Arduino nedir?

Arduino, kullanımı kolay donanım ve yazılıma dayalı açık kaynaklı bir elektronik platformudur. Arduino kartları girdileri okuyabilir - bir sensördeki ışık, bir düğmedeki parmak veya bir Twitter mesajı - ve bunu bir çıktıya dönüştürebilir - bir motoru etkinleştirebilir, bir LED'i açabilir, çevrimiçi bir şeyler yayınlayabilir. Kart üzerindeki mikrodenetleyiciye bir dizi talimat göndererek kartınıza ne yapması gerektiğini söyleyebilirsiniz. Bunu yapmak için Arduino programlama dilini (Wiring tabanlı) ve Processing tabanlı Arduino Yazılımını (IDE) kullanırsınız .

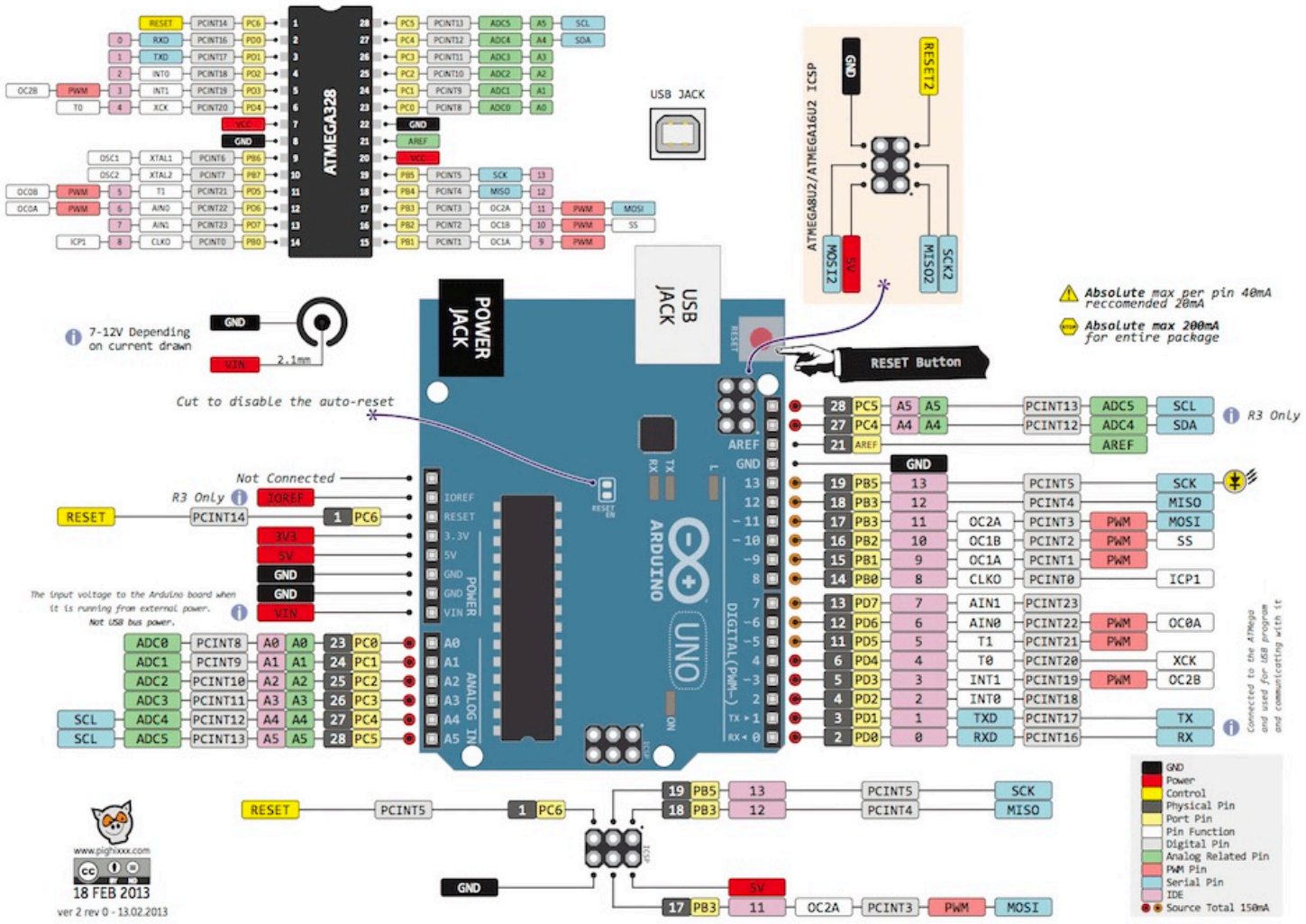
Arduino, yıllar boyunca günlük nesnelerden karmaşık bilimsel araçlara kadar binlerce projenin beyni olmuştur. Öğrenciler, amatörler, sanatçılar, programcılar ve profesyonellerden oluşan dünya çapındaki bir yapımcı topluluğu bu açık kaynak platformu etrafında toplandı ve katkıları, hem acemiler hem de uzmanlar için çok yardımcı olabilecek inanılmaz miktarda erişilebilir bilgi sağladı.

Arduino, Ivrea Etkileşim Tasarımı Enstitüsü'nde, elektronik ve programlama geçmişi olmayan öğrencilere yönelik, hızlı prototipleme için kolay bir araç olarak doğdu. Daha geniş bir topluluğa ulaşır ulaşmaz, Arduino kartı yeni ihtiyaçlara ve zorluklara uyum sağlamak için değişmeye başladı ve teklifini basit 8 bitlik kartlardan IoT uygulamalarına, giyilebilir ürünlere, 3D baskıya ve gömülü ortamlara kadar farklılaştırdı.

Neden Arduino?

Arduino, basit ve erişilebilir kullanıcı deneyimi sayesinde binlerce farklı proje ve uygulamada kullanılmıştır. Arduino yazılımı, yeni başlayanlar için kullanımı kolay, ileri düzey kullanıcılar için ise yeterince esnektir. Mac, Windows ve Linux üzerinde çalışır. Öğretmenler ve öğrenciler, düşük maliyetli bilimsel araçlar oluşturmak, kimya ve fizik ilkelerini kanıtlamak veya programlama ve robot bilimine başlamak için kullanırlar. Tasarımcılar ve mimarlar etkileşimli prototipler oluşturur, müzisyenler ve sanatçılar bunu enstalasyonlar ve yeni müzik enstrümanları ile deneyler yapmak için kullanır. Yapımcılar, örneğin, Yapıcı Fuarı'nda sergilenen projelerin birçoğunu inşa etmek için kullanırlar. Arduino, yeni şeyler öğrenmek için önemli bir araçtır. Herkes - çocuklar, hobiler, sanatçılar, programcılar - bir kitin adım adım talimatlarını izleyerek tamir etmeye başlayabilir.
Fiziksel bilgi işlem için başka birçok mikrodenetleyici ve mikrodenetleyici platformu mevcuttur. Parallax Basic Stamp, Netmedia'nın BX-24'ü, Phidgets, MIT'nin Handyboard'u ve daha birçokları benzer işlevler sunar. Tüm bu araçlar, mikrodenetleyici programlamanın karmaşık ayrıntılarını alır ve kullanımı kolay bir pakette toplar. Arduino ayrıca mikrodenetleyicilerle çalışma sürecini basitleştirir, ancak öğretmenler, öğrenciler ve ilgilenen amatörler için diğer sistemlere göre bazı avantajlar sunar:

- **Ucuz** - Arduino kartları, diğer mikrodenetleyici platformlarına kıyasla nispeten ucuzdur. Arduino modülünün en ucuz versiyonu elle monte edilebilir ve önceden monte edilmiş Arduino modüllerinin bile maliyeti 50 dolardan daha düşüktür.
- **Çapraz platform** - Arduino Yazılımı (IDE) Windows, Macintosh OSX ve Linux işletim sistemlerinde çalışır. Çoğu mikrodenetleyici sistemi Windows ile sınırlıdır.
- **Basit, anlaşılır programlama ortamı** - Arduino Yazılımı (IDE) yeni başlayanlar için kullanımı kolay, ileri düzey kullanıcıların da yararlanabileceği kadar esnektir. Öğretmenler için İşleme programlama ortamına dayalıdır, bu nedenle bu ortamda programlamayı öğrenen öğrenciler Arduino IDE'nin nasıl çalıştığına aşina olacaktır.
- **Açık kaynak ve genişletilebilir yazılım** - Arduino yazılımı, deneyimli programcılar tarafından genişletilmek üzere açık kaynak araçları olarak yayınlanır. Dil, C++ kitaplıkları aracılığıyla genişletilebilir ve teknik ayrıntıları anlamak isteyen kişiler, Arduino'dan temel aldığı AVR C programlama diline geçiş yapabilir. Benzer şekilde, isterseniz AVR-C kodunu doğrudan Arduino programlarınıza ekleyebilirsiniz.
- **Açık kaynak ve genişletilebilir donanım** - Arduino kartlarının planları bir Creative Commons lisansı altında yayınlanır, böylece deneyimli devre tasarımcıları modülün kendi versiyonunu oluşturabilir, genişletebilir ve geliştirebilir. Nispeten deneyimsiz kullanıcılar bile , nasıl çalıştığını anlamak ve paradan tasarruf etmek için modülün devre tahtası sürümünü oluşturabilir .

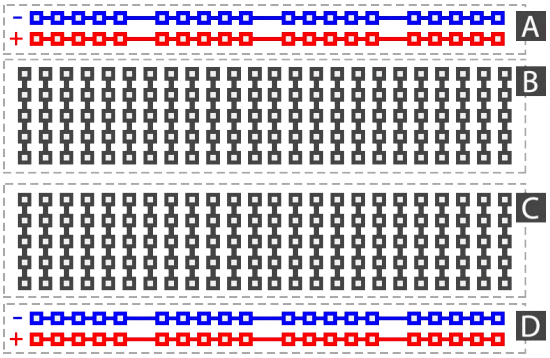

ATMEGA328
USB JACK
ATMEGA8U2/ATMEGA16U2 ICSP
POWER JACK
USB JACK
RESET Button
7-12V Depending on current drawn
Cut to disable the auto-reset
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
R3 Only
Not Connected
The input voltage to the Arduino board when it is running from external power. Not USB bus power.
Connected to the ATMega and used for USB program and communicating with it
ARDUINO
UNO
www.pighixxx.com
18 FEB 2013
ver 2 rev 0 - 13.02.2013
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA

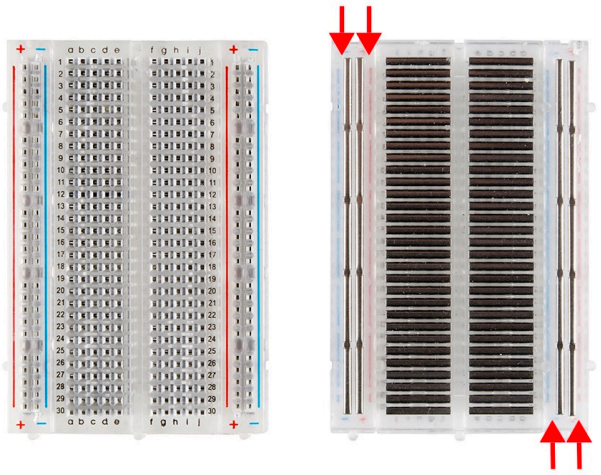
Breadboard Nedir?

Breadboard üzerinde devrelerimizi test ettiğimiz araçtır. Kurduğumuz devreleri birbirlerine lehimlemeden kolaylıkla test etmemizi sağlar. Tasarladığımız devreleri baskı devre veya delikli plaketler üzerine aktarmadan önce denememize olanak sağlar. Bu sayede devre bağlantılarını kontrol ederek bir hata olup olmadığını gözlemlemiş oluruz. Devreleri tak-çıkar şeklinde kurabildiğimiz için kullandığımız elektronik bileşenleri başka projelerde tekrar kullanma imkanı verir.

Breadboard İç Yapısı Nasıldır?

Breadboard iç yapısı dik ve yatay şekilde birbirlerine bağlı halde konumlanmış metal kıskaçlardan oluşur. Dışarıdan baktığımızda gördüğümüz kırmızı ve mavi kısımları breadboardun satır kısımlarıdır. Bu kısımlar boydan boya bir satır şeklinde iletim halindedir. Bu kısımları breadbordun iki yanında görebilirsiniz. Burada dikkat etmemiz gereken nokta bu satır bazı breadboardlarda ortadan ikiye ayrılmış durumdadır. Yani baştan sona kısa devre değildir.
Breadboardun ortada kalan kısımları da sütun boyunca yerleştirilmiş iletkenlerden oluşur. Bu kısımlar da tıpkı satırlarda olduğu gibi breadboardun iki tarafında bulunur. Tüm bu iletkenlerin üstü elektronik bileşenlerin ayaklarını yerleştirmemiz için açılmış deliklerden oluşan bir plastik ile kapalıdır.





Breadboard Nasıl Kullanılır?

Breadboard üzerindeki + ve – şeklinde belirtilmiş satırlar gerilim bağlantılarını sağlamak için kullanılır. Gerilim bağlantılarını sağladıktan sonra board üzerinde yerleştirilen bileşenlerin güç ihtiyaçları bu hatlar üzerinden karşılanır.
Orta kısımda 5 delikten oluşan sütunlar bulunmaktadır. Bu delikler aynı metal parçasının üzerinde konumlandığı için birbirleriyle iletim halindedir. Yani aynı sütun üstündeki deliklere bağlayacağımız bacaklar iletim durumuna geçer. Bu işlem sayesinde komponentleri birbirine bağlamış oluruz. Bunu iç yapısını incelediğimizde çok daha iyi anlayabiliyoruz. Board üzerinde bulunan kırmızı-mavi güç hattı ve sütunlar her iki tarafta da bulunmaktadır. Bu hatlar birbirinden ayrıktır, yani iletim halinde değillerdir.

Arduino

Serial Communication(Seri Haberleşme)

İki cihaz arasındaki çift yönlü haberleşme için kullanılır.

Seri iletişim, Mikrodenetleyici üzerindeki UART'ı (Universal asynchronous receiver transmitter - Evrensel Asenkron Alıcı/Verici) kullanan basit bir şemadır.

Tx(transmitter) -> verici Arduino->Computer

Arduinodan bilgisayara veri aktarımı için kullanılır.

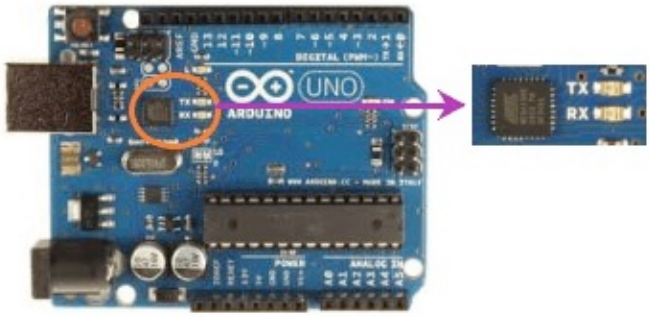
Rx(receiver)->Alıcı Computer -> Arduino

Bilgisayardan Arduinoya veri aktarımı için kullanılır.

Arduino UNO üzerinde; TX pini 1, RX pini 0 nolu pinlerdir.

- 5V for logic 1 (high)
- 0V for logic 0 (low)

UART üzerinde gönderilen her mesaj 8 bit veya 1 bayt şeklindedir, burada 1 bayt = 8 bittir.



Kodumuzda seri haberleşme için pinleri ilişkilendirdiğimizde bu iki pini (Rx ve Tx) herhangi bir amaçla kullanamıyoruz. Tx ve Rx pinleri de doğrudan bilgisayara bağlı oluyor.

Pimler, seriden USB'ye tercüman görevi gören seri Tx ve Rx çipine bağlıdır. Bilgisayarın Mikrodenetleyici ile konuşması için bir ortam görevi görür.

Serial

if (Serial)

Belirlenen Seri Portun hazır olup olmadığını kontrol etmemize yarar. Serial nesnelerinin mantıksal bir alan alanda kullanılması durumunda eğer belirtilen Seri Port hazır ise **True** eğer belirtilen Seri Port hazır değilse **False** değeri dönecektir.

Serial.begin()

serial.begin(), seri veri iletişimi için baud hızını ayarlar. Baud hızı, veri hızını saniyedeki bit cinsinden ifade eder. Arduino'daki varsayılan baud hızı 9600 bps'dir (saniyede bit). 4800, 14400, 38400, 28800 gibi diğer baud hızlarını da belirleyebiliriz.

Serial.print() ve Serial.println()

Seri porttan veri göndermek için 2 komut vardır. Serial.print(), Serial.write() komutlarıdır.
Yazdırılan veriler, araç çubuğunun sağ köşesinde bulunan seri monitörde görünür olacaktır.

- print(value)
- print(value, format)

sayıyı ve kayan noktalı sayılar için iki ondalık basamağa kadar olan değeri kabul eder.
İnput: Serial.print(15.452732)
Output: 15.45

Input: Serial.print(25, BIN)
Output: 11001

Serial

if (Serial)

Belirlenen Seri Portun hazır olup olmadığını kontrol etmemize yarar. Serial nesnelerinin mantıksal bir alan alanda kullanılması durumunda eğer belirtilen Seri Port hazır ise **True** eğer belirtilen Seri Port hazır değilse **False** değeri dönecektir.

available ()

Serial Port'tan okuma yapmak için kullanılabilir byte(karakter)'in olup olmadığını kontrol etmemize yarar. Gönderilmiş olan veriyi byte byte değerlendirir.

Serial.begin()

serial.begin(), seri veri iletişimi için baud hızını ayarlar. Baud hızı, veri hızını saniyedeki bit cinsinden ifade eder. Arduino'daki varsayılan baud hızı 9600 bps'dir (saniyede bit). 4800, 14400, 38400, 28800 gibi diğer baud hızlarını da belirleyebiliriz.

Serial.print() ve Serial.println()

Seri porttan veri göndermek için 2 komut vardır. Serial.print(), Serial.write() komutlarıdır.
Yazdırılan veriler, araç çubuğunun sağ köşesinde bulunan seri monitörde görünür olacaktır.

- print(value)
- print(value, format)

Serial.read()

Arduino'daki Serial.read(), Arduino'ya gelen seri verileri okur. Burada int veri tipi kullanılmıştır. Gelen seri verinin ilk veri baytını döndürür. Ayrıca, seri bağlantı noktasında hiçbir veri bulunmadığında -1 değerini döndürür.

Serial.write("veri")

Bu gönderme biçiminde ise veriler ikili formatta gönderilir. İkili formatta gönderim, ASCII formata göre biraz daha farklıdır. Ancak kullanım şekli print ile tamamen aynıdır.

Serial.write("veri");

Serial.available()

Bu komut tampon bellekte bulunan okunmamış verilerin saysını döndürür. Bellekte veri yok ise 0 döndürür. Tampon bellek boyutu sınırlı olduğu için gelen veriler belirli bir zaman aralığında okunmalıdır. Aksi takdirde gelen veriler eskilerin üzerine yazılır. Arduino'daki seri arabellek 64 baytı tutar

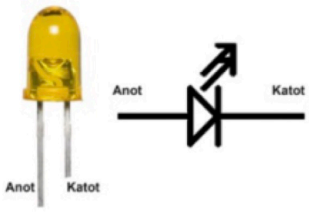
Örnek:

```
int arrivingdatabyte;
void setup( )  {
Serial.begin(9600);
}
void loop( )  {
if(Serial.available( ) > 0)  {
        arrivingdatabyte = Serial.read( );  // It will read the incoming or arriving data byte
        Serial.print("data byte received:");
        Serial.println(arrivingdatabyte);
    }
}
```

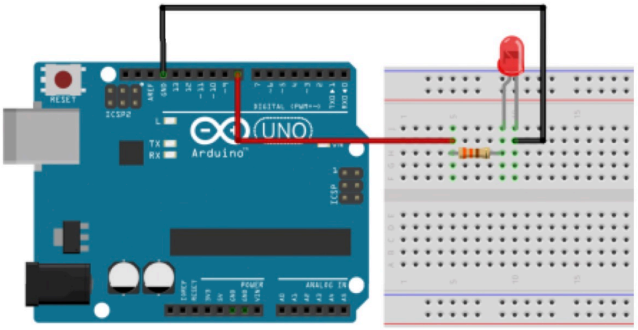
Yukarıdaki kod, Serial.available() öğesinin, 0'dan büyükse kullanılabilir bayt sayısını almak için kullanıldığını açıkça açıklar. Serial.read() işlevi, veri baytından verileri okuyacak ve veri varsa bir mesaj yazdıracaktır. alınır. Veriler seri monitörden Arduino'ya gönderilir.

LED

Led, ışık yayan diyot anlamına gelen “Light Emitting Diode” sözcüğünün baş harflerinden oluşan bir kısaltmadır. Led’lerin anot ve katot olmak üzere iki farklı bacağı vardır. Bunlardan anot pozitif(+) gerilime, katot ise negatif(-) gerilime ya da toprak hattana (GND, Ground) bağlanmalıdır.



Kırmızı	2 Volt	15mA
Sarı	2.1 Volt	15mA
Yeşil	2.2 Volt	15mA
Mavi	3.3 Volt	20mA
Beyaz	3.3 Volt	20mA



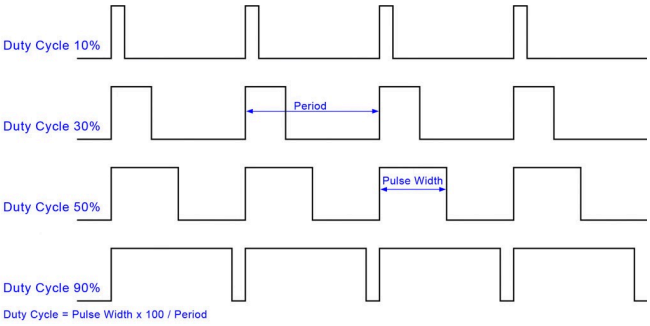
LED’lerin ideal çalışma voltajları ve ihtiyaç duydukları akım farklıdır. Bunlara göre devrenin direnç hesabı OHM yasası kullanılarak hesaplanmalı ve uygun direnç devreye bağlanmalıdır.

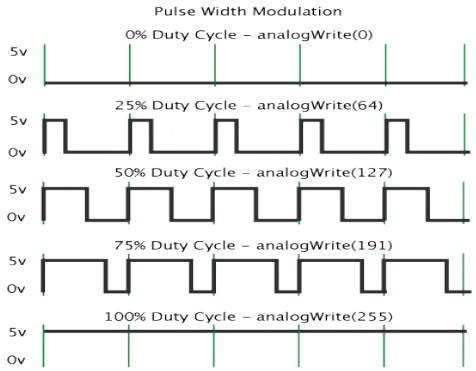
PWM(Pulse Width Modulation - Sinyal Genişlik Modülasyonu)

PWM (Pulse Width Modulation), dijital sinyalleri kullanarak analog sinyalleri temsil etmek için yaygın olarak kullanılan bir yöntemdir. PWM, bir dalganın darbe genişliğini değiştirerek ortalama voltajı kontrol etmek için kullanılır.

PWM'nin çalışma prensibi basittir: Bir sabit frekansta çalışan bir kaynak sinyali alırız (genellikle bir kare dalga). Bu kaynak sinyali, belirli bir görev döngüsü oranına (duty cycle) sahip olan darbelerden oluşur. Görev döngüsü oranı, bir darbenin yüksek seviyede (genellikle tam yüksek gerilimde) ne kadar süre kaldığını ifade eder. Diğer bir deyişle, görev döngüsü oranı, darbenin yüksek seviye süresini toplam periyot süresine bölerek elde edilir.

Duty Cycle(**Görev döngüsü):** Bir yükün ya da devrenin AÇIK olduğu sürenin, yükün ya da devrenin KAPALI olduğu süreyle kıyaslandığı oransal bir ifadedir.



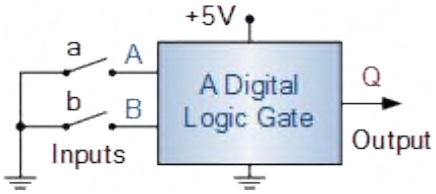


Örneğin, bir PWM sinyali için %50 görev döngüsü oranı kullanırsak, darbenin yüksek seviyede olduğu süre, toplam periyot süresinin yarısı olacaktır. Eğer görev döngüsü oranı %25 ise, darbenin yüksek seviyede olduğu süre toplam periyot süresinin dörtte biri olacaktır.

Arduino'da PWM'nin çalışma periyodu (frekansı) sabittir ve belirli bir değere sahiptir. Arduino UNO ve benzeri modellerde, varsayılan PWM frekansı 490 Hz'dir (Periyodu 2.04081633ms). Bu frekans, analogWrite() fonksiyonuyla PWM çıkışı oluşturulduğunda kullanılır. Arduino Uno'da 3, 5, 6, 9, 10 ve 11 numaralı pinler PWM özelliklerine sahiptir.

Pull Up ve Pull Down Direnç

Mantık kapıları, günümüzde dijital elektronikte yoğun bir şekilde kullanılmaktadır. Bu lojik kapıların Giriş (INPUT) ve çıkışlarının (OUTPUT) doğru sinyali durumunu ve beklenen anahtarlama koşulunu sağladığından emin olmak için özen gösterilmelidir. dijital devrenin düzgün çalışması için yüksek veya düşük olarak doğru şekilde ayarlanması gerekir.
Dijital olan devreler, mantık “0” veya mantık “1” durumu olarak adlandırılan iki mantık durumundan yalnızca birine sahip olabilir.
Bu lojik durumları, belli bir seviyenin altındaki herhangi bir voltajın “0” mantığı olarak kabul edildiği ve başka bir seviyenin üzerindeki herhangi bir voltajın “1” mantığı olarak kabul edildiği iki farklı voltaj seviyesi ile temsil edilir. Bazen bir dijital mantık kapısında veya devresinde girişler, mantık “0” veya mantık “1” girişi olarak algılanabileceği aralıkta olmayabilir. Bu durumda kapı veya devre doğru giriş değerini tanımadığı için dijital devre yanlış tetiklenebilir. Çıkışın “1” olması gereken durumda “1” veya “0” olması gereken durumda “0” olmayabilir.

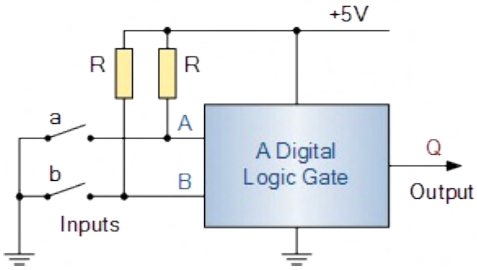


Örneğin, yandaki dijital devred, iki anahtar,” a “ve” b”, genel bir mantık kapısına girişleri temsil eder. “A” anahtarı kapalıyken, “A” girişi toprağa, (0v) veya “0” mantık seviyesine (düşük) bağlanır. Aynı şekilde “b” anahtarı kapalıyken, “B” girişi de toprağa bağlanır, “0” mantık seviyesi (düşük) ve bu bizim ihtiyacımız olan doğru durumdur.

Bununla birlikte, “A” anahtarı açıldığında, “A”, girişe uygulanan voltajın değeri ne olacaktır? “A” anahtarının açık devre olduğu ve bu nedenle “A” girişinin toprağa kısa devre yapmadığı için +5V (yüksek) olacağını varsayıyoruz ancak durum böyle olmayabilir. Giriş artık tanımlanmış bir yüksek veya düşük koşuldan etkin bir şekilde bağlanmadığından, girişin herhangi bir voltaj seviyesinde kalması mümkün olmayabilir. Bu durumda devre çalışırken sıkıntı oluşturabilmektedir. İşte bu noktada PULL-UP ve PULL-DOWN dirençleri hayatlarımızı kurtarmaktadır. Bu dirençleri devreye bağladıktan sonra kesin bir çıkış sinyal değeri elde etmek mümkün bir hale gelmektedir.

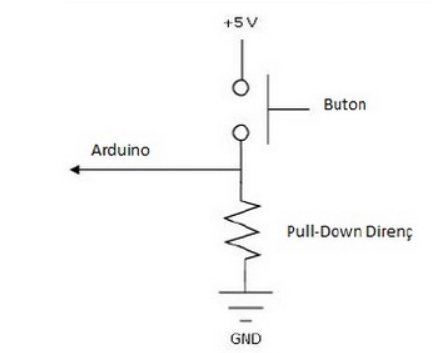
Pull-up Direnci

Dijital mantık kapılarının ve devrelerinin girişlerinin kendi kendine yer değiştiremediğinden ve yüzemediğinden emin olmanın en yaygın yöntemi, kullanılmayan pimleri sabit bir düşük “0” girişi için doğrudan toprağa (0V) bağlamaktır.veya ve nor kapıları) veya sabit bir yüksek “1” girişi (ve ve NAND kapıları) için doğrudan Vcc’ye (+5V).



Bu iki pull-up dirençleri kullanılarak, giriş etkin bir pull-up direnç üzerinden +5 V beslemeye bağlıdır. Girişler de diren aracılığıyla 5V’ye bağlıdır. Anahtar kapandığı anda giriş toprağa bağlanacaktır. Diğer taraftan da akım geçişi neredeyse hiç olmayacak kadar azalacağından (Ohm Yasası ile belirlendiği gibi) bu sayede bu durum en iyi şekilde önlenecektir. “A” veya “B” anahtarları kapatıldığında, (açık) giriş, girişte daha önce olduğu gibi bir mantık “0” koşulu oluşturarak toprağa (düşük) kısa devre yapar. Vcc ve giriş (veya çıkış) arasındaki bağlantı, bir pull-up direnci kullanmak için tercih edilen yöntem olsa da aklımıza gelen bir diğer soru ise gerekn direncin değerinin ne olduğudur.

Pull-Down Direnci

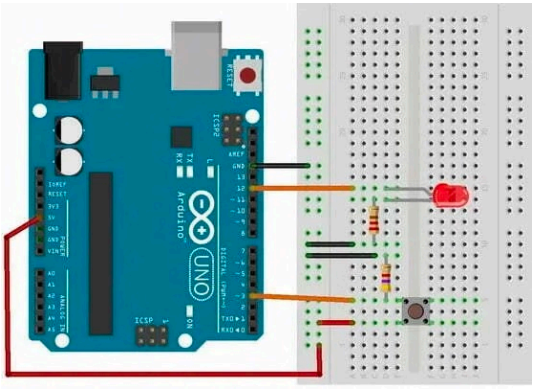


Pull down şeması yandaki gibidir. Butona basıldığında gerilim kaynağıyla Arduino’nun girişi kısa devre olur. Buton eski haline döndüğünde hat üzerinde hâlâ enerji kalır. Bu enerji butona basılmasa da devrenin belirsiz çalışmasına sebep olur. Bu enerjinin yok edilmesi için hat pull down direnciyle toprağa bağlanır.

Arduino Pull-Up ve Pull-Down Direnç Bağlama

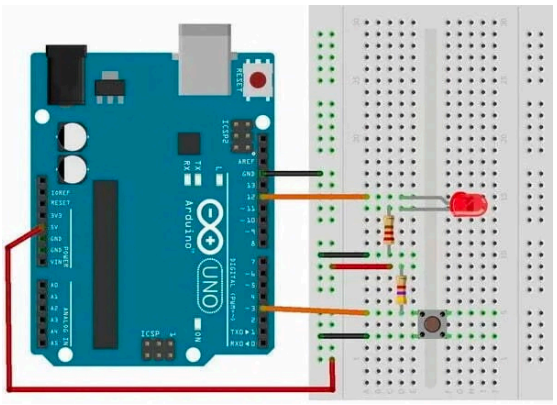
Pull Down Direnç

Düğmeye basıldığında gerilim kaynağıyla Arduino'nun girişi kısa devre olur. Elinizi düğmeden çektiğinizde hat üzerinde hâlâ enerji kalır. Bu enerji düğmeye basılmadığı durumunda bile Arduino'nun düğmeye basılmış gibi davranmasına neden olur. Bu enerjinin yok edilmesi için hat genellikle 10K ohm'luk bir direnç ile toprağa bağlanır. Bu dirence pull down direnç denir.



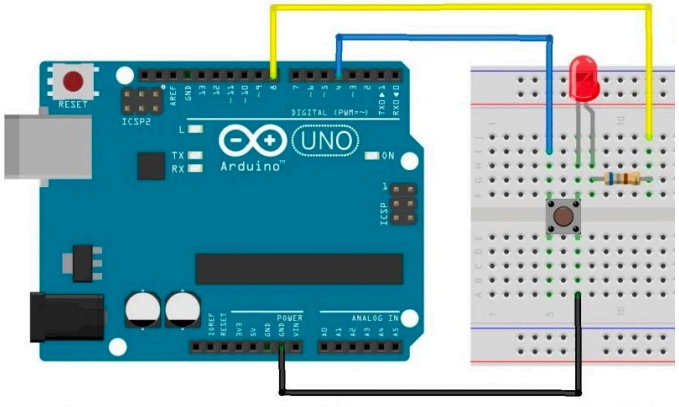
Pull-Up Direnç

Düğmeye basılmadığı durumlarda Arduino'nun giriş pini 5 volt düzeyindedir. Düğmeye basıldığında akım, Arduino'nun giriş pini yerine doğrudan toprağa ulaşmaktadır. Böylece pull-down direnç sistemini tam tersi çalışmaktadır. Arduino düğmeye basıldığında 0, düğmeye basılmadığında 1 değerini görmektedir. Pull-up direnci kullanma amacımız ise, düğmeye basıldığında toprak ve besleme hattının direkt olarak kısa devre olmasını engellemektir. Pull-down dirençte olduğu gibi pull-up dirençlerde genellikle 10K ohm olur.



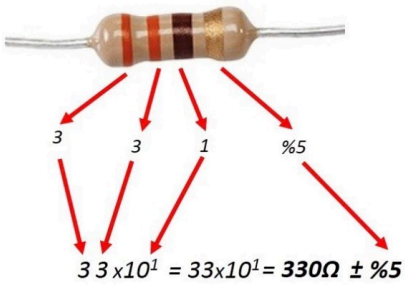
Arduino Dahili Pull-Up Direnci

Bu fonksiyonu kullanmak için pinMode() fonksiyonunda INPUT_PULLUP kullanmamız gerekiyor, Arduino ve AVR çiplerin de 20k ohm ve 50k ohm arasında pull-up direnci bulunur, bu sayede direnç kullanmadan aynı işlemi yapabiliriz.



Direnç Renk Kodları:

Renk	1. band	2. band	3. band (çarpan)	4. band(tolerans)	Sıcaklıkla değişim Katsayısı
Siyah	0	0	$\times 10^0$		
Kahverengi	1	1	$\times 10^1$	±1% (F)	100 ppm
Kırmızı	2	2	$\times 10^2$	±2% (G)	50 ppm
Turuncu	3	3	$\times 10^3$		15 ppm
Sarı	4	4	$\times 10^4$		25 ppm
Yeşil	5	5	$\times 10^5$	±0.5% (D)	
Mavi	6	6	$\times 10^6$	±0.25% (C)	
Mor	7	7	$\times 10^7$	±0.1% (B)	
Gri	8	8	$\times 10^8$	±0.05% (A)	
Beyaz	9	9	$\times 10^9$		
Altın			$\times 10^{-1}$	±5% (J)	
Gümüş			$\times 10^{-2}$	±10% (K)	
Yok				±20% (M)	



SoKaKTa SaYaMaM GiBi Ama Görürüm..

Arduino Advance I/O Fonksiyonları

1. tone()
2. noTone()
3. pulseIn()
4. pulseInLong()
5. shiftIn()
6. shiftOut()

1.Tone()

Bir pin üzerinde belirtilen frekansta (ve %50 görev döngüsünde) bir kare dalga üretir. Bir süre belirtilebilir, aksi halde dalga noTone() çağrılana kadar devam eder. Pim, sesleri çalmak için bir piezo ziline veya başka bir hoparlöre bağlanabilir.
tone() işlevinin kullanılması, pin 3 ve 11'deki (Mega dışındaki kartlarda) PWM çıkışını engelleyecektir.
31 Hz'den daha düşük tonlar üretmek mümkün değildir.

Syntax

tone(pin, frekans)
tone(pin, frekans, sure)

Parameters

pin: tonun oluşturulacağı Arduino pini.
frekans: Tonun hertz cinsinden frekansı. İzin verilen veri türü: unsigned int.
duration: milisaniye olarak tonun süresi . Allowed data types: unsigned long.

2.noTone()

tone() tarafından tetiklenen bir kare dalganın üretimini durdurur. Ton üretilmiyorsa etkisi yoktur

Syntax

noTone(pin)

3.pulseIn()

pulseIn() fonksiyonu belirtilen bir pim üzerine uygulanan bir darbenin kaç mikrosaniye süre ile HIGH ya da LOW durumunda kaldığını belirlemek için kullanılır. Geriye unsigned long türünde mikrosaniye cinsinden değer döndürür.
Örnek olarak HIGH parametresi almış ise, pin üzerine HIGH geldiğinde zamanlayıcıyı çalıştırır. Pulse değeri LOW olunca zamanlayıcıyı durdurur ve HIGH sinyalden LOW sinyale geçme süresini mikrosaniye cinsinden döndürür. Eğer timeout ile belirtilen süre içerisinde pulse sinyali konum değiştirmemiş ise geriye değer olarak 0 döndürür.
Bu işlevin zamanlaması ampirik olarak belirlenmiştir ve muhtemelen daha uzun darbelerde hatalar gösterecektir. 10 mikrosaniyeden 3 dakikaya kadar olan darbelerde çalışır.

Syntax

pulseIn(pin, value)
pulseIn(pin, value, timeout)

Parameters

pin: the number of the Arduino pin on which you want to read the pulse. Allowed data types: int.
value: type of pulse to read: either HIGH or LOW. Allowed data types: int.
timeout (optional): the number of microseconds to wait for the pulse to start; default is one second. Allowed data types: unsigned long.

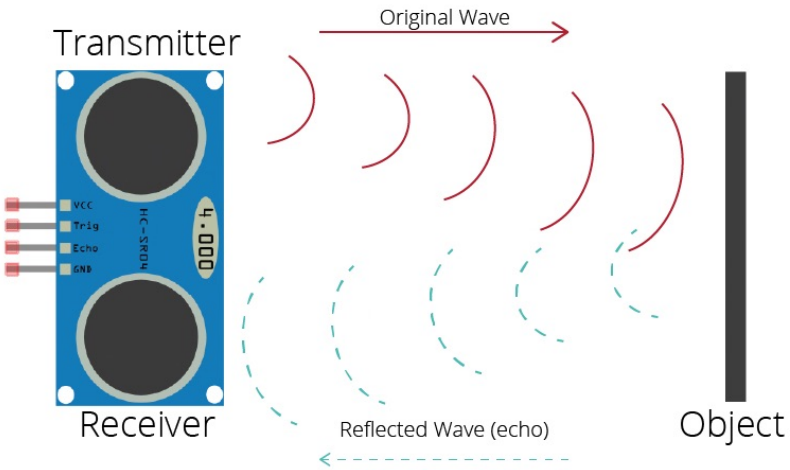
HC-SR04 ULTRASONİK MESAFE SENSÖRÜ

HC-SR04 ultrasonik sensör, bir nesneye olan mesafeyi belirlemek için sonar olarak kullanır. Bu sensör 0,3 cm hassasiyetle 2 cm ila 400 cm arasında okuma yapar. Ek olarak, bu özel modül ultrasonik verici ve alıcı modüllerle birlikte gelir.

Nasıl Çalışır?

Ultrasonik sensör, bir nesnenin mesafesini belirlemek için sonar kullanır. İşte ne olduğuyla ilgili bilgiler:

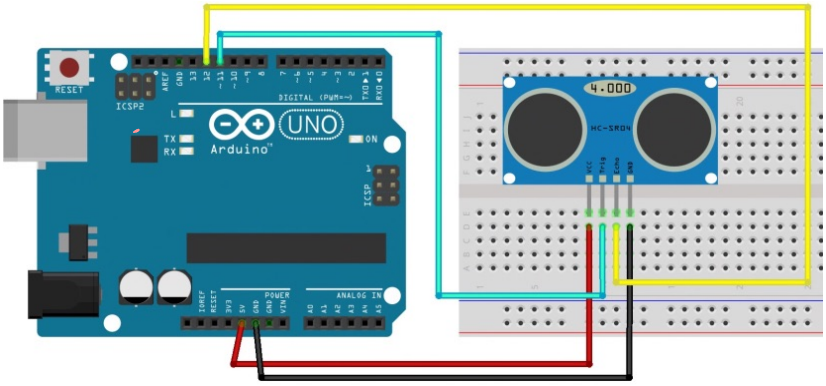
1. Ultrasonik sensör, elektrik sinyallerini ultrasonik ses dalgalarına dönüştüren bir vericiye(trig pin) sahiptir.
2. Verici, yüksek frekanslı ses dalgalarını nesneye doğru gönderir.
3. Ses dalgaları, nesneye çarpar ve yansır.
4. Yansıyan ses dalgaları, sensöre geri döner.
5. Alıcı(echo pin), geri dönen ses dalgalarını elektrik sinyallerine dönüştürür.
6. Sensör, geri dönme süresini hesaplar.



Sinyalin iletilmesi ve alınması arasındaki süre, bir nesneye olan mesafeyi sesin ortamdaki yayılma hızını kullanarak hesaplayabiliriz. Ses dalgaları nesneye ulaşır, nesneden sekip geri döndüğü için nesne ile sensör arasındaki mesafeyi 2 kez gitmiş olur. Bu yüzden süreyi 2 ye böleriz.

Formül: Mesafe = (Geçen Süre/2) * Ses Hızı

VCC	Powers the sensor (5V)
Trig	Trigger Input Pin
Echo	Echo Output Pin
GND	Common GND



TCP

TCP; bilgisayarlar arası iletişimde veri alışverişinin yanı sıra kimlik doğrulaması da sağlayan bir port türüdür.
Günümüzün bilgisayarlar arası iletişim protokollerinden en popüler olan HTTP, HTTPS, POP3, SSH, SMTP, TELNET ve FTP gibi protokoller TCP protokolünü kullanır ve her birinin kendine özel bir 0-65535 arası numarası bulunur.

UDP
UDP; ağırlıklı olarak ses ve video iletişiminde kullanılmak üzere geliştirilmiş bir port türüdür ve açılımı User Datagram Protocol olup Türkçe'ye Kullanıcı Veri Bloğu İletişim Kuralları olarak çevrilebilir.

- TCP, iletişim esnasında verileri paketler halinde ve sırayla gönderirken UDP'de bu yönetim akış sistemiyle sağlanır.

Port; bilgisayarlar arası iletişimi sağlayan ve bu iletişimde köprü görevi gören bir teknolojidir.
Port; **bilgisayar IP adreslerinin birden çok amaçla veri alışverişi yapabilmesini sağlar.**
Port, telefon numaraları örneğinden farklı olarak IP adresinin sonuna, ":" işaretinden sonra, **0'dan 65535'e kadar bir değerle ve her değer belli bir amacı temsil edecek şekilde** gelerek iki bilgisayar veya cihaz arası iletişimin kurulmasını sağlar.
Her bir amaca farklı bir port numarası atanarak aynı anda birden çok bağlantıya izin verilecek şekilde iletişim kurulmasını sağlayan port, bilgisayarlar arası iletişimin olmazsa olmazıdır.

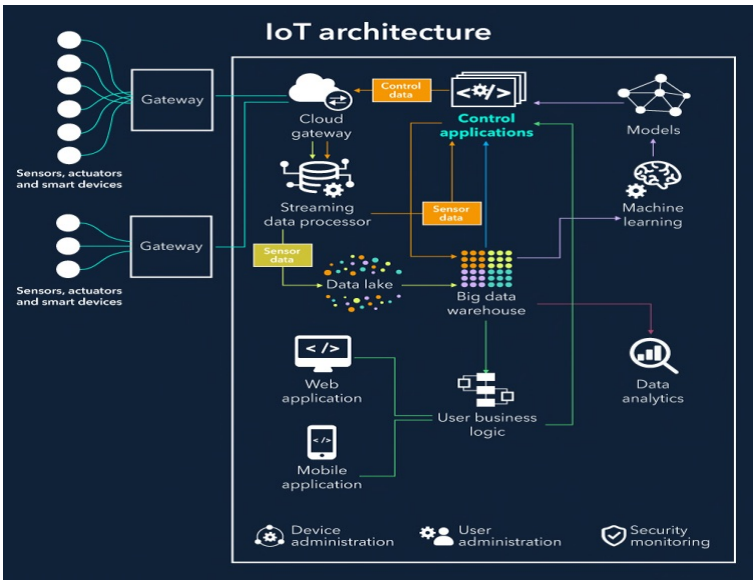
Portun kullanımı oldukça basittir. Önce bağlantı sağlanacak olan bilgisayarın IP adresi, daha sonra ise ihtiyaç duyulan iletişim protokolünün port numarası elde edilerek port kullanımı sağlanabilir.
192.168.1.1:12345

Bazı programlar; örneğin Filezilla gibi FTP istemcisi veya Putty gibi SSH/Telnet iletişimi sağlayan bir program, amacına uygun olan port numaralarını varsayılan olarak kendi kodları arasında içerir ve programların kullanılabilmesi için yalnızca IP adresini yazmak yeterli olur. Ancak varsayılan portlar her zaman geçerlidir diye bir kural yoktur çünkü port yönlendirme diye bir olay söz konusudur!

Port yönlendirme; bir iletişim protokolünün yönünü yani port numarasını değiştirmek veya bu iletişim protokolünü durdurmak üzere yapılan bir işlemdir. Adında her ne kadar yönlendirme terimi geçse de aslında bu işleme *port kontrolü* demek daha doğru olacaktır.
Port yönlendirme ile bir IP adresinin açık olan portları kapatılabilir, kapalı olan portları açılabilir veya açık olan portlar siber güvenlik gerekçesi ile farklı bir port numarası ile değiştirilebilir.

IoT

IoT mimarisi, sensör tarafından oluşturulan cihaz verilerinin büyük veri ambarında toplanmasını, depolanmasını ve işlenmesini ve cihazların aktüatörlerinin bir kullanıcı uygulaması aracılığıyla gönderilen komutları gerçekleştirmesini sağlamak için birbirine bağlı birkaç IoT sistem yapı taşından oluşur.



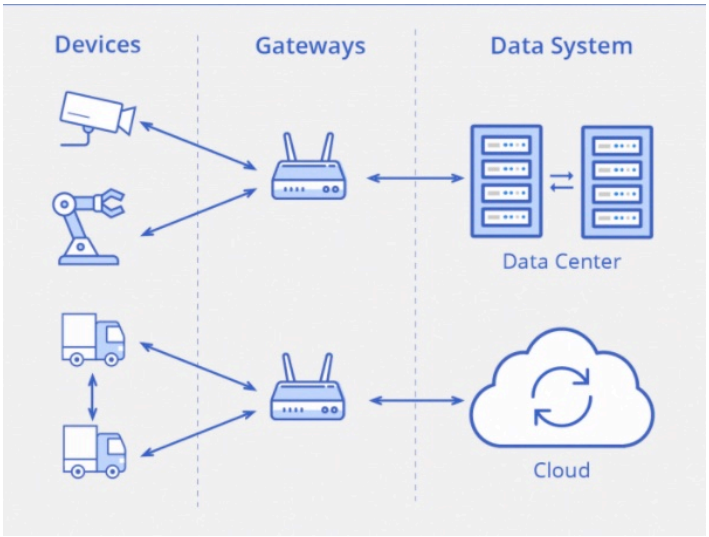
Detaylı IoT makalesi

Sensörler veriyi toplar. ağ üzerinden toplanan veriyi platformlarla paylaşabilir. Platformlarda oluşturulan senaryolara göre davranış sergileyebilir. Yada biz ağ üzerinden servera istek atıp
Thingspeak.com ya da Arduino cloud IoT kullanılabilinir.
IoT projeleri oluşturmanızı, dağıtmanızı ve izlemenizi kolaylaştıran çevrimiçi bir platformdur.

Ağ geçitleri (gateway). Veriler, ağ geçitleri aracılığıyla nesnelerden buluta gider ve bunun tersi de geçerlidir. Ağ geçidi, nesneler ve IoT çözümünün bulut kısmı arasında bağlantı sağlar, verileri buluta taşımadan önce ön işleme ve filtrelemeye olanak tanır (ayrıntılı işleme ve depolama için veri hacmini azaltmak için) ve buluttan nesnelere giden kontrol komutlarını iletir. İşler daha sonra aktüatörlerini kullanarak komutları yürütür.

IoT Bağlantı Türleri

IoT sisteminin üç seviyeli bir mimarisi vardır: cihazlar, ağ geçitleri ve veri sistemleri. Veriler, dört tür iletim kanalı aracılığıyla bu seviyeler arasında hareket eder.



Detaylı bilgi

SPI HABERLEŞME

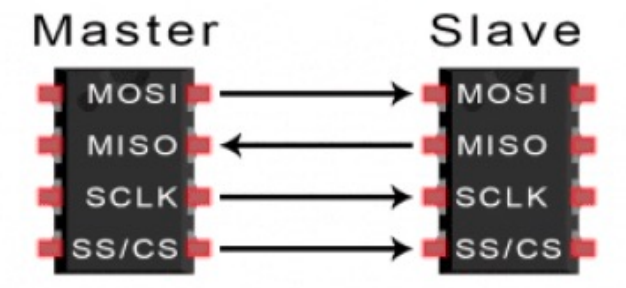
SPI Haberleşme, I2C ve UART, USB, ethernet, Bluetooth ve WiFi gibi protokollerden biraz daha yavaştır, ancak çok daha basittirler ve daha az donanım ve sistem kaynağı kullanırlar. SPI, I2C ve UART, mikrodenetleyiciler arasında ve mikrodenetleyiciler ve büyük miktarlarda yüksek hızlı verinin aktarılmasının gerekmediği sensörler arasında iletişim için idealdir.

SPI, birçok farklı cihaz tarafından kullanılan ortak bir iletişim protokolüdür. Örneğin, SD kart okuyucu modülleri, RFID kart okuyucu modülleri ve 2,4 GHz kablosuz verici/alıcıların tümü, mikro denetleyicilerle iletişim kurmak için SPI kullanır.

SPI'nin benzersiz bir avantajı, verilerin kesintisiz olarak aktarılabilmesidir. Sürekli bir akışta herhangi bir sayıda bit gönderilebilir veya alınabilir. I2C ve UART ile veriler, belirli sayıda bit ile sınırlı paketler halinde gönderilir. Başlatma ve durdurma koşulları, her paketin başlangıcını ve sonunu tanımlar, böylece veri iletim sırasında kesintiye uğrar.

SPI aracılığıyla iletişim kuran cihazlar, master-slave ilişkisi içindedir. Master, kontrol cihazıdır (genellikle bir mikrodenetleyici), bağımlı cihaz (genellikle bir sensör, ekran veya bellek yongası) master'dan talimat alır. SPI'nin en basit konfigürasyonu tek ana, tek bağımlı sistemdir, ancak bir ana birden fazla bağımlıyı kontrol edebilir (aşağıda daha fazlası anlatılmaktadır).

SPI Haberleşme Pinleri



- **MOSI (Master Output/Slave Input):** Master'ın slave'e veri gönderdiği hat.
- **MISO (Master Input/Slave Output):** Slave'in master'a veri gönderdiği hat.
- **SCLK (Saat):** Saat sinyali için gerekli hat.
- **SS/CS (Slave Select/Chip Select):** Master'ın hangi slave'e veri göndereceğini seçtiği hat.

SPI Haberleşme Nasıl Gerçekleşir?

Saat(The Clock)

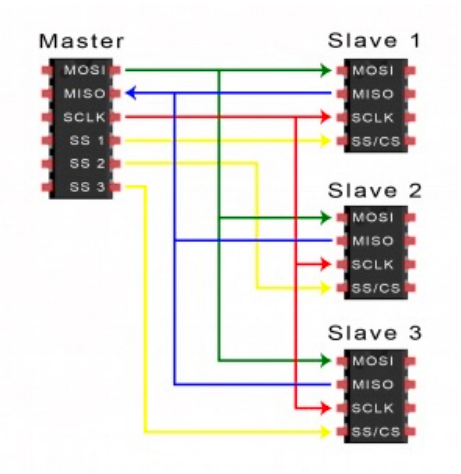
Saat sinyali, master'dan veri bitlerinin çıkışını, slave tarafından bitlerin örneklenmesine senkronize eder. Her saat döngüsünde bir bit veri aktarılır, bu nedenle veri aktarım hızı saat sinyalinin frekansı tarafından belirlenir. Master, saat sinyalini yapılandırdığı ve ürettiği için SPI iletişimi her zaman master tarafından başlatılır.
Cihazların bir saat sinyali paylaştığı herhangi bir iletişim protokolü, senkron olarak bilinir. SPI senkron bir iletişim protokolüdür.

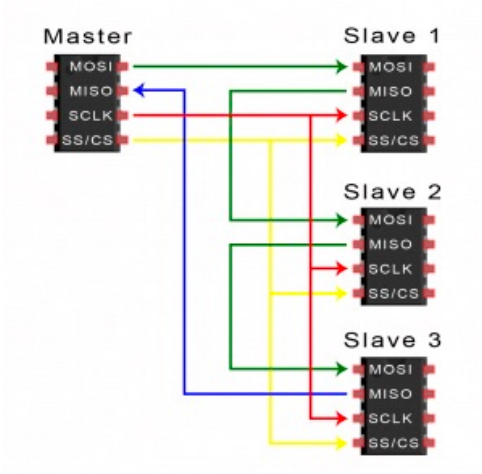
Köle Seçimi(Slave Select)

Master, slave'in CS/SS hattını düşük voltaj seviyesine ayarlayarak hangi slave ile konuşmak istediğini seçebilir. Boşta, iletim yapmayan durumda, bağımlı seçim hattı yüksek voltaj seviyesinde tutulur. Master üzerinde birden fazla CS/SS pini mevcut olabilir, bu da birden fazla slave'in paralel olarak bağlanmasına izin verir.

Çoklu Köleler(Multiple Slaves)

SPI, tek bir ana ve tek bir bağımlı ile çalışacak şekilde ayarlanabilir ve tek bir ana tarafından kontrol edilen birden fazla bağımlı ile kurulabilir. Master'a birden fazla slave bağlamanın iki yolu vardır. Master birden fazla bağımlı seçme pimine sahipse, bağımlı birimler şu şekilde paralel olarak bağlanabilir:





Yalnızca bir bağımlı seçim pimi mevcutsa, bağımlı birimler yukarıdaki gibi zincirleme bağlanabilir:

MOSI ve MISO

Master, MOSI hattı üzerinden seri olarak veriyi bit bit köleye gönderir. Slave, master'dan gönderilen verileri MOSI pininden alır. Master'dan slave'e gönderilen veriler genellikle en önemli bit önce gönderilir.

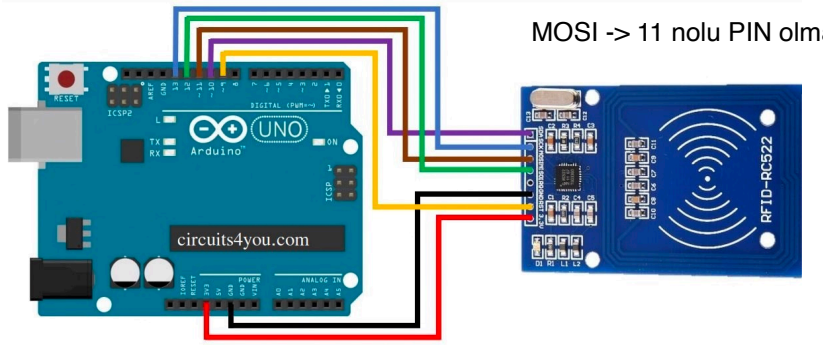
Köle ayrıca seri olarak MISO hattı üzerinden master'a veri gönderebilir. Slave'den master'a gönderilen veriler genellikle en az anlamlı bit ilk önce gönderilir.

RFID

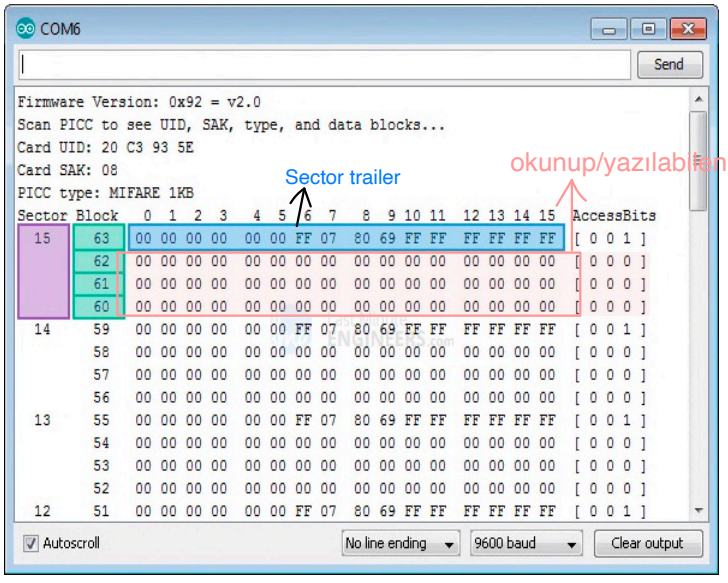
Radyo Frekansı ile Tanımlama (**RFID**) teknolojisi, radyo frekansı kullanarak nesneleri tekil ve otomatik olarak tanıma yöntemidir (bkz Otomatik Tanıma ve Veri Toplama). RFID, temel olarak bir etiket ve okuyucudan meydana gelir. RFID etiketleri Elektronik Ürün Kodu (EPC) gibi nesne bilgilerini almak, saklamak ve göndermek için programlanabilirler. Ürün üzerine yerleştirilen etiketlerin okuyucu tarafından okunmasıyla tedarik zinciri yönetimi ile ilgili bilgiler otomatik olarak kaydedilebilir veya değiştirilebilir.

RFID etiketi, radyo frekansı ile yapılan sorguları almaya ve cevaplamaya olanak tanıyan bir silikon yonga, anten ve kaplamadan meydana gelir. Yonga, etiketin üzerinde bulunduğu nesne ile ilgili bilgileri saklar. Anten, radyo frekansı kullanarak nesne bilgilerini okuyucuya iletir. Kaplama ise etiketin bir nesne üzerine yerleştirilebilmesi için yonga ve anteni çevreler.

Arduino bağlantısı yapılırken SCK -> 13 nolu PIN
MISO -> 12 nolu PIN
MOSI -> 11 nolu PIN olmalıdır.



ID Card



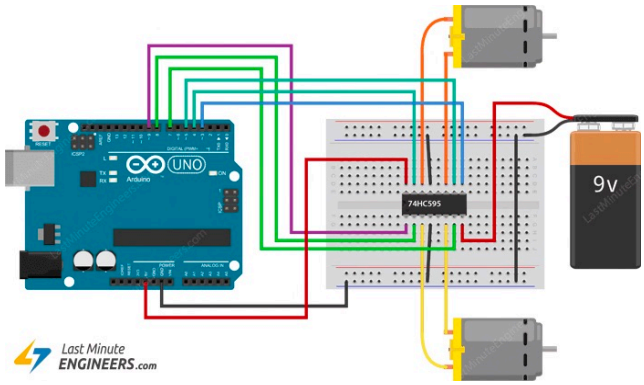
Etiketin Benzersiz Kimliği (UID), bellek boyutu ve tüm 1K bellek dahil olmak üzere etiketle ilgili tüm yararlı bilgileri görüntüler.

Etiketin 1K hafızası 16 sektör olarak düzenlenmiştir (0 ila 15) Her sektör ayrıca 4 blok olarak (0 ila 3 blok) ayrılmıştır. Her blok 16 bayt veri depolayabilir.

1. Her sektörde son bloğa (3, 7, 11.. numaralı bloklar) sector trailer denir. Bu blok, 2 anahtarı veya şifreyi saklar ve sektördeki geri kalan bloklara erişimi kontrol eder (örn. 7 numaralı blok, blok 4, 5 ve 6'yı kontrol eder). İlk 6 bayt (0..5) A anahtarıdır ve son 5 bayt (10..15) B anahtarıdır. Göreceğimiz gibi, herhangi bir blokta okumadan veya yazmadan önce, önce kimliğimizi doğrulamalıyız. iki tuştan birini (normalde A tuşu) sağlayarak bu sektör. 6, 7 ve 8 numaralı baytlar, bloklara erişilme şeklini (okuma/yazma) kontrol eden kontrol baytlarını saklar.
2. Sektör 0'ın Blok 0'ı Üretici Bloğu / Üretici Verileri olarak bilinir, IC üretici verilerini ve Benzersiz Tanımlayıcı'yı (UID) içerir.
3. Kalan bütün bloklar (1 ve 2; 4,5 ve 6; 8, 9,10; vb.) veri blokları olarak bilinirler ve gerekli kimlik doğurlama işlemlerinden sonra okuma ve yazma operasyonları için kullanılabilirler.

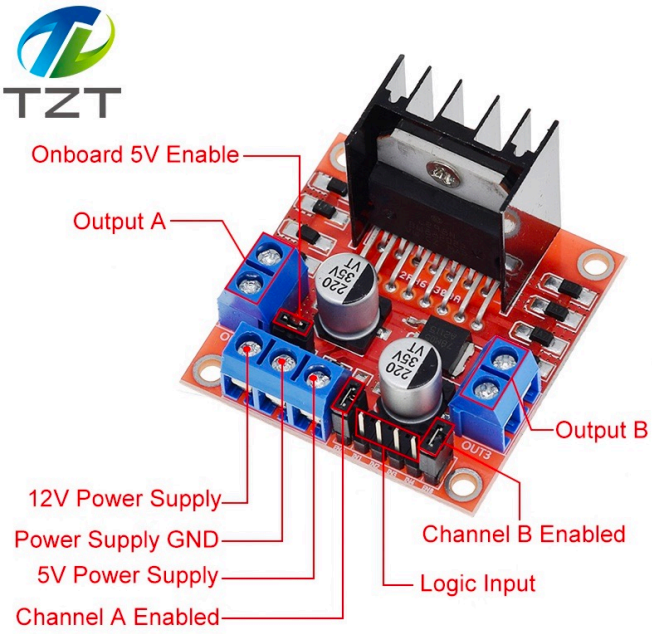
Arduino ile DC Motor Kontrolü

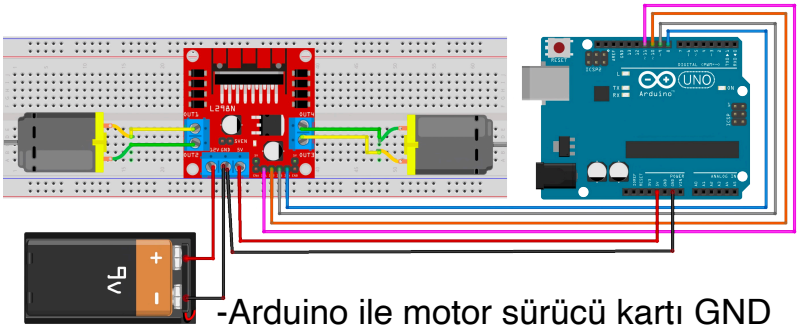
Detaylı bilgi için: https://www.robocombo.com/blog/icerik/arduino-ile-dc-motor-kontrolu
https://www.tutorialspoint.com/arduino/arduino_dc_motor.htm



L298 Motor Sürücü Kartı

https://www.youtube.com/watch?v=kE0Vrtbw4Vw&t=28s





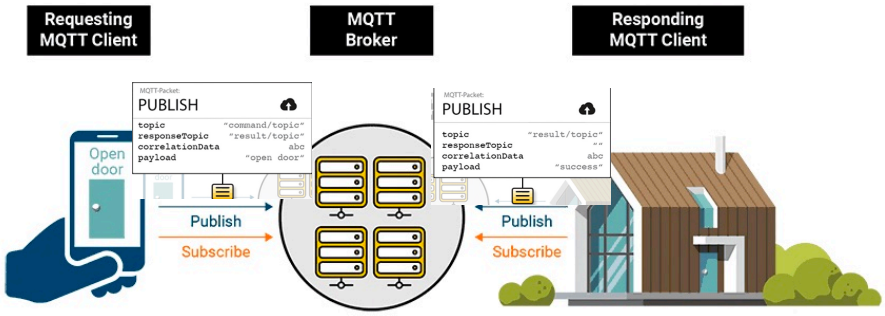
-Arduino ile motor sürücü kartı GND hattı bir olmalı
-5V pini boş olabilir eğer 5v çıkışa ihtiyacın varsa kullanabilirsin.
-EN0 ve EN1 bağlantıları motor hızını ayarlamak içindir. Pinleri arduinonun PWM pin çıkışlarına bağlayarak analog çıkış verilebilir.

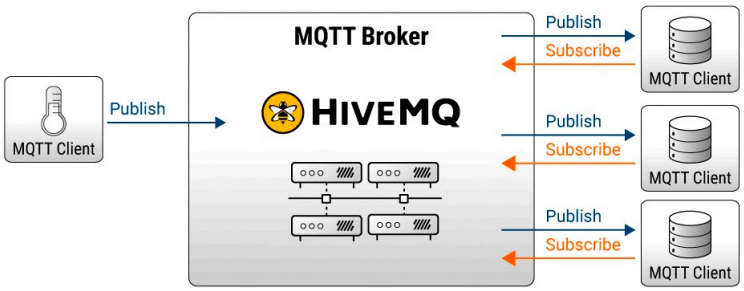
https://omerfarukyildiz.com/arduino-l298n-motor-surucu-kullanimi/#close

https://www.hivemq.com/mqtt-essentials/

MQTT(Message Queuing Telemetry Transport)

MQTT, Nesnelerin İnterneti (IoT) için en yaygın kullanılan mesajlaşma protokolüdür. MQTT, MQ Telemetri Aktarımı anlamına gelir. Protokol, IoT cihazlarının İnternet üzerinden verileri nasıl yayınlayabileceğini ve bunlara abone olabileceğini tanımlayan bir dizi kuraldır. MQTT, gömülü cihazlar, sensörler, endüstriyel PLC'ler vb. gibi IoT ve endüstriyel IoT (IIoT) cihazları arasında mesajlaşma ve veri alışverişi için kullanılır. Protokol olay odaklıdır ve yayınla/abone ol (Publish/ Subscribe) modelini kullanarak cihazları birbirine bağlar. Gönderici (Publisher) ve alıcı (Subscriber) konular (Topics) aracılığıyla iletişim kurar ve birbirinden ayrılır. Aralarındaki bağlantı MQTT broker(aracısı) tarafından gerçekleştirilir. MQTT aracısı, gelen tüm mesajları filtreler ve Abonelere doğru bir şekilde dağıtır.





MQTT-Packet:

CONNECT

contains:	Example
clientId	"client-1"
cleanSession	true
username (optional)	"hans"
password (optional)	"letmein"
lastWillTopic (optional)	"/hans/will"
lastWillQos (optional)	2
lastWillMessage (optional)	"unexpected exit"
lastWillRetain (optional)	false
keepAlive	60

mosquitto open source MQTT broker server
https://test.mosquitto.org/

open terminal
ping test.mosquitto.org

MQTT client:
https://www.hivemq.com/demos/websocket-client/