);        // stabilite için okumalar arasındaki gecikme
}
```

Devre 6:

Devre Adı: Ardunio ile buton kullanımı.

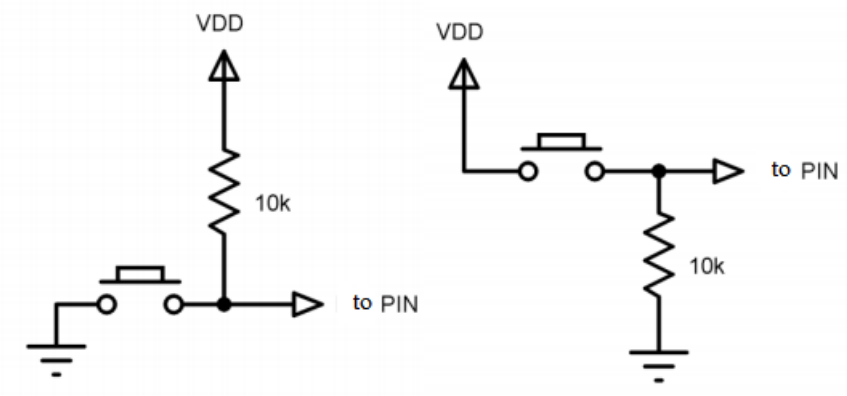
Devre Açıklaması: Pin-7'ye bağlı bir butona basıldığında, dijital pin-4'e bağlı bir (LED) yanar ve söner.




```
/*        Pin 7'ye bağlı bir butona basıldığında, dijital pin 4'e bağlı bir ışık yayan diyotu (LED) açar ve kapatır.        */

void setup()
{
  pinMode(4, OUTPUT);  // pin-4 çıkış yap
  pinMode(7, INPUT);     //  pin-7 giriş yap
}
void loop()
{
   if (digitalRead(7) == HIGH)                    //  Eğer pin-7 açık ise,
```

```
  digitalWrite(4, HIGH);          // Led'i yak,
  if (digitalRead(7) == LOW)      // Eğer pin-7 kapalı ise
   digitalWrite(4, LOW);          // Led'i söndür
}
```

NOT: IF-THEN komutu, Arduino'un giriş pinlerine butonları bağlamak için de kullanılabilir. Bu bağlantı iki farklı şekilde yapılabilir. Pull_Up bağlantısı ve Pull-Down bağlantısı.

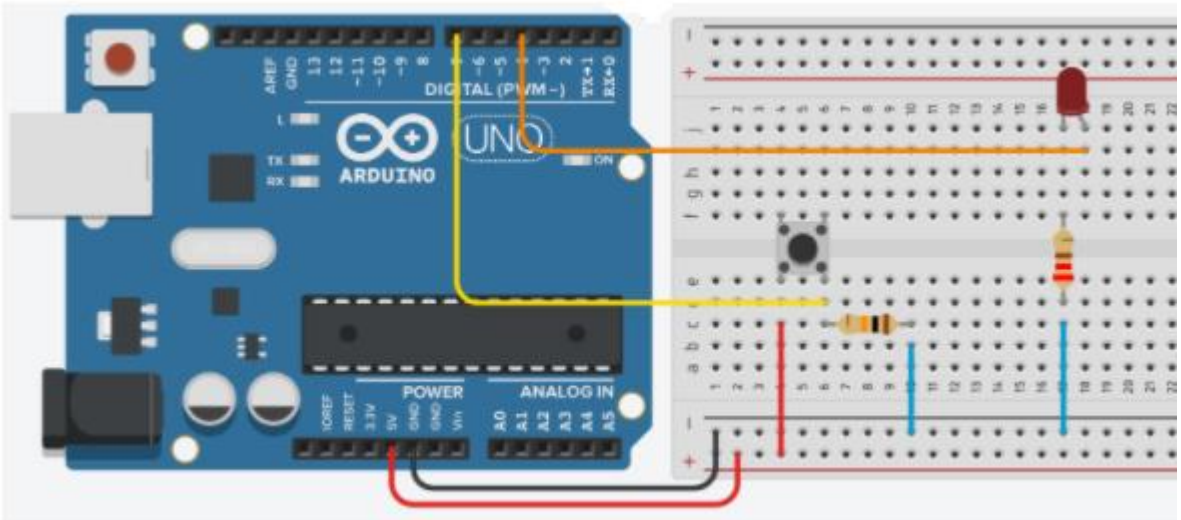


Şekil: IF…THEN komutu için kullanılabilen Anahtarlar veya Düğmeler

Devre 7:

Devre Adı: Arduino'da ELSE komutunu kullanma.

Devre Açıklaması: Pin-7'ye bağlı bir butona basıldığında dijital pin-4'e bağlı bir (LED) buton yanar ve söner. Ayrıca pinleri "int" değişkeni ile belirlemeyi öğreneceğiz.

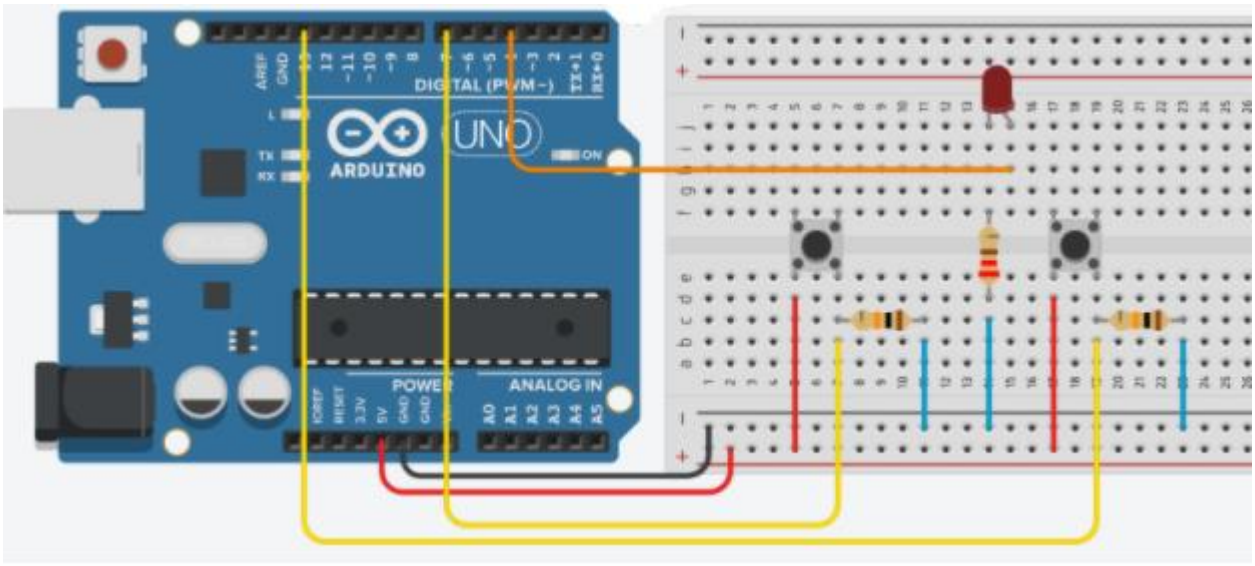


```
 int led = 4;
int buton =7;
void setup()
{
 pinMode(led, OUTPUT);
 pinMode(buton, INPUT);
}
void loop()
{
    if (digitalRead(buton) == HIGH)          // Buton açıksa
       digitalWrite(led, HIGH);              // Ledi yak
     else                                    // Değilse
       digitalWrite(led, LOW);               // Ledi söndür
}
```

Devre 8:

Devre Adı: LED'i iki butonla kontrol etme.

Devre Açıklaması: Bir buton Led'i açar, diğer buton Led'i kapatır.




```
/*      LED'i iki butonla kontrol etme

Düğmeler seri olarak 10K dirençlerle bağlanır. */

int led = 4;                    //  Pin-4 "led" olarak tanımlandı
int button1 = 7;                // Pin-7 "button1"  olarak tanımlandı
int button2 = 13;               // Pin-13 "button2"  olarak tanımlandı
void setup()
{
  pinMode(led, OUTPUT);                 //  led=Çıkış
  pinMode(button1, INPUT);              //  button1=Giriş
  pinMode(button2, INPUT);              // button2=Giriş
}
void loop()
{
  if (digitalRead(button1) == HIGH)     // Eğer button1 açıksa
     digitalWrite(led, HIGH);           // Led 'i yak
    if (digitalRead(button2) == HIGH)   // Eğer button2 açıksa
     digitalWrite(led, LOW);            // Led' söndür
}
```

ARDUINO'DA SERİ MONİTÖR NASIL KULLANILIR

Arduino IDE, çizimlerde hata ayıklamada veya Arduino'yu bilgisayarınızın klavyesinden kontrol etmede çok yardımcı olabilecek bir özelliğe sahiptir.

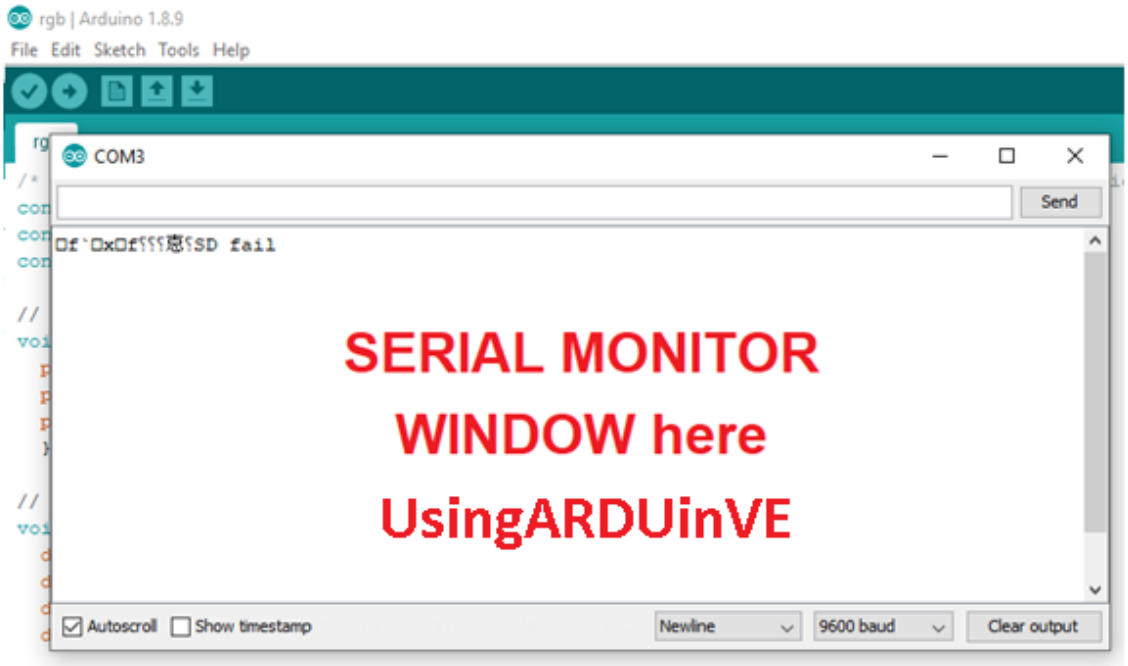
Seri Monitör, Seri verileri alıp göndererek iletişim kuran ayrı bir terminal görevi gören ayrı bir açılır penceredir. 

simgesi ile Seri Monitör gösterilir.

Seri veriler tek bir kablo üzerinden gönderilir (ancak bizim durumumuzda genellikle USB üzerinden geçer) ve kablo üzerinden gönderilen bir dizi 1 ve 0'dan oluşur. Veriler her iki yönde de gönderilebilir (Bizim durumumuzda iki kablo üzerinde).

Arduino uygulamalarında hata ayıklamak veya çalışan bir uygulama tarafından gönderilen verileri görüntülemek için Seri Monitörü kullanacaksınız. Seri Monitörü aktif hale getirebilmek için bilgisayarınıza USB ile bağlı bir Arduino'nuz olmalıdır.

IDE'deki Seri Monitör kutusuna tıklayarak Seri Monitörü açın. Aşağıdaki ekran görüntüsü gibi görünmelidir. Baud'un (hız) 9600'e ayarlandığından EMİN OLUN(sağ alt köşede bulunur). Önemli olan bizim programımızda ve burada aynı ayarlanmış olması. Buradaki default 9600 olduğu için biz kendi ayarlarımızı yapıyoruz.



Seri Monitörü Kullanmak için Ana Komutlar

Serial.begin(): Seri veri iletimi için veri hızını saniyede bit (baud) olarak ayarlar. Seri Monitör ile iletişim kurmak için ekranın sağ alt köşesindeki menüde listelenen baud hızlarından birini kullandığınızdan emin olun. Bununla birlikte, örneğin baud hızı gerektiren bir bileşenle 0 ve 1 pinleri üzerinden iletişim kurmak için başka hızlar belirleyebilirsiniz.

Kullanımı:

Serial.begin(speed);

Örnek: Serial.begin(9600);

Serial.print(): Verileri seri bağlantı noktasına insan tarafından okunabilen ASCII metni olarak yazdırır. Bu komut birçok şekilde olabilir. Sayılar, her basamak için bir ASCII karakteri kullanılarak yazdırılır. Kayan noktalı sayılar benzer şekilde ASCII basamakları olarak yazdırılır ve varsayılan olarak iki ondalık basamaktır. Baytlar tek bir karakter olarak gönderilir. Karakterler ve dizeler olduğu gibi gönderilir. Örneğin:

Serial.print(78); "78" gösterir

Serial.print('N'); "N" gösterir

Serial.print("Hello UsingARDinVET"); "Hello UsingARDinVET " gösterir

Serial.println(): Verileri seri bağlantı noktasına insan tarafından okunabilir ASCII metni ve ardından bir satır başı karakteri (ASCII 13 veya '\r') ve yeni satır karakteri (ASCII 10 veya '\n') olarak yazdırır. Bu komut Serial.print() ile aynı formları alır.

Serial.println(val);

Serial.println(val, format);

Serial.println(13, BIN); Seri Monitörde "1101" ikili değeri 15'i verir.

NOT: Serial.print ve Serial.println arasındaki tek fark, Serial.println'nin, bundan sonra seri bağlantı noktasından gönderilen şeyin bir sonraki satırda başlayacağı anlamına gelmesidir. Fark etmiş olabileceğiniz üçüncü bir yeni şey daha var. Tırnak içinde bir şey var (“). Buna dize denir.

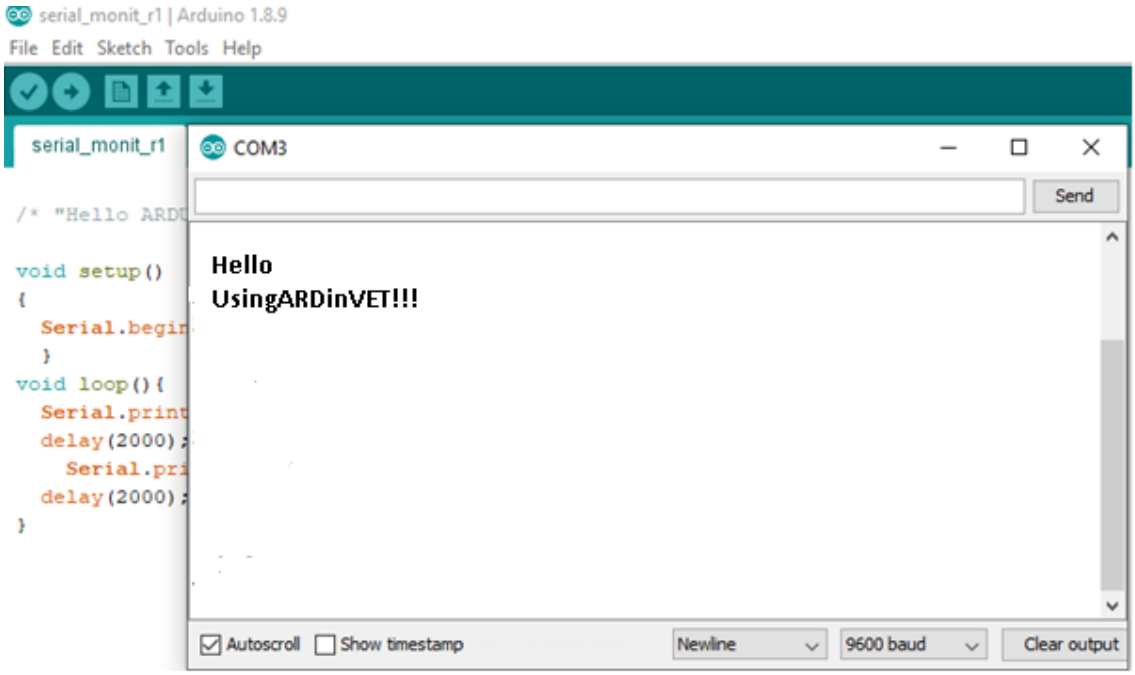
Devre 9:

Devre Adı: Seri Monitörde “Merhaba UsingARDinVET” Yazdırma Uygulaması.

Devre Açıklaması: Merhaba UsingARDinVET seri Monitörde görülecektir.

/* " Seri Monitörde Hello UsingARDinVET" Uygulaması */

```
void setup()
{
  Serial.begin(9600);     // Seri bağlantı hızı
}
void loop()
{
  Serial.println("  HELLO ");
  delay(2000);
  Serial.println("UsingARDinVET!!!");
  delay(2000);
}
```

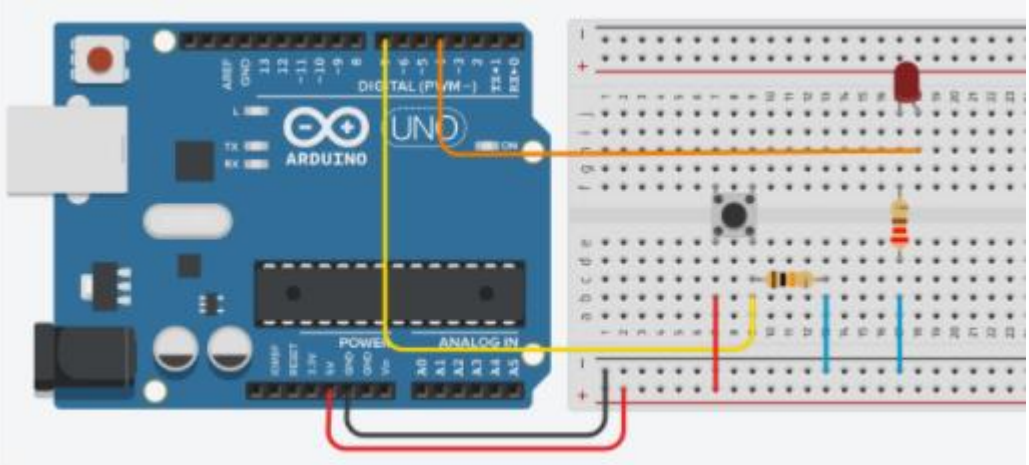


Devre 10:

Devre Adı: Seri Monitörde Düğme Durum Analizi

Devre Açıklaması: Butona basıldığında seri monitörde "Led yanıyor, LED=AÇIK" mesajı çıkıyor ve Led yanıyor.

Butona basılmazsa, seri monitörde ("Buton basılı değil") mesajı görülür ve Led yanar.




```
/*      Seri Monitörde Buton Durum Analizi      */

void setup()                    {
 pinMode(4, OUTPUT);
 pinMode(7, INPUT);
 Serial.begin(9600);          }
void loop()                       {
  if (digitalRead(7) == HIGH)        // 7 nolu pin'de sinyal varsa
   {
   digitalWrite(4, HIGH);
   Serial.println("Led is ligthing, LED=ON");
   delay(250);
   }
  else                          // Önceki şart doğru değilse
   {
   digitalWrite(4, LOW);
```

```
  Serial.println("Button is not pressed");
  delay(1000);
  }
}                                   // Seri monitörün görünümü aşağıda görülmektedir.
```

```
Button is not pressed
Button is not pressed
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Button is not pressed
```

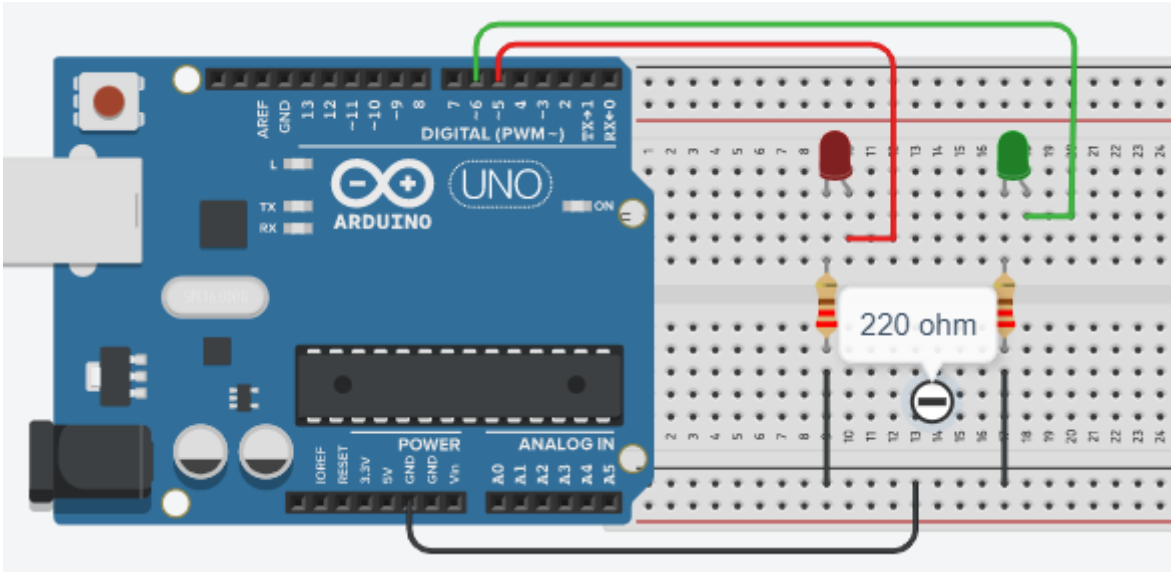
Devre 11:

Devre Adı: Seri ekrandan veri gönderme.

Devre Açıklaması: Klavyede “A” karakterine basılırsa led1 yanar.

Klavyede “3” karakterine basılırsa led2 yanar.

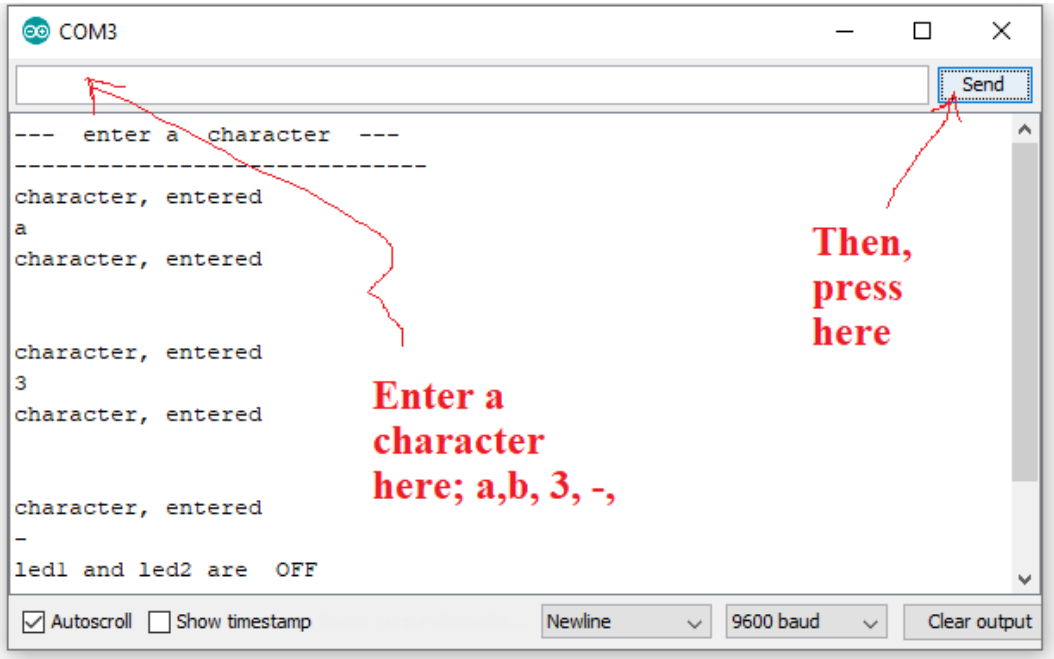
Klavyede “-” karakterine basılırsa led1 ve led2 OFF olur.




```
/*     Seri monitörden veri gönderme   */

char character;
#define  led1 5
#define  led2 6

 void setup()   {
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
Serial.begin(9600);
Serial.println ("---  enter a  character  --- ");
Serial.println ("-----------------------------");
}

void loop()
{
if (Serial.available()>0)
{
character=Serial.read();
Serial.println ("character, entered ");
Serial.println (character);

 if (character=='A') digitalWrite (led1, 1);

if (character=='a') digitalWrite (led1, 1);

if (character=='3') digitalWrite (led2, 1);

if (character=='-')
{
digitalWrite(led1,0);
```

“

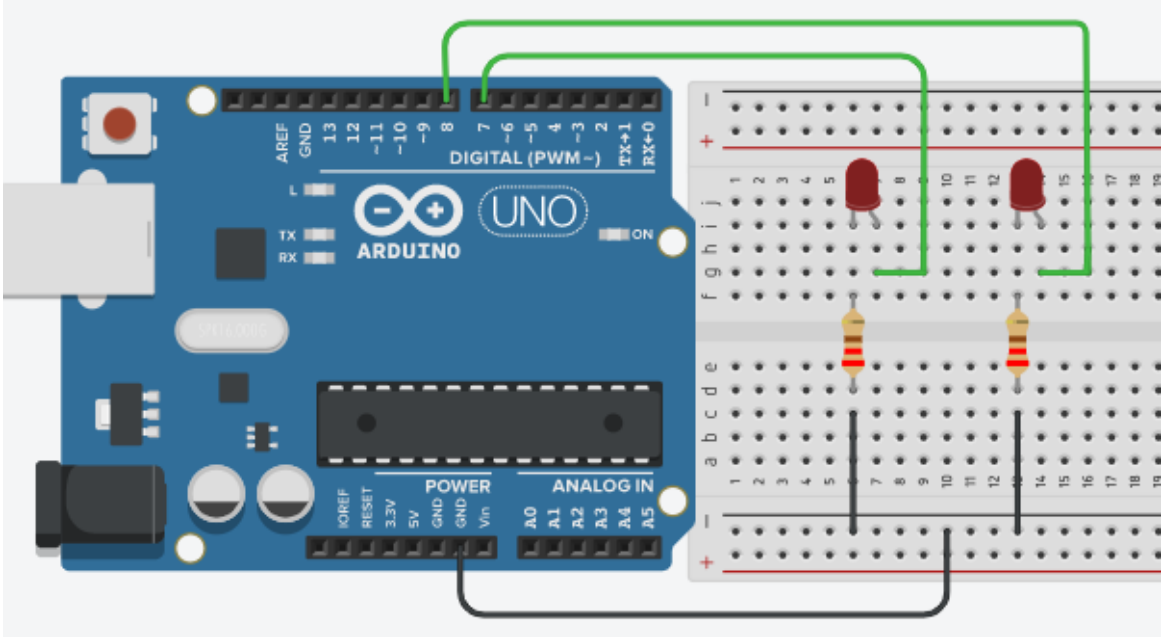
```
digitalWrite(led2,0);
Serial.println ("led1 and led2 are  OFF ");
}
}
}
```



Devre 12:

Devre Adı: FOR komutunu öğrenme uygulaması.

Devre Açıklaması: Bu uygulamada LED'ler 6 kez yanıp sönecektir.



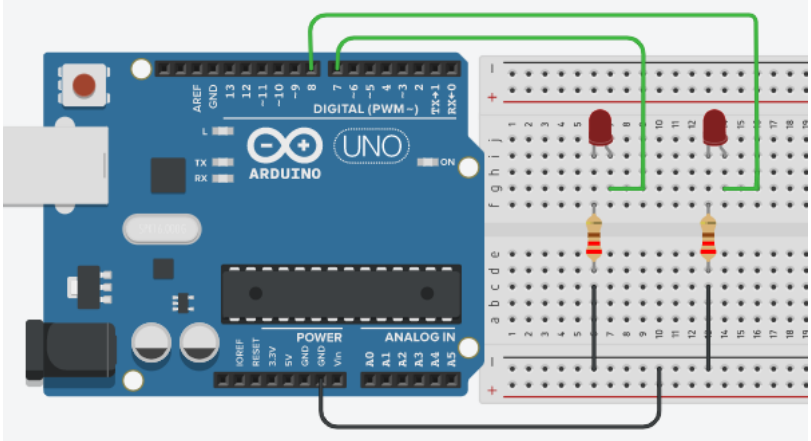

```
/*     FOR komutunu öğrenme   */

int   led1 = 7;
int   led2 = 8;

void setup() {
Serial.begin(9600);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);
}

void loop()
{
for(int i=1; i<6; i++)
{
Serial.println(i);
digitalWrite(led1, 1);
digitalWrite(led2, 1);
delay(1000);

digitalWrite(led1, 0);
digitalWrite(led2, 0);
delay(1000);
}
}
```

Devre 13:

Devre Adı: FOR komutu ve seri monitör uygulaması.

Devre Açıklaması: Bu uygulamada LED'ler 6 kez yanıp sönecektir. Seri monitörde 1, 2, 3, 4, 5,6 sayıları görünecektir. Daha sonra for döngüsünden çıkılarak while döngüsüne girilecek ve program duracaktır. Yeniden başlatmak için Reset düğmesine basılmalıdır.

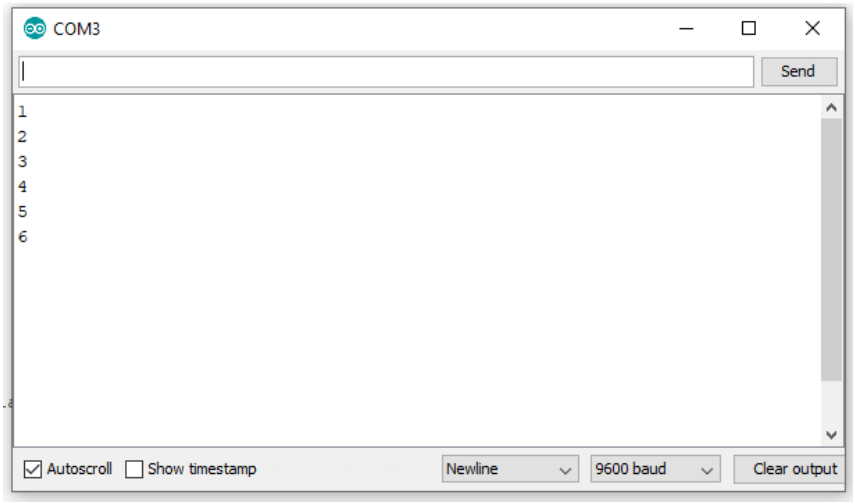
Burada önceki devre ile aynı devreyi kullanıyoruz.




```
/*     FOR komutu ve seri monitor uygulaması   */

int   led1 = 7;
int   led2 = 8;

void setup()                    {
Serial.begin(9600);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);       }

void loop() {

  for(int i=1; i<=6; i++)                 // i = 1, 2,3, 4,5, 6

  {

 Serial.println(i);                       // Seri monitöre i'nin değerlerini yaz

digitalWrite(led1, 1);
digitalWrite(led2, 1);
delay(1000);
digitalWrite(led1, 0);
digitalWrite(led2, 0);
delay(1000);
}
while(1);                    //  for döngüsü sonsuz bir döngüye girmez.
```

}

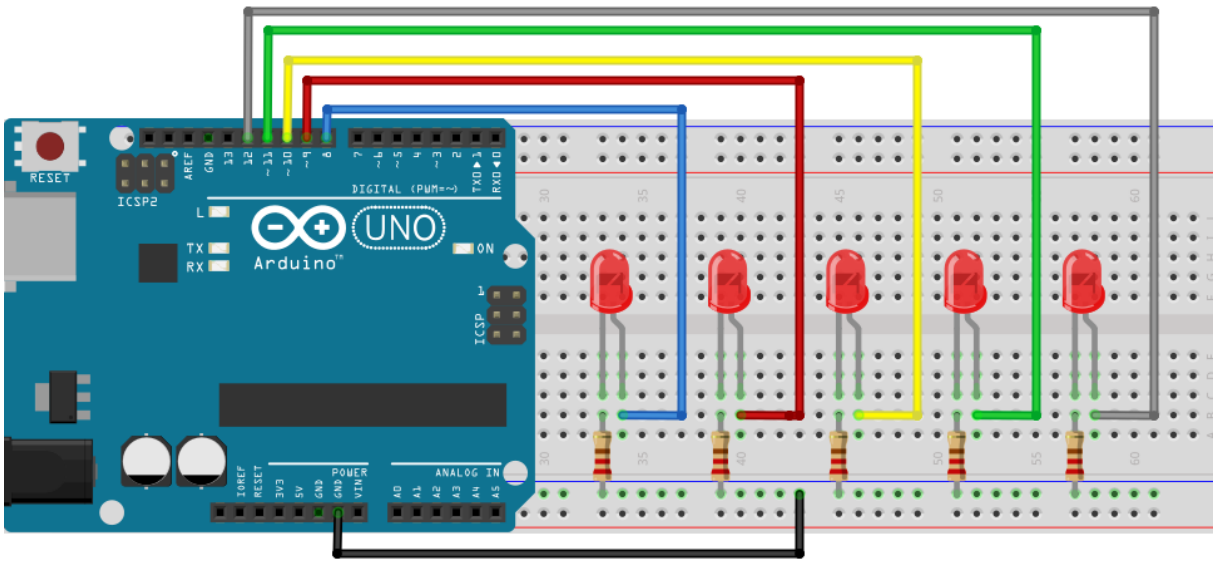
NOT: Programı yeniden başlatmak içinReset düğmesine basılmalıdır.



Devre 14:

Devre Adı: FOR komutu ile karaşimşek uygulaması.

Devre Açıklaması: Devreye bağlı 5 LED sırayla bir yöne doğru yanma/sönme işlemi yapacak, son LED'e geldiğinde ise bu işlemi ters yönde gerçekleştirecektir.



/* FOR komutu kullanarak karaşimşek uygulaması yapmak */

```
void setup() {
pinMode(8, OUTPUT);
pinMode(9, OUTPUT);
pinMode(10, OUTPUT);
pinMode(11, OUTPUT);
pinMode(12, OUTPUT);     }

void loop() {

 for (int b=8; b<=12; b++)                    // Döngü başlangıcı

  {

 digitalWrite(b, HIGH);
delay(150);
digitalWrite (b, LOW);
delay(150);

  }

  for (int b=12; b>=8; b--)                    // Döngü başlangıcı

  {

 digitalWrite (b, HIGH);
delay(150);
digitalWrite (b, LOW);
delay(150);

  }
}
```

Devre 15:

Devre Adı: Switch/case komutunu kullanarak RGB led ile rastgele renk üretimi

Devre Açıklaması: Bu uygulamada Random komutu ve Switch/Case komutu kullanılmaktadır. Bu devrede kırmızı, yeşil, mavi renkler rastgele elde edilecektir.

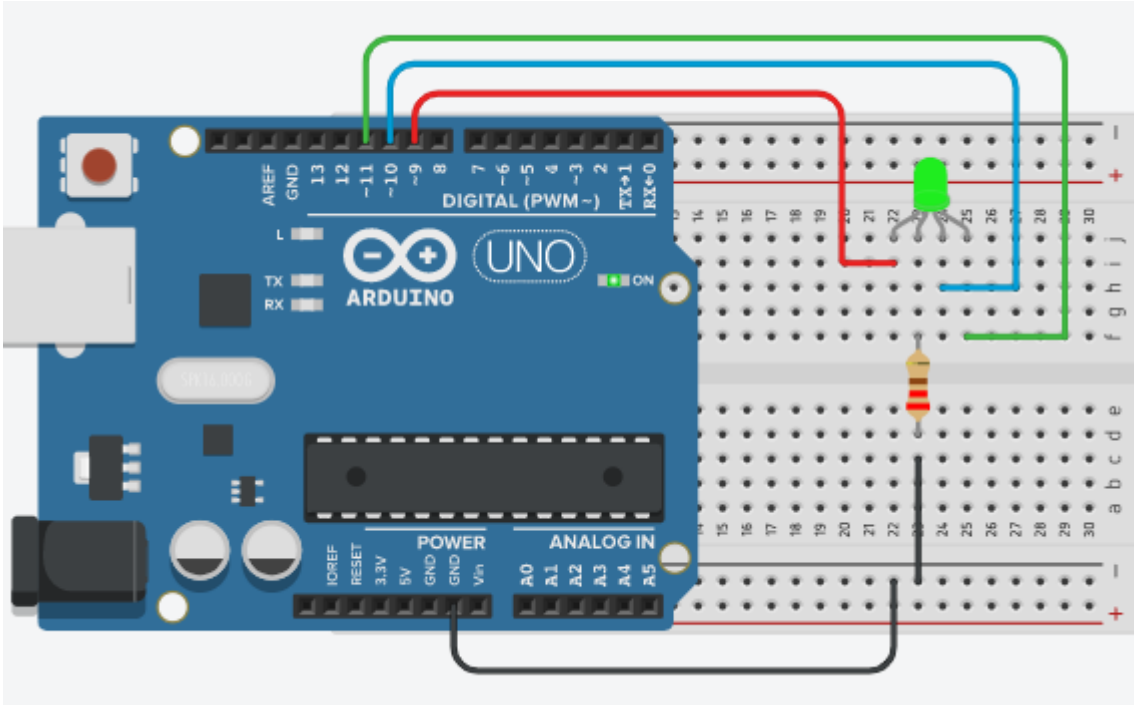
NOT:

random() : Random işlevi rasgele sayılar üretir.

Kullanımı: random(max), random(min, max)

min: rastgele değerin alt sınırı, (isteğe bağlı).

max: rastgele değerin üst sınırı.




```
/*     Switch/case komutunu kullanarak RGB led ile rastgele renk üretimi   */

#define R  9
#define G 10
#define B 11
int colour;
int dly=3000;

void setup() {
pinMode(R, OUTPUT);
pinMode(G, OUTPUT);
pinMode(B, OUTPUT);
Serial.begin(9600);
}

void loop()
{
colour=random(4);
Serial.println(colour);

switch(colour)  {
```

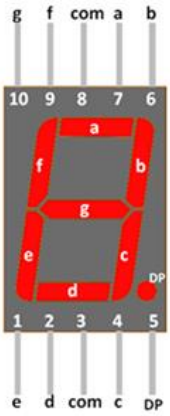
```
 case 0:
digitalWrite (R, 1);              //RedLED=ON
digitalWrite (G, 0);
digitalWrite (B, 0);
delay(dly);
break;

 case 1:
digitalWrite (R, 0);             //GreenLED=ON
digitalWrite (G, 1);
digitalWrite (B, 0);
delay(dly);
break;

 case 2:                          //BlueLED=ON
digitalWrite (R, 0);
digitalWrite (G, 0);
digitalWrite (B, 1);
delay(dly);
break;

 case 3:                          //Renk yok
digitalWrite (R, 0);
digitalWrite (G, 0);
digitalWrite (B, 0);
delay(dly);
break;
}
}
```

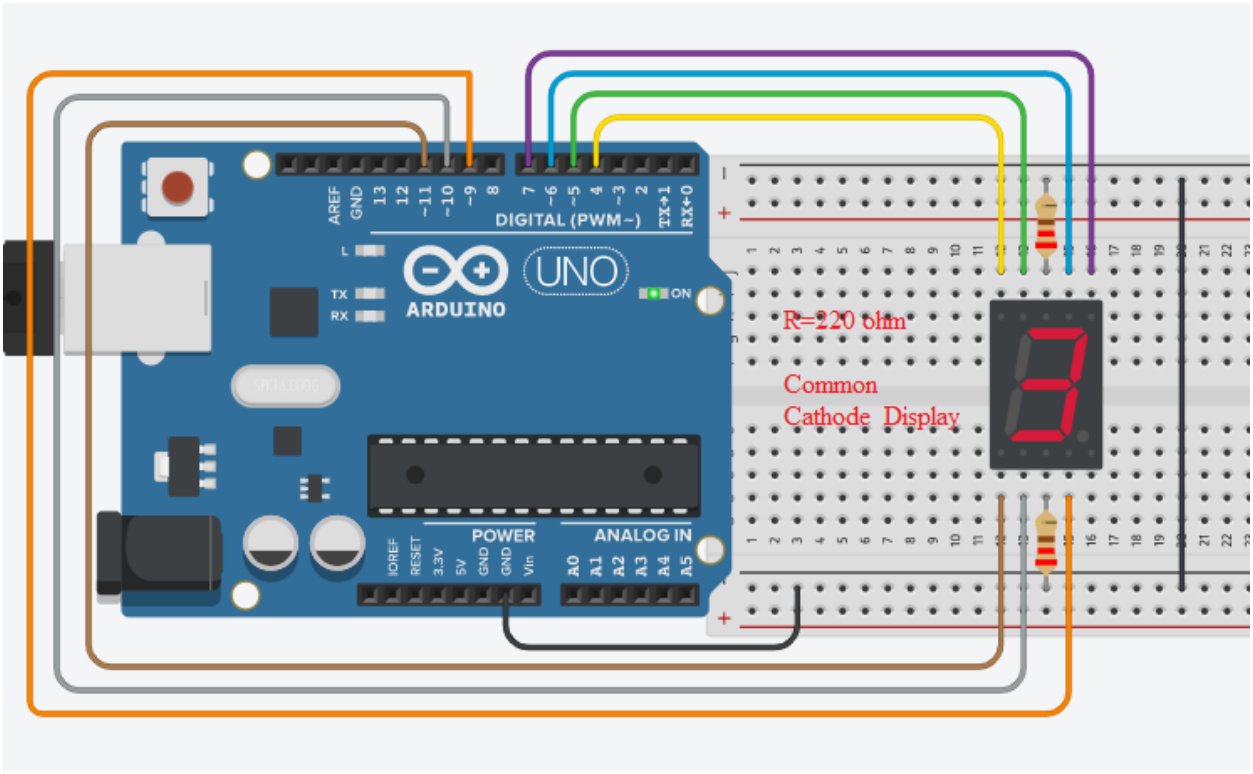
Devre 16:

Devre Adı: Segment Display 0-9 Sayıcı Devresi (Counter)

Devre Açıklaması: Ekranda bir sayı görmek için a, b, c, d, e, f, g harfleriyle temsil edilen 7 LED'den ilgili LED yanar.

NOT: Segment display(Yedi bölmeli ekran), ondalık sayıları görüntülemek için bir elektronik görüntüleme cihazıdır.






```
/*     Segment Display 0-9 Sayıcı Devresi (Counter)     */

int       a=6,   b=7,   c=9,    d=10,     e=11,   f=5,    g=4;
int number;

void setup() {
pinMode(a,  OUTPUT);
pinMode(b,  OUTPUT);
pinMode(c,  OUTPUT);
pinMode(d,  OUTPUT);
pinMode(e,  OUTPUT);
pinMode(f,  OUTPUT);
```

```
pinMode(g,  OUTPUT);
}

void loop()  {
for(number=0;  number< =9;  number++)    {
delay(1000);
switch(number)  {

 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;
}
}
}
```

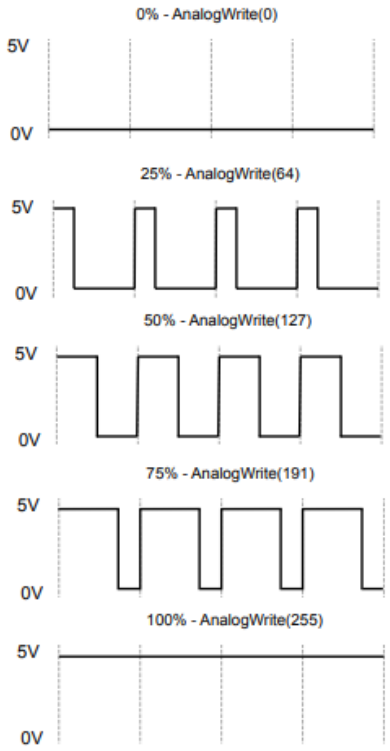
Devre 17:

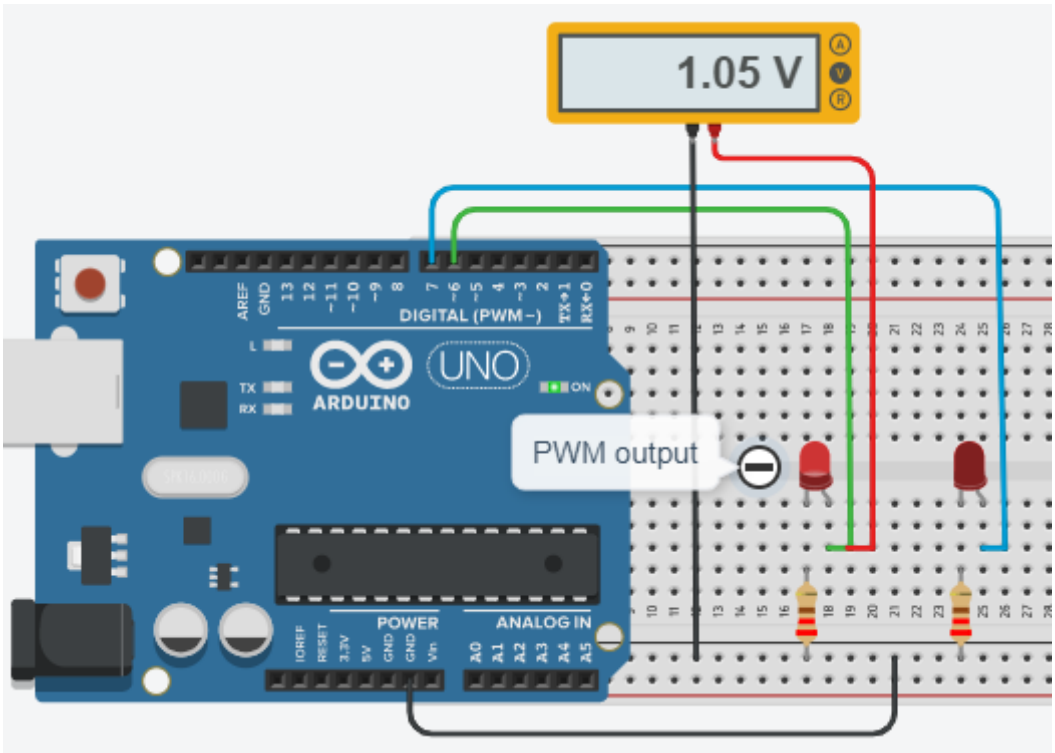
Devre Adı: PWM Tekniğinin Kullanılması

Devre Açıklaması: Pratikte PWM çıkışı olarak pin-6 ve dijital çıkış olarak pin-7 kullanılır. Böylece aralarındaki farkın gözlemlenmesi amaçlanmaktadır. PWM yöntemi ile led parlaklık ayarı, motor hız kontrolü gibi uygulamalar yapılabilmektedir.

NOT: Arduino, 500Hz'de PWM'yi (Arduino kartınızda tilde (~) ile işaretlenmiş belirli pinlerde - pin 3, 4,5,9,10 ve 11) destekler. (Saniyede 500 kez.) 0 ile 255 arasında bir değer verebilirsiniz. 0, asla 5V olmadığı anlamına gelir. 255 her zaman 5V olduğu anlamına gelir. Bunu yapmak için değerle analogWrite()'a bir çağrı yaparsınız. “AÇIK” zamanın toplam süreye oranına “görev döngüsü” denir. Zamanın yarısında AÇIK olan bir PWM çıkışının %50'lik bir görev döngüsüne sahip olduğu söylenir.

PWM'yi x/255 için açık olarak düşünebilirsiniz; burada x, analogWrite() ile gönderdiğiniz değerdir. Aşağıda, darbelerin nasıl göründüğünü gösteren bir örnek verilmiştir:





/* PWM Tekniğini analogWrite() kullanarak öğrenme */

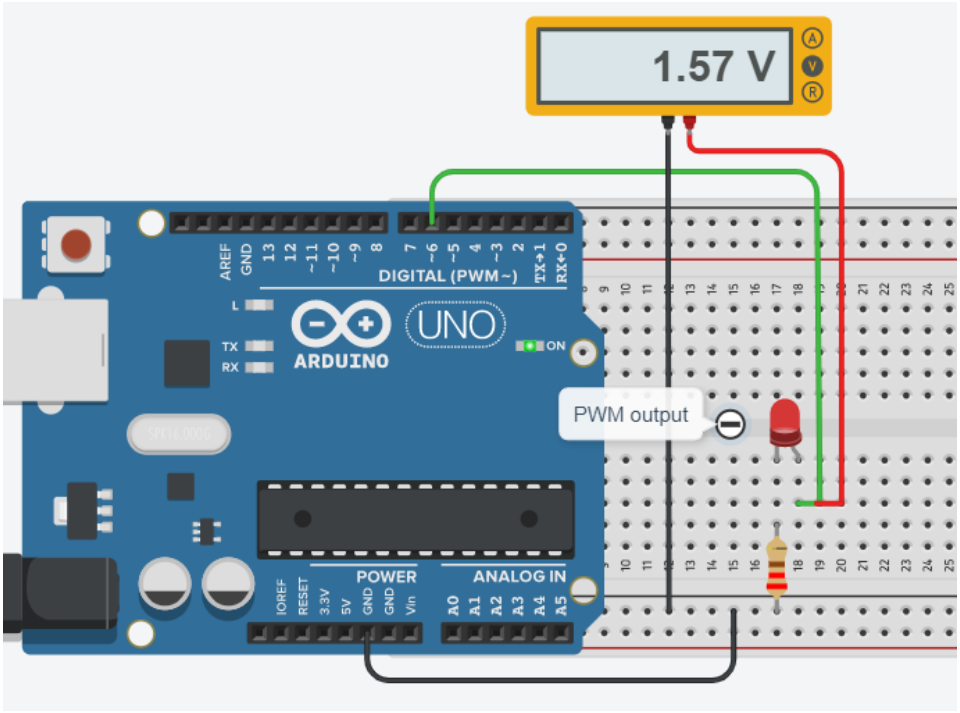
```
#define   led1  6
#define   led2  7

void setup()  {
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);      }

void loop()               {

 analogWrite(led1, 60);       //  60 değeri Led1 pinine gönderilir (1.05 V)


 analogWrite(led2,60);        //  60 değeri Led2 pinine gönderilir (1.05 V)


delay (100);                  // sadece Led1 yanar

}
```

Devre 18:

Devre Adı: Bir LED'in parlaklığını minimumdan maksimuma değiştirmek.

Devre Açıklaması: PWM tekniği kullanılarak bir LED'in parlaklığı minimumdan maksimuma değiştirilir. Burada analogWrite() ve for komutları birlikte kullanılacaktır.

.




```
/*        Bir LED'in parlaklığını minimumdan maksimuma değiştirmek          */

#define led1    6
int a;

void setup()          {
Serial.begin(9600);
pinMode(led1, OUTPUT);    }

void loop() {

 for (a=0; a<=255; a++) {

 Serial.println(a);

analogWrite(led1, a);

 delay (100);
}                }
```

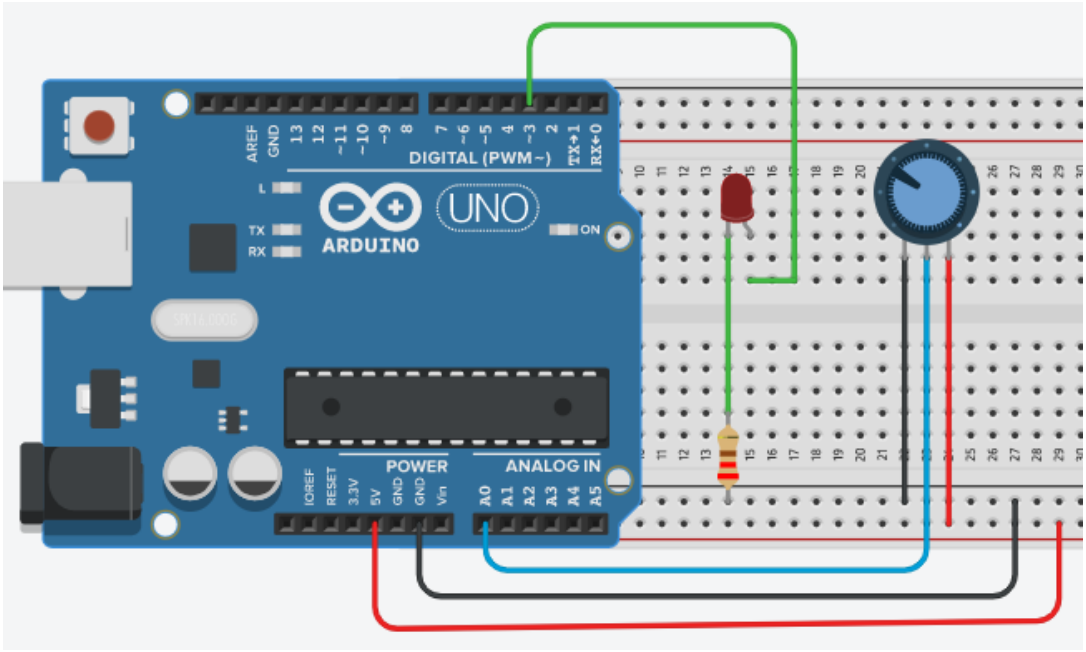
NOT: Voltmetrede 0 ile 5 Volt arasında değişen değerler görüntülenir. 0'dan 255'e kadar sayma sayıları seri monitörde görüntülenir.

.

Devre 19:

Devre Adı: Potansiyometre ile LED yakma.

Devre Açıklaması: Potansiyometre (10K) kullanılarak bir LED'in parlaklığı minimumdan maksimuma değiştirilir. Burada map() fonksiyonu kullanılır.



map() Fonksiyonu:

Bir sayıyı bir aralıktan diğerine yeniden eşler. Yani, fromLow değeri toLow ile, fromHigh değeri toHigh ile, aradaki değerler ile aradaki değerler ile eşlenir.

Aralık dışı değerler bazen amaçlandığından ve kullanışlı olduğundan, değerleri aralık dahilinde sınırlamaz. Aralıklarda sınırlamalar isteniyorsa, bu işlevden önce veya sonra Constrain() işlevi kullanılabilir.

map() işlevi tamsayı matematiği kullanır, bu nedenle matematiğin yapması gerektiğini belirttiğinde kesirler oluşturmaz. Kesirli kalanlar kesilir ve yuvarlanmaz veya ortalaması alınmaz.

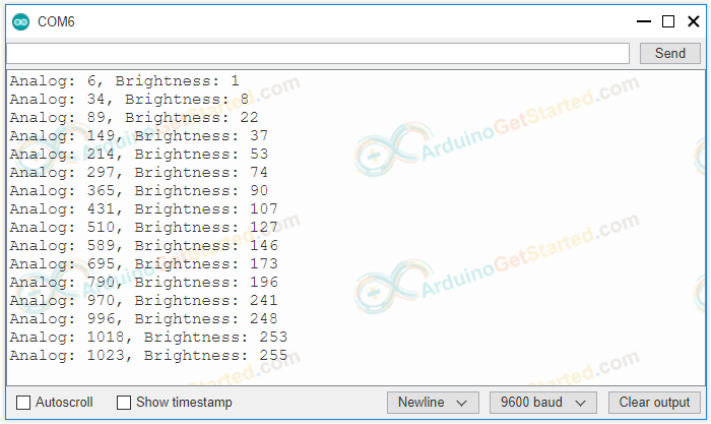
Kullanımı: map(value, fromLow, fromHigh, toLow, toHigh);

Örnek: val = map(val, 0, 1023, 0, 255);

/* Potansiyometre ile LED yakma */

```
int LED_PIN = 3;

void setup() {
Serial.begin(9600);
pinMode(LED_PIN, OUTPUT);      }

void loop() {

        int analogValue = analogRead(A0); // A0 pini üzerindeki girişi okur (0 ile 1023
                                                arasındaki değer)

        int brightness = map(analogValue, 0, 1023, 0, 255);       // parlaklığı ölçeklendirir (0 ile
        255 arasındaki değer)

        analogWrite(LED_PIN, brightness);  // pin-3'e bağlanan LED'in parlaklığını ayarlar

         Serial.print("Analog: ");              // değeri Seri monitöre yazdır

        Serial.print(analogValue);

         Serial.print(" Brightness: ");

          Serial.println(brightness);

         delay(100);

}
```

Seri monitör ekranı aşağıdadır

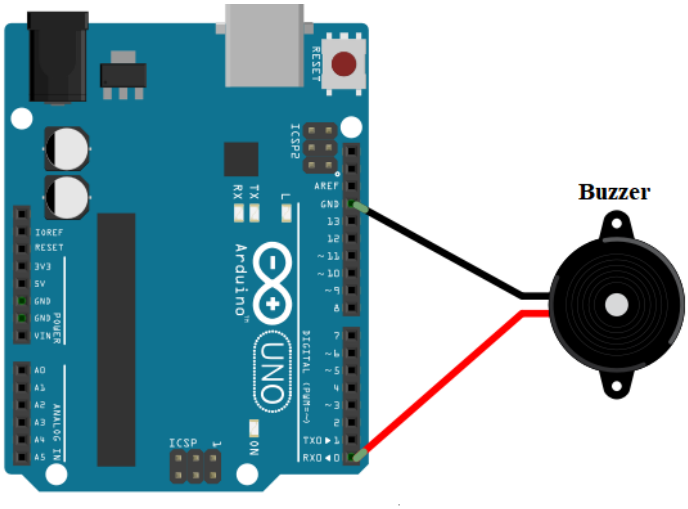


Devre 20:

Devre Adı: Buzzer kullanımı.

Devre Açıklaması: Buzzer, mekanik, elektromekanik veya piezoelektrik (kısaca piezo) olabilen bir ses sinyali cihazıdır. Buzzer'ların endüstrideki tipik kullanımları, uğultu gerektiğinde uğultu veya bip sesi çıkaran bir alarm cihazıdır.

.

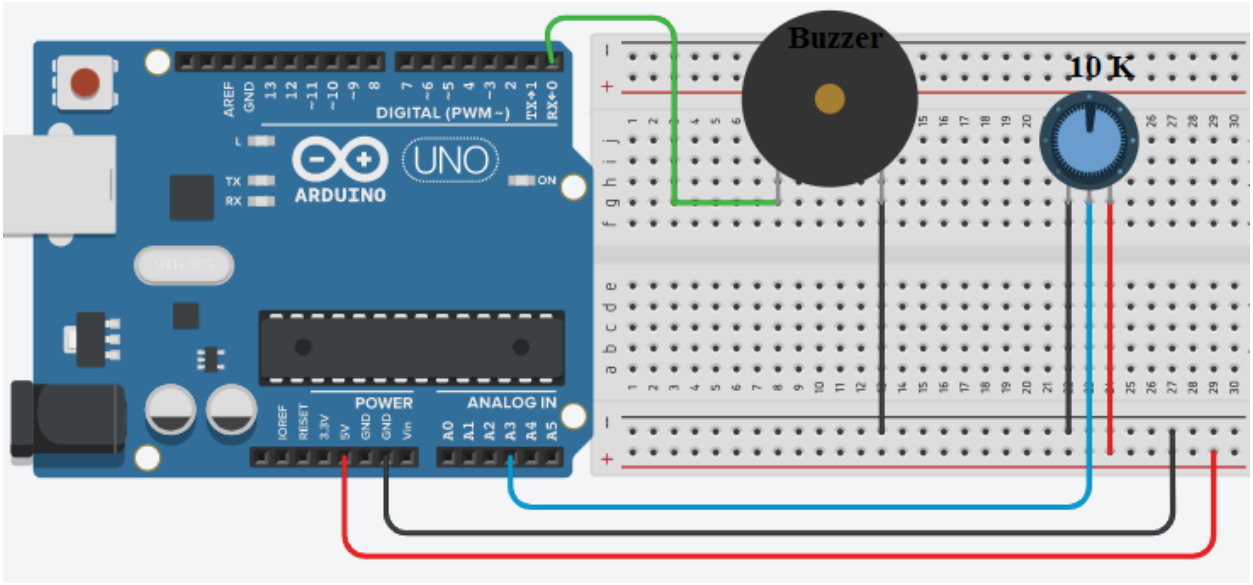



```
/*        Arduino devrelerinde buzzer kullanımı    */

#define buzzer_pin  0

 void setup()   {

 pinMode(buzzer_pin, OUTPUT);          }

void loop()        {

 digitalWrite(buzzer_pin, HIGH);

 delay(500);

 digitalWrite(buzzer_pin, LOW);

 delay(500);      }
```

Devre 21:

Devre Adı: Potansiyometre ile buzzer'ı kontrol etme.

Devre Açıklaması: Potansiyometre değeri 500'den yüksek olduğunda sesli uyarı çalar.




```
/* Potansiyometre ile buzzer'ı kontrol etme */

const int POT_PIN = A3;              // Pot'a bağlı Ardunio pini
const int BUZZER_PIN  = 0;           // Buzzer'a bağlı Ardunio pini

const int ANALOG_THRESHOLD  = 500;
int analogValue;

void setup() {

  pinMode(BUZZER_PIN, OUTPUT);       // Pin çıkış olarak ayarlandı

}

void loop() {

 analogValue = analogRead(POT_PIN);            //  analog pindeki giriş değerini oku

if(analogValue > ANALOG_THRESHOLD)
digitalWrite(BUZZER_PIN, HIGH);                 // Buzzer'ı aç

else
digitalWrite(BUZZER_PIN, LOW);                  // Buzzer'ı kapat

}
```

Co-funded by the
Erasmus+ Programme
of the European Union

Erasmus+ KA210-VET

Küçük Ölçekli Mesleki Eğitim ve Öğretim Ortaklık Projesi

Proje adı: "Mesleki Eğitimde Arduinoları Kullanma"

Proje kısaltması: "UsingARDinVET"

Proje No: "2023-1-RO01-KA210-VET-000156616"

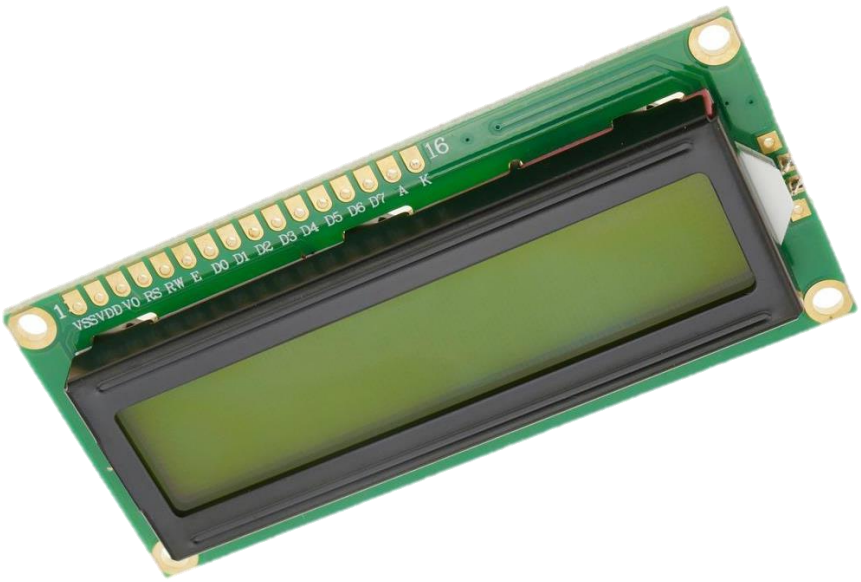
LCD Modül ve Eğitim Kiti

LCD Modül ve Eğitim Kiti

Bu modülde Arduino'nun durum mesajlarının veya sensör okumalarının LCD ekranlarda nasıl görüntüleneceğini açıklamak istiyoruz. Bunlar son derece yaygındır ve projenize okunabilir bir arayüz eklemenin hızlı bir yoludur.
LCD, Liquid Crystal Display'in kısaltmasıdır. Görünür bir görüntü oluşturmak için sıvı kristalleri kullanan bir görüntüleme birimidir. Bu özel tür kristale akım uygulandığında, opaklaşır ve ekranın arkasında kalan arka ışığı engeller. Sonuç olarak, belirli bir alan diğerine kıyasla karanlık hale gelecektir. Ve karakterler ekranda bu şekilde görüntülenir.

Arduino ile 16×2 Karakterli LCD Modülü

Arduino'nuza bağlayabileceğiniz farklı türde LCD ekranlar vardır. En yaygın olanı, Hitachi'nin HD44780 adlı paralel arabirim LCD denetleyici yongasına dayanmaktadır.
Bu LCD'ler yalnızca metin/karakterleri görüntülemek için idealdir, bu nedenle 'Karakter LCD' adı verilir. Ekranın LED arka aydınlatması vardır ve her satırda 16 karakter olmak üzere iki satırda 32 ASCII karakteri görüntüleyebilir.
Ekranda her karakter için küçük bir dikdörtgen vardır, bu dikdörtgenlerin her biri 5×8 piksellik bir ızgaradır.
Yalnızca metin görüntülemelerine rağmen, birçok boyut ve renkte gelirler: örneğin, 16×1, 16×4, 20×4, mavi arka plan üzerinde beyaz metin, yeşil üzerinde siyah metin ve çok daha fazlası.
Arduino topluluğu, HD44780 LCD'leri (LiquidCrystal Kitaplığı) işlemek için zaten bir kitaplık geliştirdi; bu yüzden onları birkaç zaman içinde arayüz haline getireceğiz.



LCD pin çıkışı aşağıdadır:
- VSS: bir topraklama pin'idir ve Arduino'nun toprağına bağlanmalıdır;
- VDD: 5 V'a bağlı;
- VD: kontrast ayarı için (potansiyometrenin merkez pin'ine bağlı);
- RS: gönderilen verilerin hangi lcd bellek bölgesinde saklandığını kontrol eder;
- R/W: okuma/yazma modunu seçmek için pin;
- E: etkinleştirilirse, LCD modülünün özel talimatları gerçekleştirmesini sağlar;
- D0 - D7: veri iletimi;
- A ve K: LCD modülüne arka ışık sağlamak için anot ve katot.

Arduino - LCD Fonksiyonları (Komutlar)

LiquidCrystal lcd() - LiquidCrystal türünde bir değişken oluşturur. Ekran 4 veya 8 veri hattı kullanılarak kontrol edilebilir. İlki ise, d0 ila d3 için pin numaralarını atlayın ve bu satırları bağlantısız bırakın. RW pini Arduino'daki bir pin yerine toprağa bağlanabilir; eğer öyleyse, bu fonksiyonun parametrelerinden çıkarın. Komut Yapısı:

LiquidCrystal(rs, enable, d4, d5, d6, d7)
LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)
LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)
LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)

Parametreler:
rs: LCD üzerindeki RS pinine bağlı olan Arduino pininin numarası
rw: LCD üzerindeki RW pinine bağlı olan Arduino pininin numarası (opsiyonel)
enable: LCD üzerindeki etkinleştirme pinine bağlı olan Arduino pininin numarası
d0, d1, d2, d3, d4, d5, d6, d7: LCD'deki ilgili veri pinlerine bağlı Arduino pinlerinin numaraları. d0, d1, d2 ve d3 isteğe bağlıdır; atlanırsa, LCD sadece dört veri hattı (d4, d5, d6, d7) kullanılarak kontrol edilecektir.
lcd.begin() - LCD ekran arayüzünü başlatır ve ekranın boyutlarını (genişlik ve yükseklik) belirtir. start() diğer LCD kitaplığı komutlarından önce çağrılmalıdır. Sözdizimi:
lcd.begin(cols, rows)

Parametreler:
lcd: LiquidCrystal türünde bir değişken

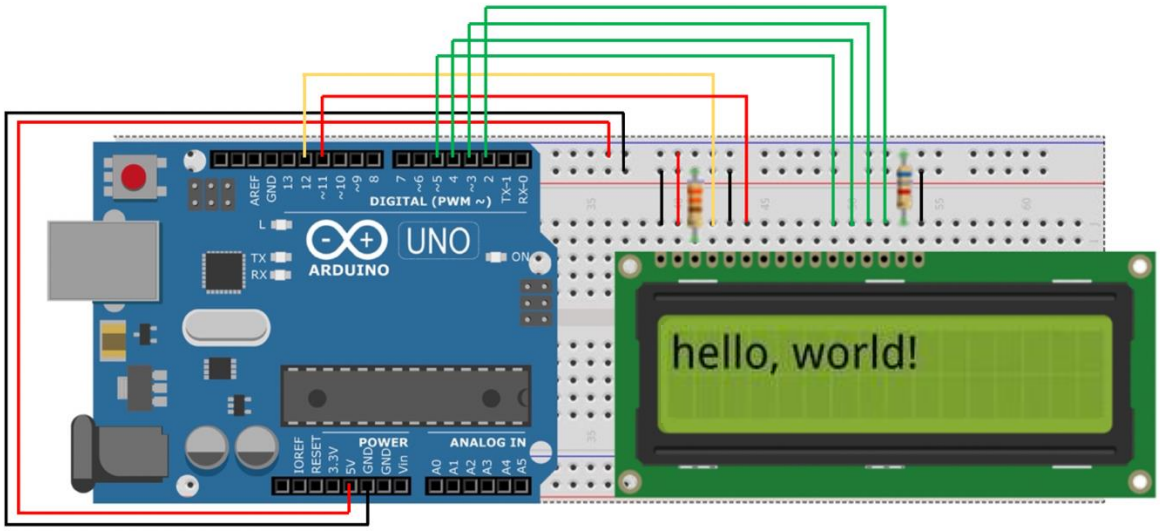
cols: ekranın sahip olduğu sütun sayısı
rows: ekranın sahip olduğu satır sayısı
lcd.print() - Metni LCD'ye yazdırır. Sözdizimi:
lcd.print(data)
lcd.print(data, BASE)
Parametreler:
lcd: LiquidCrystal türünde bir değişken
data: Yazdırılacak veri (char, byte, int, long veya string)
BASE (isteğe bağlı): sayıların yazdırılacağı taban: ikili için BIN (taban 2), ondalık için DEC (taban 10), sekizlik için OCT (taban 8), onaltılık için HEX (taban 16).
İfadeler:
byte print() yazılan bayt sayısını döndürür, ancak bu sayıyı okumak isteğe bağlıdır
lcd.setCursor() - LCD imlecini konumlandır; yani, LCD'ye yazılan sonraki metnin görüntüleneceği konumu ayarlayın. Sözdizimi:
lcd.setCursor(col, row)
Parametreler:
lcd: LiquidCrystal türünde bir değişken
col: imlecin konumlandırılacağı sütun (0 ilk sütun olmak üzere)
row: imlecin konumlandırılacağı satır (0 ilk satır olmak üzere)
lcd.clear(); - LCD ekranı temizler ve imleci sol üst köşeye konumlandırır.
Sözdizimi: lcd.clear()
Parametreler: lcd: LiquidCrystal türünde bir değişken

Devre 1:

Devre başlığı: "LCD'ye bir mesaj yazdırın"

Devre Açıklaması: Mikrodenetleyiciye bir LCD ekran bağlanarak ekrana istenilen mesajların yazdırılması mümkündür.

Not: Aşağıda açıklanan LCD işlevlerini kullanmak için LiquidCrystal.h kitaplığını eklemeniz gerekir.




```
/*  Mesaj yazdır  */
#include <LiquidCrystal.h> // kütüphane kodunu dahil et
// LCD'deki satır ve sütun sayısı için sabitler
const int numRows = 2;
const int numCols = 16;
// kütüphaneyi arayüz pinlerinin numaralarıyla başlatın
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
lcd.begin(numCols, numRows);
lcd.print("hello, world!"); // LCD'ye bir mesaj yazdırın.
}
```

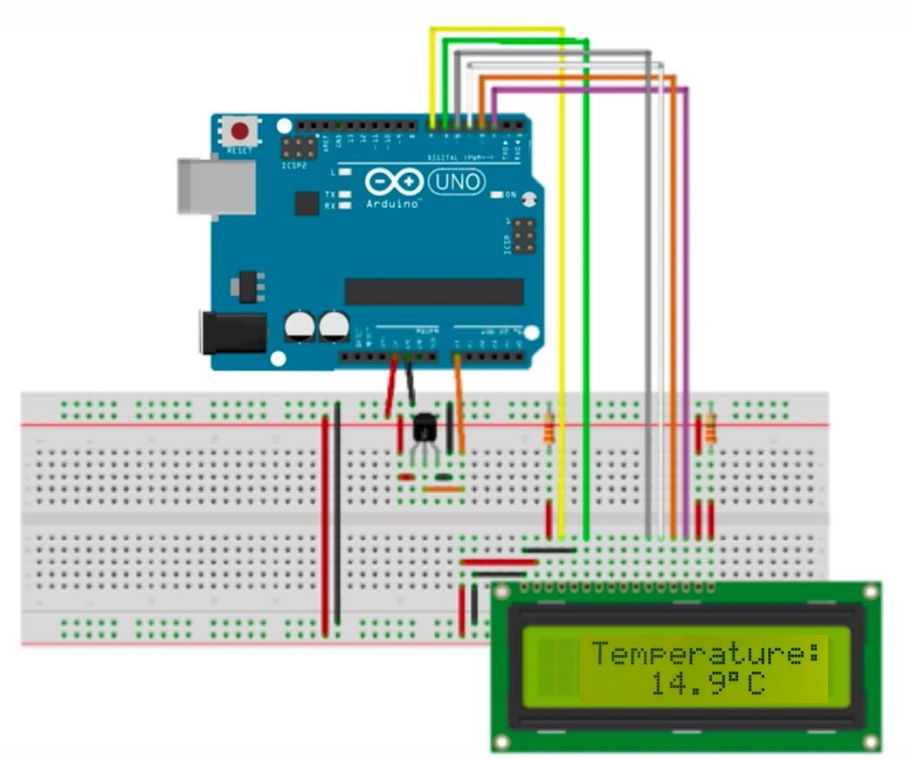
Devre 2:

Devre başlığı: "Dijital Termometre"

Devre Açıklaması: Arduino ile dijital bir termometre oluşturun. Sıcaklık, LM35 sıcaklık sensörü ile alınacak ve bir LCD ekranda görüntülenmeli ve her saniye güncellenmelidir.

LCD ekranın Vo pini, ekranda görüntülenen karakterlerin kontrastını ayarlar. Yukarıda bahsedildiği gibi GND'ye bağlı 3,3 kΩ dirence bağlanmalıdır. Ancak bazı ekranlarda Vo pinini doğrudan GND'ye bağlamanız gerekebilir.

Not: LM35 sensörünün 3 terminali vardır: biri güç kaynağı için, biri kütle için ve diğeri algılanan sıcaklığa orantılı voltaj çıkışı için, her bir santigrat derece için 10 mV'ye eşittir ve Santigrat derece olarak kalibre edilmiştir.

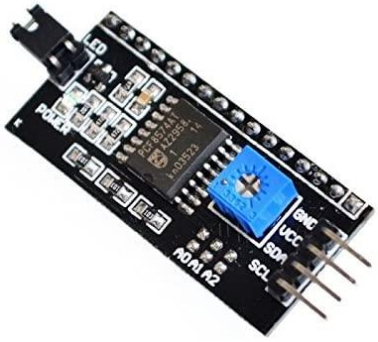



```
/*  Dijital termometre */
#include <LiquidCrystal.h> //Library to drive LCD display
```

```
#define pin_temp A0 // Sıcaklık sensörü Vout ayak bağlantı pin’i
float temp = 0; // Tespit edilen sıcaklığın saklanacağı değişken
LiquidCrystal lcd(7, 6, 5, 4, 3, 2); // Kütüphaneyi LCD ekran pin’leriyle başlatma
void setup()
{
  lcd.begin(16, 2); // Ekrandaki sütun ve satır sayısını ayarlama
  LCD  lcd.setCursor(0, 0); // İmleci ilk satırın (0. satır) ve ilk sütunun üzerine getirin
  lcd.print ("Temperature:"); İlk satıra 'Sıcaklık:' mesajını yazdırın
 //* 1.1V'de Uygulanan ADC Vref
 (sıcaklık hesaplamasında daha fazla doğruluk için)
  NEMLİ: Arduino Mega kullanıyorsanız INTERNAL'ı INTERNAL1V1 ile değiştirin */
  analogReference(INTERNAL);
}

void loop()
{
  /* sıcaklığı hesapla ==============================================*/
  temp = 0;
  for (int i = 0; i < 5; i++) { // Sonraki ifadeyi 5 kez yürütür
    temp += (analogRead(pin_temp) / 9.31); // Sıcaklık ve toplamı 'temp' değişkeninde hesaplar
  }
  temp /= 5; // Sıcaklık değerlerinin matematiksel ortalamasını hesaplar
  /*================================================================*/

  /* LCD ekranda sıcaklığı görüyorum
  ================================================*/
  lcd.setCursor(0, 1); // lcd.print(temp); İmleci ilk sütunun ve ikinci satırın
  üzerine getirin lcd.print(temp);
  // LCD ekran sıcaklığında kalıp
  lcd.print(" C"); // Ekranda bir boşluk ve 'C' yazı tipi oluşturun
 /*=============================================*/
  delay(1000); // Bir saniye gecikme (değiştirilebilir)
  }
```

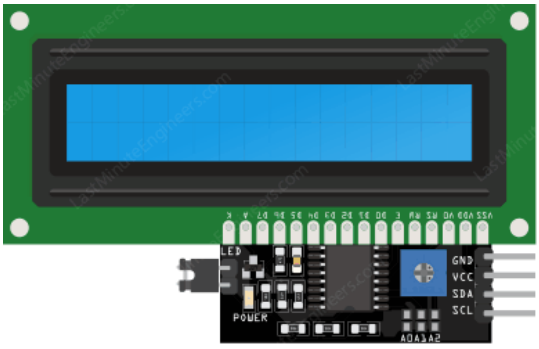
Arduino ile I2C LCD'yi Arayüzleme

Arduino ile LCD ekran bağlamak istiyorsanız Arduino üzerinde çok fazla pin tüketmeniz gerekiyor. 4 bit modunda bile, Arduino hala mevcut dijital G/Ç pinlerinin yarısı olan toplam yedi bağlantı gerektirir.
Çözüm, I2C protokolü ile arayüz oluşturan bir LCD ekran kullanmaktır. Yalnızca bir dijital I/O pin setinin parçası olamayacak ve diğer I2C cihazlarıyla paylaşılabilen iki I/O pini tüketir.
En yaygın I2C LCD ekranı, HD44780 tabanlı bir karakter LCD ekranı ve bir I2C LCD adaptöründen oluşur.



Adaptörün en önemli parçası 8-Bit G/Ç Genişletici yongası – PCF8574. Bu çip, bir Arduino'dan gelen I2C verilerini LCD ekranın gerektirdiği paralel verilere dönüştürür. Kart ayrıca ekranın kontrastında ince ayarlar yapmak için küçük bir potansiyometre ile birlikte gelir. Aynı I2C veriyolu üzerinde birden fazla cihaz kullanıyorsanız, başka bir I2C cihazı ile çakışmaması için kart için farklı bir I2C adresi ayarlamanız gerekebilir. Bu nedenle kartta üç adet lehim jumper'ı (A0, A1 ve A2) bulunur.

Bir I2C LCD, onu Arduino'ya bağlayan sadece 4 pin'e sahiptir.



Pin çıkışı aşağıdaki gibidir:

- GND: bir topraklama pin’idir ve Arduino'nun toprağına bağlanmalıdır;

- VCC: modüle ve LCD'ye güç sağlar. Arduino'nun 5V çıkışına veya ayrı bir güç kaynağına bağlayın;

- SDA: Seri Veri pinidir. Bu hat hem gönderme hem de alma için kullanılır. Arduino üzerindeki SDA pinine bağlanın;

- SCL: Seri Saat pinidir. Bu, Bus Master cihazı tarafından sağlanan bir zamanlama sinyalidir. Arduino üzerindeki SCL pinine bağlayın.

R3 düzenine sahip Arduino kartlarında, SDA (veri hattı) ve SCL (saat hattı), AREF pinine yakın pin başlıklarında bulunur. A5 (SCL) ve A4 (SDA) olarak da bilinirler.
Arduino modeline bağlı olarak doğru pinleri belirlemek için aşağıdaki tabloya bakın.

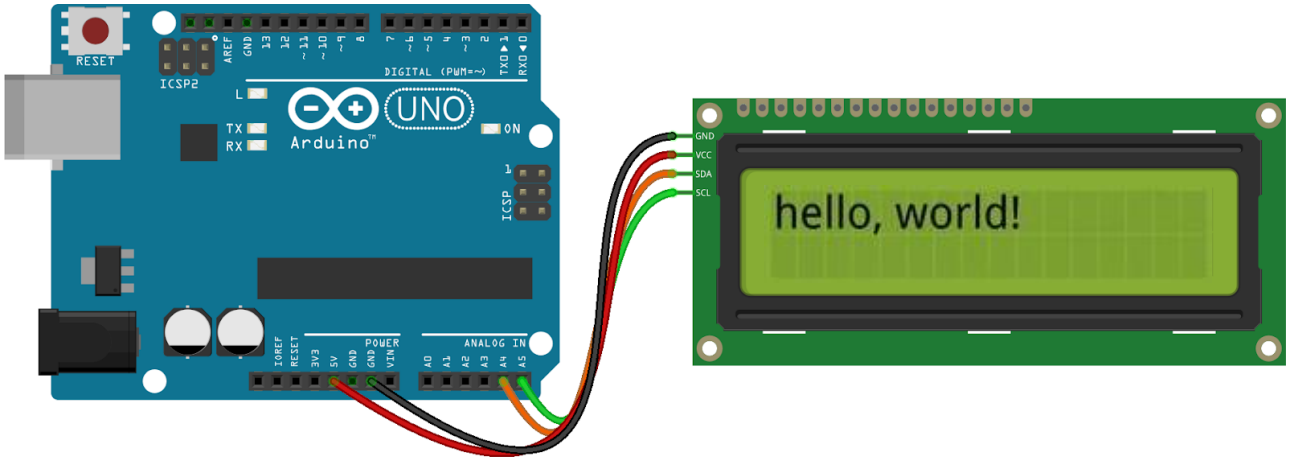
	SCL	SDA
Arduino Uno	A5	A4
Arduino Nano	A5	A4
Arduino Mega	21	20
Leonardo/Micro	3	2

Devre 3:

Devre başlığı: "I2C LCD'ye bir mesaj yazdırın"

Devre Açıklaması: Mikrodenetleyiciye sadece 4 pin ile bir LCD ekran bağlayıp ekrana istenilen mesajları yazdırmak mümkündür.

Not: LiquidCrystal_I2C adlı bir kitaplık kurmanız gerekir. Bu kitaplık, Arduino IDE'nizle birlikte gelen LiquidCrystal kitaplığının geliştirilmiş bir sürümüdür.




```
/*  I2C Ekranında bir mesaj yazdırın */
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3F,16,2);  // 16 karakter ve 2 satır ekran için LCD adresini 0x3F olarak ayarlayın
void setup() {
  lcd.init();
  lcd.clear();
  lcd.backlight();      // Arka ışığın açık olduğundan emin olun
  // LCD'nin her iki satırına da bir mesaj yazdırın.
  lcd.setCursor(2,0);   // İmleci 0 satırındaki karakter 2'ye ayarla
  lcd.print("Hello world!");
  lcd.setCursor(2,1);   // İmleci satır 1'deki karakter 2'ye taşıyın
  lcd.print("with I2C protocol!");
}
void loop() {
}
```

Arduino ile OLED Grafik Ekran Modülü

I2C protokolünü kullanmanın bir başka olasılığı da bir OLED (Organik Işık Yayan Diyot) ekranı seçmektir. Süper hafif ve incedirler ve daha parlak ve net bir resim üretirler.

Bir OLED ekran, arka ışık olmadan çalışır. Ekranın bu kadar yüksek kontrasta, son derece geniş görüş açısına sahip olmasının ve derin siyah seviyelerini gösterebilmesinin nedeni budur. Arka ışığın olmaması, OLED'i çalıştırmak için gereken gücü önemli ölçüde azaltır.

Şekilden de görebileceğimiz gibi, pin çıkışı yukarıda zaten görülen klasik dört pinli I2C arayüzüdür.

Arduino topluluğu, bu OLED ekranları işlemek için Adafruit'in SSD1306 kitaplığı gibi birkaç kitaplık geliştirdi. Kitaplığı kurmak için Çizim > Kitaplığı Dahil Et > Kitaplıkları Yönet'e gidin... Kitaplık Yöneticisinin kitaplıklar dizinini indirmesini ve kurulu kitaplıkların listesini güncellemesini bekleyin.

Arduino - OLED Grafik Ekran Modülü Fonksiyonları (Komutları)

pinMode();

display.begin()

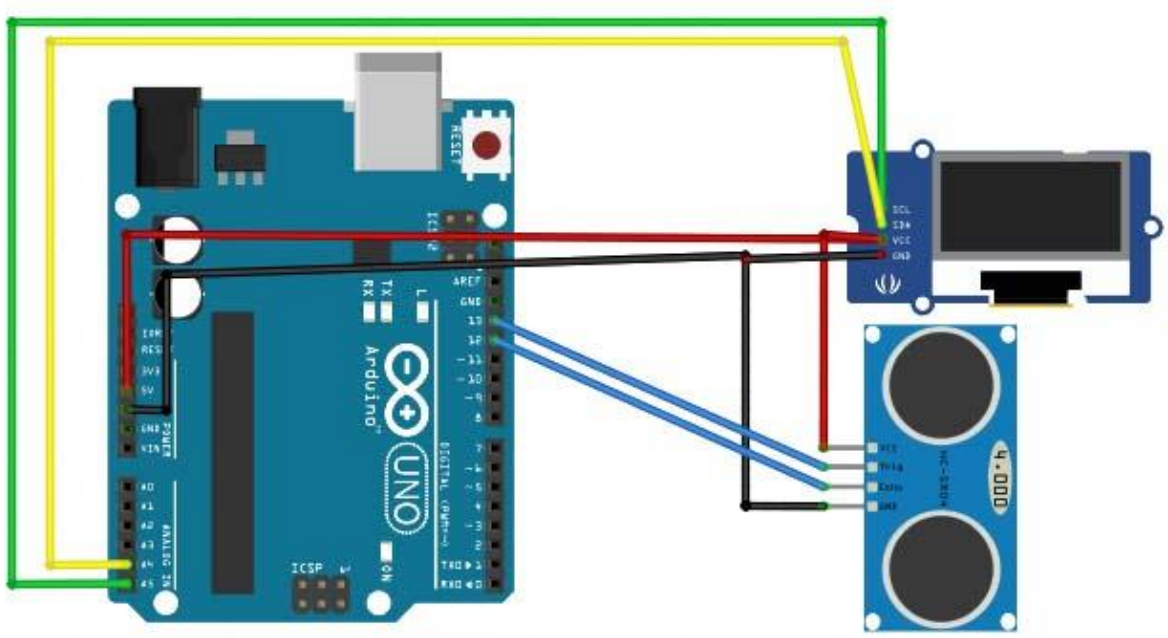
display.clearDisplay();

Devre 4:

Devre başlığı: "Mesafe sensörü"

Devre Açıklaması: Mesafe HC-SR04 ultrasonik sensör ile alınacak ve OLED ekranda görüntülenecektir.

Not: HC-SR04'te endişelenmeniz gereken yalnızca dört pin vardır: VCC (Güç), Trig (Trigger), Echo (Alma) ve GND (Ground).

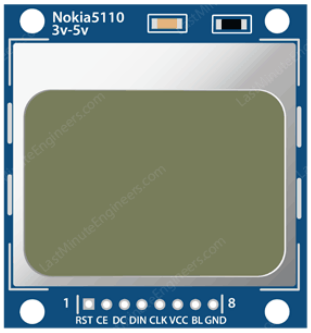



```
/*  Mesafe sensörü */

#include <SPI.h>        // Bu kütüphane, ana cihaz olarak Arduino ile SPI cihazları ile iletişim
                        kurmanıza izin verir.
#include <Wire.h>      // bu kitaplık I2C/TWI cihazlarıyla iletişim kurmanızı sağlar
#include <Adafruit_GFX.h>            // OLED ekranın kitaplıkları
#include <Adafruit_SSD1306.h>

#define CommonSenseMetricSystem
#define trigPin 13      // sensörün pin'lerini tanımlayın
#define echoPin 12
```

```
void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // I2C adresi 0x3C (128x64) ile başlat
  display.clearDisplay();

}

void loop() {
  long duration, distance;

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = (duration/2) / 29.1;
  display.setCursor(22,20);    // OLED ekran ayar imleci
  display.setTextSize(3);        // metnin boyutu
  display.setTextColor(WHITE);       // siyah yazarsan her şeyi siler
  display.println(distance);     // değişkenimizi yazdır
  display.setCursor(85,20);
  display.setTextSize(3);

    display.println("cm");

  display.display();

  delay(500);
  display.clearDisplay();

  Serial.println(distance);//debug
}
```

Arduino ile Nokia 5110 Grafik LCD Ekranı Arayüzü

Arduino'yu Nokia'nın 3310 ve 5110 cep telefonlarında kullandığına benzer küçük LCD'lerle arayüzleyebilirsiniz. bu ekranlar küçüktür (yalnızca 1,5 inç), ucuzdur, kullanımı kolaydır, oldukça düşük güçtedir (yalnızca 6 ila 7mA kadar düşük) ve metinlerin yanı sıra bitmapleri de görüntüleyebilir.



Bunlar 84×48 piksellik grafik ekranlardır. SPI'ye benzer bir seri veri yolu arabirimi aracılığıyla mikro denetleyicilere arabirim oluştururlar. LCD ayrıca kırmızı, yeşil, mavi ve beyaz gibi farklı renklerde bir arka ışık ile birlikte gelir. Arka ışık, ekranın kenarlarına yerleştirilmiş dört LED'den başka bir şey değildir.
Arduino ile arayüz oluşturan 8 pini vardır, pin çıkışı aşağıdaki gibidir:

- RST: ekranı sıfırlar. Aktif bir düşük pin anlamıdır; aşağı çekerek ekranı sıfırlayabilirsiniz. Bu pini de Arduino resetine bağlayarak ekranı otomatik olarak resetlemesini sağlayabilirsiniz;

- CE(Chip Enable): aynı SPI veri yolunu paylaşan birçok bağlı cihazdan birini seçmek için kullanılır;

- D/C(Veri/Komut): pin ekrana aldığı verinin komut mu yoksa görüntülenebilir veri mi olduğunu belirler;

- DIN: SPI arayüzü için bir seri veri pinidir;

- CLK: SPI arayüzü için bir seri saat pinidir;

- VCC: Arduino üzerindeki 3.3V volt pinine bağladığımız LCD'ye güç sağlar;

- BL(Arka Işık): ekranın arka ışığını kontrol eder. Parlaklığını kontrol etmek için bir potansiyometre ekleyebilir veya bu pini herhangi bir PWM özellikli Arduino pinine bağlayabilirsiniz;

- GND: bir topraklama pin'idir ve Arduino'nun toprağına bağlanmalıdır.

Veri iletim pinlerini herhangi bir dijital I/O pinine bağlayabilirsiniz. LCD'nin 3v iletişim seviyeleri vardır, bu yüzden bu pinleri doğrudan Arduino'ya bağlayamıyoruz. Bunun bir yolu, her bir veri iletim pinine inline dirençler eklemektir. Sadece CLK, DIN, D/C ve RST pinleri arasına 10kΩ direnç ve CE arasına 1kΩ direnç ekleyin. Arka ışık (BL) pin'i, 330Ω akım sınırlama direnci ile 3.3V'a bağlanır. Parlaklığını kontrol etmek istiyorsanız, bir potansiyometre ekleyebilir veya bu pini herhangi bir PWM özellikli Arduino pinine bağlayabilirsiniz.
Arduino topluluğu, bu NOKIA ekranlarını işlemek için Adafruit'in PCD8544 Nokia 5110 LCD kitaplığı gibi birkaç kitaplık geliştirdi. Kitaplığı kurmak için Çizim > Kitaplığı Dahil Et > Kitaplıkları Yönet'e gidin... Kitaplık Yöneticisinin kitaplıklar dizinini indirmesini ve kurulu kitaplıkların listesini güncellemesini bekleyin.

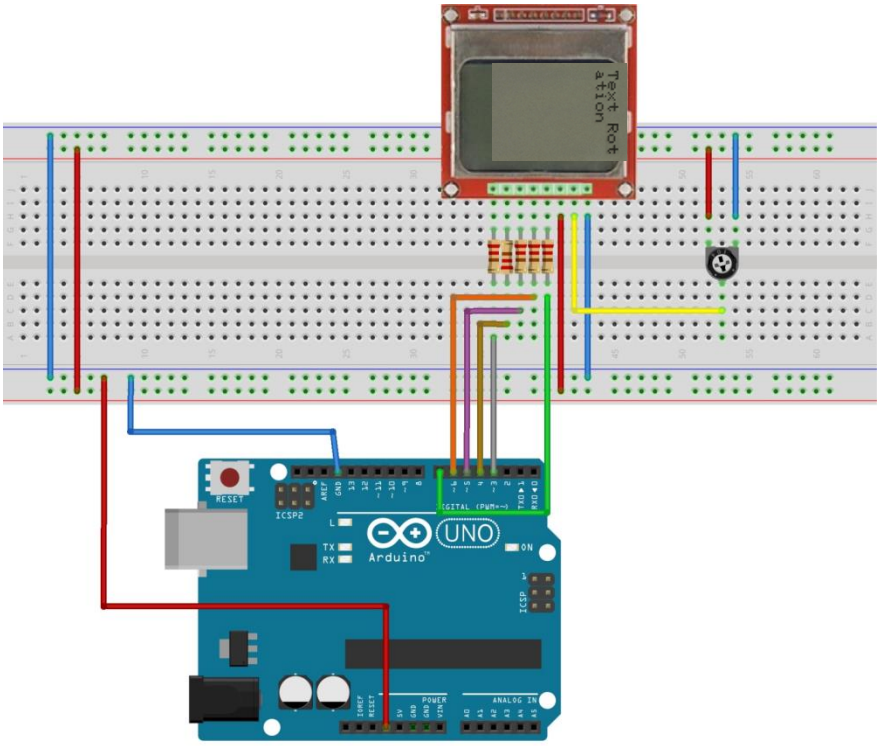
Devre 5:

Devre başlığı: "Metin Döndürme"

Devre Açıklaması: setRotation() işlevini çağırarak ekranın içeriğini döndürebilirsiniz. Ekranınızı portre modunda görüntülemenizi veya baş aşağı çevirmenizi sağlar.

Not: İşlev, 4 ana dönüşe karşılık gelen yalnızca bir parametreyi kabul eder. Bu değer, 0'dan başlayarak negatif olmayan herhangi bir tam sayı olabilir. Değeri her artırdığınızda, ekranın içeriği saat yönünün tersine 90 derece döndürülür. Örneğin:

• 0 – Ekranı standart yatay yönde tutar.
• 1 – Ekranı 90° sağa döndürür.
• 2 – Ekranı baş aşağı çevirir.
• 3 – Ekranı 90° sola döndürür.



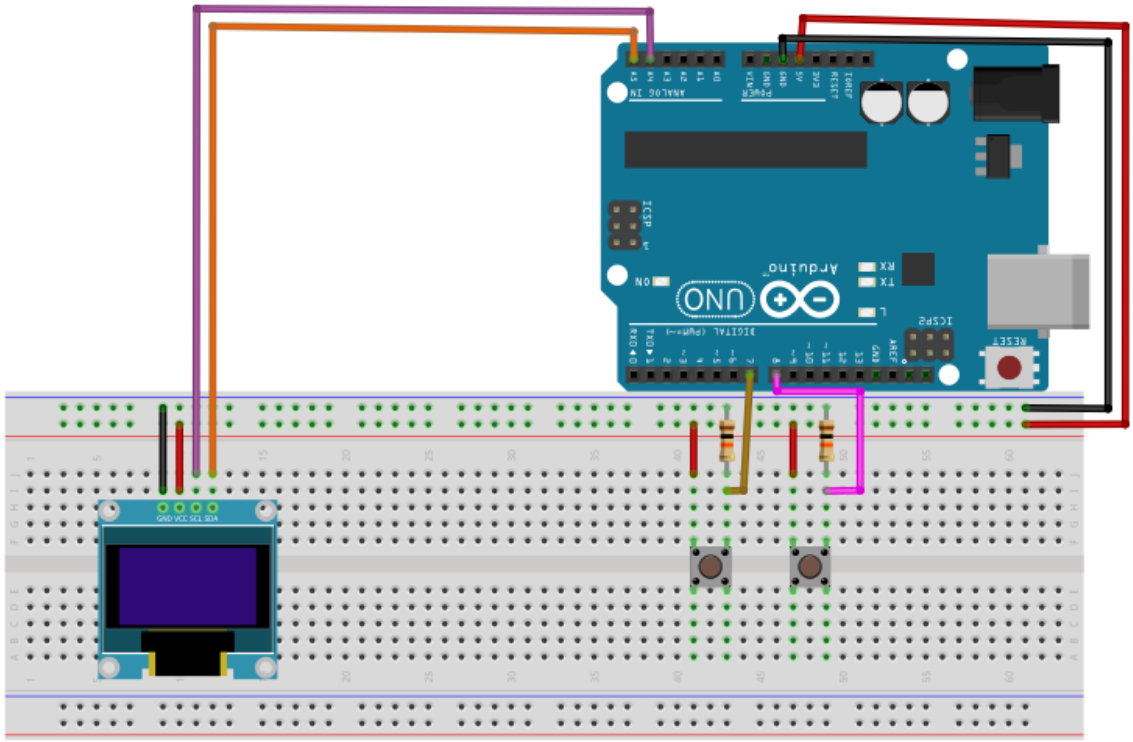

```
/*  Metin Döndürme */
#include <SPI.h>        // Bu kütüphane, ana cihaz olarak Arduino ile SPI cihazları ile iletişim
                        kurmanıza izin verir.
#include <Adafruit_GFX.h>           // OLED ekranın kitaplıkları
#include <Adafruit_PCD8544.h>
// Declare LCD object for software SPIAdafruit_PCD8544(CLK,DIN,D/C,CE,RST);
Adafruit_PCD8544 display = Adafruit_PCD8544(7, 6, 5, 4, 3);

void setup()   {
```

```
        Serial.begin(9600);

        //Initialize Display
        display.begin();

        display.setContrast(57);        // ekranı uyarlamak için kontrastı değiştirebilirsiniz

        display.clearDisplay();         // arabelleği temizle

// Text Rotation
while(1)
{
        display.clearDisplay();
        display.setRotation(rotatetext);
        display.setTextSize(1);
        display.setTextColor(BLACK);
        display.setCursor(0,0);
        display.println("Text Rotation");
        display.display();
        delay(1000);
        display.clearDisplay();
        rotatetext++;
}
}

void loop() {}
```

Devre 6:

Devre başlığı: "Düğme sayacı"

Devre Açıklaması: İki farklı düğmenin tıklanmasıyla artırılıp azaltılabilen bir sayı gösteren OLED ekran.



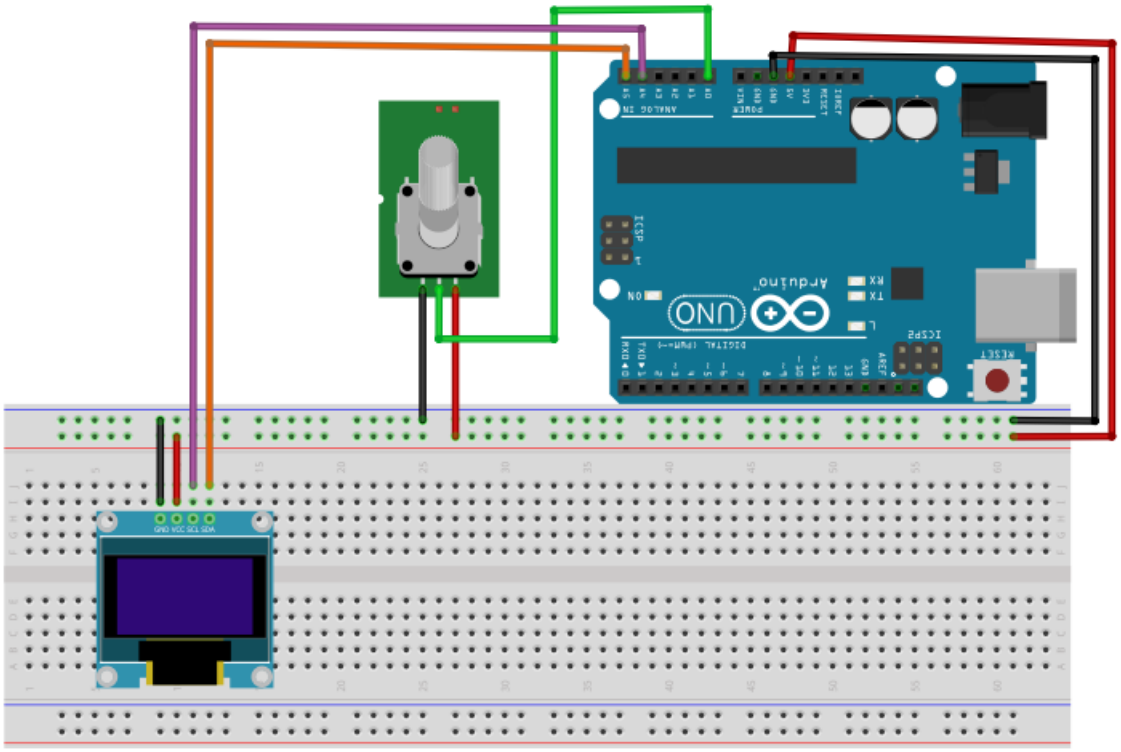
```
#include <U8glib.h> //lcd libraries
#include "U8glib.h"
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE|U8G_I2C_OPT_DEV_0);  //display model
int button_plus = 8, button_minus = 7; // bildirim pin düğmeleri
int state_plus, state_minus, number=0;
void setup() {
  pinMode(button_plus,INPUT);
  pinMode(button_minus,INPUT);
  Serial.begin(9600);
  u8g.setFont(u8g_font_fub25n);  // sayının yazılması için kullanılacak yazı tipi
}
```

```
void write_number(void) {
  u8g.setPrintPos(10,50);
  u8g.print(number);
}
void loop() {
  //buttons
  state_plus = digitalRead(button_plus);
  state_minus = digitalRead(button_minus);
  if(state_plus==1)
  {
      number++;
      delay(50);
  }
  if(state_minus==1)
  {
    number--;
    delay(50);
  }
  //write de number sul display
  u8g.firstPage();
  do {
    write_number();
  } while  ( u8g.nextPage() );
    u8g.firstPage();
}
```

Devre 7:

Devre başlığı: "Potansiyometreli sayaç"

Devre Açıklaması: Potansiyometre döndürüldükçe artan ve azalan bir sayı gösteren OLED ekran.



```
#include <U8glib.h> //lcd libraries
#include "U8glib.h"
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE|U8G_I2C_OPT_DEV_0);  //display model

int potentiometer = 0; //declaration and potentiometer
int state_pot, stop_pot, difference, number=0;

void setup() {
  Serial.begin(9600);
  u8g.setFont(u8g_font_fub25n);//font to be used for the write of the number
}

void write_number(void) {
```

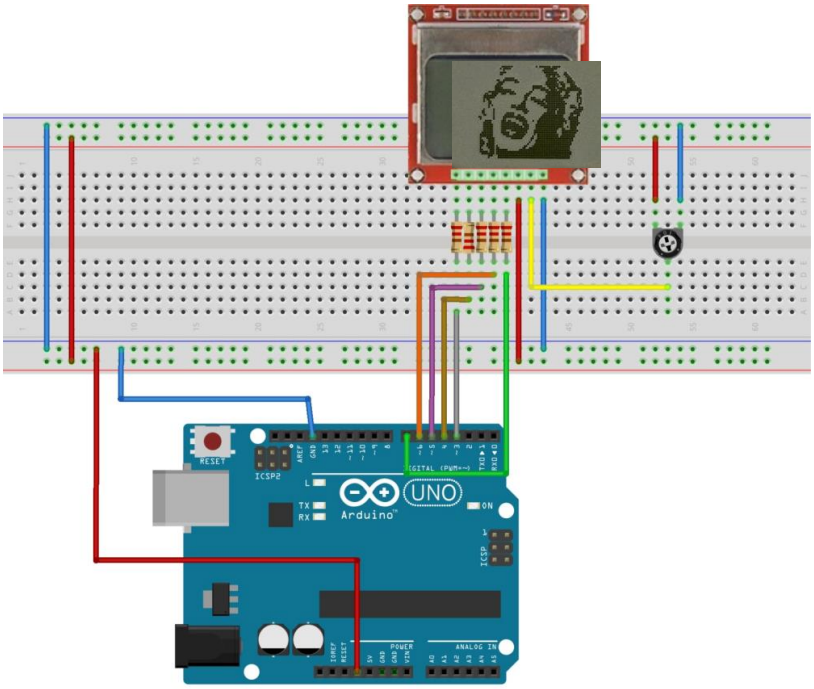
```
  u8g.setPrintPos(10,50);
  u8g.print(number);
}

void loop() {
  state_pot = analogRead(potentiometer);

  //potentiometer
  stop_pot = state_pot;//save the value when it is stop to see if the potentiometer turns clockwise
or counterclockwise
  delay(100);
  state_pot = analogRead(potentiometer);
  difference = stop_pot-state_pot;
  if((difference>2)||(difference<2))// potansiyometre dururs a işlem yapmaktan kaçınmak için
kullanılır
  {
    number=number-difference/5;// fark 0'dan büyükse saat yönünde döner aksi halde çıkarır
    delay(100);
  }

  // ekrandaki numarayı yaz
  u8g.firstPage();
  do {
    write_number();
  } while
  ( u8g.nextPage() );
    u8g.firstPage();
}
```

Devre 8:

Devre başlığı: "Marilyn Bmp Image"

Devre Açıklaması: Nokia 5110 LCD Ekranına bitmap görüntüleri nasıl çizilir. Bu örnekte Marilyn Monroe'nun bir portresi var.

Not: Nokia 5110 LCD ekranının ekran çözünürlüğü 84×48 pikseldir, bu nedenle bundan daha büyük resimler düzgün görüntülenmeyecektir. Bitmap görüntüsünü Nokia 5110 LCD ekranında göstermek için drawBitmap() işlevini çağırmamız gerekiyor. Altı parametre alır:. sol üst köşe X koordinatı, sol üst köşe Y koordinatı, monokrom bitmap bayt dizisi, piksel cinsinden bitmap genişliği, piksel cinsinden bitmap yüksekliği ve Renk. Örneğimizde bitmap görüntüsü 84×48 boyutundadır. Bu nedenle, X ve Y koordinatları 0'a, genişlik ve yükseklik ise 84 ve 48'e ayarlanmıştır.



```
/*  Marilyn Bmp Görüntüsü */
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
Adafruit_PCD8544 display = Adafruit_PCD8544(7, 6, 5, 4, 3);
// 'Marilyn Monroe 84x48', 84x48px
const unsigned char MarilynMonroe [] PROGMEM = {
0x00, 0x00, 0x00, 0x7f, 0x00, 0x02, 0xfe, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xbe, 0x00,
0x00, 0x1f, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x3f, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0xf0, 0x00, 0x00, 0x1f, 0xe1, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0,
0x00, 0x00, 0x0f, 0xf1, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0e, 0xd8, 0xe0,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x80, 0x00, 0x07, 0xe0, 0x70, 0x00, 0x00, 0x00, 0x00, 0x03,
0x3f, 0xe0, 0x00, 0x07, 0xf0, 0x78, 0x00, 0x00, 0x00, 0x00, 0x01, 0xe0, 0x70, 0x00, 0x0f, 0xee,
0x7c, 0x00, 0x00, 0x00, 0x00, 0x03, 0xc0, 0x00, 0x00, 0x0f, 0xf7, 0x1c, 0x00, 0x00, 0x00, 0x00,
0x07, 0x80, 0x00, 0x0f, 0xc7, 0xf3, 0x1e, 0x00, 0x00, 0x00, 0x00, 0x07, 0xc0, 0x00, 0x0f, 0xf3,
0xdf, 0x7f, 0x80, 0x00, 0x00, 0x00, 0x07, 0xfe, 0x00, 0x08, 0x7d, 0xef, 0xff, 0xc0, 0x00, 0x00,
0x00, 0x7f, 0xff, 0x80, 0x30, 0x0f, 0xfc, 0xe0, 0xc0, 0x00, 0x00, 0x01, 0x9e, 0x73, 0xc0, 0xe0,
0x07, 0xf8, 0xc1, 0xc0, 0x00, 0x00, 0x03, 0xfc, 0x00, 0x01, 0xc0, 0x0f, 0xfd, 0xe1, 0x80, 0x00,
0x00, 0x03, 0xf8, 0x00, 0x01, 0x9c, 0x0f, 0xff, 0xc1, 0xc0, 0x00, 0x00, 0x02, 0xc0, 0x00, 0x01,
0x9f, 0xbf, 0xfe, 0x01, 0x40, 0x00, 0x00, 0x02, 0x60, 0x00, 0x03, 0x07, 0xef, 0xff, 0x01, 0x40,
0x00, 0x00, 0x00, 0x60, 0x00, 0x07, 0x01, 0xf7, 0xff, 0x80, 0xc0, 0x00, 0x00, 0x00, 0x50, 0x01,
0xdf, 0x00, 0x7f, 0xff, 0x1c, 0x80, 0x00, 0x00, 0x00, 0x40, 0x01, 0xff, 0x00, 0x1f, 0xff, 0x1e,
0xe0, 0x00, 0x00, 0x02, 0x08, 0x00, 0x3f, 0x80, 0x07, 0xef, 0x03, 0xe0, 0x00, 0x00, 0x06, 0x08,
0x00, 0x03, 0xc0, 0x07, 0xdf, 0x07, 0xc0, 0x00, 0x00, 0x06, 0x08, 0x0f, 0x81, 0x80, 0x1f, 0xdf,
0x1f, 0x80, 0x00, 0x00, 0x03, 0x08, 0x1f, 0x98, 0x00, 0x3f, 0xfe, 0x19, 0x80, 0x00, 0x00, 0x18,
0x08, 0x3f, 0xfe, 0x00, 0x7f, 0xfe, 0x3f, 0x00, 0x00, 0x00, 0x08, 0x08, 0x30, 0x3f, 0x00, 0xff,
0xff, 0x3f, 0x00, 0x00, 0x00, 0x01, 0xe0, 0x76, 0x0f, 0x89, 0xff, 0xff, 0x9f, 0x00, 0x00, 0x00,
0x03, 0xe0, 0x7f, 0xc3, 0x81, 0xff, 0xfe, 0x9f, 0x80, 0x00, 0x00, 0x03, 0xf0, 0x7f, 0xf3, 0xc3,
0xff, 0xfe, 0x1f, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x7f, 0xfd, 0xc3, 0xff, 0xfe, 0x5e, 0x00, 0x00,
0x00, 0x03, 0xf0, 0x7f, 0xff, 0xc3, 0xff, 0xf3, 0x1e, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x71, 0xff,
0x87, 0xff, 0xe3, 0xff, 0x00, 0x00, 0x00, 0x07, 0xf0, 0x7c, 0x3f, 0x87, 0xff, 0xe3, 0xfe, 0x00,
0x00, 0x00, 0x0f, 0xf0, 0x3c, 0xff, 0x05, 0xff, 0xf3, 0xfc, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x0f,
0xfe, 0x09, 0xff, 0xf7, 0xfc, 0x00, 0x00, 0x00, 0x08, 0xf8, 0x01, 0xfc, 0x19, 0xff, 0xff, 0xf8,
0x00, 0x00, 0x00, 0x0c, 0x78, 0x00, 0x00, 0x13, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x0e, 0x78,
0x00, 0x00, 0x23, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x0e, 0xf8, 0x00, 0x00, 0x47, 0xff, 0xff,
0xf0, 0x00, 0x00, 0x00, 0x0c, 0xfa, 0x00, 0x01, 0x8f, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x08,
0x7b, 0x00, 0x03, 0x3f, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x0c, 0x3b, 0xf8, 0x0f, 0xff, 0xff,
0xff, 0xe0, 0x00, 0x00, 0x00, 0x0f, 0xbb, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00,
0x07, 0xfb, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x71, 0xff, 0xff, 0xff,
0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x41, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x00};
void setup()  {
        Serial.begin(9600);
        display.begin();
        display.setContrast(57);
        display.clearDisplay();
        // Display bitmap
        display.drawBitmap(0, 0,  MarilynMonroe, 84, 48, BLACK);
        display.display();
```

```
        // Invert Display
        //display.invertDisplay(1);
}
void loop() {}
```

Co-funded by the
Erasmus+ Programme
of the European Union

Erasmus+ KA210-VET

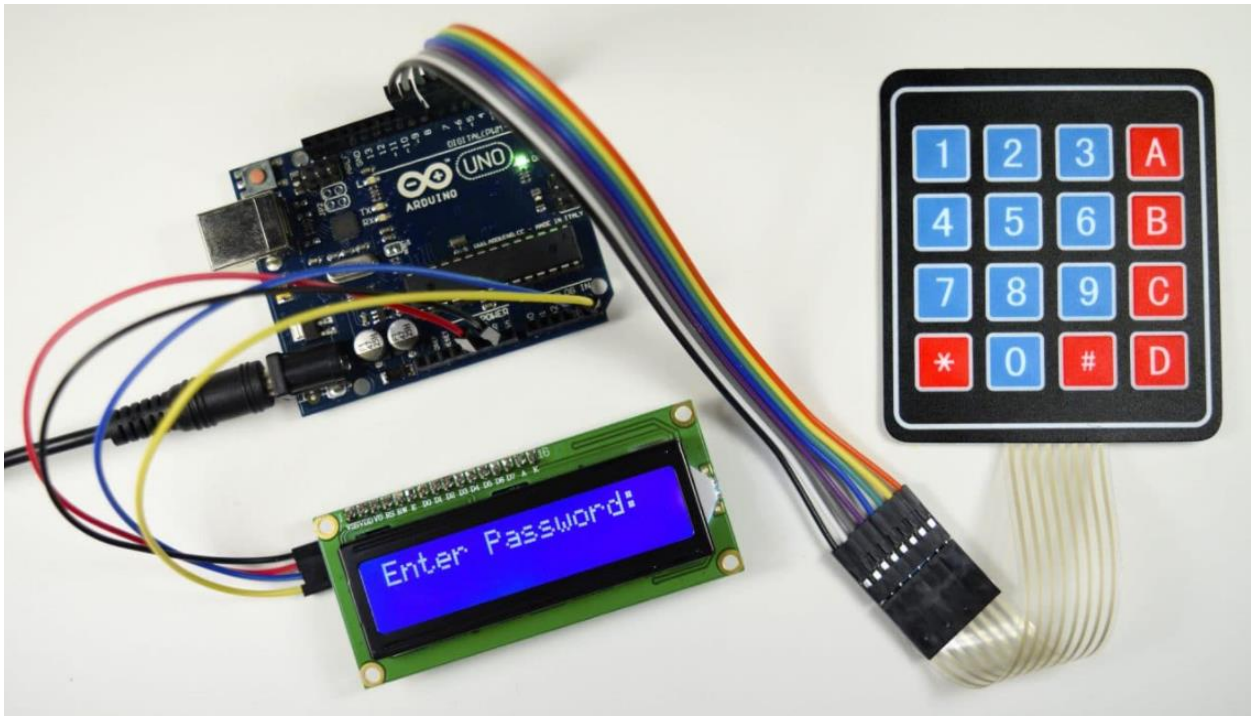
Küçük Ölçekli Mesleki Eğitim ve Öğretim Ortaklık Projesi

Proje adı: "Mesleki Eğitimde Arduinoları Kullanma"

Proje kısaltması: "UsingARDinVET"

Proje No: "2023-1-RO01-KA210-VET-000156616"

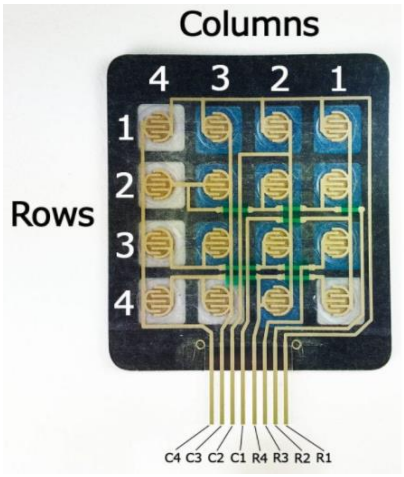
Keypad Modülü ve Eğitim Kiti



Keypad (Tuş takımı) nedir?

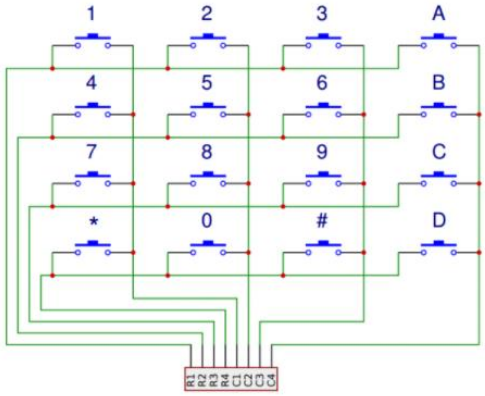
Tuş takımı, bir satır ve bir sütun arasında bağlantı sağlamak için bir anahtarlama elamanı olarak çalışan, bir matris içinde düzenlenmiş butonlar sistemidir.

Burada, 4X4 matrisli bir membran (ince) keypad kullanıyoruz. Çünkü, membran keypad incedir ve çoğu düz yüzeye yapıştırılabilmesi için yapışkan desteği vardır. Her tuşun altında bir membran anahtar bulunur. Sıradaki her bir tuş, yüzeyin altındaki iletken bir yol ile sıradaki diğer tuşlara bağlanır. Bir sütundaki her tuş aynı şekilde bağlanır - tuşun bir tarafı, iletken bir yol ile o sütundaki diğer tüm tuşlara bağlanır. Her satır ve sütun için yani toplamda, 4X4 tuş takımında toplam 8 pini için dışarıya bir bağlantı pini getirilir.	

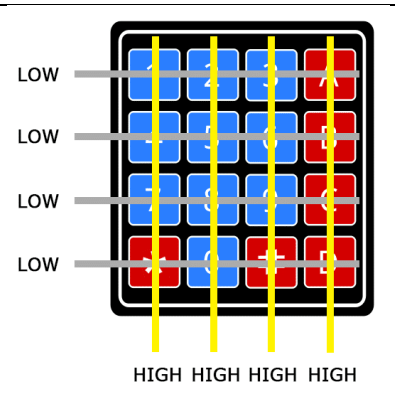
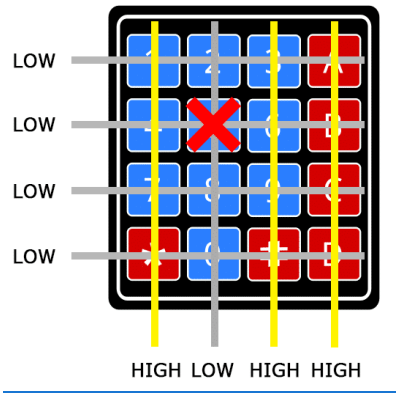


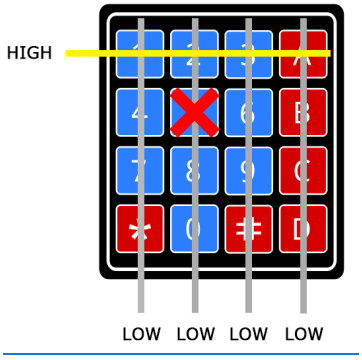
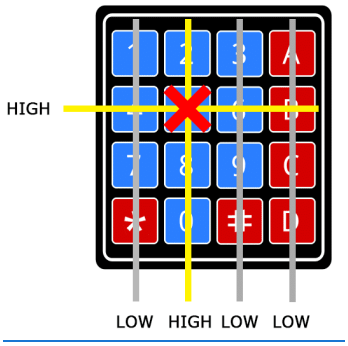
Bir tuşa basarak, bir sütun ve bir satır yolu arasındaki anahtar kapatılır akım bir sütun pini ile bir satır pini arasında akmasına izin verilir.

4X4 tuş takımı şeması, satırların ve sütunların nasıl bağlandığını gösterir:



Arduino, butona bağlı olan satır ve sütun pinini algılayarak hangi butona basıldığını algılar. Bu dört adımda gerçekleşir:

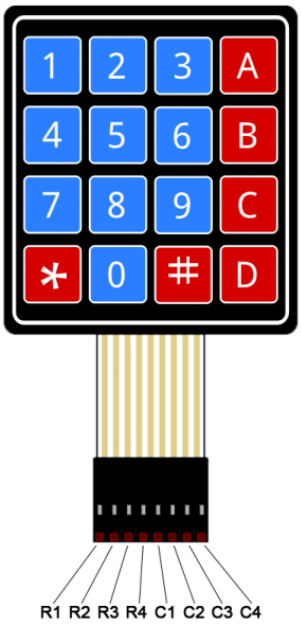
1. İlk olarak, hiçbir tuşa basılmadığında, tüm sütun pinleri YÜKSEK (HIGH) ve tüm satır pinleri DÜŞÜK (LOW) tutulur	
2. Bir tuşa basıldığında, YÜKSEK (HIGH) sütundan gelen akım, DÜŞÜK (LOW) satır pinine aktığı için, sütun pini DÜŞÜK'e (Lojik-0'a) çekilir:	

3. Arduino, artık butonun hangi sütunda olduğunu bilir. Bu yüzden şimdi sadece butonun bulunduğu satırı bulması gerekir. Bunu, satır pinlerinin her birini YÜKSEK olarak değiştirerek ve aynı zamanda YÜKSEK'e dönen sütün pinlerini tespit ederken, tüm sütun pinlerini okuyarak yapar.	
4. Sütun pini tekrar YÜKSEK'e gittiğinde, Arduino tuşa bağlı olan satır pinini bulur.	

Yukarıdaki şekilde, 2. satır ve 2. sütun kombinasyonunun yalnızca 5 numaralı tuşa basıldığı anlamına gelebileceğini görebiliriz.

TUŞ TAKIMINI ARDUINOYA BAĞLAMA

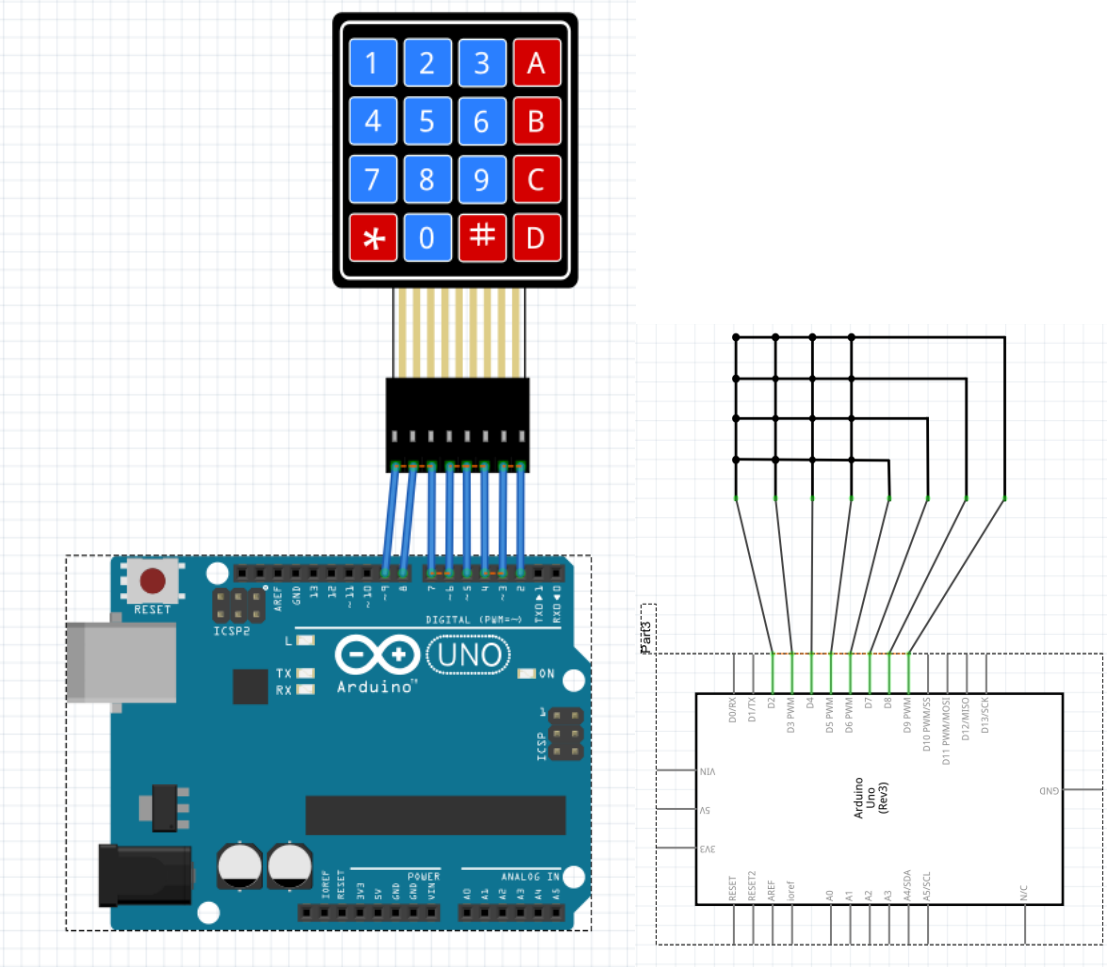
Çoğu membran tuş takımı için pin düzeni aşağıdaki gibidir:



Devre 1:

Devre Adı: Basılan tuşu Seri monitörde gösterir

Devre tanımı: Program bize basılan tuşun numarasının seri monitöre nasıl yazdırılacağını gösterir.

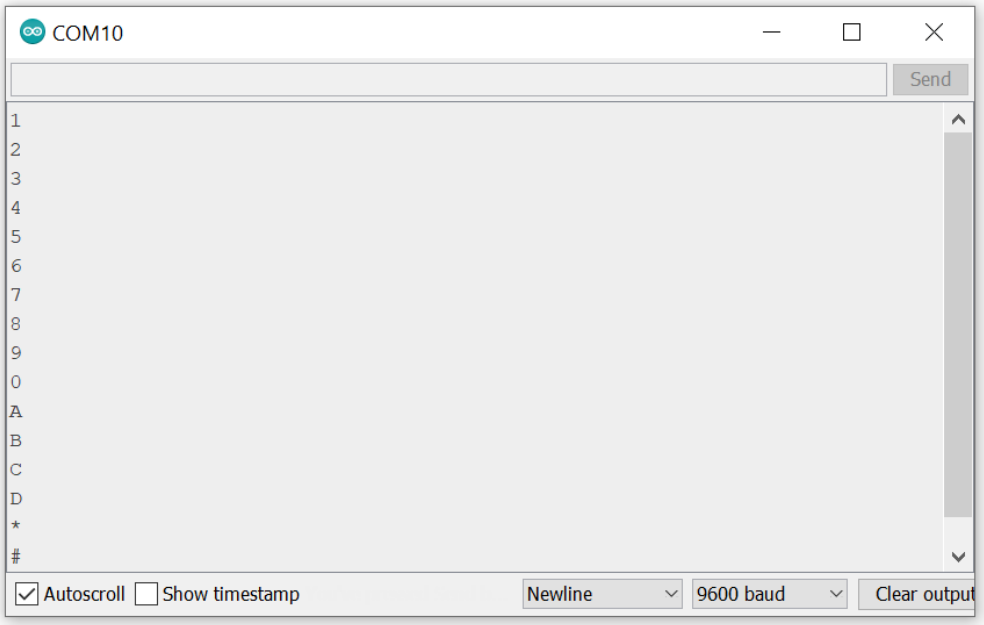


1-) Breadboard görünümü

2-) Şematik görünüm

```
/*     "Basılan tuş Seri Monitörde gösterilir"   */

#include <Keypad.h>
const byte ROWS = 4;
const byte COLS = 4;
char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};
```

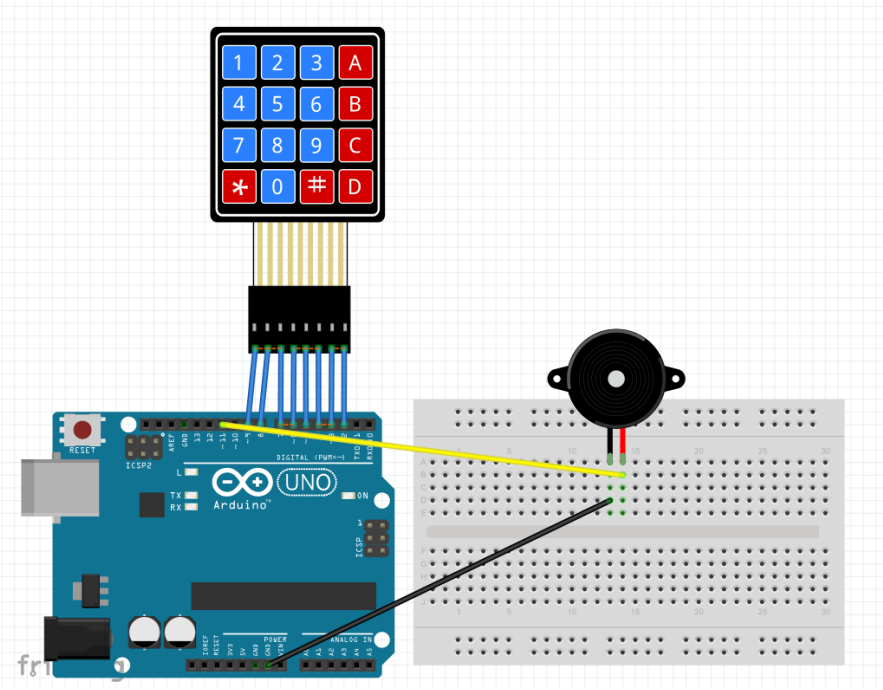
```
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
void setup(){
  Serial.begin(9600);
}
void loop(){
  char customKey = customKeypad.getKey();
  if (customKey){
    Serial.println(customKey);
  }
}
```



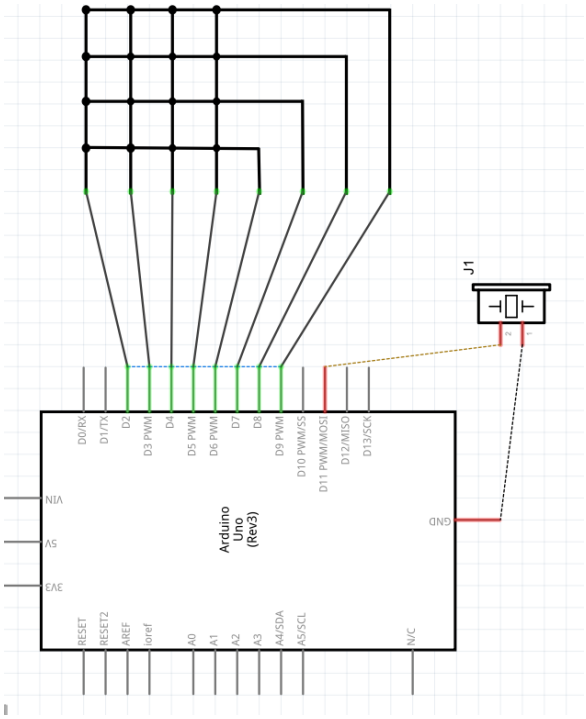
Devre 2:

Devre Adı: Arduino Keypad + Buzzer (Beep)

Devrenin tanımı: Tuş takımındaki bir tuşa basıldığında, buzzer bip sesi çıkarır.



1-) Breadboard görünümü



2-) Şematik görünüm

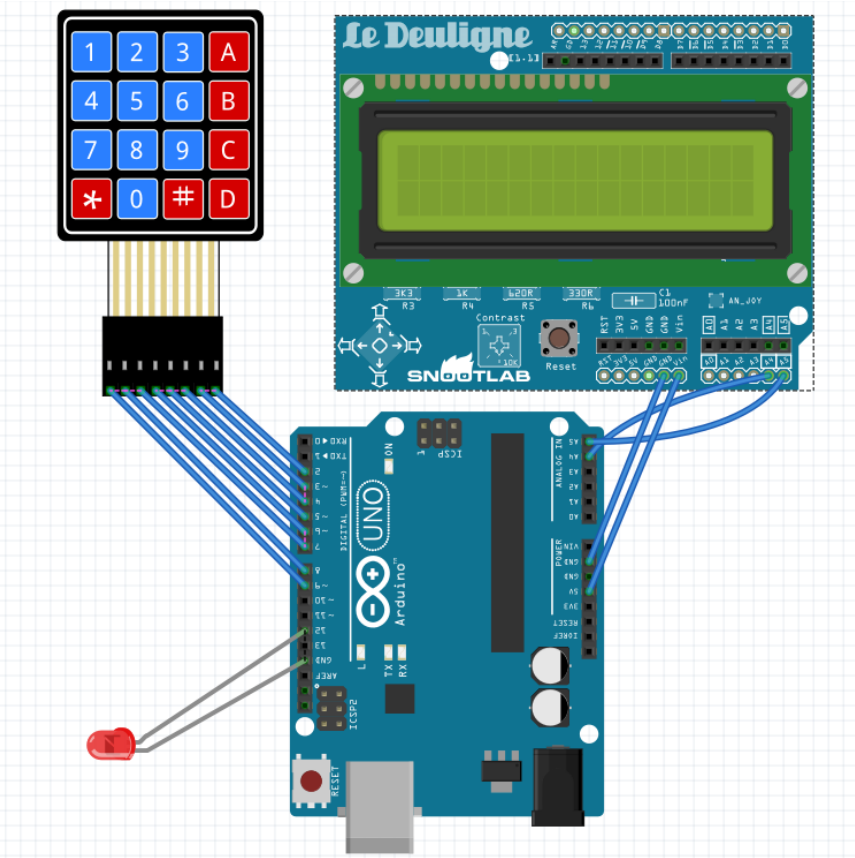
/* "Arduino Keypad + Buzer (Bip) " */

```
#include <Keypad.h>
#include <ezBuzzer.h>
const int BUZZER_PIN = 11;
const int ROW_NUM    = 4; // four rows
const int COLUMN_NUM = 4; // four columns
char keys[ROW_NUM][COLUMN_NUM] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte pin_rows[ROW_NUM] = {9, 8, 7, 6};      // Tuş takımının satır pinlerine bağla
byte pin_column[COLUMN_NUM] = {5, 4, 3, 2}; // Tuş takımının sütün  pinlerine bağla

Keypad keypad = Keypad(makeKeymap(keys), pin_rows, pin_column, ROW_NUM,
COLUMN_NUM );
ezBuzzer buzzer(BUZZER_PIN);       // Pine bağlanan ezBuzzer nesnesini oluştur;
void setup() {
  Serial.begin(9600);
}
void loop() {
  buzzer.loop(); // MUST call the buzzer.loop() function in loop()
  char key = keypad.getKey();
  if (key) {
    Serial.print(key);                 // Basılan tuş numarasını Seri monitöre yazdır.
    buzzer.beep(100);                  // 100ms beep  sesi çıkart.
  }
}
```

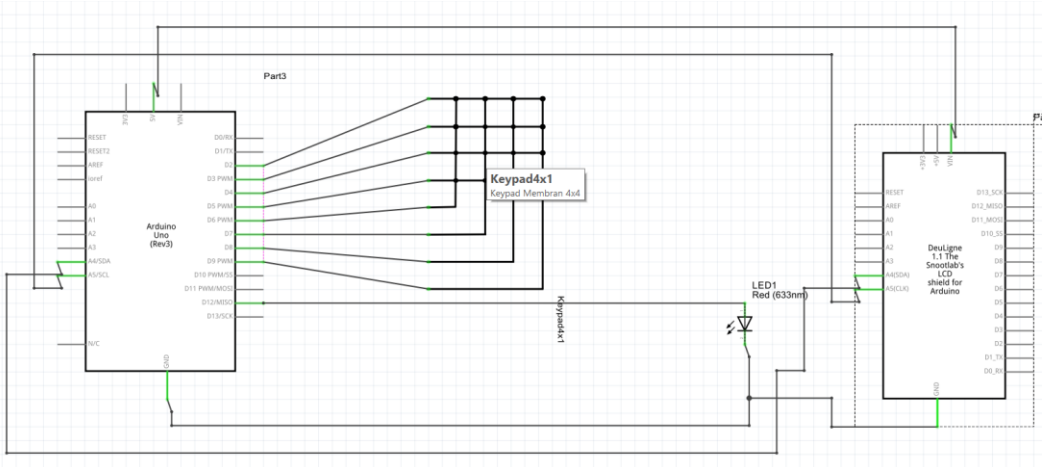
Devre 3:

Devre Adı: Arduino Kilit Açma kodu

Devrenin tanımı: Arduino bağlı bir tuş takımını ayarlayan program. Doğru kodu yazdığınızda bir LED yanacaktır.



1-) Breadboard görünümü



2-) Şematik görünüm

```
/* Arduino Kilit Açma Kodu */
#include <Keypad.h>     // Arduino IDE ile indirebileceğimiz kütüphaneler.
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#define Solenoid 12          //Gerçekte,  solenoidi kontrol eden transistörün Geyti (Beyzi),
                             //Biz bu durumumda, basit bir LED kullandık.
#include <liquidcrystal_i2c.h></liquidcrystal_i2c.h></lcd.h></wire.h></keypad.h>
#define I2C_ADDR 0x27 // LCD i2c Adress and pins
#define BACKLIGHT_PIN 3
#define En_pin 2
#define Rw_pin 1
#define Rs_pin 0
#define D4_pin 4
#define D5_pin 5
#define D6_pin 6
#define D7_pin 7
LiquidCrystal_I2C
lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);
const byte numRows= 4;        //Keypad üzerinde ki satırların  sayısı
const byte numCols= 4;        // Keypad üzerinde ki sütünların  sayısı
int code = 1234;              //Şifre buradaki
int tot,i1,i2,i3,i4;
char c1,c2,c3,c4;
//keymap, basılan tuşu satır ve sütunlara göre tıpkı ekranda göründüğü gibi tanımlar.
keypad char keymap[numRows][numCols]=
{
{'1', '2', '3', 'A'},
{'4', '5', '6', 'B'},
{'7', '8', '9', 'C'},
{'*', '0', '#', 'D'}
```

```
};
//Arduino terminallerine keypad bağlantılarını gösteren kod
byte rowPins[numRows] = {9,8,7,6}; //Rows 0 to 3
byte colPins[numCols]= {5,4,3,2}; //Columns 0 to 3
//initializes an instance of the Keypad class
Keypad myKeypad= Keypad(makeKeymap(keymap), rowPins, colPins, numRows,
numCols);
void setup()
        {
         lcd.begin (16,2);
         lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
         lcd.setBacklight(HIGH);
         lcd.home ();
         lcd.print("ROMANIA DoorLock");
         lcd.setCursor(9, 1);
         lcd.print("Standby");
          pinMode(Solenoid,OUTPUT);
         delay(2000);
        }
void loop()
{
    char keypressed = myKeypad.getKey();  //The getKey fucntion keeps the program
runing, as long you didn't press "*" the whole thing bellow wouldn't be triggered
      if (keypressed == '*')          // and you can use the rest of you're code simply
         {
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Enter Code");              //when the "*" key is pressed you can enter
the passcode
              keypressed = myKeypad.waitForKey();  // here all programs are stopped until
you enter the four digits then it gets compared to the code above
              if (keypressed != NO_KEY)
```

```
  {
   c1 = keypressed;
   lcd.setCursor(0, 1);
   lcd.print("*");
   }
 keypressed = myKeypad.waitForKey();
 if (keypressed != NO_KEY)
  {
   c2 = keypressed;
   lcd.setCursor(1, 1);
   lcd.print("*");
   }
  keypressed = myKeypad.waitForKey();
if (keypressed != NO_KEY)
  {
   c3 = keypressed;
   lcd.setCursor(2, 1);
   lcd.print("*");
   }
   keypressed = myKeypad.waitForKey();
if (keypressed != NO_KEY)
  {
   c4 = keypressed;
   lcd.setCursor(3, 1);
   lcd.print("*");
   }
  i1=(c1-48)*1000;        //Basılan tuşlar karakterlerde saklanır, sonra onlar  Int'e
```

dönüştürülür, daha sonra kodun Int'sini xxxx'i yapmak için çarpma yapılır.

```
i2=(c2-48)*100;
  i3=(c3-48)*10;
  i4=c4-48;
  tot=i1+i2+i3+i4;
```

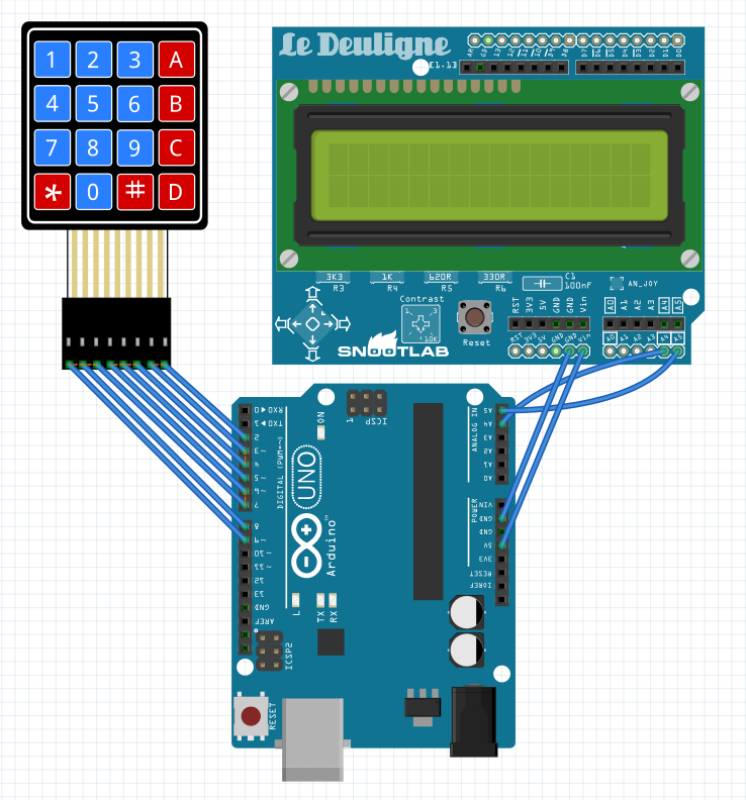
if (tot == code) //Kod doğruysa, burada ne istersen onu tetiklersin, ekrana bir mesaj yazdırman yeterlidir.

```
      {
       lcd.clear();
       lcd.setCursor(0, 0);
       lcd.print("Welcome");
       digitalWrite(Solenoid,HIGH);
  delay(3000);
  digitalWrite(Solenoid,LOW);
       lcd.setCursor(7, 1);
       lcd.print("ROMANIA DoorLock");

       delay(3000);
       lcd.clear();
       lcd.print("ROMANIA DoorLock");
       lcd.setCursor(9, 1);
       lcd.print("Standby");

      }
      else          // Kod yanlışsa, başka bir şey alırsınız.
      {
       lcd.clear();
       lcd.setCursor(0, 0);
       lcd.print("WRONG CODE");
       delay(3000);
       lcd.clear();
       lcd.print("ROMANIA DoorLock");
       lcd.setCursor(9, 1);
       lcd.print("Standby");
       }
         }
}
```

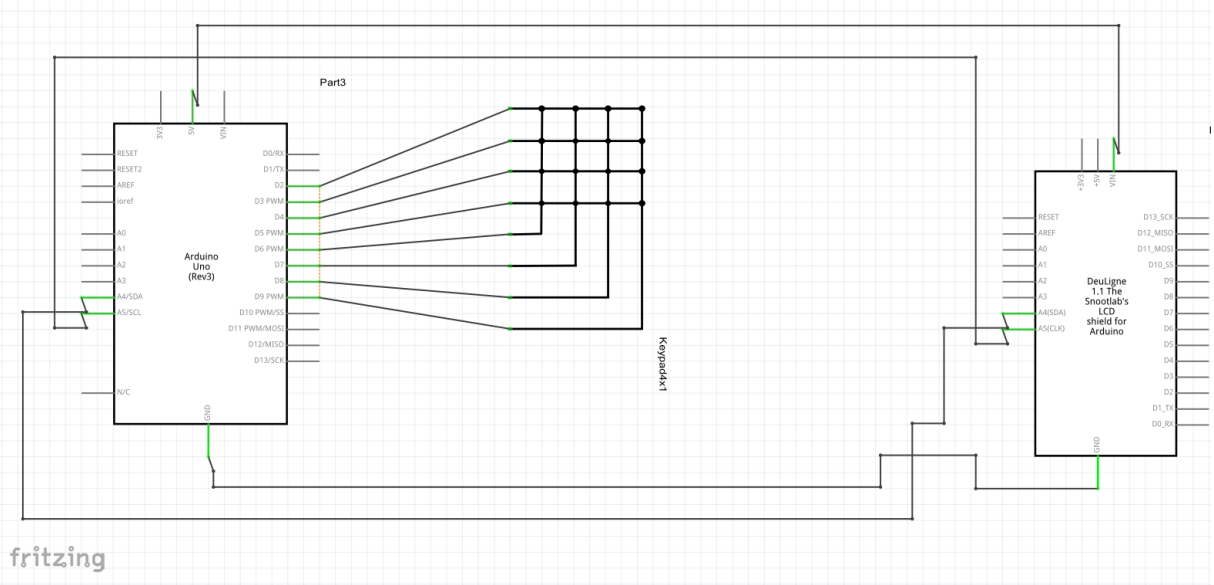
Devre 4:

Devre Adı: Arduino hesap makinası

Devrenin Tanımı: Program temel matematiksel hesaplamaları yapar.



1-) Breadboard görünümü



2-) Şematik görünüm

```
/* Arduino hesap makinası */
#include <Keypad.h>
#include <EEPROM.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#define I2C_ADDR 0x27                 // I2C adresi
#define BACKLIGHT_PIN 3               //  LCD Pinlerini tanımlama
#define En_pin 2
#define Rw_pin 1
#define Rs_pin 0
#define D4_pin 4
#define D5_pin 5
#define D6_pin 6
#define D7_pin 7
const byte ROWS = 4;                  // Dört satır
const byte COLS = 4;                  // Dört sütün
// Define the Keymap
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
```

```
 {'7','8','9','C'},
 {'*','0','#','D'}
};
byte rowPins[ROWS] = {9,8,7,6};        //0-3  satırlar
byte colPins[COLS]= {5,4,3,2};          // 0-3 sütünlar
LiquidCrystal_I2C
lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);
Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
//  Keypadi oluşturma
long Num1,Num2,Number;
 char key,action;
 boolean result = false;
 void setup()
       {
        lcd.begin (16,2);
        lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
          lcd.setBacklight(HIGH);                 //Arka ışığı yakma
       lcd.print("Calculator Ready");          // Ekran giriş mesajı
  lcd.setCursor(0, 1);                          // İmleci 0. kolon, 1. satıra ayarlama
  lcd.print("A=+ B=- C=* D=/");                 //Giriş mesajını gösterme
   delay(6000);                                 // Ekranın bilgileri göstermesini bekle
    lcd.clear();                                //Sonra mesajı sil.
                //        for(i=0 ; i<sizeof(code);i++){       // Kodu ilk kez yüklediğinizde, yoruma
devam edin
//          EEPROM.get(i, code[i]);         //Kodu yükleyin ve EEPROM'da saklamak için
değiştirin
//          }       // Ardından bunu döngü için kaldırın ve kodu yeniden yükleyin (Yalnızca bir kez
yapılır)
       }
void loop() {
  key = kpd.getKey();                   //Basılan tuş değerini bir karakterde saklamak
if (key!=NO_KEY)
```

```
DetectButtons();
if (result==true)
CalculateResult();
DisplayResult();
}
void DetectButtons()
{
    lcd.clear();                        //Sonra sil
   if (key=='*')                        //İptal Butonuna basılırsa,
   {Serial.println ("Button Cancel"); Number=Num1=Num2=0; result=false;}
       if (key == '1')                  //If Buton-1'e basılırsa
   {Serial.println ("Button 1");
   if (Number==0)
   Number=1;
   else
   Number = (Number*10) + 1;            //2 defa bas
   }
       if (key == '4')                  //Şayet  Buton-4'e  basılırsa
   {Serial.println ("Button 4");
   if (Number==0)
   Number=4;
   else
   Number = (Number*10) + 4;            //2 defa bas
   }
       if (key == '7')                  //Şayet Buton-7'e  basılırsa
   {Serial.println ("Button 7");
   if (Number==0)
   Number=7;
   else
   Number = (Number*10) + 7;            //2 defa bas
   }
     if (key == '0')
```

```
{Serial.println ("Button 0");                 //Buton-0'a basılırsa
if (Number==0)
Number=0;
else
Number = (Number*10) + 0;                     //2 defa bas
}
    if (key == '2') //Button 2 is Pressed
{Serial.println ("Button 2");
 if (Number==0)
Number=2;
else
Number = (Number*10) + 2;                     //2 defa bas
}
    if (key == '5')
{Serial.println ("Button 5");
 if (Number==0)
Number=5;
else
Number = (Number*10) + 5;                     //2 defa bas
}
    if (key == '8')
{Serial.println ("Button 8");
 if (Number==0)
Number=8;
else
Number = (Number*10) + 8;                     //2 defa bas
}
  if (key == '#')
{Serial.println ("Button Equal");
Num2=Number;
result = true;
}
```

```
    if (key == '3')
{Serial.println ("Button 3");
 if (Number==0)
Number=3;
else
Number = (Number*10) + 3;                    //2 defa bas
}
    if (key == '6')
{Serial.println ("Button 6");
if (Number==0)
Number=6;
else
Number = (Number*10) + 6;                    //Pressed twice
}
    if (key == '9')
{Serial.println ("Button 9");
if (Number==0)
Number=9;
else
Number = (Number*10) + 9; //Pressed twice
}
  if (key == 'A' || key == 'B' || key == 'C' || key == 'D')     // Sütun 4'teki butonları algılama
{
Num1 = Number;
Number =0;
if (key == 'A')
{Serial.println ("Addition"); action = '+';}
 if (key == 'B')
{Serial.println ("Subtraction"); action = '-'; }
 if (key == 'C')
{Serial.println ("Multiplication"); action = '*';}
 if (key == 'D')
```

```
   {Serial.println ("Division"); action = '/';}
   delay(100);
  }
}
void CalculateResult()
{
  if (action=='+')
    Number = Num1+Num2;
  if (action=='-')
    Number = Num1-Num2;
  if (action=='*')
    Number = Num1*Num2;
  if (action=='/')
    Number = Num1/Num2;
}
void DisplayResult()
{
  lcd.setCursor(0, 0);                    // İmleci 0. sütün, 0. satıra ayarla
  lcd.print(Num1); lcd.print(action); lcd.print(Num2);
    if (result==true)
  {lcd.print(" ="); lcd.print(Number);}     // Sonucu göster
    lcd.setCursor(0, 1);                  // İmleci 0. sütün, 1. satıra ayarla
  lcd.print(Number);                      // Sonucu göster
}
```

Co-funded by the Erasmus+ Programme of the European Union

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA210-VET

Küçük Ölçekli Mesleki Eğitim ve Öğretim Ortaklık Projesi

Proje adı: “Mesleki Eğitimde Arduinoları Kullanma”

Proje kısaltması: “UsingARDinVET”

Proje No: “2023-1-RO01-KA210-VET-000156616”

Dot (Nokta) Matrix Modulü ve Eğitim Kiti

Co-funded by the
Erasmus+ Programme
of the European Union

Dot Matrix Display Modülü

Arduino bağlamında, Nokta Matrisi, satırlar ve sütunlar halinde düzenlenmiş iki boyutlu bir LED ızgarasından oluşan bir LED ekranıdır. Her LED bağımsız olarak açılıp kapatılabilir ve bu da birden fazla LED'in eş zamanlı kontrolü yoluyla karmaşık desenler ve animasyonlar oluşturulmasına olanak tanır.

Bir LED matrisi çok çeşitli uygulamalar için kullanışlıdır. Şekil 1'de gösterilen gibi 8x8'lik bir matris, harfleri veya sayıları görüntülemek için kullanılabilir. Bu modüllerden birkaçını yan yana yerleştirdiyseniz, kayan metin içeren bir ekran oluşturabilirsiniz.

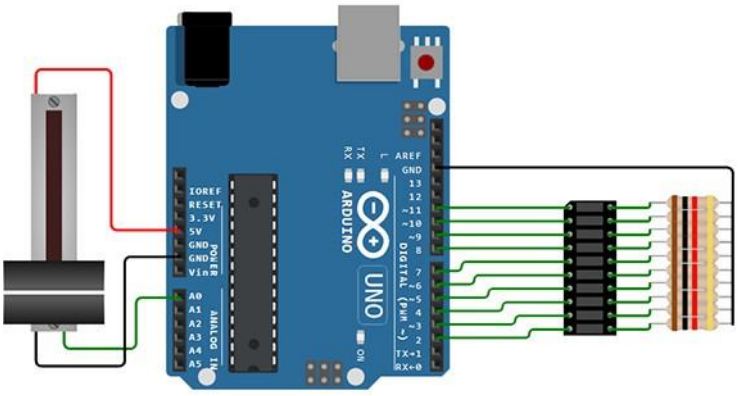
Şekil 1: 8x8'lik bir nokta matris LED .

Modülün, LED'ler ile modülün kenarları arasında çok az boşluk olacak şekilde tasarlandığını unutmayın. Bu tür matris modülleri yan yana monte edildiğinde, bir modüldeki son sütun veya satır ile bir sonraki modüldeki bitişik sütun veya satır arasındaki mesafe, modülün merkezinde bulunan LED'ler arasındaki mesafeye eşittir. Bu, büyük ekranlar oluşturmak için birden fazla modül kullanıldığında tutarlı bir aralık sağlar.

Devre 1: Bir Çubuk Grafik Oluşturma

Örneğin, tek bir sıra halinde düzenlenmiş 10 ayrı LED'den oluşan 10x1 Nokta Matrisi ekranını düşünün. Şekil 2'de gösterildiği gibi LED'leri bağlayabilirsiniz. Aşağıdaki çizim, bir dizi LED'i açar ve sayı, analog giriş portuna bağlı bir potansiyometrenin değerine orantılıdır.

```
const int analogPin = A0;// potansiyometrenin bağlandığı pin

const int ledCount = 10;// çubuk grafikteki LED sayısı


// LED'lerin bağlandığı pin numaralarının dizisi
```



Şekil 2: Çubuk Grafik

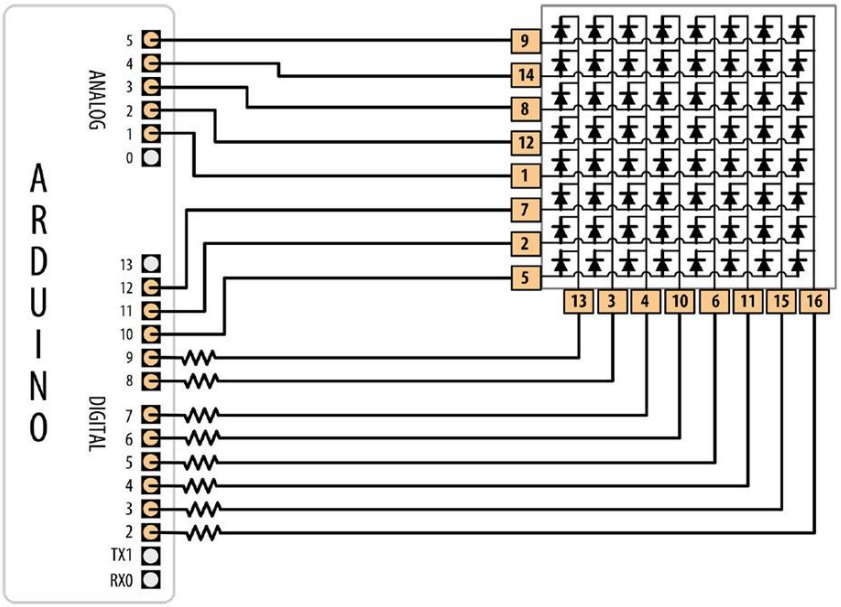
```
int ledPins[] = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11};


void setup() {
  // pin dizisi üzerinde döngü oluştur ve hepsini
  çıkışa ayarla

  for (int thisLed = 0; thisLed < ledCount; this
  Led++)

  {

    pinMode(ledPins[thisLed], OUTPUT);

  }

}
```

```
void loop() {
  // read the potentiometer:
  int sensorReading = analogRead(analogPin);
  // Sonucu 0'dan LED sayısına kadar olan bir aralığa
  eşleyin:
  int ledLevel = map(sensorReading, 0, 1023, 0, led
  Count);

  // LED dizisi üzerinde döngü:
  for (int thisLed = 0; thisLed < ledCount; thisLed++) {
    // eğer dizi elemanının indeksi led eviyesinden küçükse,
    // bu eleman için pini aç:
    if (thisLed < ledLevel) {
      digitalWrite(ledPins[thisLed], HIGH);
    }
    // Led seviyesinden yüksek tüm pinleri kapat
    else {
      digitalWrite(ledPins[thisLed], LOW);
    }
  }
}
```

LED'lere bağlı pinler ledPins dizisinde tutulur. LED sayısını değiştirmek için bu diziden eleman ekleyebilir (veya çıkarabilirsiniz), ancak ledCount değişkeninin eleman sayısıyla aynı olduğundan emin olun (ki bu pin sayısıyla aynı olmalıdır). Arduino map fonksiyonu, sensör değerinin bir oranı olarak yanması gereken LED sayısını hesaplamak için kullanılır. Kod, her bir LED'de döngüye girerek sensörün orantılı değeri LED sayısından büyükse LED'i açar.

İdeal bir dünyada, en düşük ayarındaki bir potansiyometre sıfır döndürür, ancak gerçek dünyada kayması muhtemeldir. Sensör maksimum değerdeyken tüm LED'ler yanar. Potansiyometre maksimum değerindeyken son LED'in yanmasını istiyorsanız, map için ikinci argümanı 1023'ten 1000'e veya benzeri bir değere düşürmeyi deneyin.

Devre 2: Bir LED Matrisinin Kontrolü



Şekil 3: 16 dijital pine bağlı bir LED matrisi.

Bu taslak, anotların sıralara, katotların sütunlara bağlandığı 64 LED'den oluşan bir LED matrisi kullanır (Jameco 2132349'daki gibi). Şekil 3, bağlantıları gösterir (Çift renkli LED ekranların elde edilmesi daha kolay olabilir ve ihtiyacınız olan tek şey buysa renklerden yalnızca birini çalıştırabilirsiniz).

```
const int columnPins[] = {2, 3, 4, 5, 6, 7, 8, 9
};const int rowPins[] = {10, 11, 12, A1, A2, A3,
A4, A5};int pixel = 0;

// Matriste ki 0 ila 63. LEDler


int columnLevel = 0;

// pixel değeri LED sütununa dönüştürüldü

int rowLevel = 0;

// pixel değeri LED satırına dönüştürüldü


void setup() {
```

```
  for (int i = 0; i < 8; i++) {
    pinMode(columnPins[i],
    OUTPUT); pinMode(rowPins[i],
    OUTPUT);
  }
}

void loop() {
  pixel = pixel + 1;
  if (pixel > 63) pixel = 0;
  columnLevel = pixel % 8;
  // sütun sayısına göre haritalamama
  rowLevel = pixel % 8;
  // /kesirli değeri al
  for (int column = 0; column < 8;
    column++) {digitalWrite(columnPins[
    column], LOW);
    for (int row = 0; row < 8;
      row++) { if (columnLevel >
      column) {
        digitalWrite(rowPins[row], HIGH);
      } else if (columnLevel == column && rowLevel >=
        row) {digitalWrite(rowPins[row], HIGH);
      } else {
        // Bu satırdaki bütün ledleri söndür.
        digitalWrite(columnPins[column],
        LOW);
      }
      delayMicroseconds(300);
```

```
      digitalWrite(rowPins[row], LOW);      // turn off LED
    }
    // bu sütunu eksi terminalden ayırın
    digitalWrite(columnPins[column],
    HIGH);
  }
}
```

Direnç değeri, Arduino Uno'da bir pinden geçen maksimum akımın 40 mA'yı geçmemesini sağlayacak şekilde seçilmelidir. Her bir sütun pininden sekiz LED'e kadar akım geçebildiğinden, her bir LED için maksimum akım 40 mA'nın sekizde biri veya 5 mA olmalıdır. Tipik bir küçük kırmızı matristeki her bir LED'in yaklaşık 1,8 voltluk bir ileri voltajı vardır. 1,8 voltluk bir ileri voltajla 5 mA'ya neden olan direncin hesaplanması 680Ω değerini verir. Kullanmak istediğiniz matrisin ileri voltajını bulmak için veri sayfanızı kontrol edin. Matrisin her bir sütunu, seri direnç aracılığıyla bir dijital pine bağlanır. Sütun pini düşük ve bir satır pini yüksek olduğunda, ilgili LED yanar. Sütun pini yüksek veya satır pini düşük olan tüm LED'ler için, LED'den akım geçmez ve yanmaz. For döngüsü her bir satırı ve sütunu tarar ve tüm LED'ler yanana kadar sıralı LED'leri açar. Döngü 1. sütun ve satırla başlar ve o satırdaki tüm LED'ler yanana kadar satır sayacını artırır; sonra bir sonraki sütuna geçer ve böylece döngüden her geçişte başka bir LED'i yakarak tüm LED'ler yanana kadar devam eder.

Bir satırın tamamını aynı anda yakmanız gerekmez. Aşağıdaki program, dizi boyunca ilerlerken bir seferde bir LED'i yakacaktır:

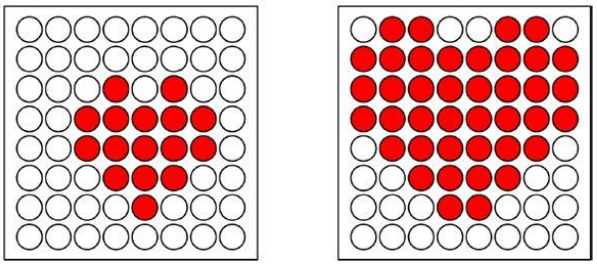
```
const int columnPins[] = {2, 3, 4, 5, 6, 7, 8, 9
};const int rowPins[] = {10, 11, 12, A1, A2, A3,
A4, A5};

int pixel = 0; //Matristeki 0 ila 63 LEDler

void setup() {
  for (int i = 0; i < 8; i++) {
```

```
    pinMode(columnPins[i], OUTPUT);
    //Bütün LED pinlerini Çıkış yap
    Mode(rowPins[i], OUTPUT);
    digitalWrite(columnPins[i], HIGH);
  }
}

void loop() {
  pixel = pixel + 1;
  if (pixel > 63) pixel = 0;
  int column = pixel / 8;                    // sütun sayısına göre haritalama
  int row = pixel % 8;                       // kesirli değeri al
  digitalWrite(columnPins[column], LOW);
  // Bu kolonu GND'ye bağlı
  digitalWrite(rowPins[row], HIGH);  // Bu satırı HIGH yap
  delay(125);                                // Az bir bekle
  digitalWrite(rowPins[row], LOW);       // Bu satırı LOW yap
  digitalWrite(columnPins[column], HIGH); // Kolonu GND'den ayır.
}
```

Devre 3: Bir LED Matrisinde Resim Görüntüleme

Bir veya daha fazla görüntüyü bir LED matrisinde görüntülemek istiyorsunuz, belki de birden fazla görüntüyü hızla değiştirerek bir animasyon yaratabilirsiniz. Bu çözüm, Şekil 3'tekiyle aynı kablolamayı kullanabilir. Çizim, kalp şeklinde düzenlenmiş LED'leri yakarak kalp atışını oluşturur. Her kalp atışı için küçük bir kalp ve ardından daha büyük bir kalp gösterilir (görüntüler Şekil 4'e benzer):

```
byte bigHeart[] = {
    B01100110,
    B11111111,
    B11111111,
    B11111111,
    B01111110,
```



Şekil 4: Her atış sırasında görüntülenen iki kalp görüntüsü.

```
    B00111100,
    B00011000,
    B00000000};

byte smallHeart[] =
    {B00000000,
    B00000000,
    B00010100,
    B00111110,
    B00111110,
    B00011100,
    B00001000,
    B00000000};
```

```
const int columnPins[] = {2, 3, 4, 5, 6, 7, 8, 9
};const int rowPins[] = {10, 11, 12, A1, A2, A3,
A4, A5};

void setup() {
  for (int i = 0; i < 8; i++) {
    pinMode(rowPins[i], OUTPUT);
    // tüm LED pinlerini çıkış
    pinMode(columnPins[i], OUTPUT);
    digitalWrite(columnPins[i], HIGH);

    // Kolon pinlerini GND'den kes/ayır
  }
}

void loop() {
  int pulseDelay = 800;
  // Vuruşlar arasında beklemek için milisaniyeler
  show(smallHeart, 80);
  // küçük kalp görüntüsünü 80 ms boyunca göster
  show(bigHeart, 160);
  // 160 ms boyunca büyük kalp takip etsin
  delay(pulseDelay);
  // Atışlar arasında hibirşey gösterme
}

// image parametresinin işaret ettiği dizide saklanan bir görüntüyü göster.
// Görüntü, milisaniye cinsinden verilen süre boyunca tekrarlanır
```

```
void show(byte * image,
unsigned long duration)
{
  unsigned long start = millis();    //animasyonun zamanlamasında
  while (start + duration > millis())
  // süre geçene kadar döngü
  {
    for (int row = 0; row < 8; row++) {
      digitalWrite(rowPins[row], HIGH);
      // connect row to +5 volts
      for (int column = 0; column < 8; column++) {
        bool pixel = bitRead(image[row],
        column);if (pixel == 1) {
          digitalWrite(columnPins[column], LOW);// Sütünu GND'ye bağla
        }
        delay Microseconds (300); // her LED için küçük bir gecikme
        digitalWrite(columnPins[column], HIGH); // Gnd'den sütunu
      }
      digitalWrite(rowPins[row], LOW);      //  LEDleri ayır.
    }
  }
}
```

The value written to the LED is based on images stored in the bigHeart and smallHeart arrays. Each element in the array represents a pixel (a single LED) and each array row represents a row in the matrix. A row consists of eight bits represented using binary format (as designated by the capital B at the start of each row). A bit with a value of 1 indicates that the corresponding LED

should be on; a 0 means o . The animation is created by rapidly switching between the arrays. The loop function waits a short time (800 ms) between beats and then calls the show function, f i rst with the smallHeart array and then followed by the bigHeart array. The show function steps through each element in all the rows and columns, lighting the LED if the corresponding bit is 1. The bitRead function is used to determine the value of each bit. A short delay of 300 microseconds between each pixel allows the eye enough time to perceive the LED. The timing is chosen to allow each image to repeat quickly enough (50 times per second) so that blinking is not perceptible.

Here is a variation that changes the rate at which the heart beats, based on the value from a sensor. You can test this using a variable resistor connected to analog input pin 0. Use the wiring and code shown earlier, except replace the loop function with this code:

LED'e yazılan değer, bigHeart ve smallHeart dizilerinde depolanan görüntülere dayanır. Dizideki her öğe bir pikseli (tek bir LED) ve her dizi satırı matristeki bir satırı temsil eder. Bir satır, ikili format kullanılarak temsil edilen sekiz bitten oluşur (her satırın başında büyük B ile gösterilir). 1 değerine sahip bir bit, karşılık gelen LED'in açık olması gerektiğini gösterir; 0, o anlamına gelir. Animasyon, diziler arasında hızla geçiş yapılarak oluşturulur. Döngü işlevi, vuruşlar arasında kısa bir süre (800 ms) bekler ve ardından önce smallHeart dizisiyle ve ardından bigHeart dizisiyle show işlevini çağırır. Show işlevi, tüm satır ve sütunlardaki her öğeyi adım adım geçerek karşılık gelen bit 1 ise LED'i yakar. bitRead işlevi, her bitin değerini belirlemek için kullanılır. Her piksel arasındaki 300 mikrosaniyelik kısa bir gecikme, gözün LED'i algılaması için yeterli zamanı sağlar. Zamanlama, her görüntünün yeterince hızlı bir şekilde tekrarlanmasına (saniyede 50 kez) izin verecek şekilde seçilir, böylece göz kırpma algılanamaz.

İşte, bir sensörden gelen değere göre kalbin atış hızını değiştiren bir varyasyon. Bunu, analog giriş pini 0'a bağlı değişken bir direnç kullanarak test edebilirsiniz. Daha önce gösterilen kablolamayı ve kodu kullanın, ancak döngü işlevini şu kodla değiştirin:

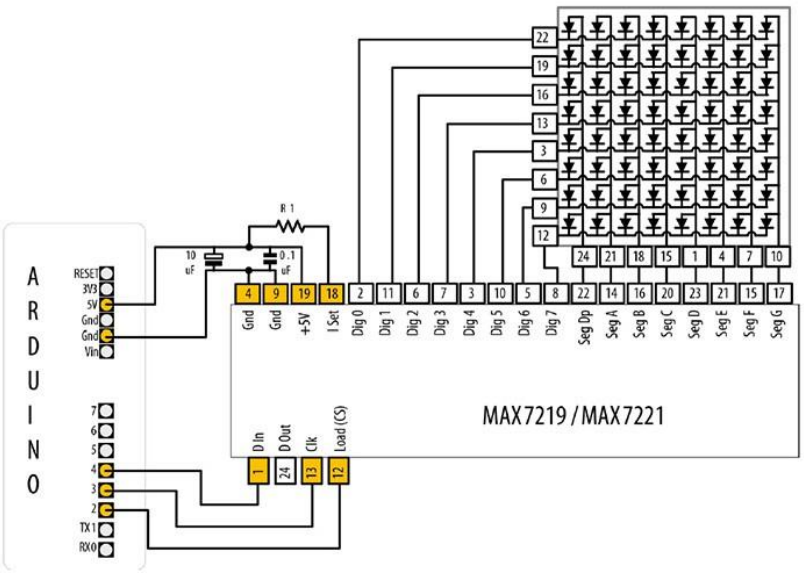
```
void loop() {
  int sensorValue = analogRead(A0); // Analog değeri oku
  int pulseRate =map(sensorValue, 0, 1023, 40, 240);
  // Atışlara dönüştür
  int pulseDelay = (60000 / pulseRate); //Vuruşlar arasında beklenecek milisaniyeler
  show(smallHeart, 80);
```

```
// 100 ms boyunca küçük kalp görüntüsünü göster
show ( big Heart ,  16 0 ) ;
//  // 200 ms boyunca büyük kalp takip etsin
delay ( pulse Delay) ;
//  vuruşlar arasında hiçbir şey gösterme
```

Bu versiyon, sensör değerini dakikadaki atış sayısına dönüştürmek için haritalama fonksiyonunu kullanarak darbeler arasındaki gecikmeyi hesaplar.

Devre 4: MAX72xx'i kullanarak bir LED Dizisini Kontrol Etme

Kontrol etmeniz gereken 8x8'lik bir LED diziniz var ve gereken Arduino pinlerinin sayısını en aza indirmek istiyorsunuz. Bir LED matrisini kontrol etmek için gereken pin sayısını azaltmak için bir kaydırma kaydı kullanabilirsiniz. Bu çözüm, bu yeteneği sağlamak için MAX7219 veya MAX7221 LED sürücü çipini kullanır. Arduino'nuzu, matrisinizi ve MAX72xx'i Şekil 5'te gösterildiği gibi bağlayın.



Şekil 5: 8x8 LED dizisini süren MAX72xx.

Bu program, metin görüntüleyebilen, ekranda nesneler çizebilen ve ekranda çeşitli dönüşümler gerçekleştirebilen MD_MAX72XX kütüphanesine dayanmaktadır. Kütüphaneyi Arduino Kütüphane Yöneticisi'nde (Arduino Library Manager) bulabilirsiniz.

.

```
#include <MD_MAX72xx. h>

// 7219 Kontrol pinleri

#define LOAD_PIN 2

#define CLK_PIN 3

#define DATA_PIN 4

// Donanımı yapılandırın

#define MAX_DEVICES 1

#define HARDWARE_TYPE MD_MAX72XX:
:PAROLA_HW MD_MAX72XX mx =

    MD_MAX72XX(HARDWARE_TYPE, DATA_PIN, CLK_PIN, LOAD_PIN,
    MAX_DEVICES);


void setup () { mx. begin (); }


void loop () {

  mx. clear (); // Displayi temizle

  // Satır ve sütunları çiz

  for (int r = 0; r < 8;
    r++) { for (int c = 0; c <
    8; c++) {

      mx. setPoint(r, c, true); //Her ledi yak
      delay (50);

    }

    // Mevcut parlaklık seviyeleri arasında geçiş yapın

    for (int k = 0; k <= MAX_INTENSITY;
    k++) {
```

```
        mx.control(MD_MAX72XX::
        INTENSITY, k);delay(100);
      }
    }
}
```

Bir matris, donanım türü, veri, yük ve saat pinleri için pin numaraları ve ayrıca maksimum cihaz sayısı (modülleri zincirlemeniz durumunda) geçirilerek oluşturulur. Döngü ekranı temizler, ardından pikselleri açmak için setPoint yöntemini kullanır. Çizim bir satır çizdikten sonra, mevcut parlaklık yoğunlukları arasında geçiş yapar ve bir sonraki satıra geçer.

Burada gösterilen pin numaraları, Adafruit'ten (parça numarası 458) temin edilebilen çift renkli 8x8 matristeki yeşil LED'ler içindir. Bu çizim, iki renkten yalnızca birini kullandığı için tek renkli bir matrisle de çalışacaktır. Matrisinizin metni ters veya beklediğiniz yönde görüntülemediğini fark ederseniz,

#define HARDWARE_TYPE MD_MAX72XX::PAROLA_HW satırındaki donanım türünü PAROLA_HW'den GENERIC_HW, ICSTATION_HW veya FC16_HW'den birine değiştirmeyi deneyebilirsiniz.

Direnç (Şekil 5'te R1 ile işaretlenmiştir) bir LED'i sürmek için kullanılacak maksimum akımı kontrol etmek için kullanılır. MAX72xx veri sayfasında bir değer aralığını gösteren bir tablo vardır. Şekil 5'te gösterilen LED matrisindeki yeşil LED'in ileri voltajı 2 volt ve ileri akımı 20 mA'dir. Direnç değerleri tablosu (MAX72xx veri sayfasından) 28kΩ'u gösterir, ancak biraz güvenlik payı eklemek için 30kΩ veya 33kΩ'luk bir direnç uygun bir seçim olacaktır. LED'ler açılıp kapatıldığında gürültü yükselmelerinin oluşmasını önlemek için kapasitörler (0,1 µF ve 10 µF) gereklidir.

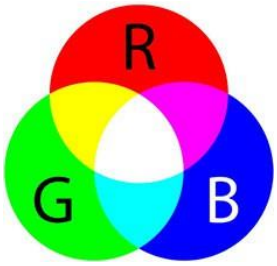
RGB LEDler

Işık Yayan Diyot (LED), içinden akım geçtiğinde aydınlanan küçük bir bileşendir. RGB LED'ler (Şekil 6) aynı prensiple çalışır, ancak dahili olarak neredeyse her renk çıktısını üretebilecek şekilde birleşebilen üç LED (Kırmızı, Yeşil ve Mavi) içerirler.

.

Şekil 6: RGB LEDler.

RGB renk modeli, kırmızı, yeşil ve mavi ışığı karıştırarak renkleri temsil etmenin bir yoludur (Şekil 7). Her renk kanalının yoğunluğu, görüntülenen genel rengi belirler. Bu birincil renkleri farklı seviyelerde birleştirmek, insan gözüyle görülebilen milyonlarca renk üretir. Örneğin, tamamen mavi ışık oluşturmak için, yeşil ve kırmızı LED'leri en düşük seviyeye ayarlarken mavi LED'i en yüksek yoğunluğa ayarlamanız gerekir. Ancak, beyaz ışık için, üç LED'in de en yüksek yoğunluklarına ayarlanması gerekir.



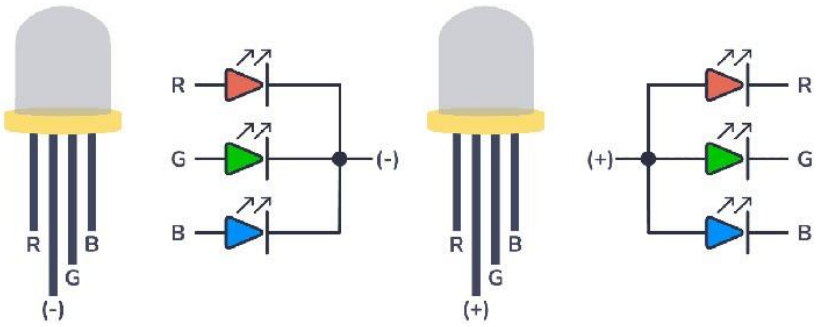
Şekil 7: RGB renk modeli.

RGB LED'ler içlerinde üç LED içerir ve genellikle bu üç LED ortak bir anot veya katot paylaşır. Bu, RGB LED'leri ortak anot veya ortak katot tipi olarak kategorize eder (Şekil 8).

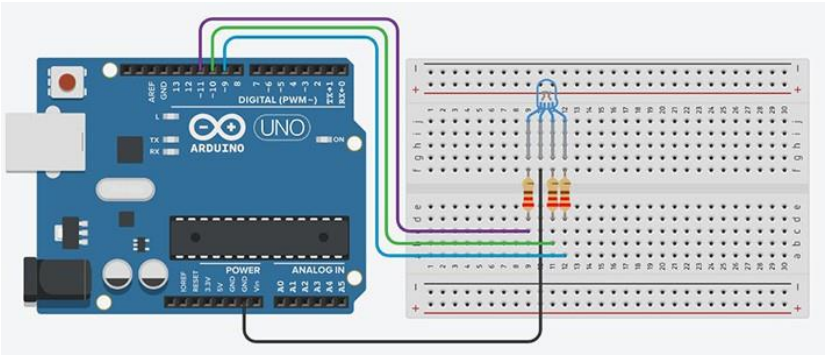
Devre 5. RGB LED kullanımı

RGB LED ile farklı renkler elde etmek için her bir dahili LED'in parlaklığını kontrol etmeniz gerekir. Bu, Arduino ile PWM sinyalleri kullanılarak gerçekleştirilebilir.Şekil-9'da gösterilen devrede katot GND'ye bağlanır ve üç anot, Arduino Kartı üzerindeki üç dijital pine 220 Ohm dirençler aracılığıyla bağlanır. Arduino'nuzda kullandığınız pinlerin PWM sinyalleri çıkışı verebilmesi önemlidir.

Aşağıdaki kod RGB LED'in birkaç renk değiştirmesini sağlayacaktır:



Şekil 8: RGB LED çeşitleri.



Şekil 9: RGB LEDi Arduinoya bağlama.

```
// PWM LED'i bildir/ata
int redLED = 9;

int greenLED =
10;int blueLED
= 11;


void setup() {
```

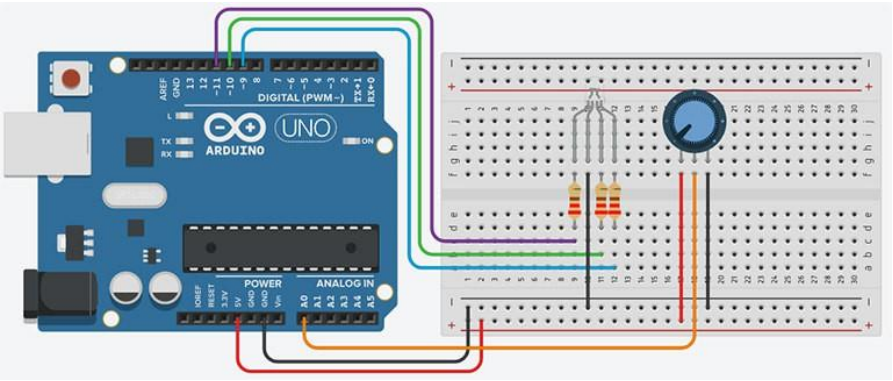
```
  // LED için pinleri Çıkış olarak bildirin
  pinMode(redLED, OUTPUT);

  pinMode(greenLED, OUTPUT);
  pinMode(blueLED, OUTPUT);

}

// Her rengin seviyesini 0 ile 255 arasında ayarlamak için basit bir fonksiyon

void setColor(int redValue, int greenValue, int blueValue)

{

  analogWrite(redLED, redValue);
  analogWrite(greenLED, green
  Value); analogWrite(blueLED,
  blueValue);

}

void loop() {
  // Bir kaç renk değiştir.

  setColor(255, 0, 0);     // Kırmızı renk

  delay(1000);

  setColor(0, 255, 0);     // Yeşil renk

  delay(1000);

  setColor(0, 0, 255);     // Mavi renk

  delay(1000);

  setColor(255, 255, 0);   // Sayı

  delay(1000);

  setColor(0, 255, 255);   // Camgöbeği,

  delay(1000);

  setColor(255, 0, 255);   // Mor
```

```
  delay(1000);
  setColor(255, 255, 255);      // White
  delay(1000);
}
```

Kurulum işlevinde, 9, 10 ve 11 numaralı pinler çıkış olarak yapılandırılır. Döngü işlevi, bir saniyelik aralıklarla farklı renkleri görüntülemek için setColor işlevini tekrar tekrar çağırır. setColor işlevi, 0 ila 255 arasında değişebilen üç parametre (kırmızı, yeşil ve mavi değerleri) alır. Bu değerler, her RGB LED kanal renginin yoğunluğunu kontrol etmek için PWM sinyalleri çıkaran analogWrite işlevinde kullanılır.

Devre 6. Potansiyometre ile RGB LED'i kontrol etme.

Bu örnekte (Şekil 10) potansiyometre düğmesini çevirdiğimizde RGB LED'in rengini değiştireceğiz.



Şekil 10: RGB LED ve potansiyometreli Arduino devresi.

Bunun için kodu yazalım.

```
#define RGB_RED_PIN 11
#define RGB_BLUE_PIN 10
#define RGB_GREEN_PIN 9
#define POTENTIOMETER_PIN A0

void setup()
```

```
{
  pinMode (RGB_RED_PIN,
  OUTPUT) ; pinMode
  (RGB_BLUE_PIN, OUTPUT) ;
  pinMode (RGB_GREEN_PIN,
  OUTPUT) ;
}

void loop ()
{
  int potentiometer Value = analogRead
  (POTENTIOMETER_PIN) ;
  int rgb Value = map( potentiometer Value , 0, 1023, 0,
  15 3 5 ); int red ;
  int blue ;
  int green
  ;

  if (rgb Value < 256)
    { red = 255 ;
    blue = rgb Value ;
    green = 0 ;
  }
  else if (rgb Value < 512
    ) { red = 511 == rgb
    Value ; blue = 255 ;
    green = 0;
  }
  else if (rgb Value < 768)
    { red = 0 ;
    blue = 255 ;
```

```
    green = rgbValue == 512;
  }
  else if (rgbValue < 1024
    ) { red = 0;
    blue = 1023 == rgb
    Value; green = 255;
  }
  else if (rgbValue < 1280
    ) { red = rgbValue ==
    1024; blue = 0;
    green = 255;
  }

  else {
    red = 255;
    blue = 0;
    green = 1535 == rgbValue;
  }

  analogWrite (RGB_RED_PIN, red);
  analogWrite (RGB_BLUE_PIN, blue
  ); analogWrite (RGB_GREEN_PIN,
  green);
}
```

Öncelikle kullanacağımız her pin için bir tanım oluşturuyoruz. Potansiyometre için bir tane ve LED'in her rengi için bir tane (kodu 3 farklı LED'i kontrol ediyormuş gibi yazıyoruz).

Void setup()'ta tüm LED'leri (aslında RGB LED'in 3 ayağını) OUTPUT moduna başlatıyoruz. Potansiyometre için yapılacak bir şey yok çünkü analog pin varsayılan olarak zaten giriş modundadır. Void loop()'ta önce potansiyometrenin değerini analogRead() ile okuyoruz. Bu bize 0 ile 1023 arasında bir değer veriyor. 1536 farklı seçenek arasından seçim yapmak istediğimiz için map() fonksiyonunu kullanarak bu değeri 0-1023 aralığından 0-1535 aralığına

dönüştürüyoruz.

Yani rengi değiştirmek için 6 farklı adımımız var. Ayrıca ilk renk ile son rengin aynı (kırmızı) olduğunu da fark edebilirsiniz. 1. adım için: kırmızıyı 255'e ayarlıyoruz ve hesapladığımız rgbValue'ye (0-1535 aralığında) göre mavi rengini 0-255 arasında artırıyoruz.

rgbValue 255'ten fazlaysa, 2 numaralı adıma geçiyoruz. Şimdi 256'dan 511'e kadar değerlerimiz var. Maviyi 255'e ayarlıyoruz ve ardından kırmızı değerini azaltıyoruz. Bunu yapmak için, rgbValue'yi bu blok için maksimum değer olan 511'den çıkarmamız gerekiyor. Örneğin, if yapısını rgbValue = 400 ile girersek, kırmızı = 511 - 400 = 111 olur.

3 numaralı adım için, maviyi 255'te tutuyoruz ve bu sefer yeşili artırıyoruz. rgbValue artık 512 ile 767 arasındadır. Yani 0'dan başlayıp 255'e ulaşmak için elde ettiğimiz her değere 512 çıkarıyoruz. 4, 5 ve 6 numaralı adımlar önceki adımlarla aynı mantığı izliyor.

Şimdi 0-255 arasında 3 farklı değişkene depolanmış 3 değerimiz var. Hesaplamadan sonra, RGB'nin her bir bacağında, kırmızı, mavi ve yeşil için karşılık gelen değerlerle 3 farklı LED'miş gibi analogWrite() kullanıyoruz.

NeoPixel LED

NeoPixels, elemanları ayrı ayrı kontrol edilebilen akıllı RGB LED şeritleridir. WS2812, WS2811 veya WS6812 sürücüleri kullanırlar ve gömülü LED'in rengini kontrol etmek için tek telli bir protokol kullanırlar. LED'ler kontrolör gövdesiyle entegredir ve çeşitli formatlarda monte edilmiş olarak satılırlar: esnek, matris, halka ve ayrı elemanlar olarak. Her hücrenin 6 pini vardır (Şekil 11):

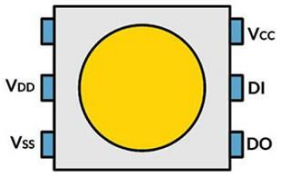
VCC: Kontrol devresi için 5V güç kaynağı;

VDD: LED için 5V güç kaynağı;

VSS: toprak;

DIN: veri girişi;

DOUT: Zincirdeki bir sonraki LED'e bağlanacak veri çıkışı.



Şekil 11: Tek bir WS2812 modülünün pin çıkışı.

NeoPixel ürünlerinde bağlantılar basitleştirilmiştir ve yalnızca üç pin gerekir:

5V: güç kaynağı için;

GND: Arduino ile paylaşılacak GND için;

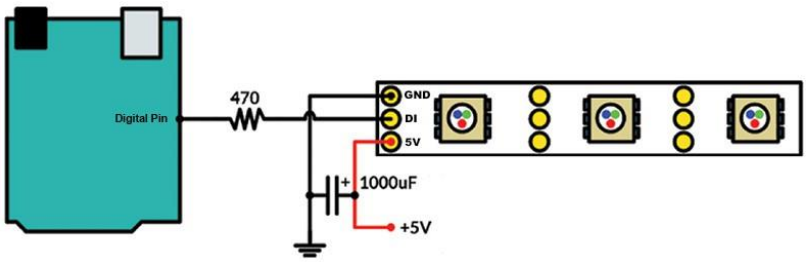
DIN: veri iletimi için.

Birçok LED'i çalıştırmak çok fazla güç gerektirir, bu nedenle 5V'luk ve LED'lerin gerektirdiği tüm akımı sağlayabilen bir güç kaynağı ünitesi veya pil kullanılmalıdır. 5V ile GND arasına, çeşitli LED'leri açmak için gereken akımı sağlamak üzere bin mikrofaradlık

bir elektrolitik kapasitör yerleştirilmesi önerilir. Veri hattı Arduino'ya 470 Ω üzerinden bağlanmalıdır (Şekil 12).

Bir NeoPixel elemanının içerebileceği LED sayısında bir sınır yoktur. Tek sınırlamalar şunlardır:

Şerit tarafından tüketilen güç, eklenen her LED için artar (her LED maksimum 60 mA gerektirir);



Şekil 12: NeoPixel ve Arduino arayüzü.

LED sayısı arttıkça tepki süresi de artar;

LED sayısı arttıkça mikrodenetleyicinin ihtiyaç duyduğu bellek de artar.

.

Kütüphaneler, bireysel LED'lerle iletişimi yönetmek için mevcuttur. Göreceğimiz yönetim kütüphanesi Adafruit tarafından üretilen Adafruit NeoPixel olarak adlandırılır ve Arduino Kütüphane Yöneticisi aracılığıyla kurulabilir. Kütüphaneyi kullanmak için, tanımı programın başına eklenmelidir:

```
#include <Adafruit_NeoPixel.h>
```

Adafruit_NeoPixel nesnesinin başlatılması, kontrol edilecek LED sayısı, iletişim için kullanılan pin ve sürücü modeli gibi bir dizi parametreyi içerir:

```
Adafruit_NeoPixel pixels = Adafruit_NeoPixel
(NUMPIXELS, PIN, NEO_RGB + NEO_KHZ800);
```

Sürücü tipi, çeşitli a aşağıdaki sinyallerle birleştirilmesiyle belirlenir

NEO_KHZ800: 800 kHz iletişim frekansını kullanır;

NEO_RGB: RGB modunda bağlı pikseller.

Pikseller şunlarla başlatılır:

:

```
pixels.begin();
```

Her bir pikselin rengini, RGB değerlerini ayarlamak için indeksini kullanarak kontrol etmek mümkündür:

```
pixels.setPixelColor(num_pixel,  0,150,0);
```

Renkler daha sonra şu şekilde iletilir:

```
pixels.show();
```

Kütüphane fonksiyonları arasında tüm LED'lerin parlaklığını ayarlama fonksiyonu da bulunmaktadır:

```
strip.setBrightness(100);
```

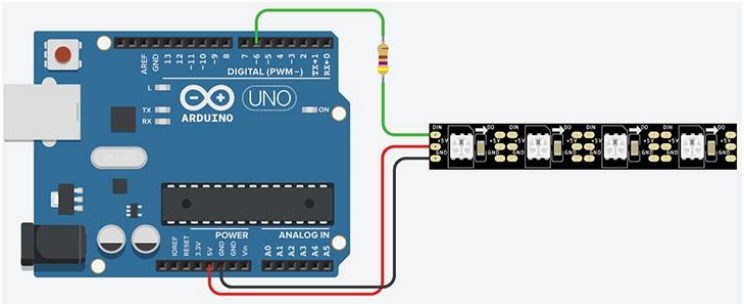
Devre 7. NeoPixel Şeridi Kullanma

Bu örnekte, NeoPixel RGB LED şeridini kontrol etmek için Arduino'yu nasıl kullanacağımızı ve NeoPixel'leri kurmak için Adafruit NeoPixel kütüphanesini nasıl kullanacağımızı öğreneceğiz. Şekil 13, NeoPixel LED şeridini Arduino kartına bağlamanın çok basit bir örneğini göstermektedir. Bir NeoPixel LED şeridini bir Arduino kartına bağlamak için üç kablo bağlayın:

Güç kaynağı (+5V) bir güç kaynağının artısına gider.

Toprak (GND) güç kaynağı toprağına gider ve ayrı bir güç kaynağından güç sağlanıyorsa ayrıca kart toprağına bağlanmalıdır.

Veri girişi (DIN) kartın herhangi bir dijital pinine gider.



Şekil 13: Arduino'ya NeoPixel şeridi bağlama.

Adafruit_NeoPixel kütüphanesi, belirli bir yoğunluk ve renge sahip belirli bir LED'i kolayca açmanıza veya kapatmanıza olanak tanır. Her LED ayrı ayrı kontrol edilebilir. Burada, ilki 0 olarak numaralandırılmış dört LED'imiz var. Örneğin, kırmızı yanmasını

sağlamak için şu komutu kullanırsınız: strip.setPixelColor(0, 255, 0, 0); Ancak, LED yalnızca strip.show() tarafından takip edilirse bu komuta yanıt verecektir.

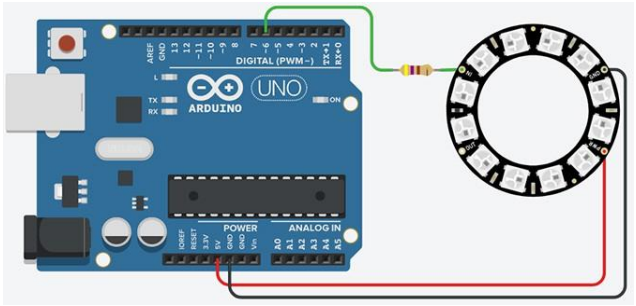
```
#include <Adafruit_NeoPixel.h>

#define LED_PIN    6

#define LED_COUNT 4

Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);

void setup() {
        strip.
        begin();s
        trip.clea
        r();strip
        .show();
}

void loop() {
        strip.setBrightness(50);
        strip.setPixelColor(0, 255
        , 0, 0);strip.show();
        delay(1000);
   strip.setPixelColor(0, 0, 0, 0);
   strip.setPixelColor(1, 0, 255, 0);
       strip.show();
       delay(1000);
   strip.setPixelColor(1, 0, 0, 0);
   strip.setPixelColor(2, 0, 0, 255);
       strip.show();
       delay(1000);
   strip.setPixelColor(2, 0, 0, 0);
   strip.setPixelColor(3, 255, 255, 255);
        strip.
        show();
```

```
        delay (10
        00);
    strip.setPixelColor(3, 0, 0, 0);
}
```

Devre 8: NeoPixel Halkasını Yakmak

NeoPixel halkası, geniş bir renk ve parlaklık kombinasyonları yelpazesine izin veren, ayrı ayrı adreslenebilir RGB LED'lerin dairesel bir düzenlemesidir.

Şekil 14'te gösterilen devrede, güç kaynağı 5V pinine, GND pini devrenin toprağına ve DIN pini Arduino kartındaki bir dijital pine bağlanır.



Şekil 14: Neopixel halkasının Arduino ile arayüzü/bağlantısı.

Aşağıdaki kodda gösterildiği gibi, NeoPixel veri girişinin bağlı olduğu pini ve halkamızda kaç tane LED olduğunu tanımlayın. Bir sonraki adım, ring adını vereceğimiz NeoPixel nesnemizi gerçekten bildirmektir. setup() fonksiyonumuzda, o halkadaki begin() fonksiyonunu çağırırız. Sonra, tüm LED'leri temizlemek için show()'u çağırırız ve son olarak, NeoPixel'imize maksimum parlaklığın ne olacağını söylemek için başlangıçta bir kez çağırdığımız setBrightness() ile parlaklığı ayarlarız.

Sonra, 0'dan halkamızdaki LED sayısına kadar çalışan bir for döngüsüyle başlayalım. Sonra, halkamızdaki her bir LED'in rengini ayarlarız. setPixelColor() fonksiyonu, argümanları sırasıyla alır: numLED, red, green ve blue. For döngüsünün her birinden geçmesi için numLED olarak i gireceğiz. Renk için, red, green ve blue için rastgele değerler kullanırız. Daha sonra ring.show()'u çağırarak halkadaki rengi gerçekten güncelleriz. Son olarak, her rengi uyguladıktan sonra 50 milisaniyelik bir gecikme ekleyerek bir yükleme animasyonu vb. oluşturuyoruz...

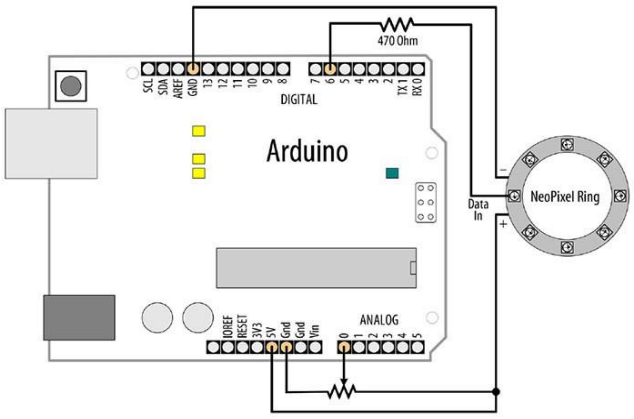
Daha sonra, aynı döngüyü alıp tersine çeviriyoruz. Bunu yapmak için, son LED'den

başlayıp 0'a kadar geriye doğru sayıyoruz. Sonra, piksel rengini 0, 0, 0'a ayarlıyoruz, bu da renk yok anlamına geliyor ve aynı iki satırı ekliyoruz.

```
#include <Adafruit_Neo Pixel .h>

#define LED_PIN   6
#define LED_COUNT 16

Adafruit_Neo Pixel ring(LED_COUNT, LED_PIN, NEO_RGB + NEO_KHZ800);

void setup
  () { rin
  g.begin (
  );
  ring.show();
  ring.setBrightness(50);
}
void loop() {
  for(int i = 0; i < ring.numPixels(); i++){
    ring.setPixelColor(i, random(255), random(255),
    random(255));ring .show();
    delay(50);
  }
  for(int i = ring.numPixels() == 1; i
    >= 0; i==){ ring.setPixelColor(i,
    0, 0, 0);
    ring.show
    ();delay (
    50);
  }
}
```

Devre 9: Controlling a NeoPixel Halka Kontolü

Bu taslak, analog bir pinden alınan okumalara göre LED renklerini değiştirmek için Adafruit Neopixels kütüphanesini (Arduino Kütüphane Yöneticisi kullanılarak yüklenmiştir) kullanır. Şekil 15, bir NeoPixel halkası ve rengi kontrol etmek için bir potansiyometrenin bağlantısını gösterir:



Şeki 15: NeoPixel halkayı bağlama

```
#include <Adafruit_NeoPixel.h>


const int sensorPin = A0; // Sensör için analog pin

const int ledPin = 6;

// LED şeridinin bağlı olduğu pin

const int count = 8;        // şeritte kaç tane LED var

// LED şeridi tanımla

Adafruit_NeoPixel leds = Adafruit_NeoPixel(count, ledPin, NEO_GRB +
NEO_KHZ800);


void setup() {

  leds.begin();      // Başla,
  başlat LED strip for (int i = 0;
  i < count; i++) {

    leds.setPixelColor(i, leds.Color(0, 0, 0));      // Her LED OFF
```

```
  }
  leds.show(); // Şeridi yeni piksel değerleriyle yenile (tümü kapalı
}

void loop() {
  static unsignedint last_reading
  == 1;
  int reading = analogRead(sensor
  Pin);
  if (reading != last_reading) { //  Değer değişirse
    // Analog okumayı Neo Pixel'in renk aralığına eşleyin
     unsigned int mappedSensorReading = map(reading,0,1023,0,65535
    );
    //Kapsamlı bir efekt yaratmak için pikselleri hafif bir gecikmeyle
    güncelleyin
    for (int i = 0; i < count; i++) {
      leds.setPixelColor(
          i, leds.gamma32(leds.ColorHSV(mappedSensorReading,
      255, 128)));leds.show();
      delay(25);
    }
    last_reading = reading;
  }
}
```

Şerit sayısındaki LED sayısını, veri hattının ledPin'e bağlı olduğu Arduino pinini ve kullandığınız LED şerit türünü (bu durumda: NEO_GRB+NEO_KHZ800) belirtirsiniz. Tek bir LED'in rengini ayarlamak için led.setPixelColor yöntemini kullanırsınız. LED sayısını (ilki için 0'dan başlayarak) ve istenen rengi belirtmeniz gerekir. LED'lere veri aktarmak için led.show'u çağırmanız gerekir. Birlikte değişmelerini sağlamak için

led.show'u çağırmadan önce birden fazla LED'in değerini değiştirebilirsiniz. Değiştirilmeyen değerler önceki ayarlarında kalacaktır. Adafruit_NeoPixel nesnesini oluşturduğunuzda, tüm değerler 0 olarak başlatılır.

NeoPixel kütüphanesi, bir tonu RGB değerine dönüştürmek için kendi işlevini içerir: ColorHSV. İlk argüman tondur, ikincisi renk doygunluğudur ve üçüncüsü parlaklıktır. Gamma32 fonksiyonu, bilgisayarların renkleri temsil etme biçimi ile insanların onları algılama biçimi arasındaki farkı telafi etmek için ColorHSV çıktısında bir dönüşüm gerçekleştirir.

Co-funded by the
Erasmus+ Programme
of the European Union

Erasmus+ KA210-VET

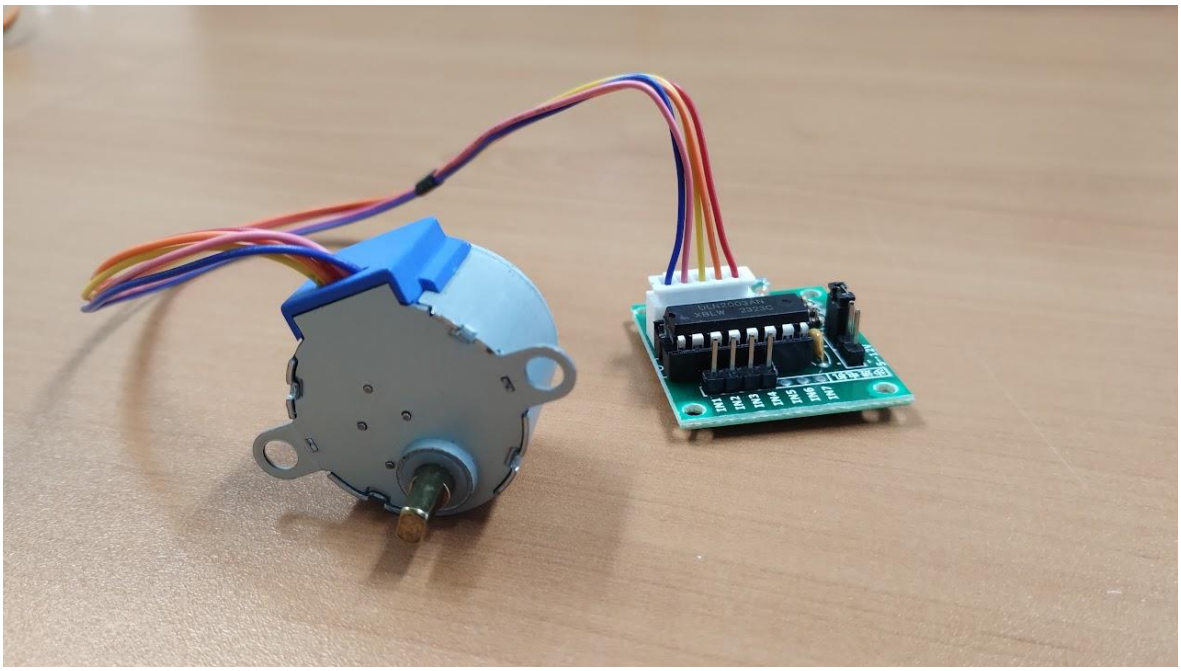
Mesleki eğitimde ve öğretimde küçük ölçekli ortaklıklar

Proje Adı: “Mesleki Eğitimde Arduino Kullanımı”

Proje Kısaltması: “UsingARDinVET”

Proje Numarası: “2023-1-RO01-KA210-VET-000156616”

(Motor modülü ve eğitim seti)



DC MOTORLAR

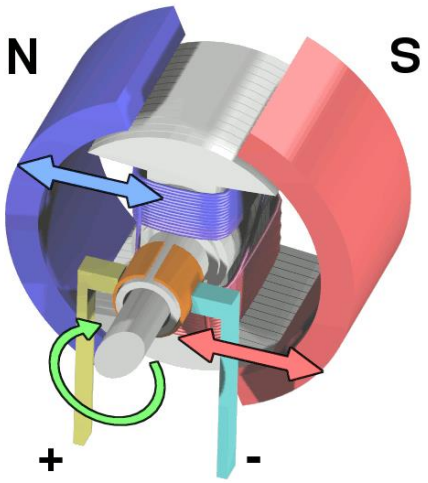
Genel bakış

DC (Doğru Akım) motoru, doğru akım elektriğiyle çalışan bir elektrik motoru türüdür. Basitliği, güvenilirliği ve sürekli dönüş sağlama yeteneği nedeniyle en yaygın kullanılan motor türlerinden biridir. DC motorlar genellikle mekanik hareket gerektiren cihazlarda, oyuncaklarda, aletlerde ve diğer çeşitli aygıtlarda bulunur.

Bu dokümanda DC motorların temel çalışma prensibini, bileşenlerini ve nasıl çalıştıklarını açıklayacağız.

DC Motor Çalışmasının Temel Prensibi

Bir DC motor, akım taşıyan bir iletkenin manyetik alana yerleştirildiğinde hareket etmesine neden olan bir kuvvete maruz kaldığını belirten elektromanyetik indüksiyon ilkesine göre çalışır. Bu kuvvete Lorentz kuvveti denir.



Temel İlke:

Bir tel bobinden geçen akım, bobinin etrafında manyetik alan oluşturur.

Bobin, harici bir manyetik alana (genellikle kalıcı mıknatıslar veya elektromıknatıslar tarafından oluşturulur) yerleştirilir.

Bobinin manyetik alanı ile dış manyetik alan arasındaki etkileşim bir kuvvet oluşturarak bobinin dönmesine neden olur.

Bu dönme hareketi motorun şaftına aktarılarak mekanik hareket sağlanır.

DC Motor Çalışmasını Etkileyen Faktörler

Bir DC motorun performansını etkileyebilecek birkaç faktör vardır:

- **Voltaj:** Bir DC motorun hızı ve torku, doğrudan kendisine uygulanan voltajla ilgilidir. Daha yüksek voltaj genellikle daha yüksek hız ve torkla sonuçlanır.
- **Hız:** Uygulanan voltajın artırılması motorun hızını artırır çünkü rotor sargılarından geçen akımı artırır, daha güçlü bir manyetik alan ve daha büyük bir kuvvet üretir.
- **Tork:** Tork veya dönme kuvveti akımla orantılıdır. Daha yüksek akım daha yüksek tork anlamına gelir.
- **Akım:** Akım, motorun ne kadar tork üretebileceğini belirler. Daha yüksek akım torku artırır ancak motor düzgün bir şekilde soğutulmazsa aşırı ısınmaya da yol açabilir.
- **Direnç:** Rotor sargılarının direnci akım akışını etkiler. Daha yüksek direnç, akabilen akım miktarını sınırlar ve bu da motorun torkunu azaltır. Öte yandan, daha düşük direnç daha fazla akımın akmasına izin vererek torku ve hızı artırır.

- **Manyetik Alan Gücü:** Statorun manyetik alanının gücü de motorun performansını etkiler. Daha güçlü manyetik alanlar stator ve rotor arasında daha fazla etkileşime neden olur, daha fazla kuvvet ve dolayısıyla daha yüksek performans üretir.

Uygulamalar

Elektrikli DC motorlar, düzgün ve sürekli bir dönüş gerektiğinde kullanılır. Düz DC motorlar voltaj kontrollüdür, böylece motorun ne kadar hız ve tork üreteceğine karar verebiliriz. Düz DC motorlar, motorun hızı veya konumu üzerinde hassas kontrol gerektiğinde kullanılamaz. Bu durumlarda, genellikle bunun yerine adım motorları veya servo motorları seçeriz.

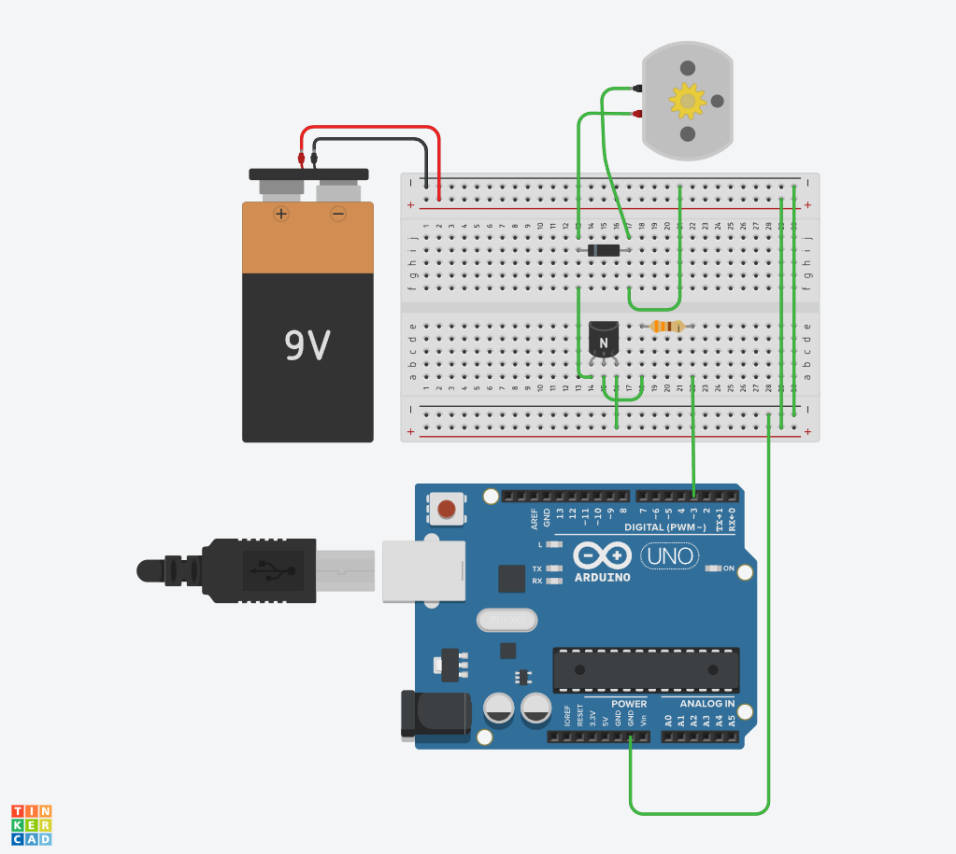
Güç gereksinimleri

Mevcut DC motorlarının çoğunun tüketimi, Arduino'nun sağlayabileceği maksimum akımdan daha yüksektir ve Arduino dijital çıkışlarına zarar verebilir, bu nedenle motora harici güç kaynağına ihtiyacımız vardır ve bu güç kaynağını Arduino'nun çıkış pinlerini kullanarak kontrol etmeliyiz. Arduino'da DC motorları kontrol etmenin en yaygın yolu bir motor kontrolörü kullanmak veya bir transistör kullanarak kendi kontrolörümüzü oluşturmaktır.

Devre 1:

Devre Adı: Transistörle DC motor sürmek

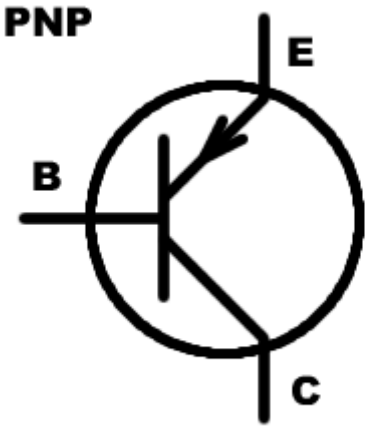
Devre Açıklaması: Bu devre, Arduino kullanarak 9V güç kaynağını kontrol etmek için bir PNP 2N2222 transistörü kullanır.



Devre 1

Bunu yapmak için şunlara ihtiyacınız olacak:

- 1 DC motor
- 1 PNP 2N2222 transistör
- 1 330Ω direnç
- 1 Zener diyot
- 1 adet 9V pil



PNP Transistörler elektronik olarak kontrol edilen bir anahtar olarak kullanılabilir. Akım, emitör pininden (E) toplayıcı pinine (C) yalnızca taban pininden (B) toplayıcıya (C) yeterli akım aktığında akacaktır. Başka bir deyişle, B pinini kullanarak E ve C pinleri arasındaki akımı kontrol edebiliriz.

Devre çok basit. Transistör, yalnızca pin 3 aktif olduğunda aküden motora 9V akım geçmesine izin verecektir. Pin 3 bir PWM pini olduğundan, analogWrite yöntemini kullanarak motorun hızını kontrol etmek için farklı darbe genişlikleri gönderebiliriz.

Diyot, motorun atalet hareketi nedeniyle devreye akım göndermesini önlemek için kullanılır. Direnç, arduino pininin çıkışını transistörün doğru giriş değerlerine uyarlamak için kullanılır.

Motoru 3 numaralı pini kullanarak kontrol etmek için kod şudur:

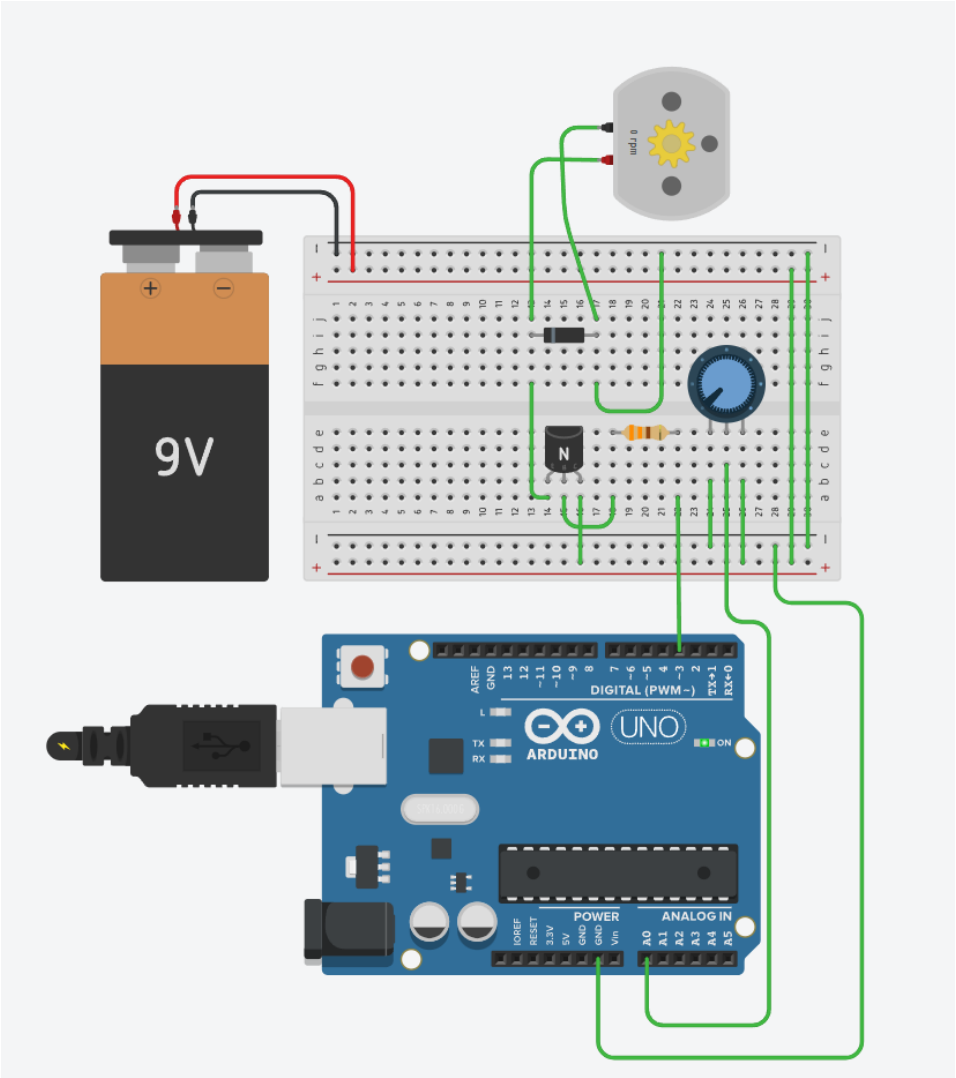
```
int motorPin = 3;                    //Motora bağlı pin

void setup()
{
pinMode(motorPin, OUTPUT);           // 3 numaralı pini çıkış olarak tanımlar
}

void loop()
{
analogWrite(motorPin, 255);          //Tam hız
delay (2000);
analogWrite(motorPin, 100);          //Orta Hız
delay(2000);
analogWrite(motorPin, 10);           //Düşük hız
delay(2000);
}
```

Devre 2:

Devre Adı: Potansiyometre kullanarak DC motor hızının kontrol edilmesi

Devre Açıklaması: Bu devre, Arduino kullanarak 9V güç kaynağını kontrol etmek için bir PNP 2N2222 transistörü kullanır. Ayrıca, arduino analog pin A0 kullanılarak bir potansiyometre okunur ve motorun hızını kontrol etmek için kullanılır.

Motorun hızını kontrol etmek için devremizi bir potansiyometre ekleyecek şekilde değiştirelim.



Devre 2

Örnek program 2: Potansiyometre ile motorun kontrolü

```
int motorPin = 3;                    //Motora bağlı pin
int controlPin = 0;                  // Pot'a bağlı pin
int speed = 0;                       //Motora atanan hız

void setup()
{
pinMode(motorPin, OUTPUT);           //Motor pinini çıkış pini olarak ayarla
}
```

```
void loop()
{
    speed= analogRead(controlPin);          //Potansiyometre değerini oku ve //hız değişkenine
                                            kaydet
analogWrite(motorPin, hız);             //Motora hız gönder
delay(500);                             //500 milisaniye bekle
}
```

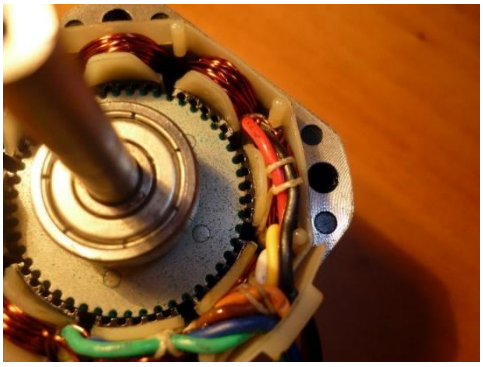
ADIM (STEP) MOTORLARI

Genel bakış

Adım motoru, ayrı adımlarla hareket eden bir tür elektrik motorudur ve bu da onu konum, hız ve yönün hassas kontrolü için ideal hale getirir. Sürekli dönen geleneksel DC motorların aksine, adım motorları tam bir dönüşü, her biri motor tasarımına bağlı olarak genellikle adım başına 0,9° ila 1,8° arasında değişen birden fazla adıma böler. Bu benzersiz özellik, kodlayıcılara veya diğer konum algılama cihazlarına ihtiyaç duymadan çok doğru ve tekrarlanabilir hareketler elde etmelerini sağlar.

Adım Motorları Nasıl Çalışır

Bir adım motorunun çalışması elektromanyetizmaya dayanır. Bir adım motorunun içinde, merkezi bir rotor etrafında düzenlenmiş birden fazla bobin veya sargı bulunur. Belirli bir bobinden akım geçirildiğinde, rotorla etkileşime giren ve rotorun belirli bir açıyla dönmesine neden olan bir manyetik alan oluşturur. Farklı bobinlere kontrollü bir düzende sırayla enerji verilerek, rotor hassas artışlarla (adımlar) hareket ettirilir.



Bir adım motorunun hareketini kontrol etmenin anahtarı, bobinlere gönderilen akım darbelerinin sırasını ve zamanlamasını düzenleyen sürücü devresinde yatar. Sürücü, aşağıdaki gibi farklı modlarda çalışabilir:

- Tam adım: Her seferinde bir bobini enerjilendirerek daha büyük bir adım boyutu sağlar (örneğin, adım başına 1,8°).
- Yarım adım: İki bobini dönüşümlü olarak enerjilendirerek daha küçük adımlara ve daha yumuşak harekete olanak tanır.
- Mikroadımlama: Her tam adımı daha küçük adımlara bölerek pürüzsüzlüğü ve çözünürlüğü artırır ve titreşimi azaltır.

Adım Motorlarının Avantajları

- **Hassasiyet ve doğruluk:** Adım motorları son derece hassastır ve bu da onları 3D yazıcılar, CNC makineleri ve robotik kollar gibi tam konumlandırma gerektiren uygulamalar için ideal hale getirir.
- **Geri bildirim gerekmez:** Adım motorları, rotorun konumu hareket ettirilen adım sayısına göre belirlendiğinden, harici geri bildirim sistemleri (örneğin kodlayıcılar) olmadan da çalışabilir.
- **Güvenilirlik:** Bu motorlar yapı olarak basittir ve bu sayede arızalanabilecek parça sayısı daha azdır.
- **Düşük maliyet:** Adım motorları nispeten ucuzdur ve bu da onları birçok uygulama için uygun maliyetli bir çözüm haline getirir.

Adım Motorlarının Dezavantajları

- **Düşük verim:** Adım motorları, özellikle yüksek hızlarda, diğer motor tiplerine kıyasla daha az enerji verimli olabilir.
- **Yüksek hızlarda tork düşüşü:** Yüksek hızlarda, adım motorları tork kaybına uğrama eğilimindedir ve bu durum bazı uygulamalarda performansı etkileyebilir.
- **Titreşim ve gürültü:** Adım motorları, özellikle düşük hızlarda veya düşük akımla çalışırken titreşim ve gürültü üretebilir.

Adım Motorları için Uygulamalar

Adım motorları, konum ve dönüşün hassas bir şekilde kontrol edilmesini gerektiren uygulamalarda yaygın olarak kullanılır, bunlar arasında şunlar yer alır:

- **3D yazıcılar:** Baskı kafasını doğru bir şekilde hareket ettirip malzemeyi katman katman oluşturmak.
- **Robotik** : Robotik kolların ve diğer hareketli parçaların hassas kontrolü için.
- **CNC makineleri** : Takımı veya iş parçasını kontrollü artışlarla hareket ettirmek için.
- **Kamera platformları** : Fotoğrafçılık ve videografide hassas pan-tilt kontrolü için.
- **Tıbbi ekipman** : Pompalar, protezler ve teşhis makineleri gibi uygulamalar için.

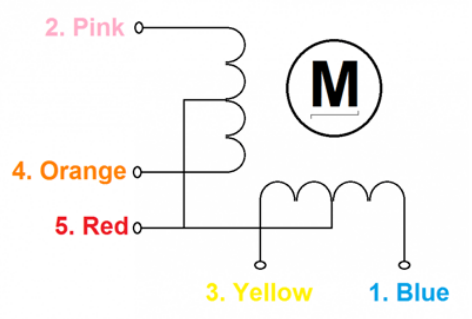
Adım motorları, hassasiyet ve kontrolün kritik olduğu birçok uygulamada olmazsa olmaz bir bileşendir. Tam dönüşü küçük, ayrı adımlara bölme yetenekleri, doğru konumlandırmaya olanak tanır ve basitlikleri, onları hem hobiciler hem de endüstri mühendisleri için popüler bir seçim haline getirir. Yüksek hızlı, yüksek verimli uygulamalar için en iyi seçenek olmasalar da, kontrol ve güvenilirlikteki avantajları onları robotik, üretim ve otomasyon gibi alanlarda vazgeçilmez kılar. Genellikle, adım motorları tasarımları, hareket eden nesnelerin yolun başlangıcına veya sonuna ulaştığını algılamak için sensörler kullanır, bu sensörlere limit anahtar sensörleri denir ve bir adım motorunun başlangıç konumunu belirlemek için olmazsa olmazdır.

Devre 3:

Devre Adı: Bir adım motorunu bir devir ileri ve bir devir geri hareket ettirme

Devre Açıklaması: Bu devre, step motora güç sağlamak ve Arduino kartına zarar vermemek için 28BYJ-48 step motor ve ULD2003 tabanlı bir kontrolcü kullanır.

Bu örnekte 28BYJ-48 adlı küçük bir adım motoru kullanacağız



Şekil 2: 28BYJ-48 Bağlantıları

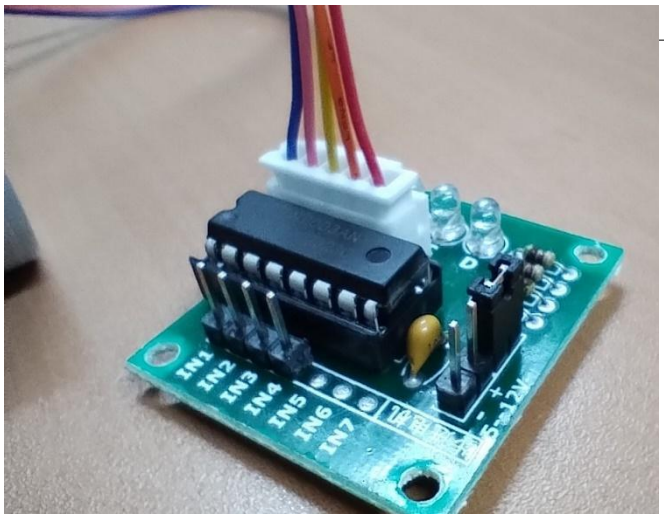
Bu motor, unipolar şema kablolamasına sahip dört bobinden oluşuyor ve genellikle Arduino'muzun I/O pinlerine zarar vermemek için her bobine akım sağlayan ULN2003 tabanlı bir devre tarafından kontrol ediliyor.

28BYJ-48 motor, yarım adım modunda kullanıldığında devir başına 64 adıma ve dahili 1:64 dişli oranına sahip olduğundan devir başına toplam 64 * 64 = 4096 adıma sahiptir.

Bu projede şunları kullanacağız:

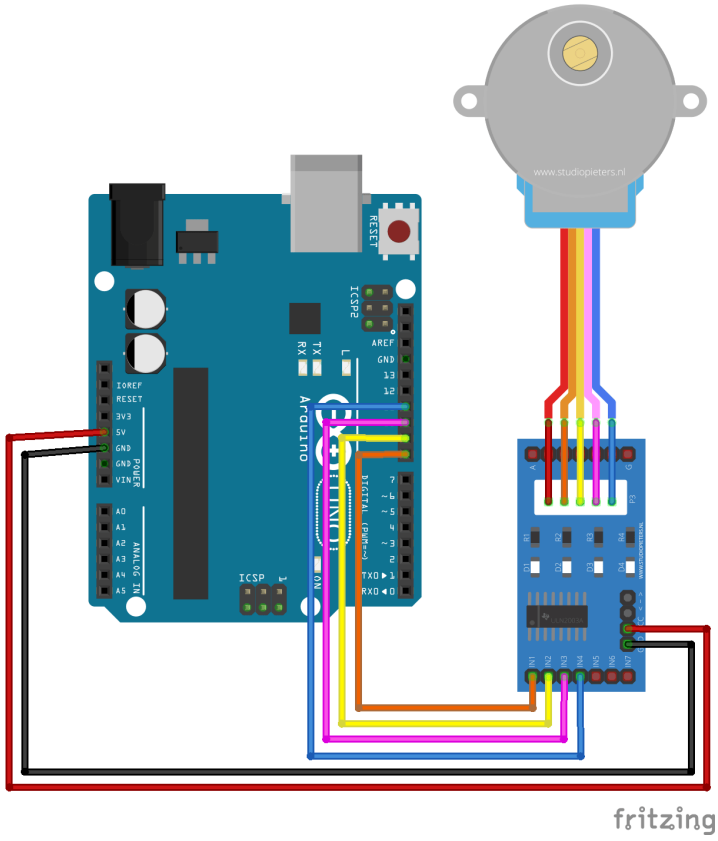
- 1 28BYJ-48 Adım motoru
- 1 ULN2003 tabanlı kontrolör
- Kablolar
- Arduino Uno kartı

Adım motorunu uygun fişi kullanarak kontrol cihazına bağlayın:



Şekil 3: Kontrol ünitesine bağlı adım motoru

Şimdi kontrolcümüzü Arduino kartına şu şemayı takip ederek bağlayalım:



Devre 3

Adım motorunu kontrol etmek için Stepper.h kütüphanesini kullanacağız. Stepper kütüphanesinin en kullanışlı yöntemleri şunlardır:

stepper(adımlar, pin1, pin2, pin3, pin4)	Aşağıdaki parametrelerle yeni bir adımlayıcı oluşturun: Adımlar: Devir başına adım sayısı Pin1 ve Pin 2: Motora bağlı pinler Pin 3 ve Pin 4: (Opsiyonel): Motorda kablo varsa motora bağlanan pinler
step(steps)	Adımlayıcıyı belirli sayıda adım hareket ettirin
setSpeed(rpm)	Motor hızını dakikadaki belirli bir devire ayarlayın. Lütfen motorunuzun maksimum hızını göz önünde bulundurun.

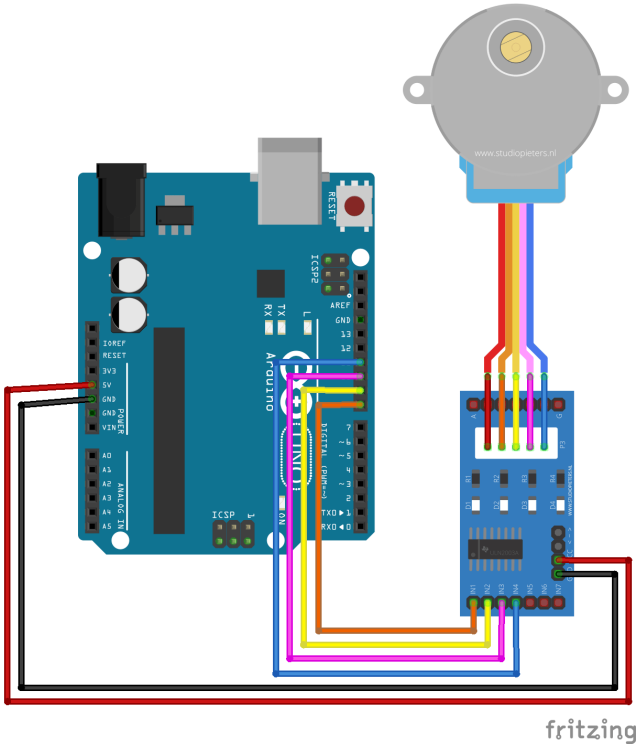
Motoru bir devir ileri ve bir devir geri hareket ettiren kod şudur:

```
#include <Stepper.h>                    // Steper kontrol kitaplığını dahil et

Stepper stepper(4096, 8, 10, 9, 11);            // Devir başına 4096 adımlı bir stepper oluştur


void setup() {

}

void loop {
stepper.step(4096);                     //Bir tam devrim
delay(5000);                            //5 saniye bekle
stepper.step(-4096);                    //Bir tam tur geriye doğru (negatif)
delay(5000);                            //5 saniye bekle
}
```

Devre 4:

Devre Adı: Bir adım motorunu saatin saniye kolu gibi hareket ettirmek

Devre Açıklaması: Aynı adım motorunu ve aynı kontrol cihazını kullanarak, kodu bir saatin saniye kolu gibi motoru hareket ettirecek şekilde değiştireceğiz: her saniye 1/60 devir.



Devre 4

Adım motorunu her saniye 1/60 tur hareket ettiren kod şudur:

```
#include <Stepper.h>                    // Steper kontrol kitaplığını dahil et

Stepper stepper(4096, 8, 10, 9, 11);            //Devir başına 4096 adımlı bir stepper oluştur

void setup() {
stepper.setSpeed(240)                   //Hızlı saniye kolu hareketi için 240 rpm hız
}
void loop {
stepper.step(68);                               //60, 4096'ya bölünür ve yaklaşık 68 adıma eşittir
                                                 //saniye başına
delay(1000);                            //her saniye motoru hareket ettiriyoruz
}
```

SERVO MOTORLAR

Genel bakış

Servo motorlar robotik, otomasyon ve açısal konumun hassas bir şekilde kontrol edilmesi gereken diğer alanlarda yaygın olarak kullanılır. Sürekli olarak dönmek yerine belirli bir konuma dönebilmeleri bakımından normal motorlardan farklıdırlar. Bu onları robotik kolları hareket ettirmek, bir kameranın konumunu kontrol etmek veya güneş panellerinin açısını ayarlamak gibi görevler için ideal hale getirir.

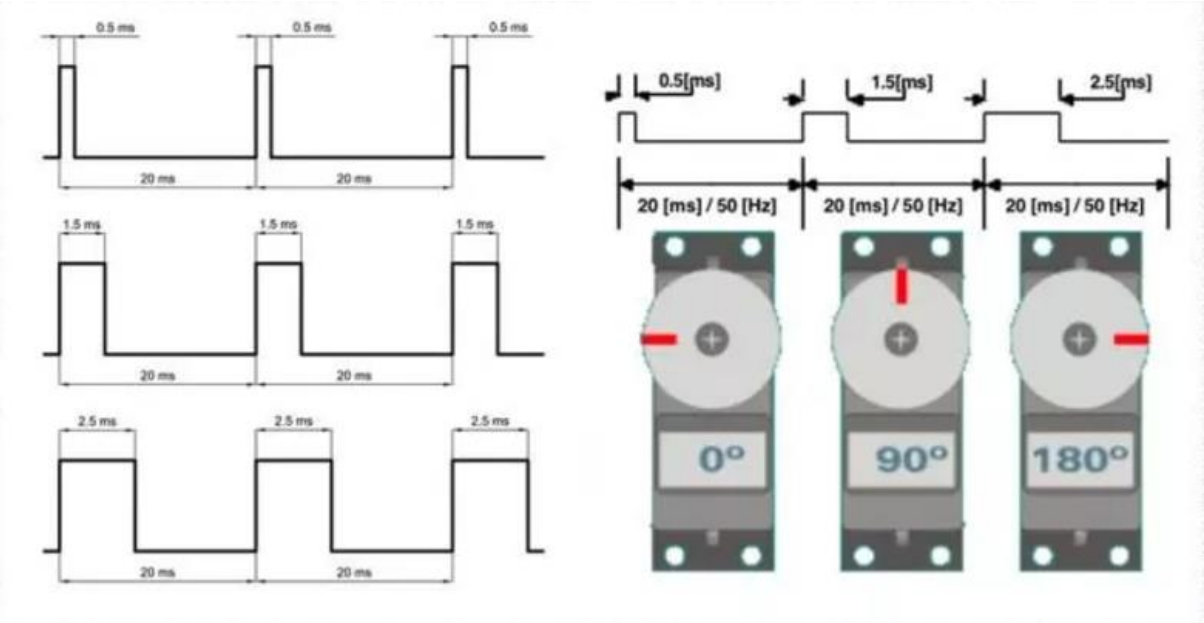
Servo motorlar nasıl çalışır?

Servo motor, hassas açısal hareket elde etmesini sağlayan bir geri bildirim sistemiyle donatılmış küçük bir motordur. Her iki yönde de sürekli dönen standart DC motorların aksine, bir servo motor yalnızca sınırlı bir aralıkta (genellikle 0° ila 180°) dönebilir ve konumu çok hassas bir şekilde kontrol edilebilir.

Servo motor üç temel bileşenden oluşur:

- Motor: Dönüşü sağlar.
- Geri besleme potansiyometresi: Motorun pozisyonunu izler.
- Kontrol devresi: Kontrol sinyalini alır ve motorun pozisyonunu buna göre ayarlar.

Servo motorlar Darbe Genişlik Modülasyonu (PWM) sinyallerini kullanarak çalışır. Arduino, servoya bir PWM sinyali gönderir ve bu da motorun konumunu sinyalin süresine göre ayarlar.



Şekil 4: PWD'nin servo motor konumunu nasıl kontrol ettiği

Kısa bir darbe (1 ms) servoyu 0°'ye ayarlayabilir.

Uzun bir darbe (2 ms) servoyu 180°'ye ayarlayabilir.

Genellikle 1,5 ms civarında bir darbe servoyu 90° pozisyonuna getirir.

Darbe genişliği değiştirilerek servo motor, çalışma aralığı içerisinde istenilen yere konumlandırılabilir.

Servo Motorların Uygulamaları

- **Robotik:** Servo motorlar robotlarda eklemleri, tekerlekleri veya aktüatörleri kontrol etmek için yaygın olarak kullanılır.
- **Kamera Gimbal'ları:** Bir kamerayı sabitlemek için, servo motorlar kameranın yönünü kontrol ederek sabit kalmasını sağlayabilir.
- **Anten Konumlandırma:** Servo motorlar, radyo ve uydu çanaklarında anten yönünü ayarlamak için kullanılır.
- **RC Araçlar:** Uzaktan kumandalı arabalar, uçaklar ve tekneler, diğer mekanik bileşenleri yönlendirmek ve kontrol etmek için genellikle servolar kullanırlar.

Servo motorlar, açıların hassas bir şekilde kontrol edilmesini gerektiren projeler için olmazsa olmaz bileşenlerdir. Arduino ile, Servo kütüphanesi sayesinde bir servoyu kontrol etmek basit bir iştir ve bu kütüphane PWM sinyalleri üretmenin karmaşıklığını ortadan kaldırır. Servoların step motorlara göre en büyük avantajlarından biri, servonun kendi konumunu algılayabilmesi nedeniyle genellikle step motor veya DC motorlar gibi sınır sensörlerine ihtiyaç duymamasıdır.

Servo motorların avantajları:

- Limit sensörlerine ihtiyaç duymadan yüksek hassasiyet.
- Dışarıdan gelen rahatsızlıklara rağmen pozisyonlarını koruyabilirler.

Servo motorların dezavantajları:

- Sürekli hareket için uygun olmayan sınırlı hareket aralığı
- Sınırlı maksimum hız

Örnek Proje: Servo Motoru Hareket Ettirme:

SG90, Arduino pinleri kullanılarak çalıştırılabilen küçük ve kullanışlı bir servo motordur. Daha büyük servoların harici bir güç kaynağına sahip olması gerekir.

Bu proje için şunları kullanacağız:

- 1 Servo motor SG90
- Kablolar
- Arduino Uno kartı

Servo kontrol pini (sarı) bir PWM dijital pinine bağlanmalıdır, biz ~3 pinini kullanacağız.

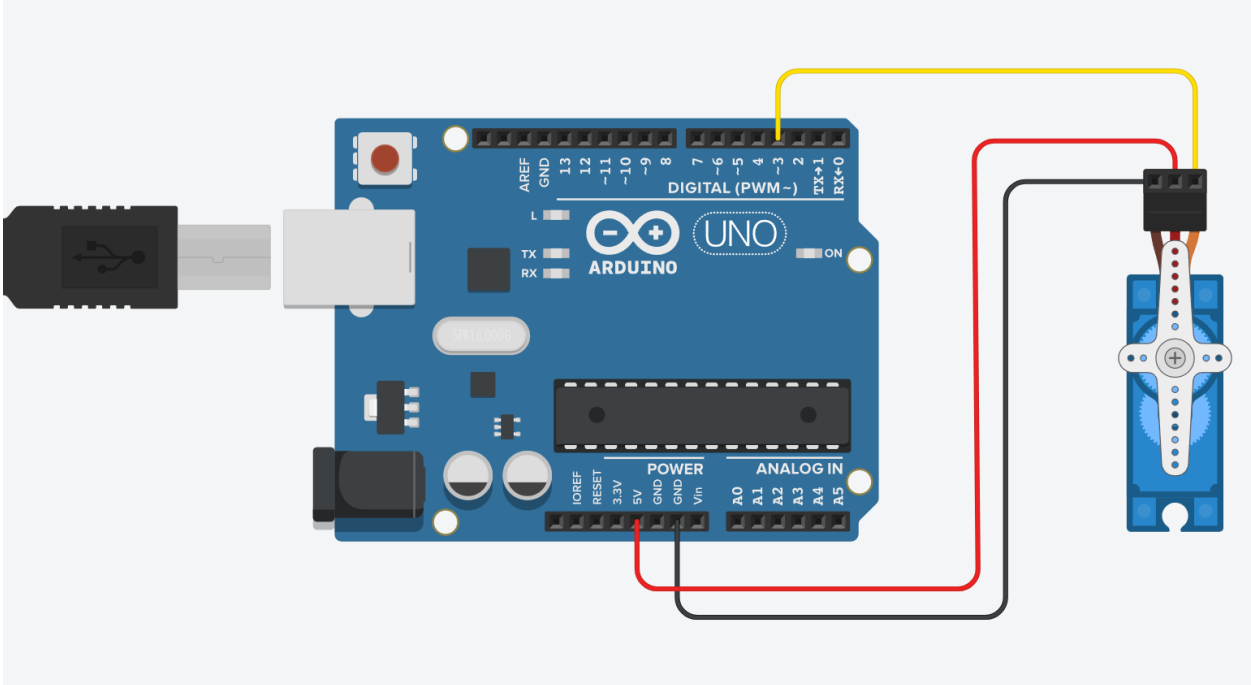
Servo'yu yönetmemize yardımcı olması için Servo.h kütüphanesini kullanacağız. En önemli Servo.h metotları şunlardır:

attach(pin, min, max)	Servo değişkenini bir pine bağlayın. Pin bir PWM dijital pini olmalıdır. İsteğe bağlı min ve max parametreleri, servonun açıyı minimum ve maksimum açıya ayarlamasını beklediği darbe genişliğini mikrosaniye cinsinden ayarlar. Min veya maks değerleri sağlanmazsa varsayılan değerler olan 544 ve 2400 milisaniye kullanılır.
Write (açı)	Servo açısını istediğiniz değere ayarlayın. Servo bu pozisyona hareket edecektir.

Devre 5:

Devre Adı: Servo motorun basit hareketi

Devre açıklaması: Arduino'nun dijital pin 3'ünü kullanarak servo motoru 90, 180 ve 0 derecelik açılarda konumlandıracağız. Servo motor güç için 5V ve GND'ye ve konumu kontrol etmek için dijital pin 3'e PWM modunda bağlanır.



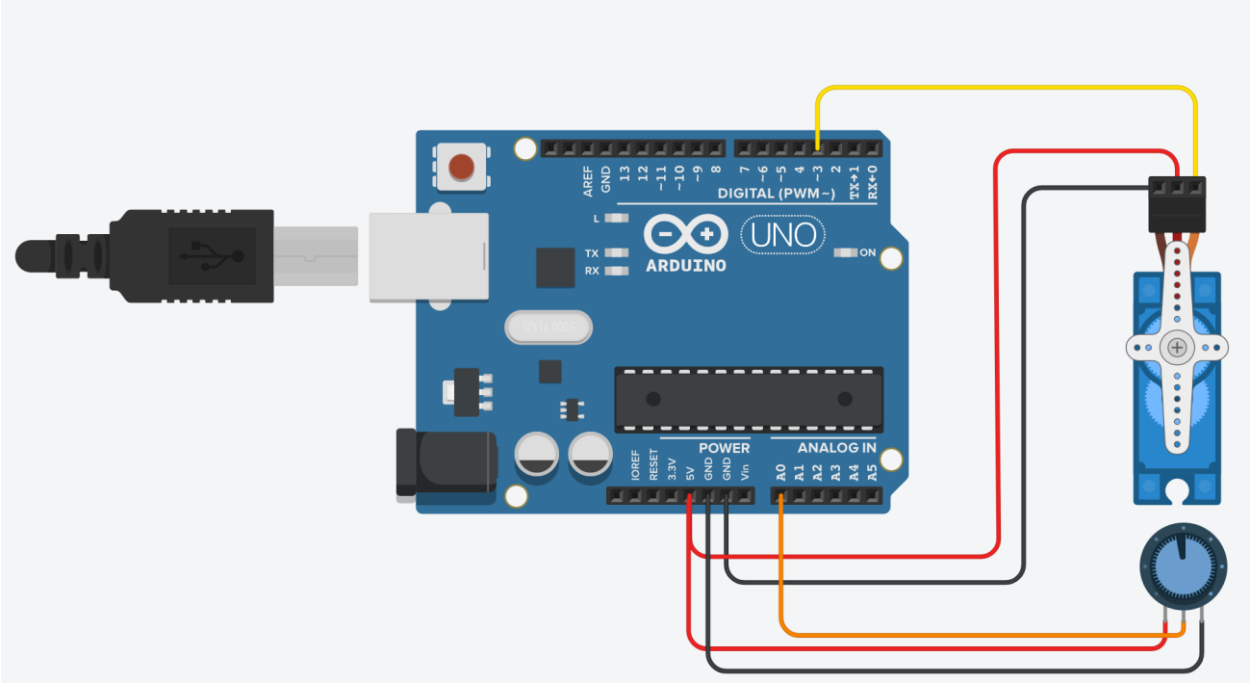
Devre 5

```
#include <Servo.h>;              //Servo kütüphanesi

Servo myServo;                   //Bir servo nesnesi oluşturmamız gerekiyor

void setup()
{
myServo.attach(3);               //Servo pinini yapılandır
myServo.write(0);                //Servoyu 0 derecelik açıya ayarla
}

void loop()
{
myServo.write(90);               //servoyu 90 derece açıyla ayarla
delay(1000);                     //1 saniye bekle
myServo.write(180);    //servoyu 180 derecelik açıya ayarla (maksimum)
delay(1000);                     //1 saniye bekle
myServo.write(0);                //servoyu 0 açıya ayarla (min)
delay(1000);                   //1 saniye bekle
}
```

Devre 6:

Devre başlığı: Potansiyometre kullanarak servo motoru kontrol etme

Devre açıklaması: Bu devrede A0 pinine bağlı bir potansiyometre servo motor pozisyonunu kontrol etmek için kullanılır. Servo bağlantıları önceki devredekiyle aynıdır ve potansiyometre Arduino kartının A0 analog pinindeki 5V topraklamaya bağlanır.



Devre 6

Bu örnekte servo pozisyonu potansiyometre tarafından kontrol edilir. Potansiyometre pozisyonunu servo pozisyonuna çevirmek için fonksiyon haritası kullanılır:

map(değer, en_küçükten, en_büyükten, en_küçükten, en_büyükten)	[from_min, from_max] ölçeğinden [to_min, to_max] ölçeğine bir değer eşleyin.

```
#include <Servo.h>;

Servo myServo;                          //Servo nesnesini oluşturur

void setup()
{
myServo.attach(3);                      //Servoyu 3 numaralı pine bağla
myServo.write(0);                       //servoyu 0 pozisyonuna konumlandır
}
```

```
void loop()
{
int position = map(analogRead(0),0,1023,0,175);     //potansiyometrenin konumunu haritala
myServo.write(position);                            //servonun bir açısına ve depolama
}                                                   //pozisyon değişkenindeki eşlenen değer
                                                    //servoya göndermeden önce
```

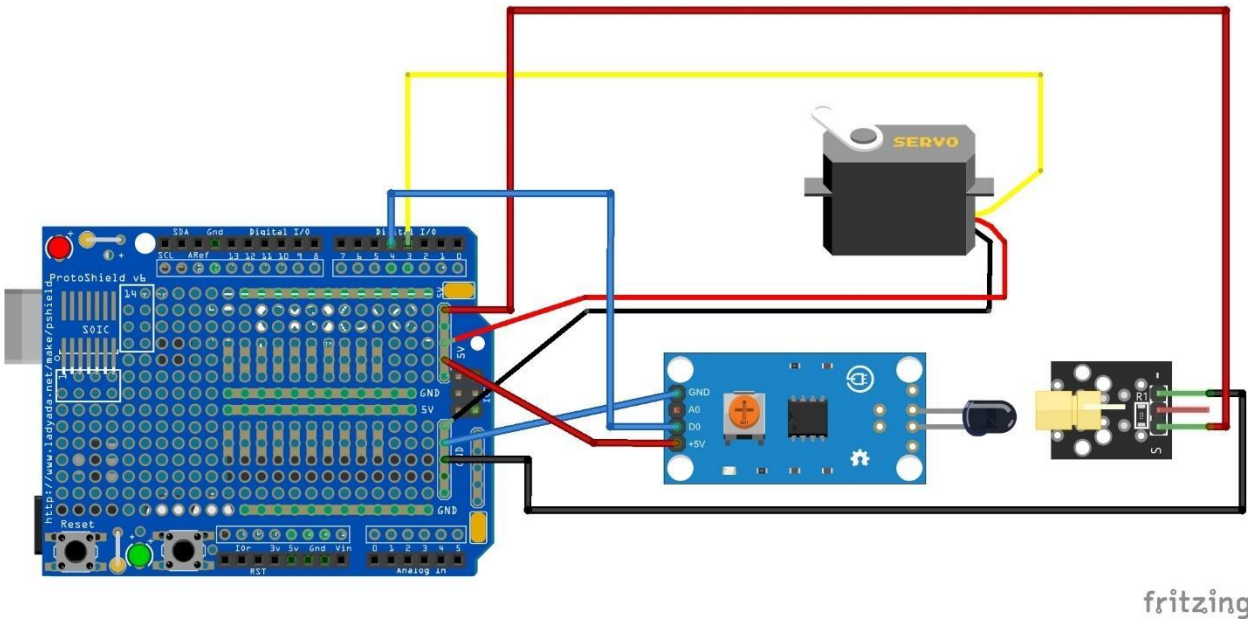
Devre 7:

Devre Adı: Lazer dedektörlü araç bariyeri

Devre açıklaması: Bir servo kullanarak bir araba bariyerini simüle edeceğiz. Kartı algılamak için bir LAZER ışık yayıcı ve bir ışık dedektörü kullanılır. Bağlantıları kolaylaştırmak için bir proto kalkan kullanılır.

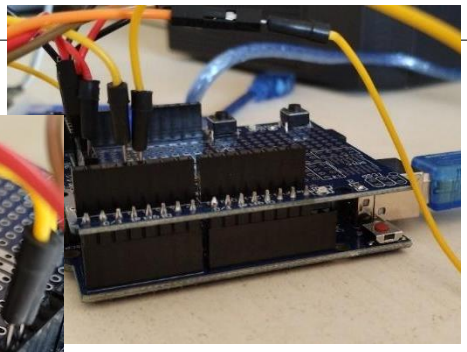
Bu proje için şunları kullanacağız:

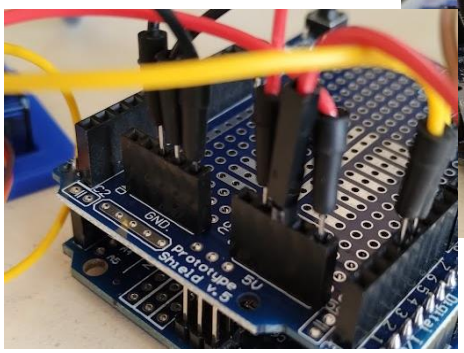
- 1 Lazer yayıcı modülü
- 1 Işık dedektörü modülü
- 1 Servo motor SG90
- 1 Proto kalkan
- Kablolar
- Destek panosu
- Bariyer
- Araba
- Arduino Uno kartı



Devre 7:

Proto shield'ı Arduino kartının üzerine bağlayın. Sensörlerin ve servo motorun güç kablolarını bağlamak için çok kullanışlı olan 5V ve GND için iki pin olduğunu lütfen unutmayın.

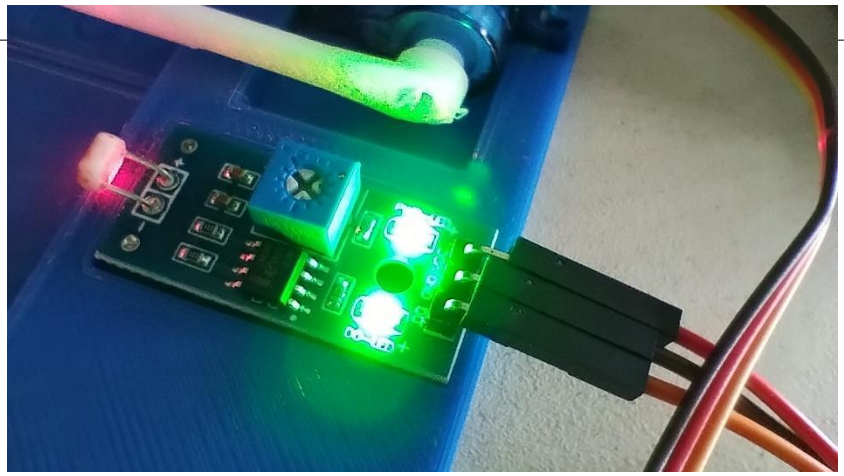




LASER modülünün - ve S pinini Proto Shield'in GND ve 5V başlıklarına bağlayın:

LAZER Yayıcı Modülü	
S	5V
Orta pin	Bağlantılı değil
-	GND

Işık dedektörü modülünün VCC, GND ve D0 pinlerini proto shield'ın ilgili pinlerine bağlayın:



Işık Dedektörü Modülü	
D0	Dijital Pin 4
GND	GND
VCC	5V

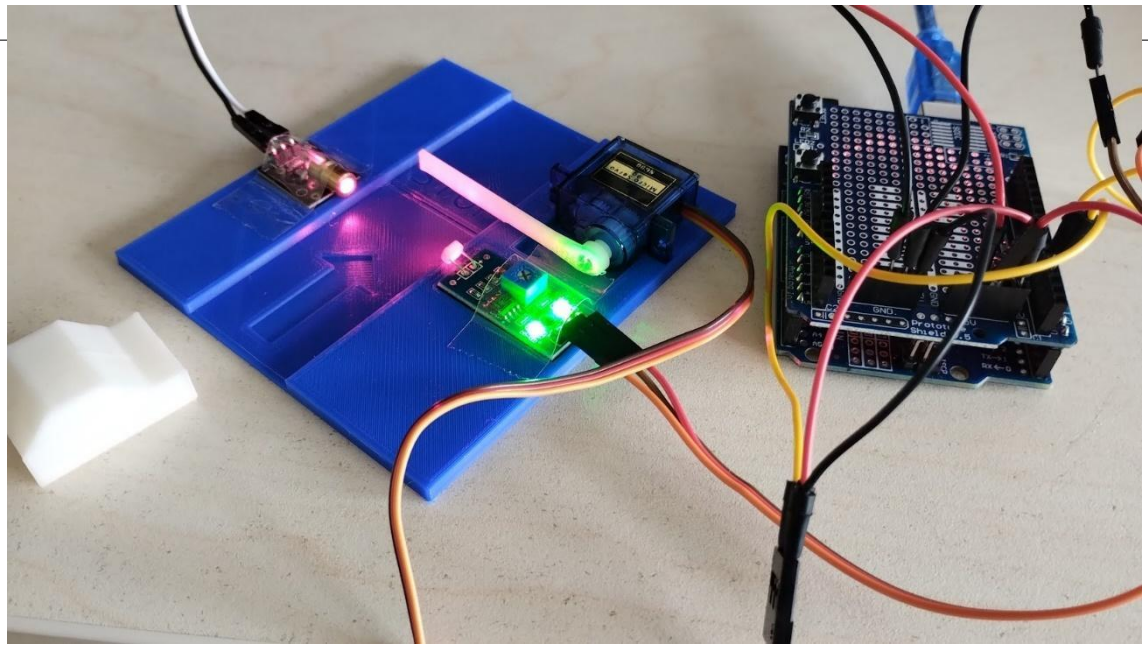
Arduino'yu açın, LAZER'i ışık sensörüne doğrultun ve ışık sensörü modülünün mavi potansiyometresindeki bir tornavidayı kullanarak ışığın hassasiyetini ayarlayın. Sol ışık yalnızca LAZER yukarıdaki resimdeki gibi sensöre çarptığında yanmalıdır.

Servo pinlerini proto shield'ın güç ve veri pinlerine bağlayın:

Servo motor	
Kahverengi	GND
Kırmızı	5V
Sarı	Dijital Pin ~3

Bariyeri servo motora takın. Servo motor doğru konumda değilse bariyerin açısını ayarlamanız gerekebilir.

Tüm parçaları destek tahtasına yerleştirin ve yerinde kalmalarını sağlamak için yapışkan bant veya mavi raptiye ile sabitleyin.



Şekil 5 Servo kontrollü bariyer

Arduino ile Servo Motor Kontrolü

Bir servo motoru Arduino ile kontrol etmek için, süreci kolay ve rahat hale getiren Servo kütüphanesini kullanabiliriz. Kütüphane, servoyu bir pine takmak, pozisyonu ayarlamak ve servoyu istenen açıya düzgün bir şekilde hareket ettirmek için işlevler sağlar.

Yanlış nesne algılamasını önlemek için program sensörü her 250ms'de bir okuyacaktır. Program, yalnızca 4 okumada LASER ile ışık sensörü arasında bir engel algılandığında bekleyen bir araba olduğunu düşünecektir.

```
#include <Servo.h>

Servo myServo;
int detections;


int sensorPin = 4;                  //Işık sensörü pimi
int servoPin = 3;                   //Servo Motor pini. PWM pini olmalı
int open = 0;                       //Açık durumda servonun konumu
int close = 128;            //Kapalı durumda servonun konumu
int threshold = 4;                  //Algılama olarak değerlendirilecek satır okumalarının sayısı
int waitToPass = 4000; // Arabanın geçmesini beklemek için gereken süre

void setup() {
myServo.attach(servoPin);       //Servo motoru başlat
myServo.write(close);           //Kapalı konuma bariyer koy
pinMode(sensorPin , INPUT);   //Sensör pinini giriş pini olarak ayarla
detections = 0;                 //Nesne algılama sayısını başlat
}
```

```
void loop() {

if (digitalRead(sensorPin)==1){          //Eğer bir şey algılanırsa
   detections = min ( threshold, detections +1);//Algılama sayısını bir artır
                                                 //eşiğe ulaşana kadar
} else {                                 //Hiçbir nesne algılanmazsa
detections = max ( 0, detections -1);    // Algılama sayısını bir azalt
                                                 //0'a ulaşana kadar
}
delay (250);                                     //Saniyede dört okuma

if (detections == threshold){            //Eğer tespitin eşik seviyesine ulaşılırsa
myServo.write(open);                     //Bariyeri aç
delay (waitToPass);                      // Arabanın geçmesini bekle
}

if (detections == 0){                            //Eşik okumaları için hiçbir nesne algılanmazsa
myServo.write(close);                    //Bariyeri kapat
}

}
```

Co-funded by the
Erasmus+ Programme
of the European Union

Erasmus+ KA210-VET

Mesleki eğitimde küçük ölçekli ortaklıklar

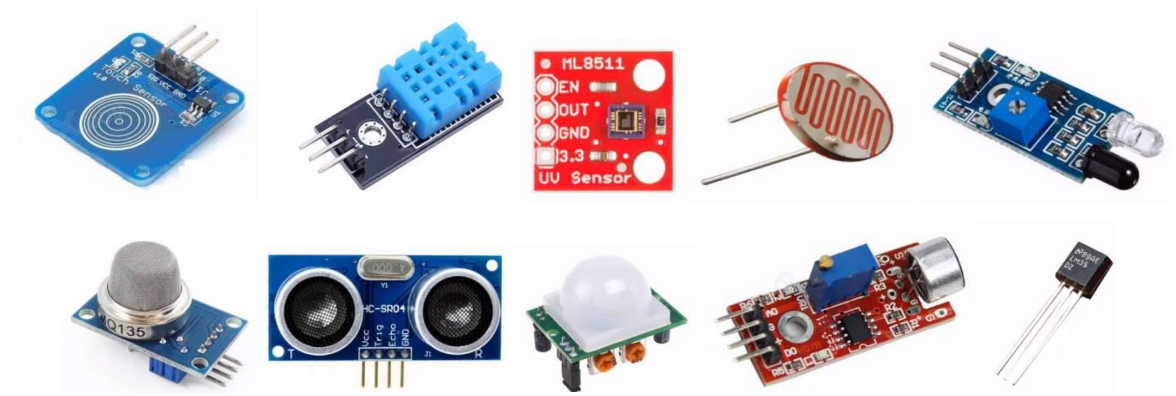
Eğitim ve Öğretim

Proje Başlığı: "Mesleki Eğitimde Arduino Kullanımı"

Proje Kısaltması: "UsingARDinVET"

Proje Numarası: "2023-1-RO01-KA210-VET-000156616"

SENSÖRLER Modülü ve Eğitim KİTİ



Sensör Modülü ve Eğitim KİTİ

Bu modülün amacı sensörlerin Arduino ile işbirliği içinde nasıl çalıştığını açıklamaktır. Ortaöğretim mesleki eğitim öğretmenleri tarafından kullanılmak üzere çeşitli Arduino tabanlı Sensör türleri sunulacak ve bunları içeren uygulamalar geliştirilecektir .

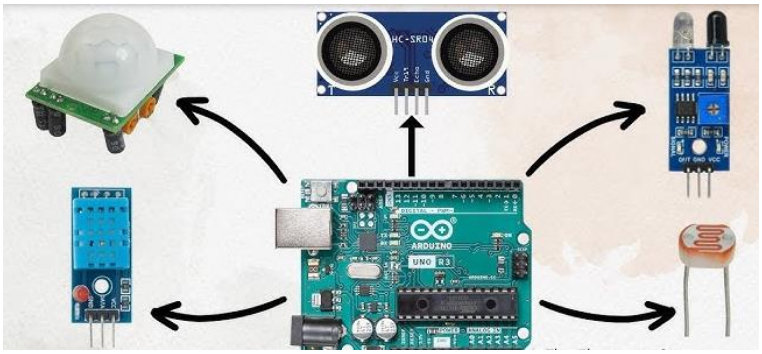
Sensörler hakkında

Sensör , Arduino kartının çevreyle etkileşime girdiği bir cihazdır. Bir sensör aslında bir giriş olarak bir sinyal alır ve bir çıkış olarak bir elektrik sinyali vererek yanıt verir.

Daha spesifik olmak gerekirse, sensörler farklı türde sinyaller alır, yani fiziksel, kimyasal veya biyolojik ve bunları akım, voltaj veya yük biçiminde bir elektrik sinyaline dönüştürerek yanıt verir. Ayrıca bir sensörün elektriksel olmayan bir değeri elektriksel bir değere dönüştüren bir çevirici olduğunu da söyleyebiliriz.

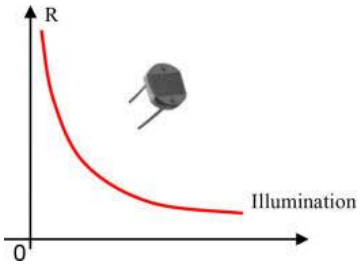
Uygulama alanına, giriş sinyaline, dönüşüm mekanizmasına ve maliyet, doğruluk veya menzil gibi sensör karakteristiklerinde kullanılan malzemeye bağlı olarak farklı tiplerde sensörler bulunmaktadır.

Dünyamız farklı tipte sensörler ve bunların basit veya daha karmaşık uygulamalarıyla dolu olduğundan, bunlara her yerde rastlayabiliriz: ofislerimizde, bahçelerimizde, alışveriş merkezlerimizde, evlerimizde, arabalarımızda, oyuncaklarımızda vb. Bu nedenle, bunların çalışma şekillerini ve çok sayıdaki olasılıklarını anlamak kritik önem taşır.



1. Fotodirenç

1.1 Fotodirenç hakkında



Fotodirenç, ışık kontrollü değişken bir dirençtir. Yüzeye düşen yüksek yoğunluklu ışık daha düşük bir dirence neden olurken, daha düşük ışık yoğunluğu daha yüksek bir dirence neden olur. En yaygın kullanımı, ışık ve karanlık etkinleştirme anahtarı olarak kullanılır. Ayrıca LDR (Işığa Bağımlı Direnç) olarak da adlandırılır .

Bir fotodirencin ışıkla nasıl tepkimeye girdiğini görelim.

1.2. Devre 1:

Devre başlığı: Işığı algılama

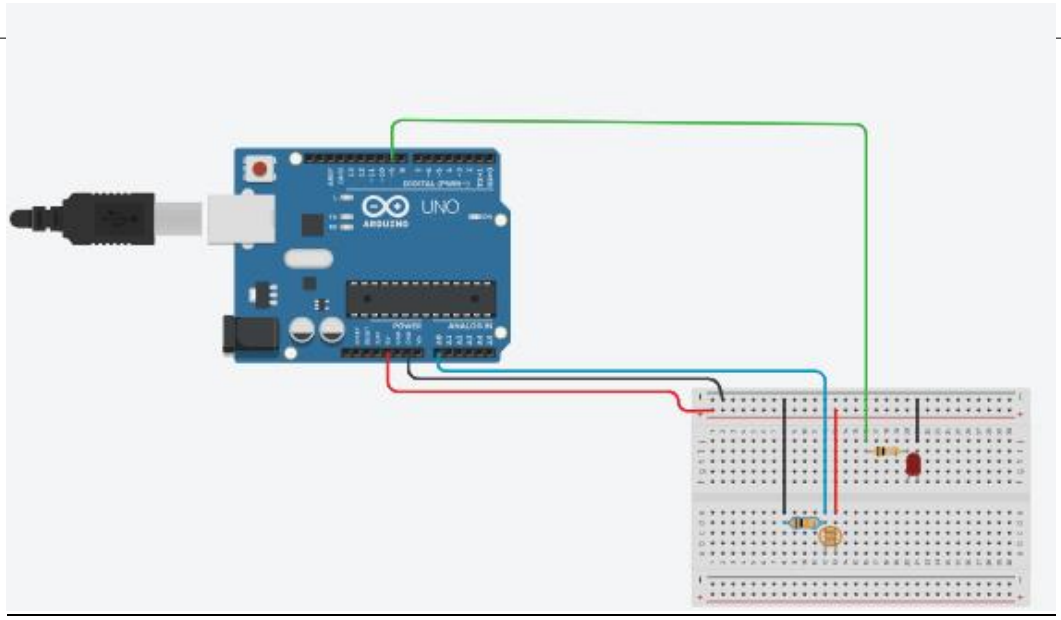
Devre açıklaması: Devre odanın parlaklığını algılar, ayarlanan sınırın altına düşerse led'i yakar

İhtiyacınız olanlar :

a) Ekmek tahtası
b) Fotodirenç
c) Direnç 220Ω* ,10KΩ **
d) 3mm led
e) Ardunio

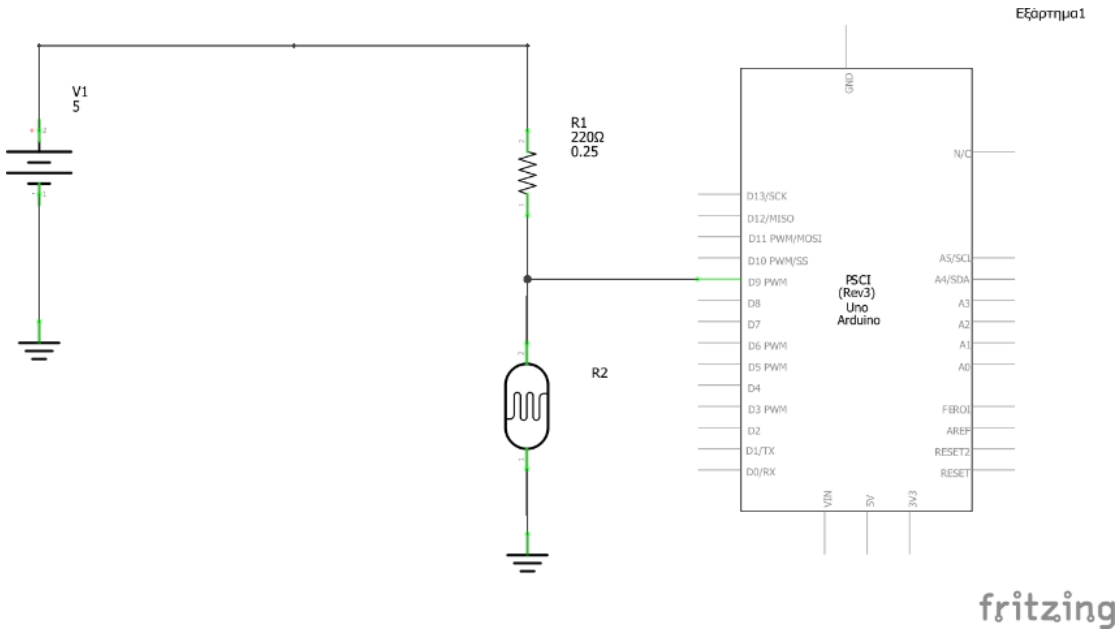
*220Ω direnç, 5 voltta LED'in standart koruma direncidir.

** 10KΩ direnç, fotodirenç değerinin önemli ölçüde azalması durumunda Arduino'yu aşırı akımdan korumak için kullanılır.



Pin 9'a bağlı LED çıkış olarak ayarlanır. Pin 9 YÜKSEK olduğunda LED ışık yayar; aksi takdirde kapalı kalır.

Fotodirenç ve 10KΩ dirençle bir voltaj bölücü oluşturuyoruz. İşlemciye gönderilecek değer, sensördeki ışığın yoğunluğuna bağlıdır.



1.3 Kod:

```
constint photoantistasi = A0; // Arduino analog pin A0'daki fotodirenç
constint ledPin=9;       // Arduino pin 9'daki LED pin

//Değişkenler
int timi ; // Fotodirençten gelen değeri sakla

voidsetup(){
pinMode(ledPin,OUTPUT);  // lepPin - 9 pinini çıkış olarak ayarla
pinMode(photoantistasi ,INPUT);//  Fotoantistasi - A0 pinini giriş olarak ayarla
Serial.begin(9600);// // seri portu açar, veri hızını 9600 bps'ye ayarlar

}

voidloop(){
value =analogRead (foton antistatik);

 //"70" değerini değiştirebilirsiniz.
 if(time >70){
      digitalWrite(ledPin,LOW);  // Led'i kapat
}
  else {
    digitalWrite(ledPin,HIGH); //Ledi aç
}

 delay(500); //Küçük gecikme
}
```

Program analog girişi okur. Değer '70'i aşarsa, LED yanar; aksi takdirde söner. Her program döngüsü, sensöre ölçüm yapması için zaman vermek üzere 500 milisaniyelik bir gecikme içerir ('delay(500)' komutu kullanılarak).

İpucu:

Kodunuzu değiştirebilir ve seri port üzerinden bilgisayarınızdan sensör değerlerini izleyebilirsiniz.

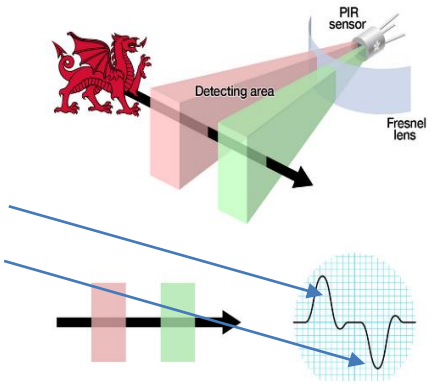
```
constint photoantistasi  = A0; // Arduino analog pin A0'daki fotodirenç
constint ledPin=9;       // Arduino pin 9'daki LED pin

//Değişkenler
int timi ; // Fotodirençten gelen değeri sakla

voidsetup(){
pinModu (ledPin ,ÇIKIŞ ); // lepPin - 9 pinini çıkış olarak ayarla
pinMode(ledPin,OUTPUT);  // Resistor - A0 pinini giriş olarak ayarla (isteğe bağlı)
Serial.begin(9600);      // // seri portu açar, veri hızını 9600 bps'ye ayarlar

}

voidloop(){
value =analogRead (foton antistatik);

  //"25" değerini değiştirebilirsiniz.
  if (zamanı >70){
    digitalWrite(ledPin,LOW);   // Led'i kapat
}
  else {
    digitalWrite(ledPin,HIGH); //Ledi aç
}

  delay(500); //Küçük gecikme
  Serial.println(timi); // değişkenin her değişim satırından sonra ekrana gelen değeri
                yazdırır .
```

2. Hareket sensörü

Hareket sensörü, kapsama alanı içindeki kızılötesi radyasyondaki değişiklikleri veya ısı ve hareketin varlığını algılayarak çalışır. En yaygın hareket sensörü türü, görüş alanındaki nesneler tarafından yayılan kızılötesi radyasyondaki değişiklikleri algılayan pasif kızılötesi (PIR) sensördür.

2.1 PIR sensörü hakkında

Kızılötesi hareket algılama sensörü (Pyroelectric InfraRed Sensors), belirli bir alanda hareket eden canlı organizmanın varlığını tespit etmemizi sağlayan bir sensördür.



Tüm canlılar kızılötesi ışınım yayarlar ve bu ışınım uygun sensörler tarafından kolayca algılanabilir.

PIR sensörü kızılötesi radyasyonu algılayan iki alandan oluşur.

Canlı bir organizmanın ilk bölgeye girmesiyle sensörün elektronik devresinde pozitif bir potansiyel farkı oluşur.

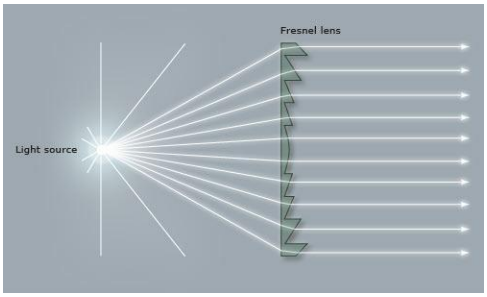
İkinci bölgeyi terk ettiğinde negatif potansiyel farkı oluşur.

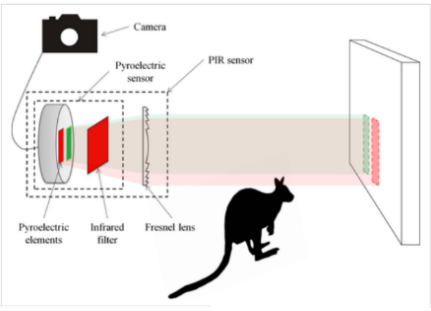
Bu voltaj sensörün elektronik devresini tetikler ve işlemciye mantıksal bir '1' gönderir (uzayda hareketin varlığı). Bu fonksiyonun sorunu, yalnızca iki algılama alanı arasındaki ince bölge olan küçük bir alanı algılamasıdır.

Alan aralığını arttırmak için sensör bir Fresnel lens ile kaplanmıştır.

Fresnel merceğini keşfeden Augustin Fresnel, deniz fenerlerinde de bu merceğin kullanımını buldu.

Özelliği, bir kaynaktan gelen ışığı belli bir doğrultuda, belli bir düzeyde yoğunlaştırmasıdır.





Burada tam tersini kullanıyoruz. Resimde görüldüğü gibi.

Bir alanda bulunan kızılötesi radyasyon, sensör üzerinde birleşir ve böylece algılayabileceği alanı artırır

2.2 Devre 2:

Devre başlığı: Hareket dedektörü

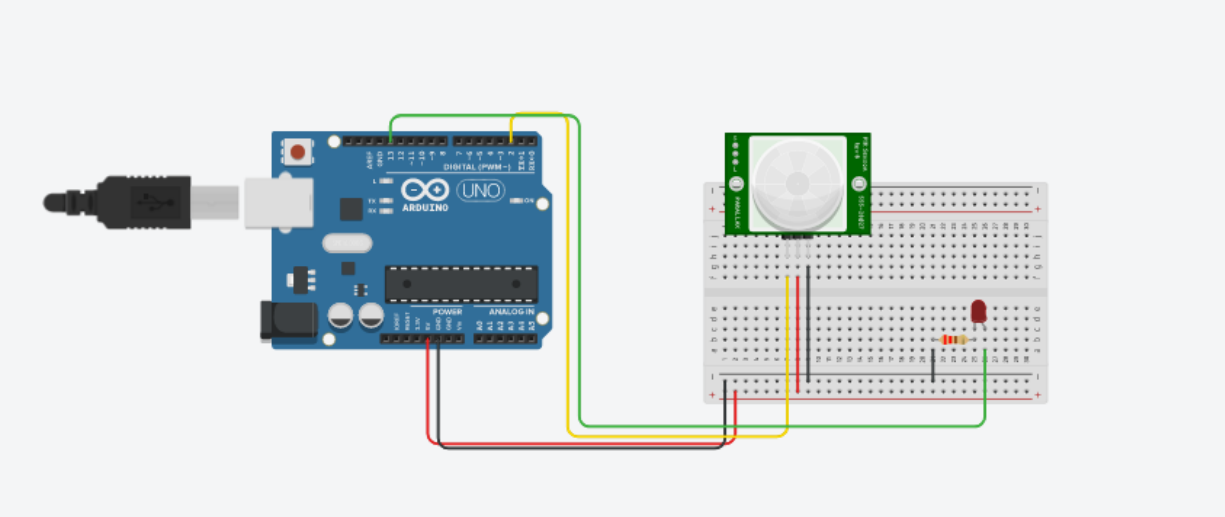
Devre açıklaması: Devre alanı izler. Hareket algılanırsa, LED'i etkinleştirir.

İhtiyacınız olanlar :

a) Ekmek tahtası
b) RIP sensörü
c) Direnç 220Ω* ,10KΩ **
d) 3mm led
e) Ardunio

*220Ω direnç, 5 voltta LED'in standart koruma direncidir.

** 10KΩ direnç, fotodirenç değerinin önemli ölçüde azalması durumunda Arduino'yu aşırı akımdan korumak için kullanılır.



Sensöre 5 voltluk bir voltaj verin. Pin sinyali Arduino'nun 2. pinine bağlanır. LED ise Arduino'nun 13. pinine bağlanır.

Sensör hareket algıladığında, bit '1' gönderir ve program LED'i yakar

PIR sensörünün üç pimi vardır: 5 voltluk bir güç kaynağına bağlı güç pimi; GR pimine bağlı toprak pimi; ve sinyal pimi. Sinyal pimi, hareket algılanmadığında 0 değeri ve sensör hareket algıladığında 1 değeri verir.

2. 3 Kod:

```
int led = 13;                    // LED'in bağlı olduğu pin
 int sensor = 2;                 // sensörün bağlı olduğu pin
 int state =LOW;                 // varsayılan olarak, hareket algılanmadı
 int val = 0;                    // sensör durumunu (değer) depolamak için değişken

 voidsetup() {
  pinMode(led,OUTPUT);           // LED'i çıkış olarak başlat
pinMode(sensor,INPUT);           // sensörü giriş olarak başlat
 pinMode(buzzer,OUTPUT);
 tone(buzzer, 1000, 2000);
Serial.begin(9600);              // seriyi başlat

}
voidloop(){
 val =digitalRead(sensor);       // sensör değerini oku
 if (val ==HIGH) {          /                  / sensörün YÜKSEK olup olmadığını kontrol
edin
      digitalWrite(led,HIGH);    // LED'i AÇ
  delay(100);                      // 100 milisaniye gecikme
  if (state ==LOW) {
    Serial.println ( "Hareket algılandı!" );
state =HIGH;                     // değişken durumunu YÜKSEK olarak güncelle
}
}
  else {
    digitalWrite(led,LOW);       // LED'i KAPATIN
    noTone(buzzer);
    delay(200);                  // 200 milisaniye gecikme

    if (state ==HIGH){
Serial.println ( "Hareket durduruldu!" );
     state =LOW;                 // değişken durumunu DÜŞÜK olarak güncelle
}
}
}
```

Program Arduino'nun 2. girişini okur. Değer '1' ise, LED yanar ve seri port üzerinden "Hareket algılandı!" görüntülenir; aksi takdirde, LED söner ve "Hareket durduruldu" görüntülenir .

3. Alev sensörü

Bina otomasyonunda önemli bir konu yangın algılamadır. İki tür sensör kullanılır: duman algılama ve alev sensörleri. Genellikle duman sensörü etkinleştirildiğinde bir alarm alırız, alev sensörü etkinleştirildiğinde ise söndürme sistemi başlar.

Yangını tespit etmenin birden fazla yolu vardır; örneğin sıcaklık değişimini tespit etmek, dumanı tespit etmek vb. Bunların hepsinde, sıcaklık değişimini tespit etmek daha doğru olacaktır çünkü bazı yangınlarda tespit edilebilir duman bile olmayacaktır.

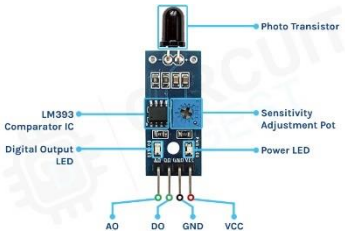
3.1 Alev sensörü hakkında

IR fotodiyot herhangi bir sıcak gövdeden gelen IR radyasyonunu algılar. Daha sonra bu değeri belirtilen bir değerle karşılaştırır.

Sensörün analog çıkışında ışınım değerini görüntüleyebiliriz veya ışınım değeri bir eşik değerine ulaştığında sensör buna göre dijital çıkışını değiştirir.

Alev sensörü modülü, bir IR fotodiyot, bir LM393 karşılaştırıcı IC ve bazı pasif bileşenleri içeren çok az sayıda bileşene sahiptir.

Modüle güç verildiğinde güç LED'i yanacak ve alev algılandığında D0 LED'i sönecektir. Hassasiyet, yerleşik trimmer direnci ile ayarlanabilir.



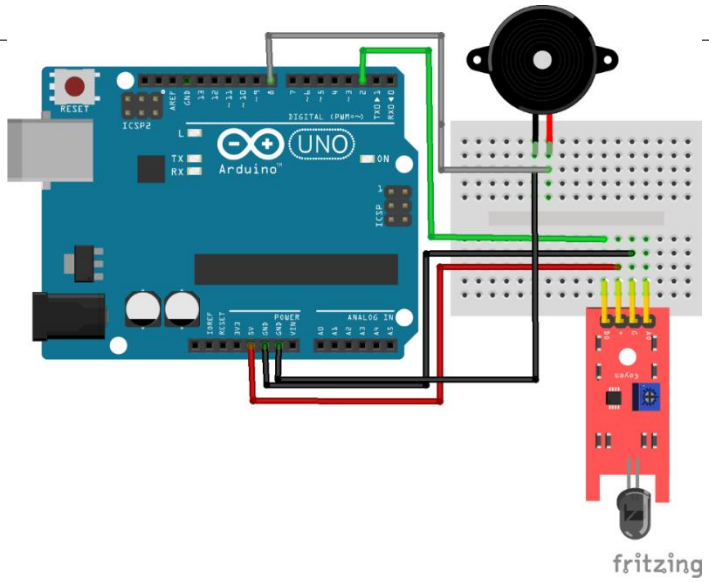
3. 2 Devre 3:

Devre başlığı: Alev dedektörü

Devre açıklaması: Devre alanı izler. Alev algılanırsa, zili etkinleştirin ve alarm çalar.

İhtiyacınız olanlar :

a) Ekmek tahtası
b) Alev sensörü
c) Pasif zil
d) Ardunio



Sensöre 5 voltluk bir voltaj verin. Dijital pin sinyali Arduino'nun 2. pinine bağlanır.Buzzer Arduino'nun 13. pinine bağlanır.

Bir alev algılandığında, sensör Arduino girişine mantıksal bir "1" gönderir ve siz, çıkış 8 üzerinden 1 K Hz ses frekansına sahip buzzer'ı etkinleştirirsiniz.

3. 3 Kod:

```
intflameSens = 2; // flameSens'i arduino pin 2'ye
intbuzzerPin= 8; //buzzer arduino pin 8'e
geçersizkurmak (){
  pinMode (alevSens ,GİRİŞ ); // Alev sensörü çıkış pinini giriş olarak bağlanmış pin olarak başlat
  pinMode ( buzzerPin ,ÇIKIŞ ); // dijital pin buzzerPin'i çıkış olarak başlat
  Seri . begin (9600); // seri iletişimi 9600 baud hızında başlat
}
geçersizdöngü (){
  eğer ( digitalRead (flameSens) == 1 )
{
    ton ( buzzerPin , 1000); // 1KHz ses sinyali gönder
    Seri . println ( "** Uyarı!!!! Yangın algılandı!!! **" );
}
  başka
{
    digitalWrite ( LED_YERLEŞTİRİLMİŞ ,DÜŞÜK ); // Led KAPALI
    noTone (buzzerPin); //sesi durdur
    Seri . println ( "Yangın Algılanmadı" );
}
  gecikme (100);
}
```

tips

tone() fonksiyonunu kullanarak farklı sesler yaratabilirsiniz. Aralarında 1000 saniyelik gecikmeyle 2400 Hz ve 2900 Hz ses frekansları üretmeyi deneyin.

4. Toprak nem sensörü

Toprak nem sensörleri tarım, peyzaj ve bahçecilikte popülerdir. Nem algılama teknolojileri, yetiştiricilere manuel sulama sistemlerine harcanan paradan ve zamandan tasarruf sağlayarak fayda sağlar.

Çiftçilerin ve bahçıvanların nem seviyelerini hızlı, hassas ve uygun fiyatlı bir şekilde kontrol etmelerine yardımcı olabilirler. Bununla, tarlanızı ne zaman sulayacağınızı veya sulama yapacağınızı kolayca belirleyebilirsiniz.

Sensör kullanılarak yapılan toprak analizi, gerçek zamanlı veri sağlayarak belirsizliği azaltır ve su gibi kaynakların nasıl yönetileceği konusunda daha bilinçli kararlar almanızı sağlar.

4.1 Toprak nem sensörü hakkında

Toprak nem sensörü basit bir şekilde çalışır.
Sensör, toprağa veya nem içeriğinin ölçüleceği yere yerleştirilen iki açık iletkene sahip çatal biçimli bir probdan oluşur.

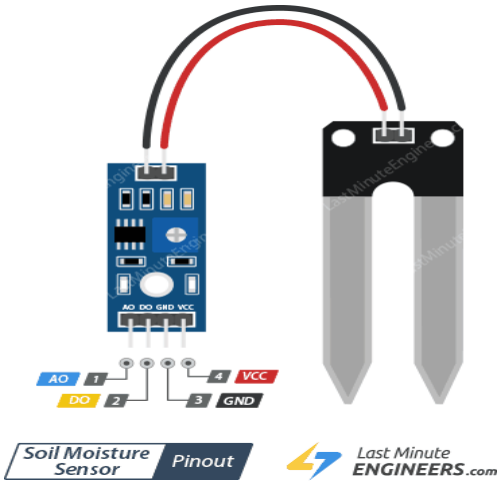
Bunlar, direnci toprağın nem içeriğine göre değişen değişken dirençler (potansiyometrelere benzer) olarak işlev görür. Bu direnç, toprak nemiyle ters orantılı olarak değişir.

Ayrıca sensör, probu Arduino'ya bağlayan elektronik bir modülü de bünyesinde barındırıyor.

Modül, Analog Çıkış (AO) pininde bulunan probun direncine bağlı olarak bir çıkış voltajı üretir.

Aynı sinyal, onu sayısallaştıran ve Dijital Çıkış (DO) pininde kullanılabilir hale getiren LM393 Yüksek Hassasiyetli Karşılaştırıcıya beslenir.

Modül, dijital çıkışın (DO) hassasiyetini ayarlamak için bir potansiyometre içerir.



Toprak nem sensörünün bağlanması için sadece dört pin yeterlidir.

AO (Analog Çıkış), toprak nem seviyesine orantılı analog çıkış voltajı üretir.

DO (Dijital Çıkış) Nem seviyesi potansiyometre ile ayarlanan eşik değerini aştığında D0 DÜŞÜK olur, aştığında ise YÜKSEK olur.

VCC sensöre güç sağlar. Sensörün 3,3 V ila 5 V arasında güçlendirilmesi önerilir. Analog çıkışın sensöre sağlanan voltaja bağlı olarak değişeceğini lütfen unutmayın.

GND topraklama pinidir

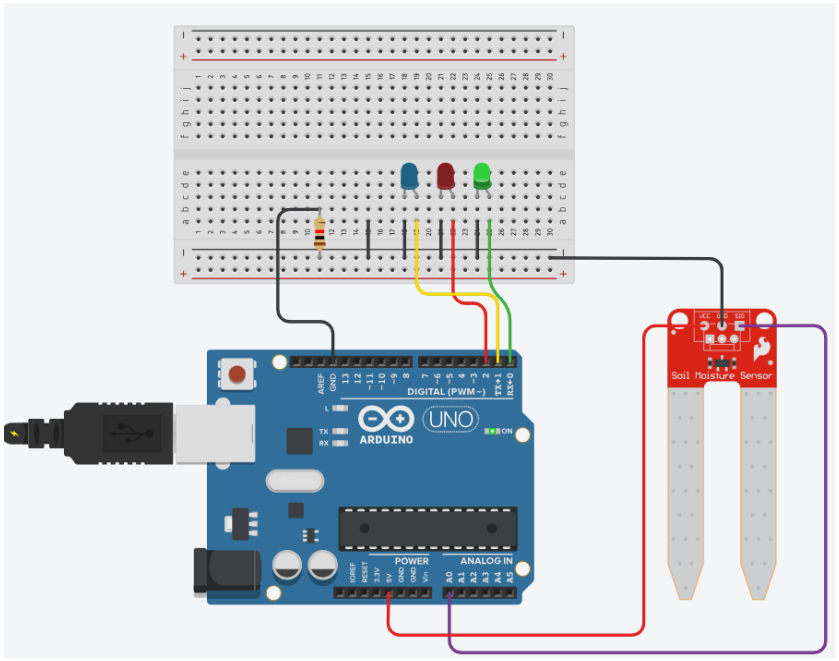
4. 2 Devre 4:

Devre başlığı: Toprak nem dedektörü

Devre açıklaması: Devre, toprak nemini belirli bir değerin yüzdesi olarak ölçer. Nem %0-30 arasındaysa kırmızı ışık yanar; %30-60 arasındaysa mavi ışık yanar ve %60'ın üzerindeyse yeşil ışık yanar.

İhtiyacınız olanlar :

a) Ekmek tahtası
b) Toprak nem sensörü
c) Yeşil, mavi, kırmızı led
d) LED'i korumak için 220Ω'luk bir direnç gereklidir
e) Ardunio



Not : Program başlangıcında sensörün maksimum nem seviyesini algılaması gerekmektedir ki, program döngüsü boyunca yüzdeleri doğru bir şekilde hesaplayabilsin.

4.3 Kod:

```
// Nem Sensörü Arduino Kodu
intgreenLight= 9; //Pinleri tanımlama
intblueLight= 8;
intredLight   = 10;
floatmaximumMoistureLevel;//Yüzde hesaplamaları için maksimum nem seviyesi ve mevcut
                                    nem seviyelerine ihtiyaç duyulacaktır
```

```
floatcurrentMoistureLevel;// = Nem seviyesi

voidsetup() {
pinMode(8,OUTPUT); //8. pini çıkış olarak başlat
pinMode(9,OUTPUT); //9. pini çıkış olarak başlat
pinMode(10,OUTPUT); //pin 10'u çıkış olarak başlat
pinMode (A1,INPUT); //A1 Toprak Nem Sensörü için kullanılan pindir
delay(100);
    maximumMoistureLevel =analogOkuma (A1)
      digitalWrite(greenLight,   HIGH); //programın başlatıldığını göstermek için tüm ledler 2sn
                                                    boyunca yanar.
      digitalWrite(blueLight,HIGH);
   digitalWrite(redLight,HIGH);
   delay(100);
   digitalWrite(greenLight,   LOW);
   digitalWrite(blueLight,LOW);
   digitalWrite(redLight,LOW);
   Serial.begin(9600);
}

void   loop() {
    if (maximumMoistureLevel/currentMoistureLevel<= 0.3) //eğer nem seviyesi %30'un
                                                                altındaysa
{
  digitalWrite(greenLight,   LOW);
   digitalWrite(blueLight,LOW);
   digitalWrite(redLight,HIGH);   //Kırmızı ışığı yak, ancak zili çalma
}
      elseif (maximumMoistureLevel/currentMoistureLevel   <= 0.8
      &&maximumMoistureLevel/currentMoistureLevel> 0.3) // nem seviyesi %30 ile %60
      arasındaysa
    {
   digitalWrite(greenLight,LOW);
   digitalWrite(blueLight,   HIGH);//Sadece sarı ışığı aç
   digitalWrite(redLight,LOW);
}
      else   //Aksi takdirde nem seviyesi %60'ın üzerindedir ve bu nedenle yeterince iyidir
{
   digitalWrite(greenLight,HIGH);///Yeşil ışığı aç
   digitalWrite(blueLight,   LOW);
   digitalWrite(redLight,LOW);

}
  currentMoistureLevel=analogRead(A1);//ölçümlerin yenilenmesi

  delay(500); //Programı aşırı yüklememek için kısa gecikme
   Serial.println("maximumMoistureLevel");/// Maksimum nem seviyesini 0-1023 arasında bir
                                        okuma olarak görebilmeniz için
Serial.println(maximumMoistureLevel);
   Serial.println("currentMoistureLevel");/// Sadece nem seviyesini 0-1023 arasında bir okuma
olarak görebilmeniz için
```

```
  Serial.println(currentMoistureLevel);
}
```

ipuçları

Nem sensörünün bir dezavantajı vardır: nemli bir ortamda sürekli olarak çalıştırıldığı için ömrü kısadır. Bu nedenle, sensörü yalnızca ölçüm yapmanız gerektiğinde çalıştırmalısınız **.**

5. Aktif yangın koruması (AFP)

Aktif yangın koruması (AFP) hakkında

Aktif Yangın Koruması (AFP), yangın durumunda etkili bir şekilde işlev görmek için bir tür eylem gerektiren bir sistem kümesini ifade eder. Bu eylemler, yangın söndürücü kullanımı gibi manuel olarak veya sprinkler sistemi gibi otomatik olarak çalıştırılabilir. Yöntemden bağımsız olarak, bu sistemlerin çalışması için bir miktar eylem gereklidir.

Tipik bir aktif yangın koruma sistemi yangını algılar, alarmı çalar ve ardından otomatik söndürme sistemini etkinleştirir.

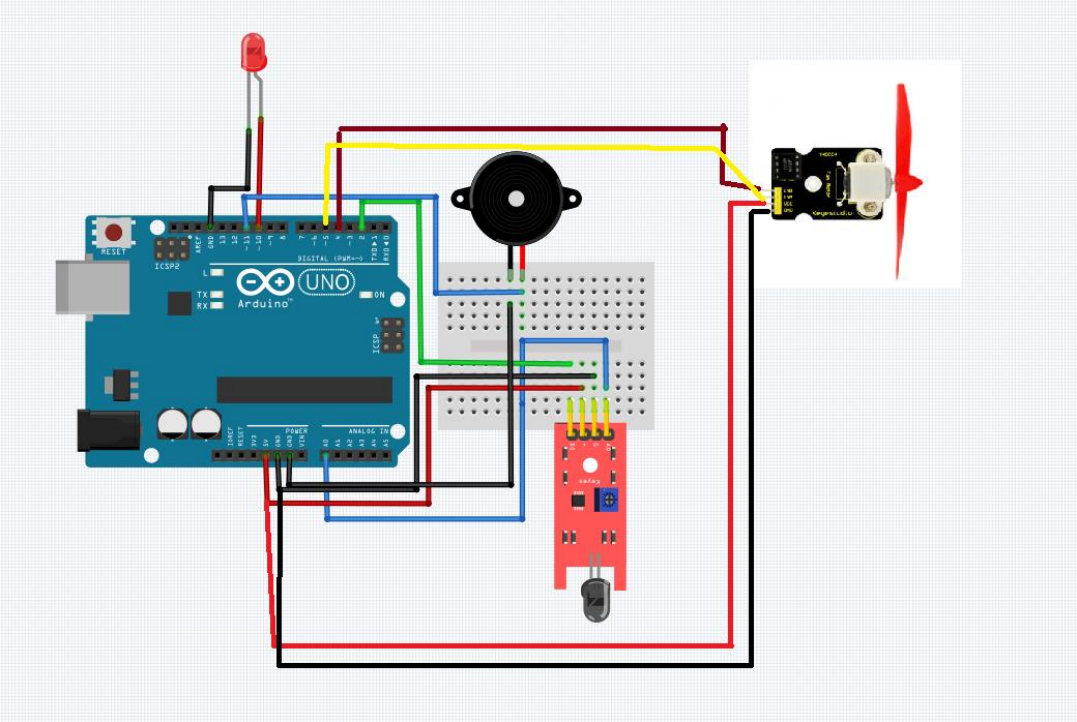
5.2 Devre 5:

Devre başlığı: Aktif yangın koruma (AFP)

Devre açıklaması: Devre, bir alev sensörü kullanarak yangını algılar. Bir alev algılandığında, hem sesli hem de görsel bir uyarı etkinleştirilir ve alarm dört kez çalar (sesli uyarı + LED). Alev devam ederse, söndürmek için fan 4 saniye boyunca açılır. Alev devam ederse, işlem kendini tekrar eder.

İhtiyacınız olanlar:

a) Ekmek tahtası
b) alev sensörü
c) Kırmızı led
d) LED'i korumak için 220Ω'luk bir direnç gereklidir
e) pasif buzzer modülü
f) modül fanı
g) Ardunio
h) çay ışığı



Pinout devresi

Pin 11 →buzzer

Pin 10 →Kırmızı led

Pin4 →INA

Pin 5 →INB

Not: Devredeki yeni bileşen, dört pini olan fan modülüdür: Vcc , Gnd , INA ve INB. 5 voltluk bir voltaj Vcc'ye bağlanırken , toprak Gnd'ye bağlanır. INA ve INB pinleri motor kontrolü için kullanılır. INA düşük ve INB düşük olduğunda motor durur. INA düşük ve INB yüksek olduğunda motor saat yönünde döner.

5.3 Kod:

```
intflameSens=A0;/// alev sensörü arduino pin 2'ye
intbuzzerPin= 11; //buzzer arduino pin 8'e
intledPin = 10; //LED arduino pin 8'e
int INA = 4; // motor kabuğu INA pinini arduino pin4'e
int INB = 5; // motor kabuğu INB pinini arduino pin5'e
int flame = 0; // sensör durumunu (değer) depolamak için değişken

voidsetup(){
  pinMode(flameSens,INPUT); // Alev sensörü çıkış pinini arduino'ya giriş olarak başlat
  pinMode(buzzerPin,OUTPUT); // buzzer pinini çıkış olarak başlat
  pinMode(ledPin,OUTPUT); // led pinini çıkış olarak başlat
  Serial.begin(9600); // seri iletişimi 9600 baud hızında başlat
  pinMode(INA,OUTPUT); // led pinini çıkış olarak başlat
  pinMode(INB,OUTPUT); // led pinini çıkış olarak başlat
}

voidloop() {
flame=analogRead(flameSens);
     Serial.println("flame arxh");
   Serial.println(flame); //sensör ölçüm ekranı

if(flame <100) // sensör değerini oku
{
 for(inti=1; i<=4; i++) // alarm 4 kez çalar
{
    digitalWrite(ledPin,HIGH);  //LED'i aç
   tone(buzzerPin,2400);   //buzzer 2400 KHz'de ses çıkarır
   delay (1000);
       digitalWrite(ledPin,LOW); //  LED'i kapat
    tone(buzzerPin,2900); //buzzer 2900 KHz'de ses çıkarır
delay (1000);
Serial.println("flame loop");
   Serial.println(flame);
}
   digitalWrite(INA,LOW); // fanı saat yönünde aç
  digitalWrite(INB,HIGH); // fanı saat yönünde aç
Serial.println("flame moter");
  Serial.println(flame);
  delay(4000);
}
digitalWrite(INA,LOW); //fanı saat yönünde aç
  digitalWrite(INB,LOW); // fanı saat yönünde aç
   noTone(buzzerPin);
  Serial.println("flametelos"); //sensör ölçüm ekranı
  Serial.println(flame);
}
```

Operasyon

Alev algılandığında alarm dört kez çalar (buzzer + LED), ardından fan 4 saniye boyunca çalışır.

6. Otomatik bitki sulama sistemi

Otomatik bitki sulama sistemi kullanmak, bahçenizle ilgilenirken su tasarrufu yapmanın etkili bir yoludur. Bu sistem, bitkilerin ihtiyaç duyduğu tam su miktarını sağlar; elle sulama veya bir sprinkler sistemi kullanmanın aksine, bu durum kolayca aşırı sulama veya yetersiz sulama ile sonuçlanabilir.

6.1Devre başlığı: Otomatik bitki sulama sistemi

Devre açıklaması : Bu devre toprak nem seviyesini ölçer. Nem seviyesi önceden belirlenmiş bir eşiğin altına düşerse, vana açılarak suyun bitkiye akmasına izin verir ve yeşil LED yanar. Nem seviyesi belirtilen eşiği aştığında, vana sulamayı durdurmak için kapanır ve mavi LED yanar.

İhtiyacınız olanlar:
a) Ekmek tahtası

b) nem sensörü (modüllü)

c) Led yeşil, led mavi

d) LED'i korumak için 220Ω direnç gereklidir.

e) solenoid valf

f) küçük su tankı

g)Arduino

h) bitkili küçük saksı

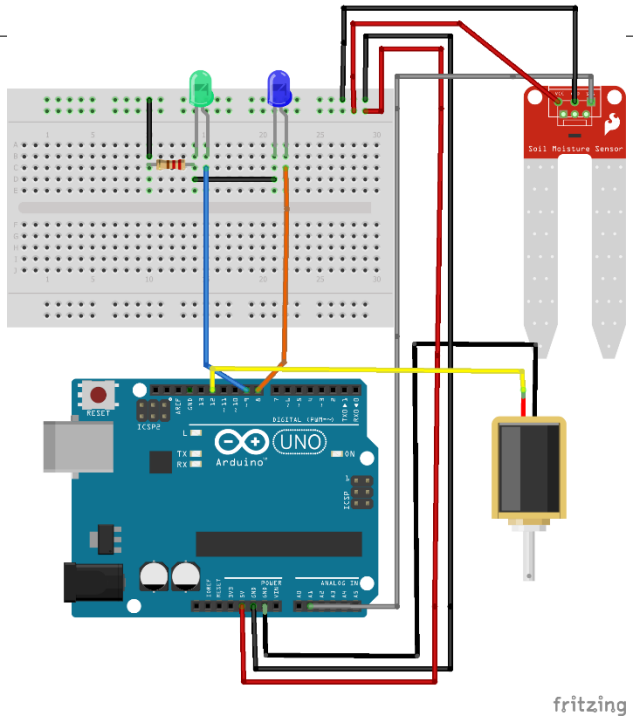
Pinout devresi

Pin 8 →led, yeşil

Pin 9 →mavi led

A1 →analog giriş nem sensörü

Pin 12 →valfi



6.3 Kod

```
// Nem Sensörü Arduino Kodu

intgreenLight= 9; //Pinleri tanımlama
intblueLight= 8; //Pinleri tanımlama
intmoistureLevel=A1; //Pinleri tanımlama
intvalvePin=10; //Pinleri tanımlama
intmaxMoistureLevel ;  //Maksimum nem seviyesi
intminMoistureLevel ;  //Maksimum nem seviyesi
intcurrentMoistureLevel ; // mevcut nem seviyesi


voidsetup() {
  pinMode ( blueLight , OUTPUT );  // pini çıkış olarak başlat
  pinMode ( greenLight , OUTPUT );  //9. pini çıkış olarak başlat
   pinMode ( nemSeviyesi ,GİRİŞ );  //A1 Toprak Nem Sensörü için kullanılan pindir
   pinMode ( vanaPin ,ÇIKIŞ );  //9. pini çıkış olarak başlat
Serial.begin(9600););//  seri iletişimi 9600 baud hızında başlat:
}
void   loop() {
   maxMoistureLevel = 700;  //değişkeni 700 olarak ayarla
   minMoistureLevel = 400; //değişkeni 400 olarak ayarla
   currentMoistureLevel = analogRead ( nemSeviyesi ); //Değişkene geçerli değeri atayın
  if ( currentMoistureLevel >1000) //nem seviyesi 1000'in altına düşerse yeşil ve mavi ışıklar
yanar
{
   digitalWrite ( greenLight , HIGH );  //Işığı aç
   digitalWrite ( blueLight , HIGH );  //Işığı aç
   digitalWrite ( valvePin , LOW );   //Işığı kapat
```

```
}
  elseif((currentMoistureLevel>minMoistureLevel)&&(currentMoistureLevel<1000)) //nem seviyesi %30 ile %60 arasındaysa
{
    digitalWrite(greenLight,HIGH);

   digitalWrite(blueLight,   LOW); // led ışığı açık
   delay(3000);
   digitalWrite(valvePin,HIGH); // vanayı aç
delay(500);
    digitalWrite(valvePin,LOW); // vanayı kapat
   delay(300);
    digitalWrite(greenLight,LOW); // led ışığı kapalı
}
 if (currentMoistureLevel>maxMoistureLevel){ //nem seviyesi %30'un altındaysa
   digitalWrite(blueLight,HIGH);
}

   delay(500);  //Kısa gecikme
Serial.println("MoistureLevel");
  Serial.println(currentMoistureLevel);
}
```

7. Güvenlik ışığı

Güvenlik ışıkları, genellikle bir binanın dışında bulunan ve birisi veya bir şey onlara yaklaştığında yanan elektrik ışıklarıdır. Hırsızlar erken saatlerde eve girmeye çalışırlar, ancak güvenlik ışıkları yandığında komşular onları fark edebilir. Bu sistem, bir parlaklık sensörü eklenerek daha verimli hale getirilmelidir ; ışık yalnızca çevredeki ışık düşük olduğunda yanacak ve enerji tasarrufu sağlayacaktır.

7.1 Devre başlığı: Güvenlik ışığı

Devre açıklaması: Bu devre iki sensöre sahiptir: bir hareket sensörü (PIR) ve bir parlaklık sensörü. İzleme alanının parlaklık seviyesi düşük olduğunda ve hareket algılandığında, ışık üç saniye boyunca yanacaktır.

İhtiyacınız olanlar:
a) Ekmek tahtası

b) fotodirenç

c) Modül Led

d)10K direnç

e) sensör PIR

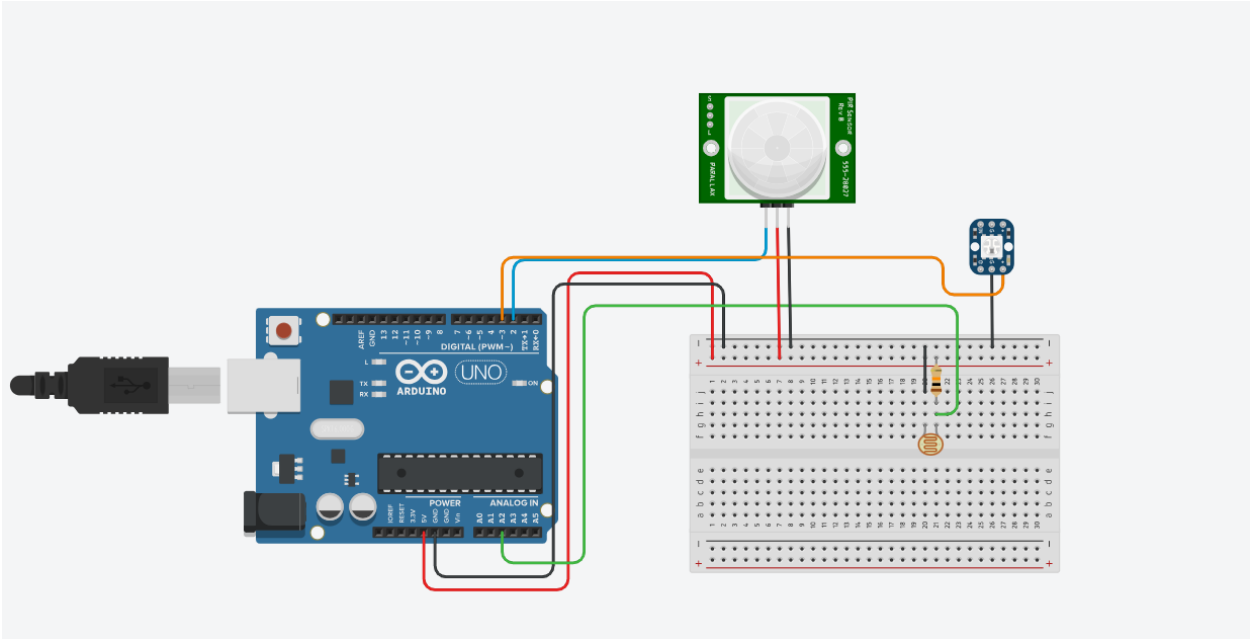
g) Arduino

Pinout devresi

Pin 2 →PIR girişi

Pin A2 →giriş parlaklık sensörü

Pin 3 →Led



7.3 Kod :

```
//Sabitler
constintpResistor= A2;   // Arduino analog pin A2'deki fotodirenç
constintledPin=3;        // Arduino pin 9'daki LED pin

//Değişkenler
int value;               // Fotodirençten gelen değeri depola (0-1023)

voidsetup(){
pinMode(ledPin,OUTPUT);  // lepPin - 9 pinini çıkış olarak ayarla
pinMode(pResistor,INPUT); //  pResistor - A0 pinini giriş olarak ayarla (isteğe bağlı)
Serial.begin(9600);      //arduino'nun dizüstü bilgisayarla iletişim kurma hızı
}

voidloop(){
value =analogRead(pResistor); //sensörden analog verileri okur

  if (value < 700){
digitalWrite(ledPin,LOW);  //Led'i kapat
}
else{
digitalWrite(ledPin,HIGH); //Ledi aç
}

  delay(500);              //Küçük gecikme
 Serial.println(value);       //seri monitörde yazdırır
}
```

8. Uygulama

Tüm sensör projeleri ilgili yapıya uygulanabilir. Bu yapının pinout'u, Bireysel projenin pinout'uyla aynıdır.

Pinout devresi

Pin A0 →alev sensörü

Pin 2 →PIR çıkışı

Pin 3 →Led modülü

Pin4 →INA motoru

Pin 5 →INB motoru

Pin 8 →led, yeşil

Pin 9 →mavi led

Pin 10 →Kırmızı led

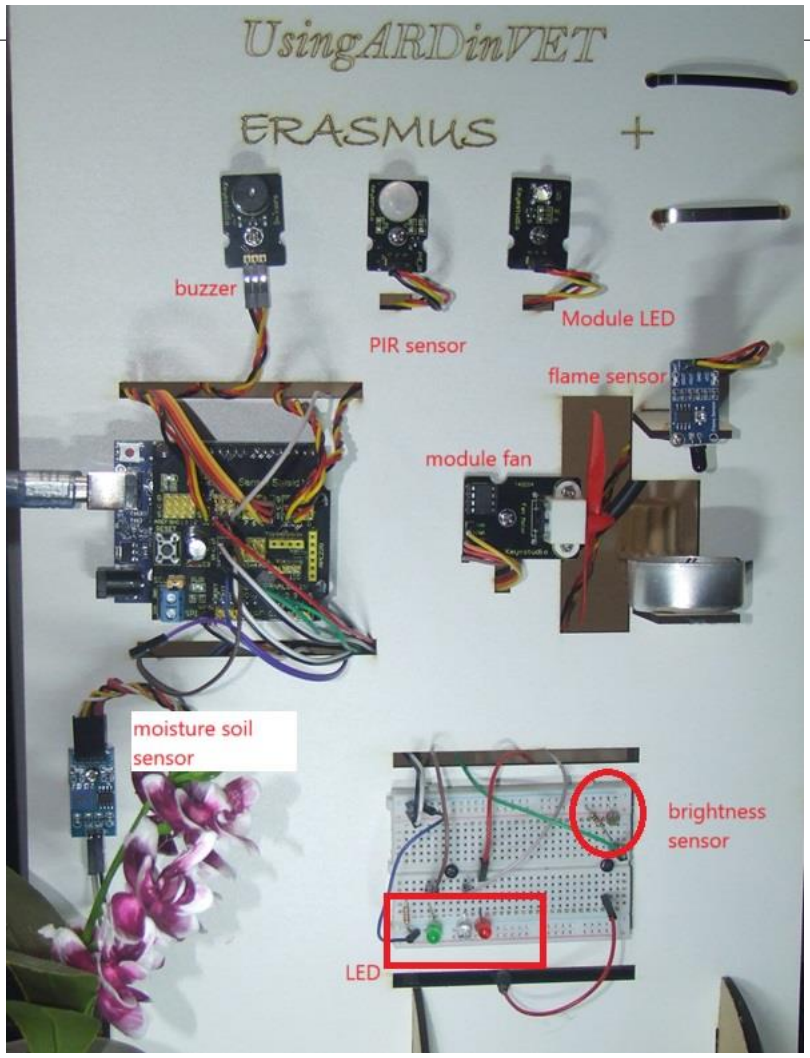
Pin 11 →buzzer

Pin 12 →valfi

Pin A0 →analog giriş alev sensörü

Pin A1 →analog giriş nem sensörü

Pin A2 →analog giriş parlaklık sensörü



Otomatik bitki sulama sistemi, aktif yangın koruma, güvenlik aydınlatması projelerini tek tek programlarını yüklemeye gerek kalmadan **aynı anda** çalıştırabilirsiniz .

```
// programın tamamı
intsensorPir= 2;              // Arduino pin 2'deki Sensör PIR
intledPin=3;                  // Arduino pin 9'daki LED pin
int INA=4;                    //  motor kabuğu INA pinini arduino pin4'e
int INB=5;                    //  motor kabuğu INB pinini arduino pin5'e
intblueLight= 8;              //Pinleri tanımlama
intgreenLight= 9;             //Pinleri tanımlama
intredLedPin=10;              //Pinleri tanımlama
intbuzzerPin= 11;             //Pinleri tanımlama
intvalvePin=10;               //Pinleri tanımlama
intflameSens=A0;              //alev sensörü arduino pin A0'a
intmoistureLevel=A1;          //nem sensörü arduino pin A1'e
intsensorPhotores= A2;        // Arduino analog pin A2'deki fotodirenç
intmaxMoistureLevel;          //maksimum nem seviyesi için değişken
```

```
intmaxMoistureLevel;                //minimum nem seviyesi için değişken
intcurrentMoistureLevel ;              // mevcut nem seviyesi için değişken
intvaluePhoto ;                  //fotodirençten gelen değeri depolamak için değişken (0-1023)
intvaluePir ;                    // sensör PIR'sinden (0-1023) gelen değeri depolamak için değişken
int flame = 0;                   // sensör durumunu (değer) depolamak için değişken

voidsetup() {
    pinMode(ledPin,OUTPUT);       // lepPin'i çıkış olarak ayarla
    pinMode(sensorPhotores,INPUT);// sensörPhotores pinini giriş olarak ayarla
    pinMode(sensorPir,INPUT);     // sensörPhotores pinini giriş olarak ayarla
    pinMode(flameSens,INPUT);     //Alev sensörü çıkış pinini arduino'ya giriş olarak başlat .
    pinMode(buzzerPin,OUTPUT);   // buzzer pinini çıkış olarak başlat
    pinMode(redLedPin,OUTPUT);   // kırmızılaştırılmış pini çıkış olarak başlat
    pinMode(INA,OUTPUT);         // INA fan pinini çıkış olarak başlat
pinMode(INB,OUTPUT);         //INB fan pinini çıkış olarak başlat
    pinMode(blueLight,OUTPUT);    // mavi ışık pinini çıkış olarak başlat
    pinMode ( greenLight , OUTPUT ); // yeşil ışık pinini çıkış olarak başlat
    pinMode(moistureLevel,INPUT);//A1 Toprak Nem Sensörü için kullanılan pindir
    pinMode(valvePin,OUTPUT);      //9. pini çıkış olarak başlat
    maxMoistureLevel=700;          //değişkeni 700 olarak ayarla
    minMoistureLevel=400;          //değişkeni 400 olarak ayarla
Serial.begin(9600);            // seri iletişimi 9600 baud hızında başlat:
}
voidsecurityLight(){                  //güvenlik ışığı için yöntem
   digitalWrite(ledPin,HIGH);              //  LED'i aç
   delay(5000);
   digitalWrite(ledPin,LOW);               //  LED'i kapat
}
void flameSens1(){                     //ateş için yöntem
   for(inti=1; i<=4; i++)              // alarm sesi 4 kez
{
      digitalWrite(redLedPin,HIGH);    //  LED'i aç
      tone(buzzerPin,2400);           //buzzer 2400 KHz'de ses çıkarır
      delay (1000);
      digitalWrite ( redLedPin , LOW ); // LED'i kapat
      tone (buzzerPin , 2900); //buzzer 2900 KHz'de ses çıkarır
      delay (1000);
Serial.println("flame loop");    //'alev' değişkenini görüntüle
Serial.println(flame);
}
      digitalWrite(INA,LOW);          // fanı saat yönünde aç
      digitalWrite(INB,HIGH);         // fanı saat yönünde aç
      noTone(buzzerPin);              //  buzzer'ı durdur
      delay(4000);
      digitalWrite(INA,LOW);          // fanı kapat
      digitalWrite(INB,LOW);          // fanı kapat

}
void watering(){                      //sulama yöntemi
    if (currentMoistureLevel>1000)        //eğer nem sensörü yerde değilse
```

```
{
        digitalWrite ( greenLight , HIGH ); //Işığı aç, sensörün yerde olmadığını gösterir
        digitalWrite ( blueLight , HIGH ); //Işığı aç/kapat , sensörün yerde olmadığını gösterir
        digitalWrite(redLedPin,LOW);      //Işığı kapat , sensörün yerde olmadığını gösterir
}
    elseif((currentMoistureLevel>minMoistureLevel)&&(currentMoistureLevel<1000))//eğer nem seviyesi min seviyenin altındaysa
{
        digitalWrite(greenLight,HIGH);    // led ışığı açık
        digitalWrite(blueLight,   LOW);    // led ışığı kapalı
        delay(3000);
        digitalWrite(redLedPin,HIGH);      // vanayı aç
        delay(500);
        digitalWrite(redLedPin,LOW);       // vanayı kapat
        delay(300);
        digitalWrite(greenLight,LOW);      // led ışığı kapalı
}
          if (currentMoistureLevel>maxMoistureLevel){     //nem seviyesi minimum seviyenin üzerindeyse
       digitalWrite ( blueLight , HIGH ); // led ışığı açık
}

      Seri . println ( " MoistureLevel " ); // nemi görüntüle
      Seri . println ( currentMoistureLevel );
}



voidloop() {
  valuePhoto=analogRead(sensorPhotores);       //Değişkene geçerli değeri atayın
   valuePir=digitalRead(sensorPir);             //Değişkene geçerli değeri atayın
flame=analogRead(flameSens);                   //Değişkene geçerli değeri atayın
  currentMoistureLevel=analogRead(moistureLevel); //Değişkene geçerli değeri atayın
if (valuePhoto> 800&&valuePir==HIGH)         //karanlık ve hareket varsa
{
 securityLight();
}
   if(flame <450)                            //eğer yangın varsa
{
   flameSens1();
}


 if((currentMoistureLevel>minMoistureLevel)&&(currentMoistureLevel<1000)) //eğer nem seviyesi minimum seviyenin altındaysa ve //eğer nem sensörü yerde ise
{
watering();
}
   Serial.println("valuePhoto");          // valuePhoto ' değişkenini görüntüle
  Serial.println(valuePhoto);             // valuePhoto ' değişkenini görüntüle
  Serial.println("flame");                // valuePhoto ' değişkenini görüntüle
```

```
  Serial.println(flame);                    //  valuePhoto ' değişkenini görüntüle
  delay(500);
}
```

Ekler:

Co-funded by the
Erasmus+ Programme
of the European Union

Daha fazla bilgi için lütfen proje web sitemizi ziyaret edin:

www.ardinvet.com

"Bu proje Erasmus+ Programı kapsamında Avrupa Komisyonu tarafından desteklenmektedir. Ancak burada yer alan görüşlerden Avrupa Komisyonu sorumlu tutulamaz."

“This project is Funded by the Erasmus+ Program of the European Union. However, European Commission cannot be held responsible for any use which may be made of the information contained there in”

KAYNAKÇA:

1. Arduino: Arduino programlama, Arduino rehber kitabı, ipuçları, püf noktaları ve daha fazlası dahil olmak üzere yeni başlayanlar için en iyi Arduino rehberi! Craig Newport;

2. Mikrodenetleyicilere ve programlanabilir mantık denetleyicilerine giriş - Yazar Viorel-Constantin Petre, Yayımcı: Matrixrom Publishing House;

3. Arduino: Arduino ile Başlarken ve Projelerle Temel Programlama (Arduino Programlamayı Öğrenmek İçin Gelişmiş Yöntemler) - Ernest Leclerc;

4. Arduino Programlama – Editura Nelly B.L. International Consulting LTD., Mart 2020.