

# An Example of Book Class

for L<sup>A</sup>T<sub>E</sub>X Class

11 Haziran 2018



# İçindekiler

<b>1</b>	<b>Giriş</b>	<b>1</b>
1.1	Gömülü Sistem Tanımları . . . . .	1
1.2	Önemi . . . . .	3
1.3	Geçmiş ve Geleceği . . . . .	4
1.3.1	Gartner Hype Cycle . . . . .	4
1.3.2	Edge Computing . . . . .	9
1.3.3	Fog Computing . . . . .	14
1.3.4	Cloud Computing . . . . .	15
1.3.5	IoT (Internet of Things) . . . . .	22
1.3.6	Endüstri 4.0 . . . . .	34
<b>2</b>	<b>Genel Sistem Mimarisi</b>	<b>41</b>
2.1	İşleyiciler . . . . .	41
2.1.1	VonNeumann Mimarisi . . . . .	41
2.1.2	İşlemci, Mikroişlemci ve Mikrodenetleyici . . . . .	42
2.1.3	Mikroişleyicilerin İç Yapısı . . . . .	44
2.1.4	Mikroişleyicilerin Sınıflandırılması . . . . .	56

2.2 Bellek . . . . .	65
2.2.1 RAM - Random Access Memory (Rastgele Erişimli Bellek) . . . . .	65
2.2.2 ROM (Read Only Memory) . . . . .	69
2.3 Çevre Birimleri . . . . .	71
2.3.1 Seri İletişim . . . . .	71
2.3.2 Paralel İletişim . . . . .	82
2.3.3 Network . . . . .	87
2.3.4 DAC ve ADC . . . . .	95
2.4 Enerji Kaynakları . . . . .	106
2.4.1 Batarya . . . . .	107
2.4.2 Enerji Hasatlama . . . . .	109
2.5 Programlama . . . . .	111
2.5.1 Gömülü Yazılım Geliştirme . . . . .	111
2.5.2 Programlamaya Giriş . . . . .	115
2.5.3 BIOS . . . . .	118
2.5.4 Firmware . . . . .	119
<b>3 Geliştirme Ortamları</b> . . . . .	<b>127</b>
3.1 İşletim Sistemi Olmayan Geliştirme Ortamları . . . . .	127
3.1.1 Arduino IDE . . . . .	127
3.1.2 Cypress . . . . .	132
3.1.3 ESP . . . . .	133
3.2 İşletim Sistemi Olan Geliştirme Ortamları . . . . .	134

3.2.1	Linux . . . . .	134
3.2.2	Android . . . . .	142
3.2.3	Windows . . . . .	143
<b>4</b>	<b>Sensörler</b>	<b>145</b>
4.1	Sensör Nedir? Ne İşe Yarar? . . . . .	145
4.1.1	Sensör Nasıl Çalışır? Çalışma Prensibi Nedir? . . . . .	145
4.2	Sensör Çeşitleri . . . . .	146
4.2.1	Beslenme ihtiyaçlarına göre sensörler . . . . .	146
4.2.2	Giriş büyüklüklerine göre sensörler . . . . .	146
4.2.3	Çıkış büyüklüklerine göre sensörler . . . . .	147
4.3	Işık Sensörleri . . . . .	147
4.4	Hareket Sensörleri . . . . .	148
4.5	Sıcaklık Sensörleri . . . . .	148
4.6	Nem Sensörleri . . . . .	149
<b>5</b>	<b>Gömülü Sistemlerde Güvenlik</b>	<b>151</b>
5.1	Giriş . . . . .	151
5.2	Veri İletimindeki Güvenlik . . . . .	152
5.2.1	Veri Şifreleme . . . . .	152
5.2.2	Açık Anahtar Algoritması . . . . .	153
5.2.3	Dijital İmza . . . . .	154
5.2.4	Dijital Sertifika . . . . .	154
5.2.5	Sertifika Hiyerarşisi . . . . .	155

5.2.6	Cihaz Üzerinde Anahtar Kabul Protokol Örneği . . . . .	155
5.3	Cihaz İçindeki Güvenlik . . . . .	155
5.3.1	Güvenli System On Chip . . . . .	156
5.3.2	Dahili RAM ve Güvenli Process . . . . .	157
5.3.3	Güvenli BootLoader ve Kod İmzalama . . . . .	158
5.3.4	Şifreleme ve Deşifreleme Motoru . . . . .	159
5.3.5	Sistem Zamanı . . . . .	160
5.4	Güvenli Veri İletimi için Kişisel Teknolojiler . . . . .	160
5.5	Cihaz Üretimi Sırasında Sertifika ve Anahtar İdaresi . . . . .	161
5.6	Sonuç . . . . .	162
<b>6</b>	<b>Örnek Geliştirme Ortamları</b>	<b>163</b>
6.1	Arduino . . . . .	163
6.2	TI LaunchPad . . . . .	167
6.3	Cypress . . . . .	169
<b>7</b>	<b>Projeler</b>	<b>173</b>
7.1	Kart Okuma Sistemi . . . . .	173
7.2	Ev Otomasyonu Sistemi . . . . .	180
7.3	Seri haberleşme (I2C) . . . . .	186
7.4	7 Segmentli Gösterge . . . . .	186
7.5	Engelden Kaçan Robot . . . . .	192
7.6	ESP32 WiFi-Bluetooth Modül . . . . .	198
7.7	ESP8266 (ESP-01) WiFi Modülü . . . . .	202

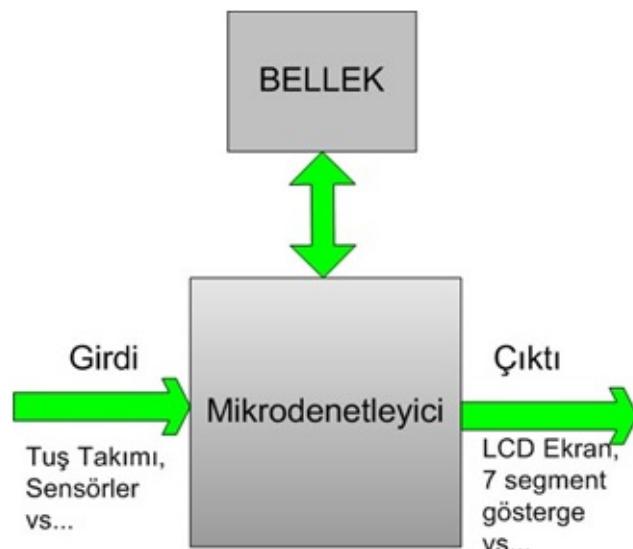
7.8 LDR Işık Sensörü . . . . .	206
7.9 Joystick Modülü . . . . .	208
7.10 LM35 Sıcaklık Sensörü . . . . .	209
7.11 Arduino Boy Ölçer . . . . .	212
7.12 Arduino Uzaktan Joystick Kontrollü Araba . . . . .	217
7.13 Piyano . . . . .	221
7.14 Anroid Telefon ile Lamba Kontrolü . . . . .	223
7.15 Akıllı Kablosuz Zar . . . . .	226
7.16 LCD Ekran ve Butonlarla Şifre Girişi . . . . .	239
7.17 Morse Alfabe Çeviricisi . . . . .	244
7.18 Tilt Mouse . . . . .	249
7.19 HC-05 Bluetooth Serial Module . . . . .	256
7.20 RFID-RC522 ile Kapı Kiliti . . . . .	263
7.21 Arduino Web Server LED Kontrol . . . . .	269
7.22 Arduino ve Ethernet ile Fan Kontrolü . . . . .	272
7.23 LM35 ve ESP8266 ile İnternet Termometresi . . . . .	286
7.24 Seri Haberleşme (UART) . . . . .	292
7.25 Multikopter . . . . .	294



# Bölüm 1

## Giriş

### 1.1 Gömülü Sistem Tanımları



Gömülü sistemler bilgi işlem sistemleridir, ancak kullanıcı arabirimini (UI) içermeyen (örneğin, gomülü sistemin tek bir görevi gerçekleştirecek şekilde tasarlandığı aygıtlarda) karmaşık grafik kullanıcı arabirimlerine (GUI) sahip olmak arasında değişebilir; mobil cihazlarda olduğu gibi. Kullanıcı arayüzleri düğmeler, LED'ler, dokunmatik ekran algılama ve daha fazlasını içerebilir.

Bazı sistemler uzak kullanıcı arayüzlerini de kullanır.

### **Gömülü sistem donanımı (mikroişlemci tabanlı, mikrodenetleyici tabanlı)**

Gömülü sistemler mikroişlemci veya mikrodenetleyici tabanlı olabilir. Her iki durumda da, ürünün kalbinde gerçek zamanlı işlemler için hesaplama yapmak üzere tasarlanmış bir entegre devre (IC) vardır. Mikroişlemciler mikrodenetleyicilerden görsel olarak ayırt edilemezler, ancak mikroişlemci sadece merkezi işlem birimi (CPU) uygular bu nedenle bellek çipleri gibi diğer bileşenlerin eklenmesini gerektirir, mikrodenetleyiciler kendi kendine yeten sistemler olarak tasarlanmıştır. Mikrodenetleyiciler sadece bir CPU değil, aynı zamanda flash bellek, RAM veya seri iletişim portları gibi bellek ve çevre birimleri içerir. Mikrodenetleyiciler tam sistemleri uygulama eğiliminde olduklarıdan, genellikle daha karmaşık görevlerde kullanılırlar. Mikrodenetleyiciler, örneğin, taşıtların, robotların, tıbbi cihazların ve ev aletlerinin operasyonlarında kullanılır. Mikrodenetleyici kabiliyetinin yüksek ucunda, RAM, sistem hızı ve benzerleri açısından kesin bir tanımlama bulunmamakla birlikte, çip üzerine sistem terimi (SoC) sıkılıkla kullanılır.

### **Gömülü sistem yazılımı**

Tipik bir endüstriyel mikrodenetleyici, tipik bir kurumsal masaüstü bilgisayarla karşılaşıldığında oldukça toydur ve genellikle daha basit, az bellek kullanan bir program ortamına bağlıdır. En basit cihazlar çiplak metal üzerinde çalışır ve çip CPU'nun makine kod dili kullanılarak doğrudan programlanır.

Ancak, çoğu zaman, gömülü sistemler, özellikle gerçek zamanlı işletim ortamlarının sunulması gereken, işletim sistemine veya gömülü kullanımına uygun dil platformları kullanır. Tasarımcılar, SoC'lerde bulunanlar gibi daha yüksek çip kabiliyet seviyelerinde, sistemlerin genellikle yeterince hızlı olduğuna ve "gerçek zamana yakın" yaklaşımların uygun olduğu reaksiyon süresinde küçük değişimlere karşı toleranslı görevlere giderek daha fazla karar verdiler. Bu durumlarda, Linux işletim sisteminin çıkarılmış sürümleri yaygın olarak dağıtılr, ancak EmbeddedJava ve Windows IoT (eski Windows Embedded) dahil olmak üzere gömülü sistemler üzerinde çalışmak üzere ayrıstırılmış başka işletim sistemleri de vardır. Genellikle, programların ve işletim sistemlerinin gömülü cihazlarda saklanması için flash veya yeniden yazılabilir flash bellekten birini kullanır.

**Gömülü sistemlerde hata ayıklama**

Masaüstü bilgisayar ortamlarında çalışan programcılar, hem geliştirilmekte olan kodları çalıştırabilen hem de geliştirme kodunun eylemlerini takip eden hata ayıklayıcı uygulamalarını çalıştırabilen sistemlere sahip olsalar da, gömülü sistem programcılarına genellikle bu tür lüksler verilmemektedir.

Bazı programlama dilleri, ilkel etkileşimli hata ayıklama işlemi doğrudan çip üzerinde mevcut olan mikroişlemciler üzerinde yeterli verimlilikle çalışır. Ayrıca, işlemciler genellikle bir JTAG veya benzeri bir hata ayıklama bağlantı noktası üzerinden denetlenebilen CPU hata ayıklayıcılarına sahiptir.

Ancak, birçok durumda, gömülü sistemlerin programcıları, seri veya başka bir bağlantı noktası yoluyla hedef sisteme ayrı bir hata ayıklama sistemi ekleyen araçlara ihtiyaç duyarlar. Bu senaryoda, programçı, bir masaüstü bilgisayardaki yazılımin hata ayıklamasında olduğu gibi, geleneksel bir kişisel bilgisayarın ekranındaki kaynak kodunu görebilir. Ayrıca, sık kullanılan bir yaklaşım, yazılımdaki fiziksel çipi taklit eden bir PC üzerinde yazılım çalıştırırmak ve böylece yazılımin performansını gerçek, fiziksel bir çip üzerinde çalışmış gibi ayıklamaktır.

Genel anlamda, gömülü sistemler test ve hata ayıklama işlemlerine daha fazla ilgi göstermişlerdir çünkü gömülü kontrol kullanan çok sayıda cihaz, güvenlik ve güvenilirliğin en öncelikli olduğu durumlarda kullanılmak üzere tasarlanmıştır.

**Nesnelerin interneti (Internet-of-Things), gömülü sistem tabanına dayanıyor**

Bazı gömülü sistemler nispeten basit olsa da, artan bir sayı ya insani karar vermeyi destekliyor ya da bir insanın sağlayabildiği kapasiteyi sunuyor. Örneğin, dronlarda kullanılanlar da dahil olmak üzere bazı havacılık sistemleri, sensör verilerini bütünlüğe getiriyor ve bu bilginin, insan türünden daha hızlı çalışmasına ve yeni türdeki çalışma özelliklerine izin verebiliyor.

## 1.2 Önemi

Gömülü bir sistem, daha büyük bir mekanik veya elektrik sistemi içinde belirlili bir tanımlanmış işlevle sahip bir bilgisayar sistemidir. Ortak kullanımında birçok cihazı kontrol ederler. Düşük güç tüketirler, küçük boyuttadırlar ve maliyeti birim başına düşüktür. Modern gömülü sistemler genellikle mik-

rodenetleyicilere dayanır. Bir mikrodenetleyici, işlemci çekirdeği, bellek ve programlanabilir giriş ve çıkış çevre birimleri içeren tek bir entegre devrede küçük bir bilgisayardır. Gömülü sistem belirli görevleri yerine getirmeye adadığı için, ürünün boyutunu ve maliyetini azaltmak ve güvenilirliği ve performansı artırmak için optimize edilebilirler. Çevremizdeki hemen hemen her elektronik cihaz bir Gömülü Sistem, dijital saatler, MP3 çalarlar, Çamaşır Makinesi, Güvenlik Sistemi, tarayıcı, yazıcı, cep telefonu, Asansörler, ATM, Satıcı Makineleri, GPS, trafik ışıkları, Uzaktan Kumanda, Mikrodalga Fırın ve birçok şey. Gömülü sistemlerin kullanımı neredeyse sınırsızdır, çünkü her gün gömülü bilgisayarları bir çok şekilde kullanan piyasaya yeni ürünler tanıtmaktadır. Gömülü Sistemler Bilim'de bir devrim yarattı. Aynı zamanda Nesnelerin İnterneti'nin (IoT) bir parçası (nesnelerin, hayvanların veya insanların benzersiz tanımlayıcılarla ve insandan insana ya da insandan bilgisayara gereksinim duymadan bir ağ üzerinden veri aktarma becerisine sahip olduğu bir teknoloji. etkileşim.) Sizin için kolaylaştırılmış. (Örneğin, gideceğiniz yere giden bir trende oturuyorsunuz ve evinizden yaklaşık elli kilometre uzaktasınız ve aniden fan değiştirmeyi unuttuğunuzu anlıyorsunuz. Endişelenmeyin, sadece yapmanız gereken bu teknolojiyi kullanan cep telefonunuzun bir tuşuna basmak) Nesnelerin İnterneti (IoT). Bu IoT hakkından iyi olan sadece bir örnek. Kirlilik Seviyelerini izleyebiliriz, sezon ve hava şartlarına göre sokak ışıklarının yoğunluğunu kontrol edebiliriz. IoT, ebeveynlere akıllı telefonlarında bebeğin nefes alma, cilt ısısı, vücut pozisyonu ve aktivite seviyesi hakkında gerçek zamanlı bilgi sağlayabilir ve hayatını kolaylaştıracak başka birçok uygulamada kullanılabilir.

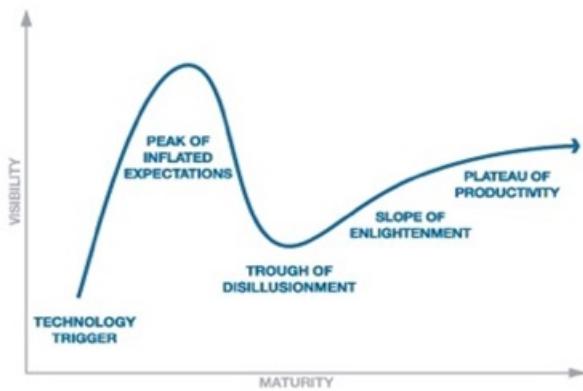
### 1.3 Geçmiş ve Geleceği

#### 1.3.1 Gartner Hype Cycle

Bilgi teknolojileri üzerinde araştırmalar gerçekleştiren ve bu alanda önde gelen araştırma firmalarından olan, Gartner her sene "Hype Cycle" olarak adlandırılan ve farklı sektörlerdeki teknolojilerin evrelerine yönelik bir grafik yayılmamakta. Teknoloji ile tetiklenen konuları, bekłentilerin en üst düzeye çıktığı noktaları, bekłentilerin düşmesi ile doğru modellerin bulunması ve ürünleşmesi fazlarını analiz eden "Hype Cycle" grafiği bir teknoloji ya da uygulamanın zaman içinde nasıl gelişeceği konusunda fikir verirken; bunun spesifik iş hedefleri kapsamında nasıl hayatı geçirileceği ve doğru yatırı-

ların ve stratejilerin planlanması adına da önemli bir görüş sağlıyor. Son yayınlanan Gelişen Teknolojiler (Emerging Technologies) "Hype Cycle 2017" grafiği de yeni teknolojilerin pazardaki algularına ve statülerine yönelik ipuçlarını paylaşmakta. Gartner, bu çalışmasını, teknolojinin kullanışlılığı, ticari boyutu ya da insanların gerçekten bu teknolojileri ilerde kullanıp kullanmayacaklarından ziyade; teknolojinin "pazar açısından değer algısı" olarak değerlendiriyor. Gartner'in Hype Cycle grafiği alttaki grafikte gösterildiği şekilde, 5 farklı evreden oluşuyor.

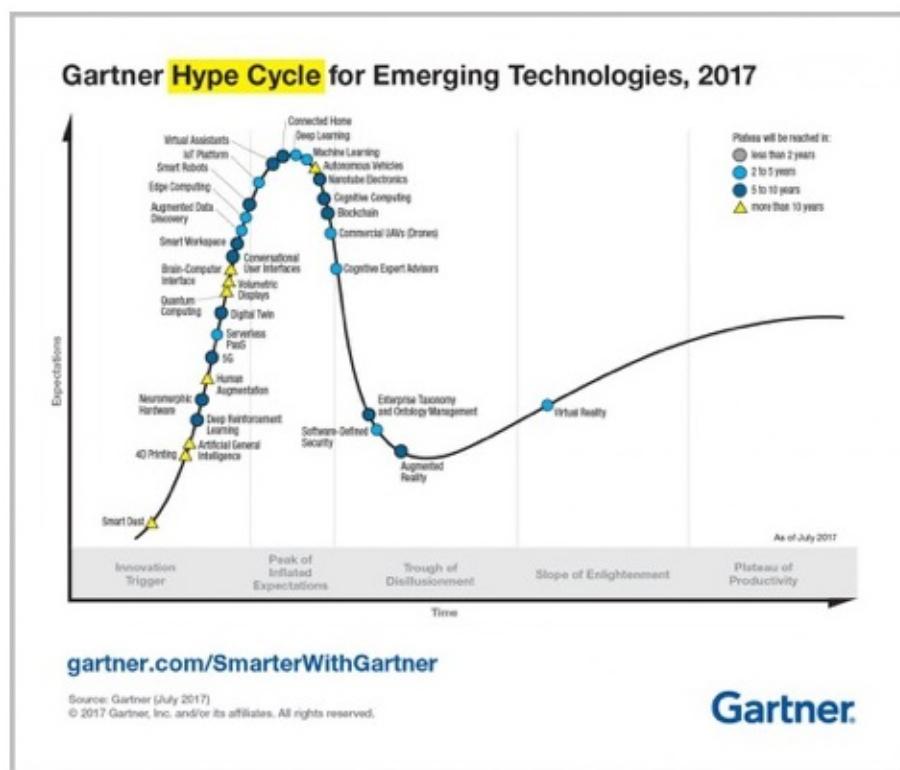
#### Gartner Teknoloji ilerleme Döngüsü (Gartner Hype Cycles Phases)



- 1 - Teknolojik Tetikleme  
(Technology Trigger)
- 2 - Beklentilerin Tepe Noktası  
(Peak of Inflated Expectations)
- 3 - Hayal Kırıklığı Oyuğu  
(Trough of Disillusionment)
- 4 - Aydınlanma Eğimi  
(Slope of Enlightenment)
- 5 - Verimlilik Platosu  
(Plateau of Productivity)

Her sene, Gartner Hype Cycle grafiğini açıkladığında, özellikle "peak of inflated expectations"da (beklentilerin tepe noktası) yer alan teknolojiler, diğerlerine kıyasla çok daha ön plana çıkıyor. Sektörlerinin önde gelen firmaları,

“next big thing” (bir sonraki büyük proje) olarak adlandırdığı çalışma ya da projelerini seçerken, özellikle bu bölümde yer alan teknolojilere odaklanıyorlar. Bu açıdan, bu sene bekłentilerin tepe noktasında yer alan teknolojileri bundan sonra çok daha sık duyacağımız kesin.



2017'nin Hype Cycle grafiği 3 yeni gelişen teknoloji olarak nitelendirilen, ancak mega trend haline gelmesi beklenen teknolojileri öne çıkarmakta: Yapay zeka (Artificial Intelligence – AI) her yerde; görünmez, sürükleyici deneyimler ve dijital platformlar. Önümüzdeki dönemde, sadece iş hayatına değil, günlük yaşama da oldukça etkisi olacak olan bu mega trendleri detaylı olarak incelemek ve üzerinde düşünmekte yarar var. Bu teknolojilerin doğru uygulanması ve pazara sunulması yüksek derecede rekabet avantajı da sağlamaaktır.

### **Her yerde AI:**

AI destekli sürücüsüz araçların potansiyel etkilerini bir düşünün: Kazaları azaltabilir, trafiği daha düzenli hale getirebilir ve insanların ev-iş dengelerini daha verimli hale getirebilir. Sürücüsüz araçlar, AI, IoT (Internet of Things – Nesnelerin Interneti) ve diğer gelişen teknolojilerin, paylaşım ekonomisi

(sharing economy) gibi ekonomik trendler ile entegrasyonu doğrultusunda, pazarı disrupt (yıkıcı) edecek, yeni iş modellerinin öne çıkması olası. Uber, bir teknoloji start-up'ının doğru iş modeli ve kurgu ile özel araçların hakim olduğu bir sanayiyi, “ulaşımı bir servis haline getirerek” (transportation as a service) nasıl dönüştürebildiğine en güzel örnek. Son birkaç yıldır, medyada sürücüsüz araçlar ile ilgili oldukça büyük bir odak oluşması ve bu alana yönelik sürekli artan haberler, beklentilerin de oldukça yükselmesine neden oldu.

Medyada sürücüsüz araçlarla ilgili övgü dolu hikayelerin sürekli ön planda yer alması, beklentilerin de oldukça artmasına neden oldu. AI'ın sürücüsüz araç konseptinde oldukça önemli bir yeri olması, bu teknolojinin ve makine öğrenim algoritmalarının önemli derecede gelişmesine de yol açmış oldu. AI ile birlikte algılama, görüntüleme ve harita üzerinde yönlendirme konusunda gelişmeler artarken; standardizasyon ve maliyetler hala teknolojinin yayılmasına yönelik bariyerler olarak kendini göstermektedir. Bu sorunlara yönelik çözümlerin geliştirilmesi ile birlikte, AI'ın önumüzdeki 10 yıl içerisinde en yıkıcı teknolojilerden biri olmayı südüreceği aşikar. AI ile birlikte işletmeler, sahip oldukları verileri en verimli şekilde kullanarak yeni durumlara uyum sağlayabilecek ve daha önce karşılaşmadılar sorunları çözebilecekler.

#### **Şeffaf ve Sürükleyici Deneyimler (Transparently Immersive Experiences):**

Bu sene gerçekleşen Facebook'un F8 Konferansı'nda, Zuckerberg, AR (Augmented Reality – Artırılmış Gerçeklik) ve sosyal medya arasındaki bağlantıyı farklı bir boyuta taşıyan bir platformu tanıttı. Bu durum, tüketicilere yönelik AR kullanımına yönelik senaryoların tekrar ateşlenmesine neden oldu (Apple'ın son yazılım geliştiricilere yönelik düzenlediği konferansta tanıttığı AR Kit'in de bu konuda oldukça önemli bir etkisi bulunuyor tabii). Sanal dünyayı gerçek dünya elementleri ile bütünlestiren teknolojinin gerçekten de pek çok sektör açısından büyük bir potansiyeli bulunuyor.

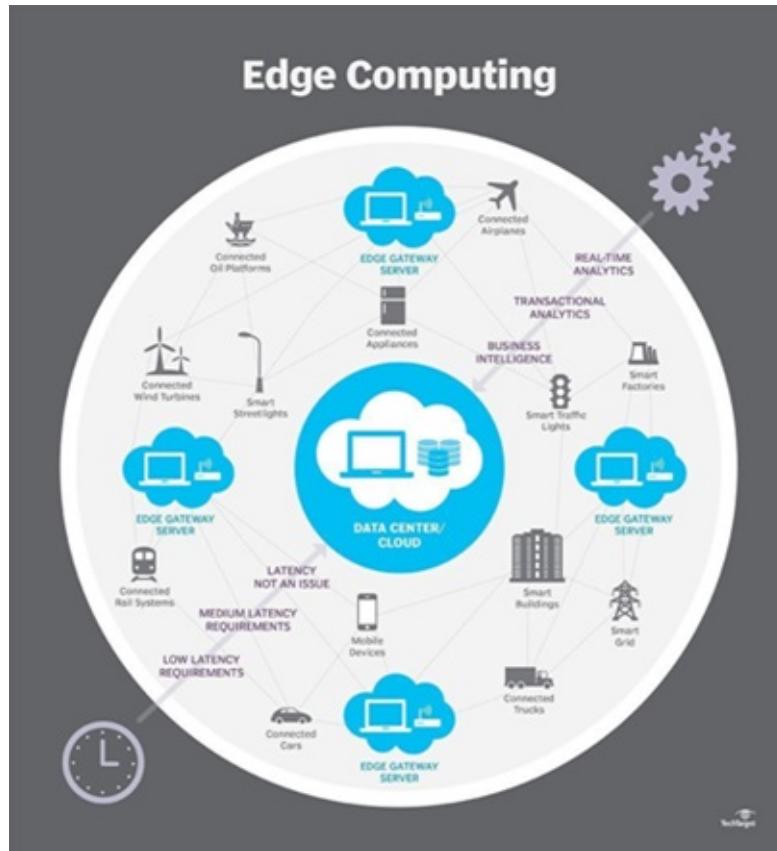
Sürükleyici deneyimler söz konusu olduğunda; teknoloji insanlar, işletmeler ve nesneler arasında şeffaflık getirmekte. Teknolojinin daha uygulanabilir, bağlamalı ve akışkan hale gelmesi, insan merkezli olmasının da önünü açmaktadır. Bu alanda, AR'ın dışında, dijital teknolojiler, bağlantılı evler, sanal gerçeklik (virtual reality – VR) ve 4D baskı gibi konular da işletmelerin odağında yer almaya başlayacak.

#### **Dijital Platformlar:**

Bitcoin ve Ethereum'un artan popularitesiyle birlikte blockchain'in dijital

dünyada artan etkisi, bundan böyle de blockchain’ı sıkça duyacağımızı göstermekte. Bununla birlikte blockchain teknolojisinin uygulama alanları genellikle pilot aşamasında. Ancak, Gartner'a göre de blockchain in bütün endüstrilere yönelik yıkıcı etkisi olacak ve pek çok sektörü yeniden düzenleyecek.

Dijital şirketler, bölgelere ayrılmış teknik altyapılardan ekosistem uyumlu platformlara evriliyor. İşletmeler, platform tabanlı iş modelleri oluşturma konusunda ve bu hareketin desteklenmesi için hangi teknolojiye ihtiyaç duyulduklarını düşünmek zorundadırlar. Bu alandaki diğer teknolojiler arasında 5G, dijital ikizler, IoT platformları ve kuantum hesaplama yer alıyor.<sup>1</sup>



### 1.3.2 Edge Computing

Edge computing, veri toplama ve kontrol fonksiyonlarını, yüksek bant genişliğine sahip içerik deposunu ve uygulamaları son kullanıcıya yakınlaştırıyor. Daha büyük bir bulut bilişim mimarisinin bir parçası olarak da bir ağın (internet veya özel ağ) mantıksal bir uç noktasına yerleştiriliyor. Edge computing'in tartışılanacak üç temel uygulaması mevcuttur.

1. Bir toplama ve kontrol noktası olarak yerel "nesnelerden" çok büyük boyutlarda bilgi toplama aracı

2. Bir içerik dağıtım ağının bir parçası olarak, bant genişliği açısından yoğun

<sup>1</sup><http://www.dipnot.tv/gartner-hype-cycle-2017de-one-cikan-yeni-teknoloji-trendleri/90489/>

icerikler için bir yerel depolama ve dağıtım sağlayıcısı

3. Bulut hizmetlerini çoğaltmayı ve veri merkezini genel buluttan izole etmeyi sağlayacak şirket içi bir uygulama ve süreç aracı

### **UYGULAMA 1: YÜKSEK BANT GENİŞLİĞİNE SAHİP İÇERİĞİN DAĞITIMI**

Gecikme, bir veri paketinin iletiliği an ile hedefine ulaştığı (tek yön) ve geri döndüğü (gidiş-dönüş) an arasında geçen süredir. Verilerin çoğu tek yönlü seyahat etse de bunu ölçmek neredeyse imkansızdır. Tek bir noktadan gidiş-dönüş süresinin en yaygın gecikme ölçümleri olmasının nedeni de budur. 100 milisaniyeden (ms) az gidiş-dönüş gecikmeleri normaldir ve arzu edilen süreden 25 ms daha kısalıdır. Bant genişliği, verilerin ağ üzerindeki iletim hızı anlamına gelir. Ağ iletişimini, donanımlarının maksimum hızları üreticileri tarafından yayınlanmaktadır. Ancak, belirli bir ağda elde edilen gerçek hız, hemen hemen her zaman tepe değerinden daha düşüktür. Aşırı gecikme, verilerin ağın kapasitesini doldurmasını engelleyen trafik sıkışıklığına neden olur. Ağ bant genişliğinde gecikmenin etkisi, bir trafik ışığı gibi geçici (birkaç saniye süren) ya da tek şeritli köprü gibi kalıcı olabilir. Ağ tıkanıklığının en büyük olası nedeni, yüksek bant genişliğine sahip video içerikleridir. VoD, 4K TV ve video akışı, hızlı büyüyen, yüksek bant genişliğine sahip uygulamalarıdır. Şimdi ve gelecekte, yüksek bant genişliğine sahip içeriklerin akışını hızlandırmak amacıyla ağ sıkışıklığını rahatlatmak için servis sağlayıcıları, bilgisayarlardan oluşan bir sistemi, içerikleri kullanıcıya daha yakın olarak önbelleğe alan internet üzerinde birbirine bağlıyor. Bu da, içerikleri birden fazla sunucuda çoğaltarak ve içerikleri yakınlığa bağlı olarak kullanıcılara yönlendirerek sayısız kullanıcıya hızlı bir şekilde dağıtılmemesini sağlar. İçeriği önbelleğe alan bu bilgisayarlar, edge computing'e bir örnektir.

### **UYGULAMA 2: IoT TOPLAMA VE KONTROL NOKTASI OLARAK EDGE COMPUTING**

Her şeyin (şehirler, tarım, otomobiller, sağlık, vb) akıllı olmasını sağlayacak teknolojiler, gelecekte çok büyük miktarda Nesnelerin İnterneti (IoT) sensörlerinin kullanılmasını gerektirecektir. Bir IoT sensörü, internete bağlanan bir IP adresine sahip, bilgisayar dışı bir düğüm ya da nesne olarak tanımlanır. Sensörlerin fiyatı düşmeye devam ettikçe, bağlı IoT nesnelerinin sayısı bir roket hızıyla artacaktır. Cisco, IoT'nin 2020 yılı itibarıyle internete bağlı 50 milyar cihazdan oluşacağını tahmin etmektedir.<sup>3</sup> IoT ile çalışmalar şu şe-

kilde otomatize edilebilir: Durumu veya davranışını izlemek amacıyla fiziksel varlıklar (makineler, ekipmanlar, cihazlar, tesisler, araçlar) hakkında otomatik olarak bilgi toplama Bu bilgileri, süreçleri ve kaynakları optimize etmek amacıyla görünürlük ve kontrol sağlamak için kullanma Makineden Makineye (M2M) terimi, hem kablolu hem de kablosuz sistemlerin aynı türdeki diğer cihazlarla iletişim kurmasına izin veren teknolojileri ifade eder. M2M, IoT'nin ayrılmaz bir parçası olarak kabul edilir ve akıllı şehir temasında geniş bir uygulama yelpazesine sahip olması dolayısıyla da sektörde ve işletmelere genel olarak çok çeşitli faydalalar sunar. Sensör verilerinden yararlanılmasını içeren Endüstriyel Nesnelerin İnterneti (IIoT), makineden makineye iletişim kontrolü ve otomasyon teknolojileri, büyük miktarda veri ve ağ trafiği üretir. Tescilli endüstriyel IT sistemleri ve ağ iletişim teknolojileri, IP (Internet Protokolü) ağları üzerinden haberleşen temel ticari IT sistemlerine geçiş yapmaktadır. Petrol ve gaz keşif aramaları, bu IoT uygulamasına bir örnektir. Petrol keşif aramaları sırasında iş tesislerini inceleyen, "havadan veri toplama botu" olarak adlandırılan birden fazla drone, büyük miktarlarda veriyi yüksek çözünürlüklü video şeklinde üretir. İş tesislerinde büyük kamyonlar, vinçler ve döner kepçeler ile çalışmalar koordine etmek zordur. Trafik yönetiminde gecikme yöntemleri, videoLU gözetim için insanlı helikopter kullanmaktadır. İnsansız hava aracı olan drone'lar ise iş tesislerini içinde 24 saat fotoğraflayarak tesis yöneticilerine kaynaklarının nasıl kullanıldığına dair bir dakikaya kadar olan görüntüler sağlayabilirler. Edge computing kullanmak, drone'ların gerçek zamanlı veri iletimi yapmasını ve talimatları zamanında almasını sağlar.

### UYGULAMA 3: ŞİRKET İÇİ UYGULAMALAR

IT ve ağlarının kullanılabilirliğini muhafaza etme ve artırma ihtiyacı hemen hemen her zaman en öncelikli endişe konusudur. Bulut bilişim her zaman merkezi bir mimari olmuştur. Edge computing, bulut bilişimi daha dağıtık bir bilişim bulut mimarisine dönüştürür. Temel avantajı, herhangi bir bozulmanın tüm ağ yerine sadece ağdaki bir noktaya sınırlı kalmasını sağlamasıdır. Örneğin bir DDoS (Distributed Denial of Service) saldırısı ya da uzun süreli bir elektrik kesintisi, merkezi bir bulut veri merkezi üzerinde çalışan tüm uygulamalar yerine, edge computing cihazıyla ve bu cihazda çalışan yerel uygulamalarla sınırlı olacaktır. Şirket dışı bulut bilişime geçiş yapmış olan şirketler, daha fazla yedeklilik ve kullanılabilirlik için edge computing avantajlarından yararlanabilir. İş kritik uygulamalar veya işletmenin temel fonksiyonlarını gerçekleştirmek için gereklili uygulamalar şirket içinde çoğaltılabilir. Küçük bir kasabanın çok büyük bir paylaşılan su kaynağını temel su kaynağı olarak kullanması şeklinde düşünebiliriz. Bu su kaynağının, ana

kaynak veya dağıtım şebekesinde yaşanan bir aksaklık nedeniyle kesintiye uğradığı durumlar için kasabada bir acil su tankı bulunmaktadır.

## EDGE COMPUTİNG TÜRLERİ

Edge computing'in genel olarak üç çeşidi vardır.

### Yerel Cihazlar

Tanımlanmış ve belirlenmiş bir amacı karşılamak için ölçeklendirilmiş cihazlar. Dağıtım anlıktır. Ev veya küçük ofis uygulamaları için uygundur. Bir binanın güvenlik sistemini çalıştırmak (Intel SOC cihazı) veya yerel video içeriğini bir DVR'da depolamak, örnek olarak verilebilir. Başka bir örnek ise yerel bir cihaz olan ve genellikle SOAP ya da REST gibi bulut depolama API'lerini çeviren bir ağ cihazı ya da sunucusu olan bir bulut depolama ağ geçididir. Bulut depolama ağ geçitleri, kullanıcıların uygulamaları bulutun içine taşımadan, bulut depolamayı uygulamaların içine entegre etmesine olanak tanır.

### Yerelleştirilmiş (1-10 Kabin) Veri Merkezleri

Bu veri merkezleri, önemli işleme ve depolama yetenekleri sağlar ve mevcut ortamlarda hızlı dağıtım yapar. Genellikle önceden tasarlanmış ve daha sonra yerinde monte edilmiş siparişe göre yapılandırılmış sistemler şeklinde mevcuttur. Yerelleştirilmiş veri merkezlerinin başka bir çeşidi ise bir fabrikada monte edildikten sonra tesise bırakılmış prefabrik mikro veri merkezleridir. Bu tek panolu sistemler, yağmur geçirmez, korozyona dayanıklı, yangına dayanıklı, vb. gibi sağlam pano donanımlarına sahip olabilir veya bir ofis ortamı için normal IT panoları şeklinde sağlanabilir. Tek kabinli çeşitler, mevcut bina, soğutma ve güç ekipmanlarından yararlanarak, yeni bir dedike tesis inşa etmenin getirdiği yatırım maliyetlerinden tasarruf etmenizi sağlar. Kurulum, binanın güç ve fiber kaynağına yakın bir konum seçilmesini gerektirir. Çok kabinli çeşitleri, ölçüye sayesinde daha çok özellikli ve esnektir, ancak daha fazla planlamaya ve daha uzun bir kurulum süresine ihtiyaç duyar ve kendi özel soğutma şeklini gerektirir. 1-10 kabinli bu sistemler, düşük gecikme gerektiren ve/veya yüksek bant genişliğine sahip ve/veya ilave güvenlik veya kullanılabılırlik ihtiyacı olan geniş bir uygulamalar tabanı için uygundur.

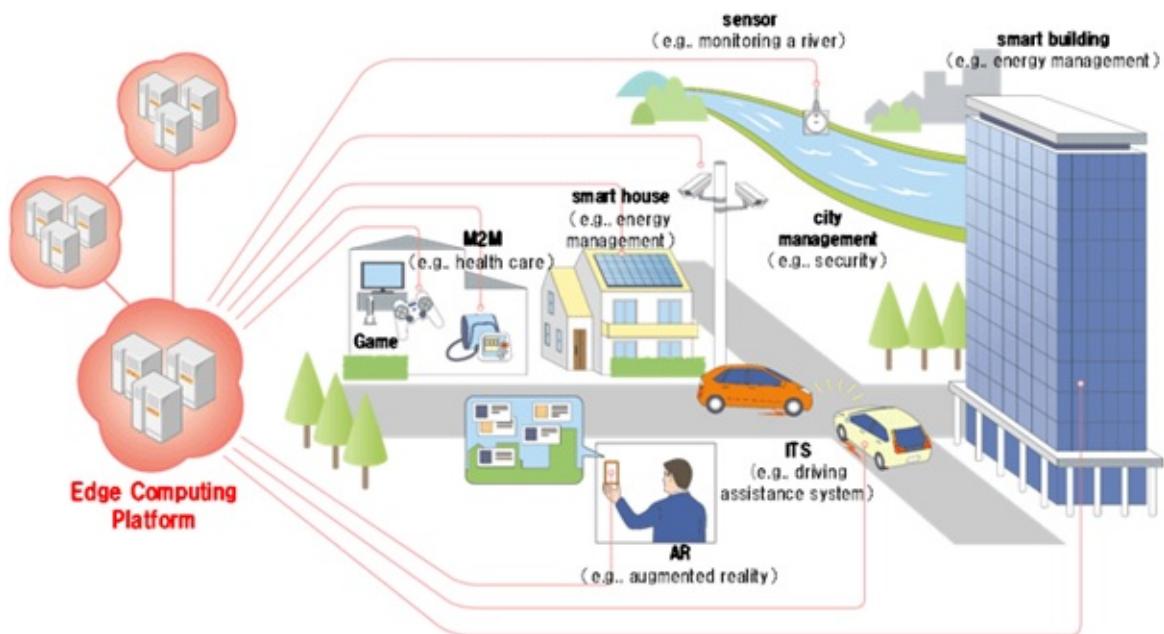
### Bölgesel Veri Merkezleri

10'dan fazla kabine sahip ve kullanıcılarla ve veri kaynaklarına merkezi bulut veri merkezlerinden daha yakın yerlerde bulunan veri merkezlerine bölgesel veri merkezleri denir. Ölçekleri sayesinde, 1-10 kabinli yerel veri merkezlerine göre daha fazla işleme ve depolama özelliğine sahiptir. Prefabrik bile olsalar, olası inşaat ihtiyacı, izin ve yerel uyumluluk konuları nedeniyle yerel veri

merkezlerine göre inşa edilmeleri daha uzun zaman alacaktır. Ayrıca, dedike güç ve soğutma kaynakları da gerekecektir. Gecikme, aradaki atlama sayısının yanı sıra kullanıcılara ve verilere olan fiziksel yakınlığa bağlı olacaktır.

### Sonuç

Edge computing gecikme sorunlarını çözer ve şirketlerin bir bulut bilişim marisinden yararlanarak fırsatları avantaja dönüştürebilmelerini sağlar. Bant genişliği açısından yoğun videolardan oluşan iş yükleri, ağ bağlantılarında titkanıklıklara ve gecikmelere neden olur. Edge veri merkezleri, bant genişliği açısından yoğun içerikleri son kullanıcıya ve gecikmeye duyarlı uygulamaları verİYE yaklaştırır. Bilişim gücü ve depolama yetenekleri, taşıma süresini kısaltmak ve kullanılabilirliği artırmak için doğrudan ağır ucuna eklenir. Edge computing çeşitleri arasında, yerel cihazlar, yerel veri merkezleri ve bölgesel veri merkezleri bulunmaktadır. Yerelleştirilmiş 1-10 kabinli çeşitleri, gelecek teki IoT uygulama talepleri ile uyumlu bir dağıtım hızı ve kapasitesi sağlar. Bunlar, ister sipariş üzerine yapılandırılmış, ister prefabrik olsun hızlıca tasarlanıp kolayca kurulabilir.<sup>2 3</sup>



<sup>2</sup><http://www.makinatek.com.tr/arsiv/yazi/161-edge-computing-dinamikleri-ve-avantajlari>

<sup>3</sup><https://medium.com/@rshariffdeen/edge-computing-vs-fog-computing-5b23d6bb049d>

### 1.3.3 Fog Computing

Sis bilişim (Fog Computing), akıllı cihazların ürettikleri veriyi merkezi bir sunucuya gönderip de işlenmesini sağlayan mimarinin aksine, önce yerel bir noktada analiz edilmesini ve ihtiyaç olunan kadarının merkezi sunuculara gönderilmesini öneren mimaridir. Örneğin bir evdeki akıllı yanın sensörlerinin tümünün ürettikleri tüm verileri İnternet üzerinden bir bulut sistemine göndemesi yüksek bant genişliği ihtiyacı doğuracaktır. Bunun aksine sis bilişim örneğinde, üretilen veri aynı evdeki bir sistemde analiz edilerek, yanının durumunda merkezi sistemlere iletilmesi daha doğru bir yaklaşım olacaktır. Bu mimarinin avantajları:

- Verinin üretildiği yere daha yakında işlenmesi
  - düşük bant genişliği ile çalışabilirlik
  - merkezi sunucuya bağımlılığın azalması
  - güvenlik ve gizlilik olarak sıralanabilir.<sup>4</sup>

Bulutun Yetemediği Sınırlarda Sis Kullanacağız

Bulut (Cloud) teknolojiler bile hayatımızda çok yeniyken, bu teknolojilerin verimlilikleri, yararları ve gereklilikleri halen pek çok işletme tarafından soruluyorken, 2015 yılının Kasım ayında ARM, Cisco, Dell, Intel, Microsoft ve Princeton Üniversitesi'nin destekleriyle kurulan OpenFog Consortium olaya bambaşka bir bakış açısı getirdi. Bulut teknolojisinin günlük hayatımıza yansımاسının en basit örneğini, cep telefonlarımızda tutmadığımız birçok uygulama bilgisinin kablosuz bağlantılarla buluttan avcumuzun içine servis edilmesinde görebiliriz. Bu akış/iletişim süresince çok büyük yapıların kritik bilgilerinin acil olarak iletilmesi ya da işlenmesi gerektiği noktalarda verinin kat ettiği uzun yollar ya da yol boyunca uğradığı duraklar sorun olmaya başlamakta. IoT teknolojilerinin yaygınlaşması ve ürettikleri verilerin büyümesi ile bu sorunun ne kadar çok karşımıza çıkabileceğini siz de görebilirsiniz. OpenFog Consortium da bu gibi sorunları aşmak için kuruldu.

## Sis Bilişimini IoT'ye Katkısı Ne Yöndedir?

Akıllı sensörlerin, aygıtların ve IoT'nin yaygınlaşması ile birlikte elde edilen verilerden ortaya çıkacak analizlerin doğru veri ile doğru lokasyonda, networkte ve zamanda yapılması ihtiyacı daha da arttı. Sis teknolojisinin bu ortamda sunduğu hizmetler ise sunlar:

- Verinin buluta gönderilip işlenmesi ve depolaması esnasında maliyeti düşer.

<sup>4</sup><http://www.wiki-zero.net/index.php?q=aHROcHM6Ly9lb1i53aWtpcGVkaWEub3JnL3dpa2kvRm9nX2NvbXB1dGluZw>

şürmek,

- Verinin oluşturduğu kaynağı ya da ortamı terk etmemesi istendiğinde, veri güvenliğini sağlayarak bulut iletişimini minimize etmek,
- Gerçek zamanlı analizlerde ve karar aşamalarında hız sağlamak,
- Çok kritik görülen bazı durumlarda olası bulut iletişiminin sağlanamamasından kaynaklanacak riski minimize etmek.<sup>5</sup>

#### 1.3.4 Cloud Computing

Bulut bilişim (cloud computing), bilgisayarlar ve diğer cihazlar için, istediği zaman kullanılabilen ve kullanıcılar arasında paylaşılan bilgisayar kaynakları sağlayan, internet tabanlı bilişim hizmetlerinin genel adıdır. Bulut bilişim bu yönyle bir ürün değil, hizmettir; temel kaynaktaki yazılım ve bilgilerin paylaşımı sağlanarak, mevcut bilişim hizmetinin; bilgisayarlar ve diğer aygıtlardan elektrik dağıticılara benzer bir biçimde bilişim ağı (tipik olarak Internet'ten) üzerinden kullanılmasıdır.<sup>6</sup>

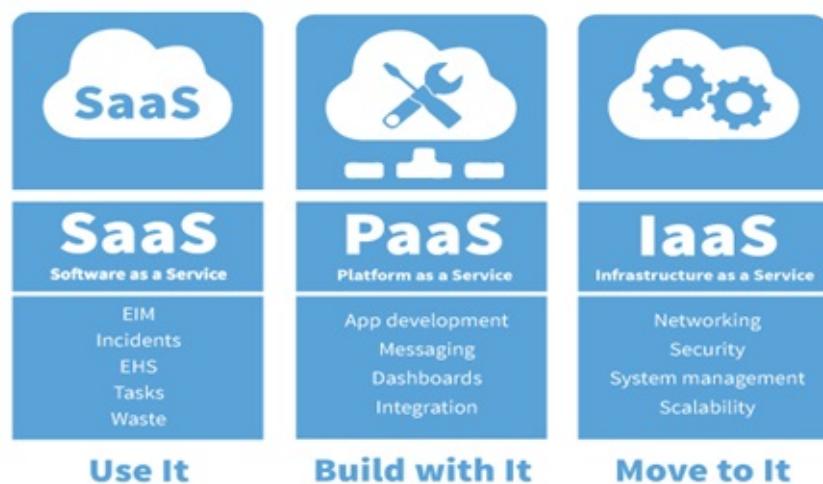
Bulut bilişim fikrinin temelleri 1950'li yıllarda atılmıştır. İnternet devlerinden olan Amazon, veri merkezlerini modernize ederek bulut bilişimin gelişmesinde anahtar bir rol oynayarak ilk gerçek bulut bilişim hizmeti olan Amazon S3'ün 2006 yılında hizmete girmesini sağladı. 2008'in ortalarına gelindiğinde, Gartner (Danışmanlık ve Araştırma Şirketi) bulut bilişimi bilgi teknoloji hizmetleri sektöründe hem kullanıcılar hem de tedarikçiler arasındaki ilişkiyi değiştirebilecek potansiyeli işaret etti. 2008 den bu yana Dünya'da yaygın bir şekilde kullanılmaya başlandı. Son birkaç yıldır ülkemizde yaygınlaşan bu teknoloji oldukça hızlı bir şekilde büyümeye devam ediyor.



<sup>5</sup><https://proente.com/bulut-un-yetmedig%C3%86i-yerlerde-sis-kullanacag%C3%86iz/>

<sup>6</sup><http://www.wiki-zero.net/index.php?q=aHR0cHM6Ly90ci53aWtpcGVkaWEub3JnL3dpa2kvQnVsdXRfYmlsaSVDNSU5Rmlt>

Günümüz teknolojisinde ki mevcut cihazlarda kullanıcılar her geçen gün daha fazla kişisel veri ve data saklamak istediği için barındırma kapasitesi büyük sorunlara sebep olmaktadır. Bununla birlikte cihazların özellikleri, kapasiteleri gittikçe artıyor. Bilgisayar, notebook, netbook, ve taşınabilir akıllı cihazların teknoloji ve kapasitesinin artmasıyla orantılı olarak fiyatlar da yükseliyor. Tüm bu sorunlara çözüm olarak ortaya çıkan Bulut (Cloud) Teknolojisi, internet üzerinden, erişimde bulunulan yazılım uygulamaları, veri depolama hizmeti ve işlem kapasitesi olarak tanımlanmaktadır. En düşük kapasiteli cihazla bile istenilen yerden istenildiği zaman her tür bilgiye, kişisel veriye ulaşmayı sağlıyor. Tüm bu işlemler için, dijital bir ağ aracılığıyla çoklu sunucu bağlantısı gerçekleştiriyor. Bulut teknolojisinin üç yapitaşı ise SaaS (Software as a Service); yazılımı servis olarak sunma, PaaS (Platform as a Service); platform hizmeti ve IaaS'tır (Infrastructure as a Service); sunucu altyapı hizmeti.



Bulut teknolojisi şirketler, üniversiteler vb. büyük kuruluşlar tarafından kurulur ve paylaşılır. Bu teknolojiyi kullanmak, kişisel bilgisayarların yükünü azaltır ve çeşitli sayıda uygulama, bulut sunucusu tarafından sağlanır. Genellikle, kullanıcılar uygulamaları bilgisayarını indirmek ve yüklemek istemezler. Tüm işlemler ve depolamalar, bulut sistemi tarafından sağlanır. İnternet üzerinde barındırdığımız tüm uygulama, program ve verilerimizin sanal bir makine üzerinde yani en çok kullanılan adıyla bulutta depolanması ile birlikte internece bağlı olunan cihaz ile her lokasyon da bu bilgilere, programlara ve verilere kolaylıkla ulaşım sağlanabilir.



### Bulut Bilişim'in Geliştirme Modelleri

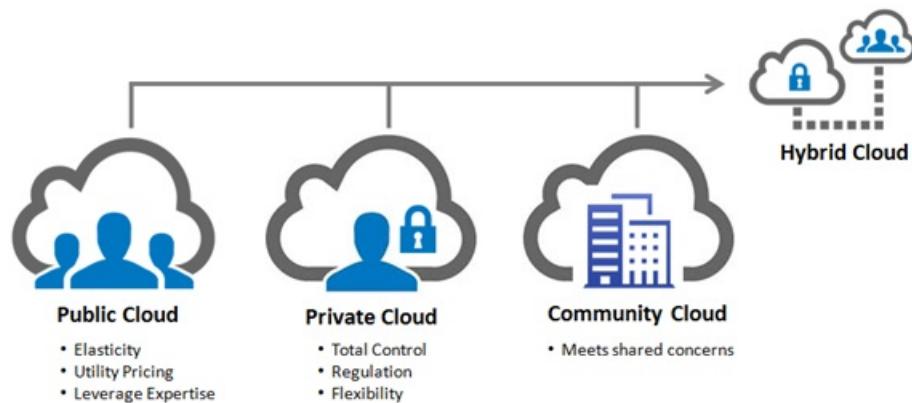
4 ayrı çeşidi ile karşımıza çıkan bu teknoloji farklı alanlarda,farklı biçimlerde kullanılmaya olanak sağlıyor.

**Public Cloud (Genel Bulut):** İnternet üzerindeki sunucular ile kurulan bir bulut teknolojisi. Küçük ve orta ölçekli şirketlerde kullanacağınız kullandığınız kadar ödeme yapılan bu modele örnek olarak, elektronik postalar gösterilebilir.

**Private Cloud (Özel Bulut):** Bilgileri önemli olan büyük şirketlerin tercih ettiği bir bulut teknolojisidir. Tüm bilgiler kurucunun elinin altındadır ve erişim güvenliği ve gizliliği yüksektir. Microsoft bunu size Hyper-V ve System Center Ürün Ailesi yardım ile sağlamaktadır.

**Hybrid Cloud (Melez Bulut):** Public ve Private Cloud'un birleşiminden ortaya çıkan bulut teknolojisidir.Şirketlerin hacmine göre birleşim oranlarında farklılıklar görülebiliyor.

**Community Cloud (Topluluk Bulut):** Birkaç şirket ile ortak kullanılan hizmetleri barındıran bulut teknolojisidir. Topluluk üyeleri uygulama ve verilere erişebilmektedir.



### Bulut Teknolojisi'nin Getirdiği Avantajlar

- Bulut bilişim sistemleri API'ler ile hızlı kullanım kolaylığı sağlıyor.
- Daha fazla depolama alanı, hızlı veri transferi ve bu yedekleme üzerinde maliyet tasarrufu yapabilme gibi bir takım olanaklar sağlıyor.
- Sürekli olarak artan verilerin arşivlenmesi, kullanıcıların yetki ve takibi gibi konuların oluşturduğu alt yapı karmaşası ortadan kalkıyor.
- Bulut teknolojisi yazılımları web tarayıcıları üzerinden çalıştığından, bilgisayar, tablet, akıllı telefon ve Smart TV'ler de kullanılarak platform bağımlılığından koruyor.
- Bulut yazılım hizmetini veren şirketlerin verilerinin tutulduğu serverları 7/24 yazılım ve donanımsal olarak güvenlik tedbirlerini aldıklarından dolayı ana bilgisayardan daha güvenlidir.

Kısaca; **bulut bilişim çok daha ucuza, kurulum gerektirmeden, her yerden çalışmayı desteklen bir hizmettir.**

### Bulut Teknolojisi'nin Dezavantajları

- Bulut teknolojisi servisi kullanarak veri saklanması, kullanıcının verilerini riske atması bilgi güvenliğini ve kullanıcı gizlliğini sağlayamamaktadır. Güvenlik açıkları oldukça fazladır.

- Ülkelerin ekonomik durumlarından dolayı dijital bölünmeyi artıracak, bu da uluslararası, politik ve ekonomik sorunlar doğuracaktır.
- En önemli sorun ise depolanan verilere ulaşılabilmesi için internet bağlantısının olması gerekmektedir. Yani internet olmayan durumlarda bilgilerimize erişmek söz konusu değildir. İnternete bağlı olarak düşük hızlı internete sahipseniz veri alış-veriş hızınız da o derecede daha yavaş olacaktır.
- Hizmetlerinin gelişmesiyle birlikte donanımsal ve yazılımsal bakım ve tamir maliyetlerinin azalacak olması ve buna bağlı olarak da bu işi yapan Bilgi Teknolojisi (BT) uzmanlarının iş sahalarının daralması durumu da son dezavantajlardan birisidir.

### Big Data ( Büyük Veri) ve Bulut İlişkisi

**Big Data** (büyük veri) ; zamanla elde edilen, yapılandırılmış ya da yapılandırmamış, yani henüz geleneksel yöntem veya araçlarla işlenerek kullanılabilir hale getirilmemiş verilerdir. Kısaca bilgisayarın işleyemeyeceği kadar büyük veriler anlamına geliyor diyebiliriz.



Son birkaç yılda verilerin % 90'ı üretildi. Veriler mobil cihazlarımız, yazılım

kayıtları, kameralar, mikrofonlar, sosyal medya, internetteki tüm hareketlerimiz şu anda bilgi akışında işlenmek üzere depolanıyor. Kısa bir sürede neredeyse tüm hareketlerin bir bilgi akışı olarak sunuculara yönlendirileceği düşünülmektedir. Bulut bilişim olanaklarının, depolama ve bilişim gücü sınırlarını ortadan kaldırması büyük verinin önünün açılmasına yardımcı olmuştur. Son yıllarda verilerin, boyut, çeşitlilik ve karmaşıklık anlamında sürekli büyümeye devam edecek olması, büyük veri konusunu bir sorun olmaktan çıkarıp bulut bilişimle birlikte bir çözüm odağı haline gelmesi sağlanmıştır.

Teknolojinin, nesnelerin interneti, big data ve bulut bilişim ile çıktığı bu yolda, kullanıcılar kaçırmaz imkanlar sunacağı benziyor. İlerleyen zamanlarda geliştirilecek yeni teknolojiler sayesinde bu ailenin büyümesi hedefleniyor.

### Bulut Teknolojisi Hizmeti Veren Platformlar

- Dropbox (<http://dropbox.com>)
- Google Drive (<http://drive.google.com>)
- SkyDrive (<https://skydrive.live.com>)
- Cloud (<https://cloud.google.com>)
- Yandex.Disk (<http://disk.yandex.com>)
- Turkcell Akıllı Bulut (<http://turkcellakillibulut.com>)
- TTNET Bulut (<http://ttnetbulutu.com>)
- Ubuntu One (<https://one.ubuntu.com>)
- Box(<http://box.com>)

### Bulut Bilişim Uygulamaları

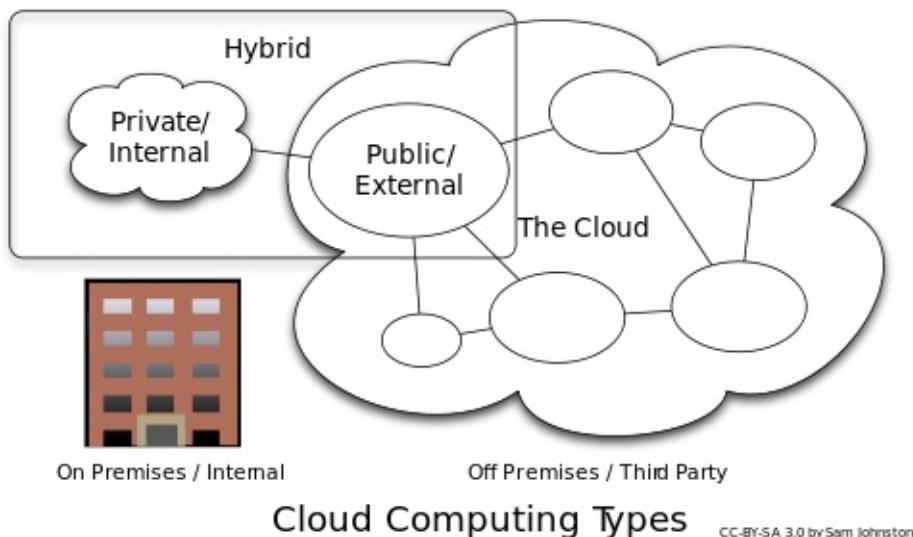
- Karelport.com: Bulut üzerinden hizmetler, fırsatlar ve sosyal paylaşım olanakları sunan online iş çevresi
- Amazon Elastic Compute Cloud (Amazon EC2): Ölçeklenebilir ve kullanım kapasitesi kadar ücretleme yapılan Amazon.com'un bulut işlem gücü servisi

- Fizy.com: Bulut üzerinde müzik dinleme servisi
- Google Apps: Ofis, veri ve iletişim amaçlı online uygulamalar
- Salesforce.com: Salesforce'un müşteri ilişkileri yönetimi (CRM) servisi

### Türkiye'de bulut bilişimi

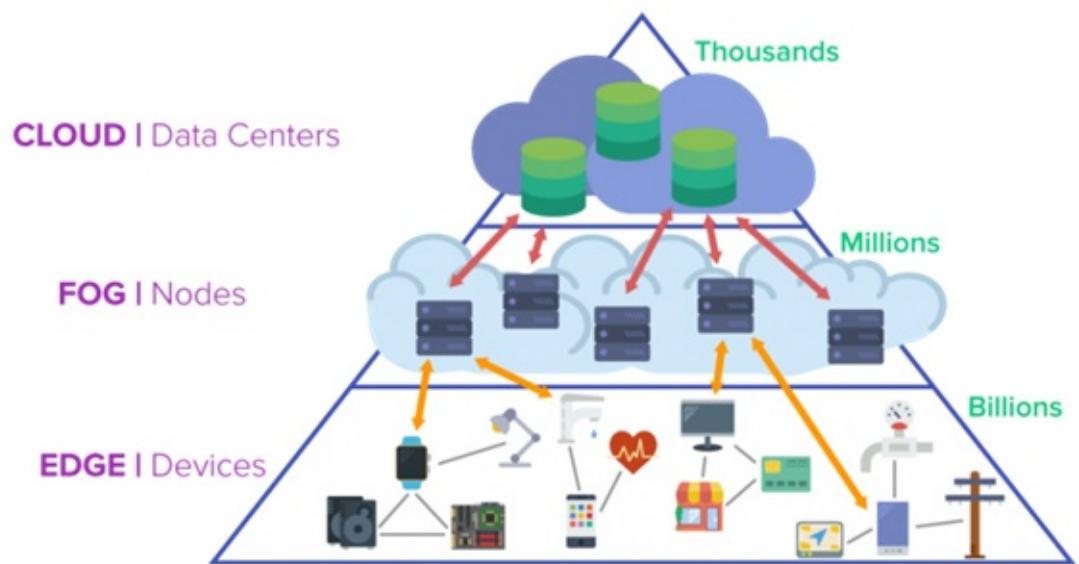
Türkiye'de bulut bilişime yönelik yoğun bir ilgi söz konusudur, Kamu sektöründe Aile ve Sosyal Politikalar Bakanlığının yaptığı çalışmalar dikkat çekicidir. Fakat bulut bilişim alanında verilerin güvenliği konusunda herhangi bir yasal düzenleme mevcut değildir.

Bussiness Software Alliance (BSA) tarafından yayınlanan küresel çapta bulut bilişimin gelişiminde etkili olan ülkeler sıralamasında 24 ülke arasında Türkiye 17. sıradadır.<sup>7</sup> <sup>8</sup>



<sup>7</sup><http://www.endustri40.com/bulut-bilisim-cloud-computing-nedir/>

<sup>8</sup><https://erpinnews.com/fog-computing-vs-edge-computing>



### 1.3.5 IoT (Internet of Things)

IoT'nin açılımı “Internet of Things” olup, Türkçede “Nesnelerin İnterneti” olarak isimlendirilmektedir.



IoT, veri analizi ve alışveriş yapmak üzere elektronik bir sistem, yazılım ve sensörlere sahip cihazlardan oluşan güçlü bir ağı ifade eder. Yani çeşitli haberleşme protokolleri sayesinde birbirleri ile haberleşen ve birbirine bağlanarak, bilgi paylaşımı yapan akıllı bir ağ oluşturmuş cihazları temsil etmektedir. Günlük hayatta kullandığımız her nesne internete bağlı olup, bir MAC

veya IP adresine sahiptir. Bu şekilde birbirleri ile sürekli haberleşme halinde olmaları mümkündür.



1991 yılında Cambridge Üniversitesi 'ndeki yaklaşık 15 akademisyenin kahve makinesini görebilmek için kurduğu kameralı sistem o günün koşullarında değerlendirildiğinde ufuk açıcı bir uygulamaydı. 2001 yılına kadar kullanılan sistem, kahve makinesinin görüntüsünü dakikada üç kez bilgisayar ekranlarına gönderiyordu. Çevrim içi ve gerçek zamanlı olması sebebiyle "Nesnelerin İnterneti" kavramının ilk örneği olarak tarihte yerini aldı.

Kevin Ashton tarafından 1999 yılında ortaya atılan bu kavram ilk başlarda RFID etiketleri sayesinde radyo frekansı üzerinden birbirleriyle haberleşen cihazları kapsıyordu. Önerdiği sistem, "Internet of Things" kavramını ortaya çıkarılan radyo dalgaları ve sensörlere dayalı bir küresel sistem standarı idi. Ancak gelişen teknoloji ile konsept çok daha geniş bir vizyona erişti. Artık internete bağlı olan cihazlar, yalnızca akıllara ilk gelen bilgisayar, telefon, tablet gibi araçları değil, evimizdeki eşyaları, yoldaki trafik ışıklarını, fabrikalarda üretim yapan makineler gibi çoğu cihazı kapsamaktadır. Yani IoT, şu anda internete bağlanan, veri toplayan ve paylaşan milyarlarca fiziksel cihaza işaret eder. Bu cihazlar internete bağlı olup, birbirleriyle de haberleşebilmektedir.

### Günümüzdeki Durum nedir?

Yapılan araştırmalara göre bugün interne'ye 10-11 milyar cihazın bağlı olduğu tahmin edilmekte ve bu rakamın 2020 yılına gelindiğinde 50 milyar cihaz seviyesine çıkması öngörülmektedir. Aynı araştırmalara göre, 2003 yılında dünyada kişi başına düşen birbirleriyle bağlantılı cihaz oranı 0,08 iken bu oranın 2020 tahmini ise 6,48 'dir. Ayrıca 2020 yılında, 20 adet tipik ev cihazının üreteceği bilgi trafiğinin, 2008 yılında üretilen tüm internet trafi-



ından daha fazla olacağı tahmin edilmektedir.

Table 1: IoT Units Installed Base by Category (Millions of Units)

Category	2016	2017	2018	2020
Consumer	3,963.0	5,244.3	7,036.3	12,863.0
Business: Cross-Industry	1,102.1	1,501.0	2,132.6	4,381.4
Business: Vertical-Specific	1,316.6	1,635.4	2,027.7	3,171.0
<b>Grand Total</b>	<b>6,381.8</b>	<b>8,380.6</b>	<b>11,196.6</b>	<b>20,415.4</b>

Source: Gartner (January 2017)

Akıllı bileklikler, akıllı saatler, akıllı gözlükler, akıllı tshirtler, akıllı raketler, ev otomasyon sistemleri, akıllı arabalar ve birçoğu. Nesneler üzerine yapılan yatırımlarla, hepsi yakın bir zamanda Wi-Fi ve Bluetooth teknolojisi ile internete bağlanıyor olacak. Bağlanılamayan bölgelerde de Mark Zuckerberg'in Internet.org projesi altında hayatımıza dahil etmeyi planladığı dronelar devreye girecek. Akıllı ürünler de internete yakınlarında bulunan modemler dışında dronelar sayesinde de girebilecek ve kullanıcılara bildirileri aktarabilecek.

Internet of Things kavramı altında geliştirilen ürünler genellikle mobil cihazlar ve tabletlerle birlikte çalışmaktadır. Her nesnenin mobil üzerinden erişilebilen bir uygulaması olmakta ve böylece bu uygulama sayesinde nesneler bildirimleri uygulamalara aktarılabilmektedir. Mesela Google tarafından satın alınan akıllı ürün Nest, evde olmadığınız durumlarda evin sıcaklığını

uygulama üzerinden kontrol edebilmenizi sağlar. Uygulamayı, eve yerleştirilen Nest ile entegre ederek bunu gerçekleştirebiliyorsunuz.



### IoT Çalışma Prensibi

Yerleşik sensör'lere sahip cihazlar ve nesneler, farklı cihazlardan verileri bütünlüştiren ve en değerli bilgileri belirli ihtiyaçlara yönelik olarak oluşturulan uygulamalarla paylaşmak için analitik uygulayan bir Nesnelerin İnterneti platformuna bağlanır. Bu güçlü IoT platformları, hangi bilgilerin yararlı olduğunu ve nelerin güvenli bir şekilde göz ardı edilebileceğini tam olarak belirleyebilir. Bu bilgi kalıpları tespit etmek, önerilerde bulunmak ve oluşabilecek sorunları tespit etmek için kullanılabilir.

Örneğin, bir araba imalat işletmemiz varsa, hangi istege bağlı bileşenlerin (örneğin deri koltuklar veya alaşım jantlar) en popüler olduğunu bilmek isteyebiliriz. Nesnelerin İnterneti teknolojisini kullanarak şunlar yapılabilir;

- Bir showroom'da hangi alanların en popüler olduğunu ve müşterilerin en uzun süre en çok nerede bulunduğu tespit etmek için sensörleri kullanın.
- Hangi bileşenlerin en hızlı sattığını belirlemek için mevcut satış veriline bakın.

- Satış verilerini stokla otomatik olarak hizalayın, böylece popüler öğeler stokta kalmaz.
- Popüler ürün stokta kalmadığında, otomatik olarak stok bilgisini satış verilerinin kaynağı ile hizalayın.

Bağlı cihazlar tarafından toplanan bilgiler, zaman ve paradan tasarruf etmeye yardımcı olan, gerçek zamanlı bilgilere dayanarak hangi bileşenlerin üzerine stoklanacağı konusunda akıllı kararlar vermemizi sağlar.

Gelişmiş analitik tarafından sağlanan anlayışla, süreçleri daha verimli hale getirme gücü beraberinde gelmektedir. Akıllı nesneler ve sistemler, belirli görevleri, özellikle bunlar tekrarlayan, sıradan, zaman alıcı ve hatta tehlikeli olduğunda otomatikleştirebileceğiniz anlamına gelmektedir.

Şöyledir bir senaryo düşünelim;

- Şehir dışında bir toplantıınız var ve toplantıya trenle gideceksiniz.
- Müşteri sabahki toplantıyı 45 dakika ileri alıyor.
- Arabanız deposunu kontrol ediyor ve tren istasyonuna gidebilmeniz için yakıt alması gerektiğini biliyor. Bu yakıt doldurma işleminin tahmini 5 dakika süreceğini hesaplıyor.
- Aynı anda trafik durumu kontrol ediliyor ve istasyona giden yolda kaza olduğunu öğreniliyor, alternatif yol ise size 15 dakika kaybettirecek.
- Tren ise 20 dakika rötar yapacağını bildiriyor.

Bunların hepsi yatağınızın baş ucundaki saatte bildiriliyor ve;

- Alarminız 5 dakika daha geç çalışıyor, bu da 5 dakika daha çok uyumanızı sağlıyor.
- Saatiniz arabaniza haber veriyor ve evden çıkış saatinizden 5 dakika önce çalışmaya başlamasını ve motoru ısıtmasını söylüyor.
- Ayrıca kahve makinesine de 5 dakika daha geç çalışmasını söylüyor.



Özetle her şeyin interne bağılı olduğu ve İnternet ‘in hava, su kadar önemli olduğu zamanlara doğru ilerliyoruz. Bu makineler oldukça akıllı ve kendilerini daha akıllı hale getirmek üzere programlanmışlar. Dış dünyadan aldıkları veriyi birbirleri ile paylaşarak hem daha çabuk öğreniyorlar hem de birçok fonksiyonu daha kolay gerçekleştirir hale geliyorlar. Aynı zamanda, donanım ve yazılım geliştirmeleri ile kapasiteleri artırmak da her zaman mümkün. Bu makineler hayatımızın her aşamasında bizi bilgilendirici ve yönlendirici olacaklar, hayatımızın her anından veri topluyor olacaklar. Öyle ki bir zaman sonra en büyük danışmanlarımız hatta karar vericilerimiz olacaklar.

### **IoT Güvenliği Hakkında**

Güvenlik, IoT ile ilgili en büyük sorunlardan biridir. Bu sensörler birçok durumda son derece hassas verileri toplar. Örneğin kendi evinizde ne söyler ve yaparsınız? Bunu güvende tutmak tüketicinin kişisel güvenliği için hayatı önem taşımaktadır, ancak şu ana kadar IoT ’nin güvenlik sicili çok zayıftır. Çok fazla sayıda IoT cihazı, güvenlik ve veri aktarımındaki verileri şifrelemek gibi temel güvenlik konularına çok az önem vermektedir.

Yazılımdaki kusurlar, (eski ve iyi kullanılmış kodlar bile) düzenli olarak saptanmaktadır, ancak birçok IoT cihazı yamaya desteklenme yeteneğinden yoksundur, bu da kahıcı olarak risk altında oldukları anlamına gelir. Hackerlar artık yönlendiriciler ve web kameraları gibi IoT cihazlarını etkin bir



şekilde hedefleyebilmektedir. Çünkü güvenlik seviyesinin yetersizliği, bunları devasa botnetlere dönüştürmeyi ve yuvarlanması kolaylaştırıyor.

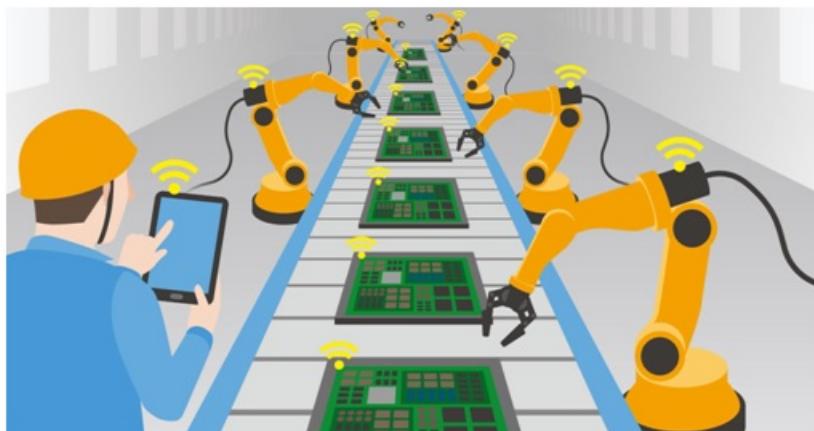
Kusurlar, buz dolapları, fırınlar ve bulaşık makineleri gibi akıllı ev cihazlarına hackerlar tarafından erişimi açık bırakmış oldu. Araştırmacılar, internete bağlı bazı akıllı saatlerin, bilgisayar korsanlarının kullanıcının yerini tespit etmesine, konuşmaları dinlemesine ve hatta kullanıcıyla iletişim kurmasına olanak tanıyan güvenlik açılarını içerdigini saptadılar. Kolaylıkla saldırıyla uğramış 100.000 web kamerası buldu.

Bir cihazı akıllı yapma maliyeti göz ardı edilebilir olduğunda, bu problemler sadece daha yaygın ve çekilmez hale gelecektir.

IoT, dijital dünya ile fiziksel dünya arasındaki uçurumu kapatır, bu da cihazlara girişin tehlikeli, gerçek dünya sonuçlarına sahip olabileceği anlamına gelir. Bir elektrik santralindeki sıcaklığı kontrol eden sensörlerle girmek, operatörleri feci bir karar vermeye zorlayabilir, sürücüsüz bir aracın kontrolünü ele geçirmek de felaketle sonuçlanabilir.

**Farklı sektörlerde IoT'nin kullanımı****1. Üretim için IoT**

- Üretim ekipmanını izleyin Sektörel IoT'yi kullanarak süreçlerini geliştirin. Sensörlerden ve ileri düzey analizden yararlanarak hangi parçaların bakıma ihtiyacı olduğunu tahmin edin ve üretim süresinden çalan plansız kapalı kalma süresini azaltın.
- Müşteri ekipmanını izleyin Ürettiğiniz ekipman için tahmine dayalı bakım ve performans izleme olağanı sunan yeni iş modelleri oluşturarak daha zengin bir müşteri deneyimi sunun.
- Saha hizmetini geliştirin Saha hizmeti zamanlamasını geliştirmek için sensör verilerine erişin ve olası sorunlar büyük bir soruna dönüşmeden önce doğru teknisyen ve araçların sevk edildiğinden emin olun.



Şekil 1.1: Üretim için IoT

**2. Akıllı şehirler için IoT**

- Enerji kullanımını optimize edin Kullanım izleme ve akıllı şebeke-ler uygulayarak güvenilir, verimli ve daha yeşil bir enerji aktarımı sunarken müşterilerin ödediği ücretleri düşürün.

- Daha güvenli şehirler oluşturun Daha iyi düzenlenmiş trafik, daha etkili acil durum yönetimi, polis ve benzeri acil müdahalelerde daha hızlı yanıtlama için altyapıları bağlayın.
- Akıllı binalar oluşturun Bina cihazlarını ve sistemlerini bağlayarak bina sahiplerine, çalışanlarına ve sakinlerine daha verimli bir operasyon ve denetim olanağı sunun.
- Saha hizmetini geliştirin Bozuk sokak lambalarını tamir etmekten trafik ışıklarının bakımı ve çöp kamyonu rotalarını optimize etmeye her alanda kamu hizmetlerinin verimliliğini artırın.



Şekil 1.2: Akıllı şehirler için IoT

### 3. Taşımacılık için IoT

- Araç performansının kalıcı olmasını sağlayın Bakım gereksinimlerini izleyip tahmin ederek ve kapalı kalma süresi sırasında olası sorunları düzelterek hem araçlarınızın hem de işletmenizin hareket halinde kalmasını sağlayın.
- Araçlarda IoT'yi etkinleştirin İşletmenize değerli içgörüler sağlarken müşterilerin sürüs deneyimini dönüşüme uğratan IoT özellikli, yenilikçi araç özellikleri ve çözümleri oluşturun.
- Filo operasyonlarını optimize edin Gerçek zamanlı veriler ve uyarılarla lojistiği kolaylaştırarak teslimat rotalarını optimize edin,

performansı takip edin ve gecikme benzeri sorunlara yanında yanıt verin.

- Trafığın akıcı kalmasını sağlayın Taşımacılık altyapılarının yönetilmesine, yol koşullarının değerlendirilmesine ve tıkanıklığın azaltılmasına yardımcı olması için gerçek zamanlı trafik verilerini izleyin ve işleyin.



Şekil 1.3: Taşımacılık için IoT

#### 4. Perakende için IoT

- Yiyecek hizmetleri için IoT Verimliliği artırarak operasyon maliyetinizi azaltın. Kalite ve güvenliği izleyin, ekipman bakımını geliştirin, malzemeleri takibe alın.
- Otomatlar için IoT Operasyonlarınızı pürüzsüz hale getirin. Stok ve malzeme kullanımını izleyin ve makine bakım gereksinimlerini daha iyi tahmin edin.
- Süpermarketler ve perakende mağazaları için IoT İşletmenizin büyümeye hızını ve marka sadakatini artırın. Stoğu ve kullanıcı davranışlarını izleyip çeşitli ürünler önerin.



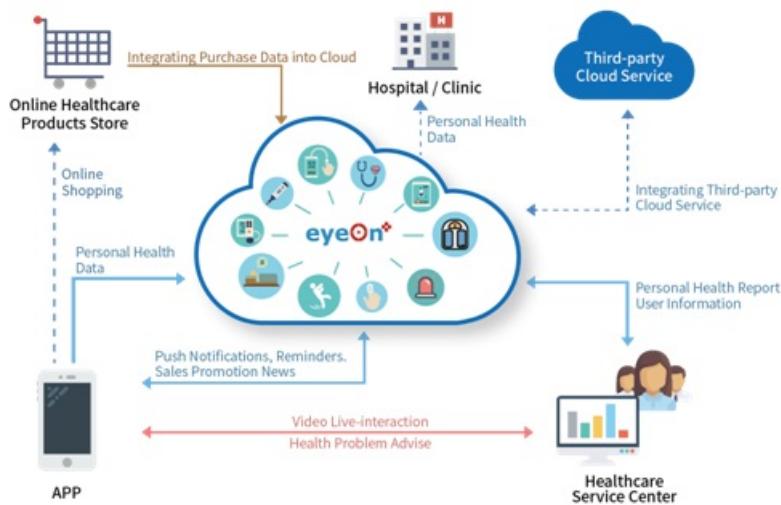
Şekil 1.4: Perakende için IoT

- Konaklama için IoT Operasyonlarınızı optimize edin ve gelirleri artırın. Oda kullanımını izleyin, konuk tercihlerine göre uyarlayın ve malzemeler tükenmeden önce haber alın.
- Stadyumlar için IoT Seyircileri gerçek zamanlı istatistikler ve izleme aracılığıyla en sevdikleri spor etkinliklerine bağlayarak seyirci deneyimini geliştirin.

##### 5. Sağlık Hizmetleri için IoT

- Hasta bakımını izleyin Hastaların en rahat oldukları yer olan kendi evlerinde bakımmasına imkan tanıyın. Giyilebilir sensörler sayesinde doktorlar bir hastanın sağlık durumunu gerçek zamanlı olarak uzaktan izleyebilir.
- Tıbbi varlıklarızı izleyin HIPAA uyumlu bir bulut platformunda malzeme ve ilaçları izleyip yöneterek personelinizin aramaya daha az, hastalarla vakit geçirmeye daha çok zaman ayırmamasına yardımcı olun.
- Hayati öneme sahip ekipmanın bakımını yapın Tahmine dayalı bakım ile sorunları henüz gerçekleşmeden düzelterek kritik tıbbi cihazların hastaların en çok ihtiyaç duyduğu an kullanıma hazır olmasını sağlayın.
- Ekipman kullanımını izleyin Hastane yatağı sensörleri kullanmadan oda sıcaklığını ve el yıkama istasyonlarını izlemeye kadar birçok alanda ekipmanların nasıl kullanıldığını izleyerek hastaların

genel sağlık durumunu iyileştirir.



Şekil 1.5: Sağlık Hizmetleri için IoT

### IoT Çözümleri

Yaklaşık yirmi yıldır yaşamımızı yeniden şekillendirmekte olan IoT, yanıt verebilen çözümlere, inovatif新产品 ve nihayetinde yeni ve muhteşem iş yapma yollarına zemin hazırlar.

- Bağlantılı fabrika
- Uzaktan izleme
- Tahmine dayalı bakım
- Bağlı saha hizmeti
- Bağlı araç
- Akıllı video
- Enerji
- Akıllı binalar

- Akıllı Şehirler
- Sağlık
- Otomotiv
- Endüstri
- Perakende
- ...vs.

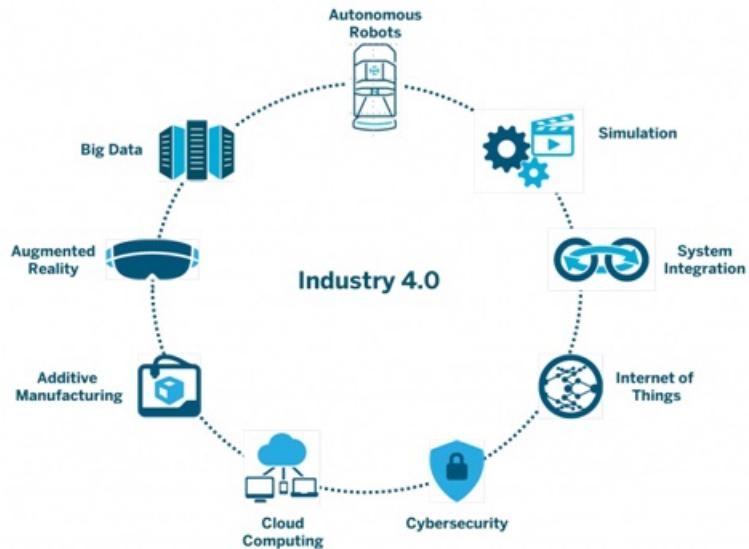


### 1.3.6 Endüstri 4.0

#### Endüstri 4.0 nedir?

Sürekli gelişen sanayimin 4. evresi: "Aydınlık Gelecek, Karanlık Fabrikalarda."

Buhar gücüyle çalışan mekanik sistemlerin kullanımını ile başlayan endüstri devrimi, elektriğin icadı, elektronikleşme ve bilgi teknolojilerinin endüstriye entegrasyonu ile günümüzdeki halini almıştır. Tarih boyunca sektörel gereksinimlerin; hızlı, güvenilir ve yenilikçi bir anlayış ile karşılamaya çalışan endüstri, hızla gelişen teknoloji olanakları sayesinde şimdilerde yeni bir sanayi devrimi olan Endüstri 4.0'in başlangıcında. Endüstri 4.0, modüler yapılı

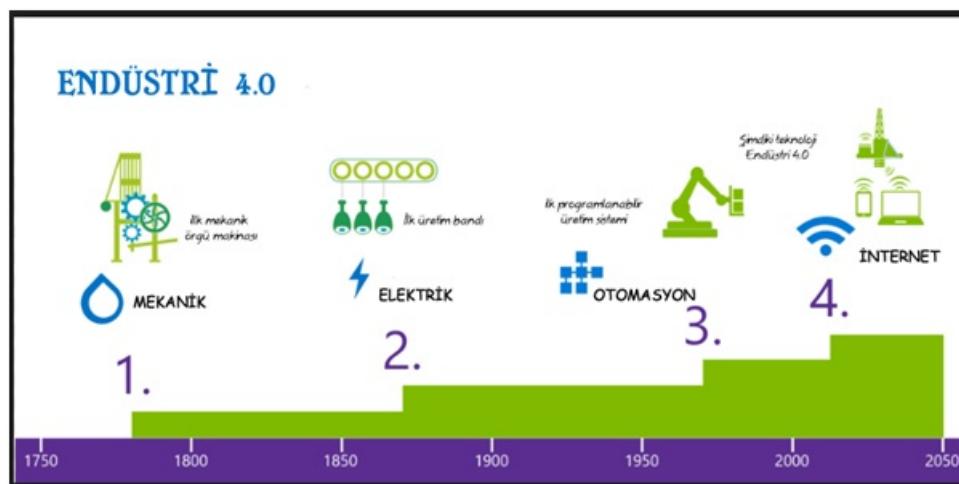


akıllı fabrikalarda, fiziksel işlemleri siber-fiziksel sistemler ile izleyerek, nesnelerin birbirleriyle ve insanlarla iletişime geçmesini ve bu sayede de merkezi olmayan kararların verilmesini hedeflemektedir. Endüstri 4.0'ın amacı insan-dan arındırılmış bir fabrika değil, daha çok bilgi teknolojileri kabiliyetlerinin kullanımıyla insan odaklı, bütün paydaşlara katma değeri yüksek organizasyonu oluşturmaktır. Sanayi temellerini oluşturmaya devam eden Türkiye gibi gelişmekte olan ülkeler, geçmiş oluşumlarını değiştirmeyi gelişmiş ülkeler gibi çok düşünmeden, belirledikleri Endüstri 4.0 Vizyonları doğrultusunda daha hızlı yatırım yapabilir ve daha rekabetçi bir konuma gelebilirler.

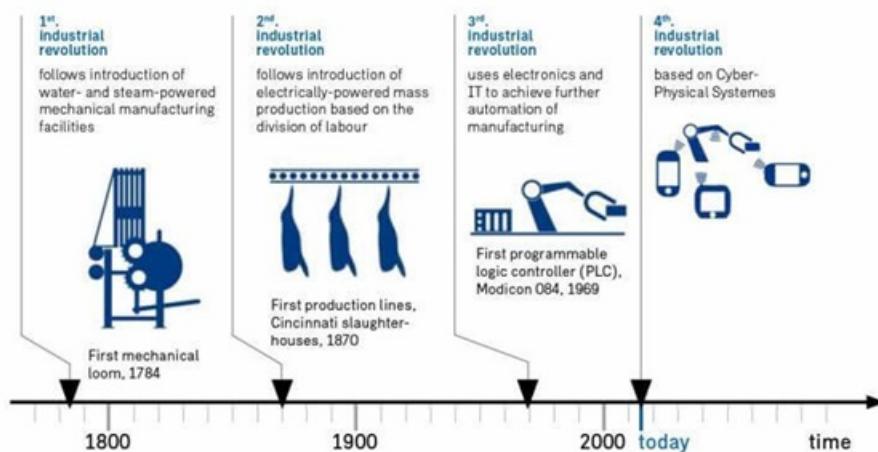
İlk kez 2011 yılında gerçekleştirilen Hannover Fuarı 'nda adı duyulan Endüstri 4.0, Alman Federal Hükümeti 'nin sağladığı desteklerle günümüz sanayisinde yerini almıştır. İleri gelen teknoloji devleri ABD ve Japonya gibi ülkeler bu endüstriyi desteklemiş ve gelecek hedeflerini Endüstri 4.0'a uygun bir şekilde planlamışlardır.

Endüstri 4.0 genel hatlarıyla,

- Robotların üretimi tamamen devralması
- Yapay zekanın gelişimi
- Üç boyutlu yazıcılarla üretimin fabrikalardan evlere inmesi



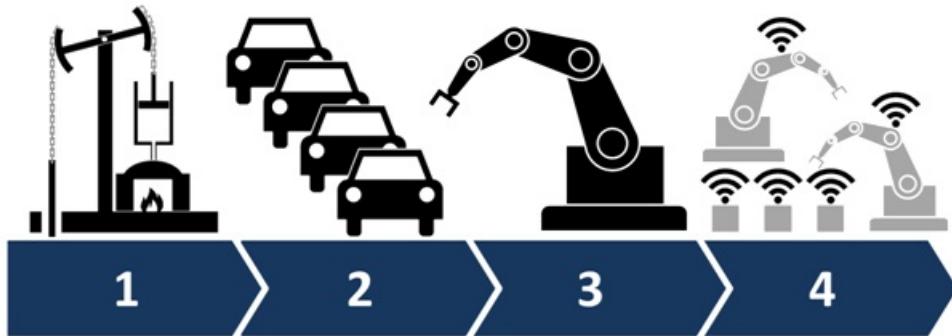
- Devasa miktarda ki bilgi yiğimini veri analizleriyle ayıklanıp değerlendirilmesi ve daha birçok yeniliklerle incelenebilir.



### Endüstri 4.0 Öncesi Sanayi Devrimleri

- Endüstri 1.0 Su ve Buhar gücünün kullanılmasıyla 1750-1830 yılları arasında İngiltere 'de çıkmıştır.
- Endüstri 2.0 Elektrik Enerjisinin üretimi ve dağıtımıyla başlamıştır.

## 3. Endüstri 3.0 Elektronigin ve Bilgisayarın üretimi

**Endüstri 4.0 'in Bize Sağladığı Faydalar**

Endüstri 4.0 'in en büyük amacı,

- Birbirleriyle haberleşen,
- Sensörlerle ortamı algılayabilen,
- Veri analizi yaparak ihtiyaçları fark edebilen robotların üretimi devralıp hata oranlarını en aza indirerek daha sağlam, daha kaliteli, daha ucuz, daha hızlı, daha az maliyetli ve daha az israf yapan, makinelerin internet ve mobil cihazlar sayesinde izlenip yönetildiği bir seri üretim yapmaktadır. Böylece fabrikalarda insan gücünden tasarruf edilerek teknolojik makineler üretimde yerini alacaktır.

4. Sanayi Devrimi daha çok fabrikaları etkileyebilecek gibi görünse de aslında gelecekteki sosyal hayatımıza bile etkileyebilecek bir yeniliktir. Üç boyutlu yazıcıları sadece sanayide değil, evlerimizde dahi kullanabilecek konuma geleceğiz. Kendi ihtiyaçlarımızı başkaları tarafından yapılan ürünlerle karşılamak yerine, kendi hayal gücümüzü kullanarak istediğimiz ürünü evimizde üretebilecek ve evimizi minik bir fabrikaya dönüştürebileceğiz. Günümüzde yaygın olan marka bağımlılığı gelecekte yerini fayda bağımlılığına bırakacaktır. Gelecekte hangi marka kıyafeti giydigimiz değil yerine hangi faydalı kıyafeti giydigimiz önem kazanacaktır ve bu faydalı kıyafetleri kendimiz evimizde üretebilir konuma geleceğiz.

**Endüstri 4.0 'ın Olumsuz Yönleri**



Endüstri 4.0'ın sahip olduğu güzel yanlarının dışında, hayatımıza zorlaştıracak olumsuz yanları da mevcuttur. Robotların üretimi devralmasıyla insan gücüne duyulan ihtiyaç azalacak ve robotlar bir anlamda insanları işlerinden kovacaktır. Bu durum sadece fabrikalardaki mavi yakalılar için değil beyaz yakalılar içinde bir risktir çünkü yapay zeka ile robotları kodlayabilen robotlar ve tasarım yapabilen robotlar, üretimi devralacaktır.

Endüstri 4.0 ile birlikte yeni meslekler ortaya çıkacağı öngörülse bile artan dünya nüfusu nedeniyle bu durum işsizliğe çare olamayacaktır.



## Endüstri 4.0 'nın Global Dünyaya Etkisi

Endüstri 4.0'in gelişmesiyle artan üretim hızı ve ürünün kalitesi rekabet için yeterli olmayacağı ve en çok üreten değil müşterinin isteğini en çok karşılayan

galip gelecektir. Apple'in dünyanın en büyük şirketi olması ve eski dünya devi Nokia'yı piyasadan silmesi bu duruma en güzel örnektir.

Müşterinin isteğini en güzel belirleme yolu ise veri analizidir. Internetin hayatımıza girmesiyle oluşan devasa bilgi yığını analiz edip en iyi şekilde yorumlayan gelecekte galip gelecek olanlardır. Endüstri 4.0 aynı zamanda Google ve Facebook gibi şirketlerinde üretime girmesini sağlayacak ve endüstride zorlu bir rekabetin başlamasına neden olacaktır.

4. Sanayi Devrimi aynı zamanda ülkeler arasındaki rekabeti artıracak ve Çin Halk Cumhuriyeti gibi hem insan gücü bakımından hem de sahip olduğu akıllı fabrikalarla gelecekte tahtı devralacaktır. Artan üretim hızının yanında Çin, üretim kalitesini da artırarak, günümüzde herkesin kalitesiz olarak gördüğü ürünler, gelecekte en kaliteli ürünlerin başına geçecektir.



INDUSTRY 4.0

### Türkiye 'nin Endüstri 4.0 'daki Yeri

Türkiye, dünyanın onde gelen üretim merkezlerinden biridir ve üretim kapasitesi Türkiye endüstrisini cazip kilsa bile gelecekte robotların üretimi devralmasıyla insan gücüne olan ihtiyaç azalacak ve yabancı şirketlerin yatırımlarını kendi ülkelerine yapmalarını sağlayacaktır. Bu nedenle ülkemizin üretim merkezi yerine, inovasyon merkezi olarak gelişen global pazarda kendine yer bulması gerekmektedir.

Bu nedenle Türkiye'nin önünde zorlu bir süreç mevcuttur. 2. ve 3. Sanayi Devrimi arasında bir evrede bulunan ülkemizin, 10 ila 15 yıl içerisinde tamamen Endüstri 4.0 girileceği düşünüldüğünde gelişen teknolojiyi yakalayıp

rekabet edebilecek konuma gelmelidir. Bu anlamda Türkiye'nin mühendisliği kız istemek için bir araç olarak kullananlara değil, ülkesini gelişen teknolojiye ayak uyduracak mühendislere ihtiyacı vardır. Sonuç olarak, Endüstri 4.0 ge-



leceğimizi iyi ve kötü yönleriyle doğrudan etkileyecektir. Gelecekte içerisinde insan olmayan ve işiğa ihtiyaç duymayan robotlarla çalışan fabrikalar bizi beklemekte ve insanoğlu artık robotlarla yarış içine girmeye hazırlanmalıdır. Yapmamız gereken Endüstri 4.0 'dan kaçmak değil ona en iyi şekilde uyum sağlamaktır.

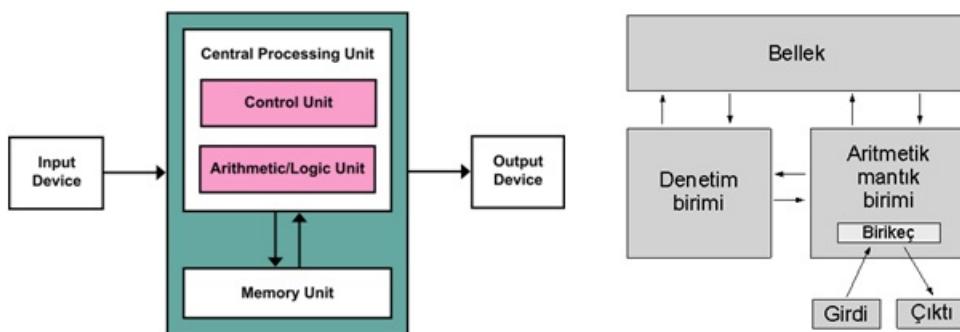
## Bölüm 2

# Genel Sistem Mimarisi

### 2.1 İşleyiciler

#### 2.1.1 VonNeumann Mimarisi

VonNeumannmimaris veri ve komutları tek bir yiğincak (depolama) biriminde bulunduran bilgisayar tasarı örneğidir. Paralel mimariler dışında Turing makinesi'nin ilkelerini uygulayan her bilgisayarı tanımlamak için kullanılır. Merkezi işlem biriminin bağımsızlığı dolaylı olup, "saklı yazılım bilgisayarı" (stored program computer) ile eşanlamlı olarak kullanılır.



### 2.1.2 İşlemci, Mikroişlemci ve Mikrodenetleyici

Hesaplamada, bir işlemci veya işlem birimi, bazı harici veri kaynağı, genellikle bellek veya başka bir veri akışı üzerinde işlemleri gerçekleştiren bir elektronik devredir. Terim genellikle bir sistemdeki merkezi işlemciye (merkezi işlem birimi) başvurmak için kullanılır, ancak tipik bilgisayar sistemleri (özellikle SoC'ler) bir dizi özel "işlemciyi" birleştirir.

Basit bir teknik tanım ile, mikroişlemci, yarıiletken tek bir tümdevre üzerine yerleştirilmiş CPU ya denir.(CPU on Chip). Mikroişlemciler genel-amaçlı cihazlar olup birçok uygulama için uygundur. Bir mikroişlemci, temel olarak CPU'nun fonksiyonlarını yerine getirdiği için pek çok kere, CPU ile mikroişlemci eş anlamba kullanılmaktadır.

Bir mikroişlemci tümdevresi üzerine kurulan bilgisayar, mikrobilgisayar (microcomputer) olarak adlandırılır. Bir mikrobilgisayarın mikroişlemcisi, I/O ve hafıza birimleri, teknolojiye ve uygulamalara göre devamlı değişim göstermektedir. Günümüzde mikroişlemciler çoğunlukla bilgisayarlarda kullanılmaktadır. Teknolojinin gelişimi doğrultusunda, daha önce mikroişlemci tümdevresi dışında olan pek çok giriş/çıkış ve hafıza alt birimleri, CPU üzerine taşınmıştır.

1970'lerin ortalarından itibaren mikroişlemciler, mikrobilgisayarların doğusunu mümkün kılmıştır. Bundan önce, tipik olarak elektronik ana işlem birimleri, sadece birkaç transistöre eşdeğer büyük, ayrık anahtarlama (switching) aygıtları (daha sonra small-scaletümdevreler) kullanılarak yapıliyordu. İşlemciyi, bir ya da birkaç large-scale tümdevre (binlerce veya milyonlarca ayrık transistörün eşdeğeri) içine gömmekle işlemci gücünü büyük ölçüde düşürdü. 1970'lerin ortalarında tümdevrelerin doğusuyla mikroişlemci, diğer bütün türleri değiştirip, ana işlem biriminin yapımında en yaygın yol oldu.

Performansın yıllar boyu sürekli artışı söz konusu olunca, mikroişlemcilerin evrimi Moore Kanunu'na uyar. Bu kanun bir tümdevrenin karmaşıklığının, en düşük bileşen maliyetine göre her 24 ayda iki katına çıktığını söyler. Bu görüşün doğruluğu 1970'lerin başından beri kanıtlanmıştır. Hesap makineleri için sürücü olarak başladıkları alçakgönüllü yolculukta, güçlerindeki sürekli artış, mikroişlemcilerin diğer bilgisayar biçimleri arasında dominant olmasını sağladı. Günümüzde, en büyük ana bilgisayarlardan, en küçük el bilgisayarlara kadar her sistem çekirdeğinde mikroişlemci kullanılmaktadır.

## Özel Amaçlı Mikroişleyiciler

“Mikroişlemci” terimi, geleneksel olarak bir tek-çip veya çoklu-çip ana işlem birimini veya bir-çipte-sistemi (System-on-a-chip (SoC)) işaret ederken; aynı teknolojiyle birkaç tip özelleşmiş işlem aygıtı üretildi. En yaygın örnekler mikrodenetleyicilerdir: Sayısal Sinyal İşleyiciler (Digital Signal Processors (DSP)) ve Grafik işleme birimleri (Graphics processing units (GPU)). Bunların çoğu örneği, ya programlanabilir değil ya da kısıtlı programlama yeteneklerine sahiptir. Örneğin, genelde 1990'lardaki GPU'lar büyük ölçüde programlanabilir değildi ve ancak yakın zamanda programlanabilir vertex-shader gibi sınırlı yeteneklere kavuştular. “Mikroişlemci” ile ne tanımlandığı konusunda evrensel bir konsensüs yoktur fakat aksi özellikle belirtilmemişse terimin bir özel amaçlı işlemciyi değil genelde bir genel amaçlı ana işlem birimini işaret ettiğini varsaymak yanlış olmaz.

Mikrodenetleyici (microcontroller) bir tümdevre üzerinde üretilen bilgisayara (Computer on Chip) denir. Bir mikrodenetleyici tümdevresinde bulunan hafıza ve giriş/çıkış alt sistemleri, bu işlemcinin birçok uygulama içinde gömülü (embedded) olarak gömülü olarak doğrudan ve tek başına, bir mikroişlemciye göre çok daha basit ve ucuz arabirim (interface) teknikleriyle kontrol amaçlı olarak kullanılmasını sağlar.

Mikrodenetleyiciler örneğin, otomobillerde motor kontrol, elektrik ve iç panel kontrolü; kameralarda ışık ve odaklılama kontrolü gibi amaçlarla kullanılmaktadır. Ayrıca bilgisayarlar, telefon ve modem gibi çeşitli haberleşme cihazları ve özel amaçlı elektronik kartlarda da mikrodenetleyiciler yoğun olarak kullanılmaktadır. Bu kadar geniş uygulamalarda kullanılan mikrodenetleyiciler tümdevre üzerinde yer alan çok değişik donanım özelliklerini sunmaktadır. Bu özelliklerden bazıları şunlardır: Paralel ve seri I/O portları, zamanlayıcı/sayıcılar, ADC ve RAM, ROM, EPROM ve EEPROM gibi değişik kapasitelerde ve özelliklerde hafıza birimleri.

Mikroişlemciler mimari (architecture) yapılarına göre farklılık gösterir. Ortak bir mimariye sahip işlemciler, aynı komutları tanıdıklarını için aynı programı çalıştırabilirler. Bir mikroişlemcinin tanıdığı komutlar, yani komut kümesi (instruction set) o mikroişlemci mimarisinin en temel özelliklerinden biridir. Diğer önemli bir özellik mikroişlemcinin saklayıcı kümesidir (register set). Saklayıcılar, mikroişlemcinin çalışması sırasında geçici verilerin saklandığı hafıza hücreleridir. Bu hafıza hücreleri işlemcinin içindedir. Farlı komut ve saklayıcı kümesine sahip mikroişlemciler genelde birbirlerinin programlarını

çalıştıramazlar.

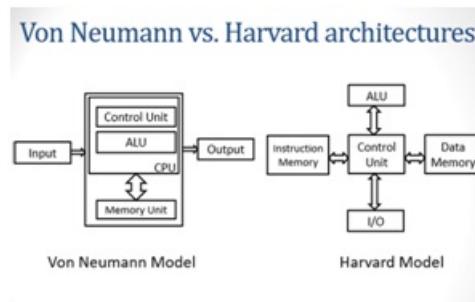
Mikroişlemciler ve mikrobilgisayarların sınıflandırılmasında en temel ölçü, mikroişlemcinin tümdevre üzerinde işlem yaptığı en uzun verinin bit sayısı, yani kelime uzunluğuudur (wordlength). 4-bit işlemci olan 4004(1971) ve 8-bit işlemci olan 8008(1972) ‘den başlayarak mikroişlemciler ve mikrobilgisayarlar için 4-bit, 8-bit, 16-bit, 32-bit ve 64-bit gibi veri uzunluk standartları doğmuştur.

### Mikrodenetleyici Çeşitleri

Piyasada birçok çeşit mikrodenetleyici bulunmaktadır. Mimarilerine göre mikrodenetleyiciler ikiye ayrılmaktadır.

**Von Neumann Mimarisi :** Programkomutarı ve veriler aynı bellekten alınarak tek bir yol üzerinden işlemciye gönderilir; önce komut, daha sonra da veri işlenir. Geçmişte bu mimari tercih edilse de şu anda yerini Harvard almıştır. Bu mimaride gecikmeler meydana gelmektedir.

**Harvard Mimarisi :** Genellikle bu mimari tercih edilmektedir. Verilere ve komutlara farklı yollardan ulaşılır, bu sayede çalışması daha hızlıdır.



#### 2.1.3 Mikroişleyicilerin İç Yapısı

Mikroişleyici, hafıza ve giriş çıkış birimlerini bulunduran yapının geneline mikrobilgisayar; CPU’yu bulunduran entegre devre çipine ise mikroişlemci denir. Aslında mikroişlemciler en basit halleriyle bilgisayarlardır ve 3 temel bölümde incelenirler.

- CPU(Merkezi İşlem Birimi) (Central Processing Unit)

- Hafıza (Memory)
- Donanım (Giriş çıkış birimleri)

### **Merkezi İşlem Birimi ( Central ProcessUnit )**

Merkezi işlem birimi (Türkçe kısaltması MİB veya Orijinal, terminolojik kısaltması CPU) bir bilgisayarın en önemli parçalarından biridir. Çalıştırılmakta olan yazılımın içinde bulunan komutları işler. İşlemci terimi genelde MİB için kullanılır. Mikroişlemci ise tek bir yonga içine yerleştirilmiş bir MİB'dir. Genelde, günümüzde MİB'ler mikroişlemci şeklindedir.

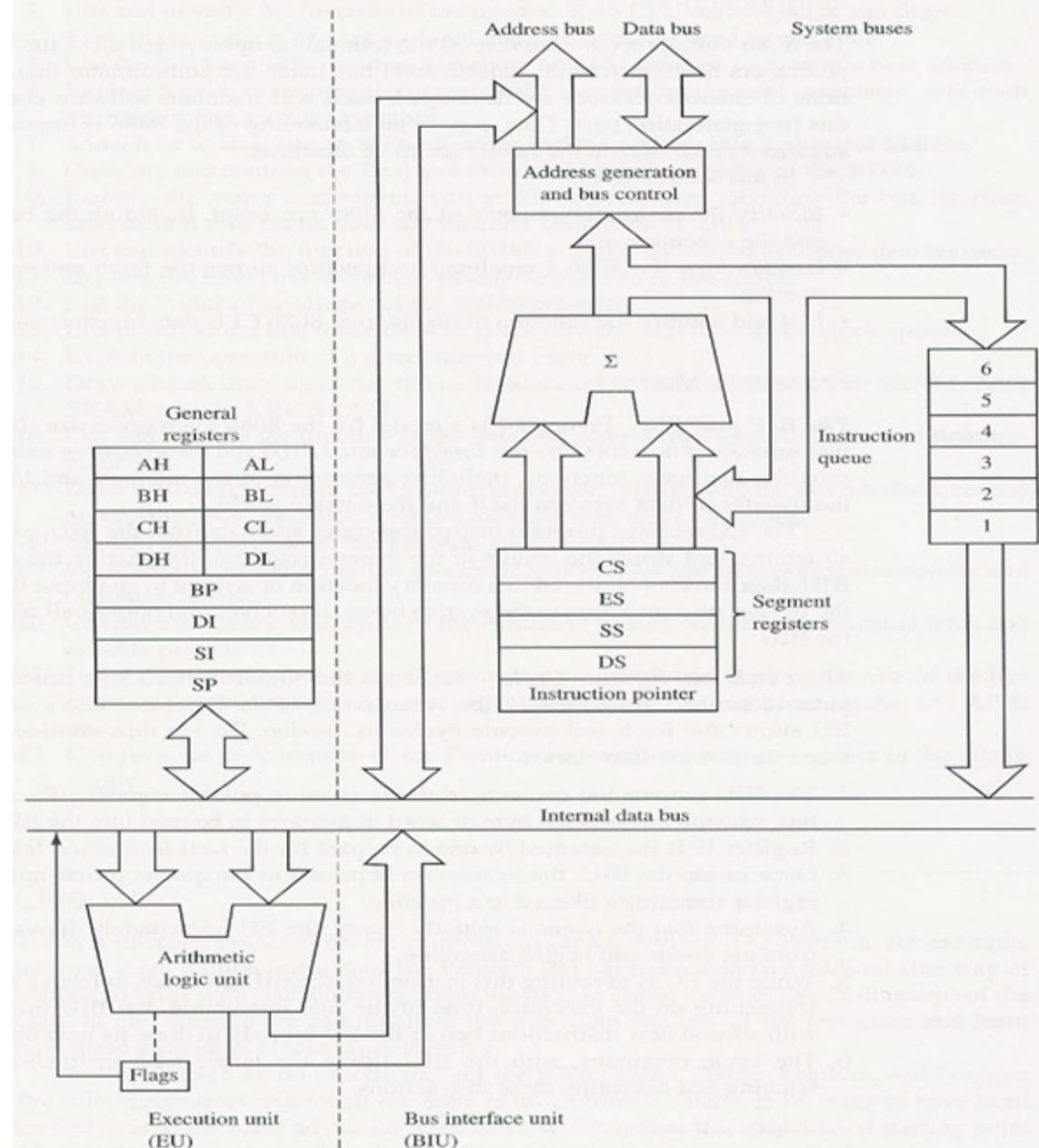
Merkezi işlem birimi (MİB) (İngilizce Central ProcessUnit (CPU)), veya başitçe işlemci, dijital bilgisayarların veri işleyen ve yazılım komutlarını gerçekleştiren bölümündür. İşlemciler, ana depolama ve giriş/çıkış işlemleri ile birlikte bilgisayarların en gerekli parçaları arasında yer alır. Mikroişlemciler ise tek bir yonga içine yerleştirilmiş bir merkezî işlem birimidir. 1970'lerin ortasından itibaren gelişen mikroişlemciler ve bunların kullanımı, günümüzde MİB teriminin genel olarak mikroişlemciler yerine de kullanılması sonucunu doğurmuştur.

Merkezi işlem birimi aritmetik ve mantıksal işlem yapma yeteneğine sahiptir. Giriş ve çıkış birimleri arasında verilen yazılım ile uygun çalışmayı sağlar. MİB, makine dili denilen düşük seviyeli kodlama sistemi ile çalışır; bu kodlama sistemi bilgisayarın algılayabileceği işlem kodlarından oluşur. Bir mikroişlemcinin algılayabileceği kodların tamamına o işlemcinin komut kümesi denir.

Merkezi işlem birimi aritmetik ve mantıksal işlemleri Aritmetik Mantık Birimi (AMB) aracılığıyla yapar. Bunun dışında virgülü sayılarla daha rahat hesap yapabilmesi için bir Kayan Nokta işlem birimi (FPU) vardır. Mikroişlemcinin içerisinde bulunan küçük veri saklama alanlarına yazmaç denir.

İlk Merkezi İşlem Birim'leri (MİB) daha büyük, bazen türünün tek örneği bilgisayarlar için özel olarak tasarlanmışlardı. Ancak belirli bir uygulama için özel MİB tasarımlının masraflı olması bir veya birçok amaç için yapılan kitlesel olarak üretilmiş işlemcilerin gelişmesine yol açtı. Bu standartlaşma eğilimi ayrık transistörlü ana sistemler ve mini bilgisayarlar döneminde başladı ve entegre devrelerin (ED) popülerleşmesiyle giderek hız kazandı. ED,

**Figure 3.1** Processor model for the 8086 microprocessor. A separate execution unit (EU) and bus interface unit (BIU) are provided.



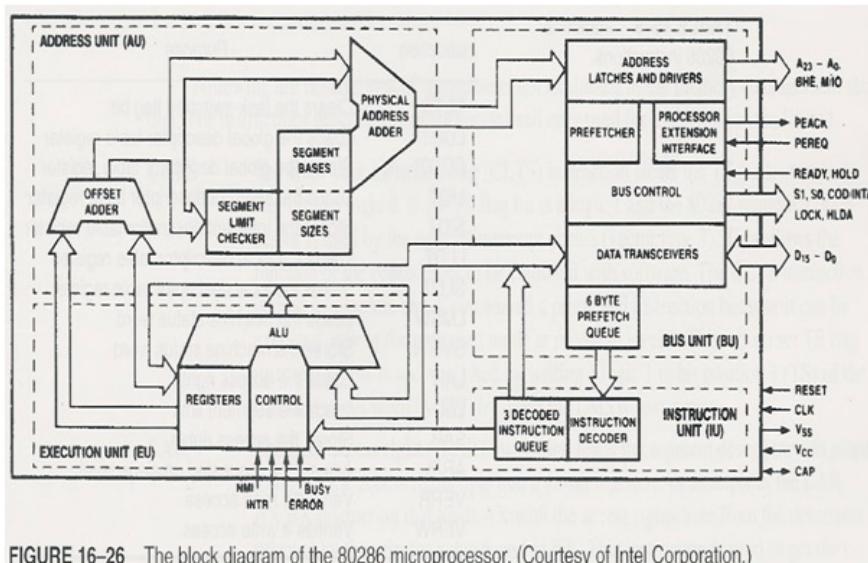
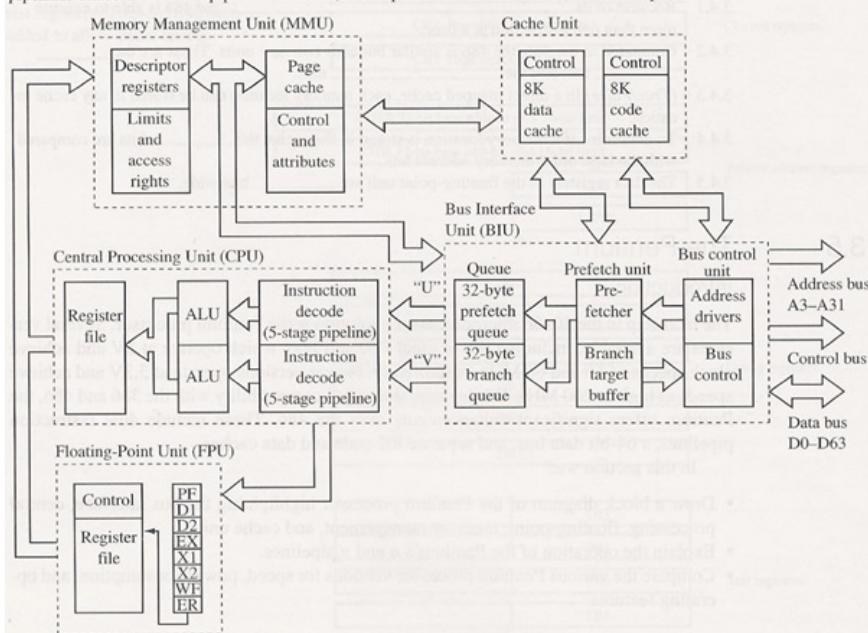


FIGURE 16-26 The block diagram of the 80286 microprocessor. (Courtesy of Intel Corporation.)

Figure 3.23 Processor model for the Pentium. The BIU supplies instructions to the CPU via two pipelines called the *u* and *v* pipes. In addition, two separate 8K data and code caches are provided.



giderek daha karmaşık ve nanometre ile ölçülebilecek MİB'lerin tasarılanmasına ve üretilmesine olanak verdi. MİB'lerin küçülmesi ve standartlaşması, modern hayatı dijital cihazların varlığını ilk bilgisayar örneklerinin sınırlı uygulamalarının çok ötesinde arttırdı.

VonNeumann mimarisine uygun olarak tasarlanırsa, en az bir kontrol birimi (CU), aritmetik mantık birimi (ALU) ve işlemci yazmaçları içerir. Bazı bağamlarda, ALU ve yazmaçlara işlem birimi (processingunit) de denir.

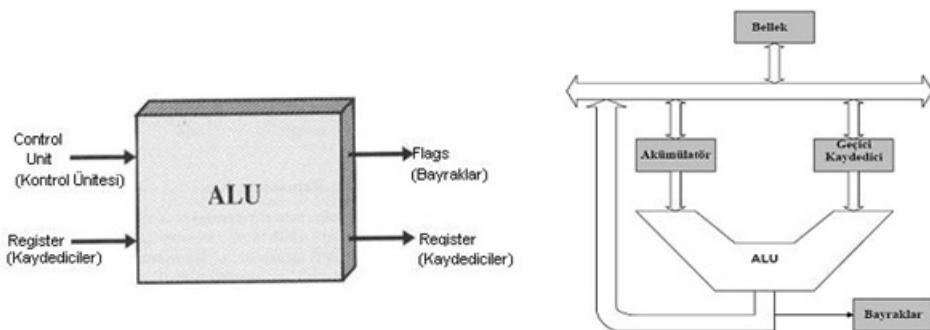
- CPU, mikroişlemcilerin beyni olarak bilinir ve bilgisayarların değişik birimleri arasındaki veri akışı ve veri işleme görevlerini yerine getiren kısımdır. Veri akışını CPU'nun alt birimlerinden kontrol kısmı yerine getirir.
- Hafızadan okunan komutları çözer ve komut tarafından belirlenen işlemi yerine getirir.
- Veri işlemenin çoğu CPU içindeki ALU'da (Aritmetik mantık ünitesinde) gerçekleştirilir.
- Sayısal aritmetik işlevler, lojik işlemler ve kontrol CPU'nun temel işlevleri arasındadır.
- CPU'lar bilgileri geçici olarak registerlarda depolarlar. CPU içerisindeki registerlar 8,16,32, veya 64 bitlik olabilirler.
- Hafızada bulunan program CPU'ya yapmasını istediği işlem için komutlar sağlar. CPU ise bu program komutlarını (instruction), hafızadan bulup çağrımak (fetch) ve olayları yürütmekle(execute) yükümlüdür.
- CPU'lar ALU birimine sahiptir ve bu birimlerinde matematiksel ve lojiksel işlemler yaparlar.
- Her CPU program sayacına(instructionpointer,flags) sahiptir ve bu sayacı var olan görev yerine getirilince devreye girer ve kendini bir artırarak bir sonraki komutun adresini gösterir.
- Bu program sayacının içeriği adres yolları üzerinde istenen komutu okuyacak,bulacak ve çağıracak şekilde dir.
- CPU içerisinde komut çözücü(instructiondecoder) birimide mevcuttur. Bu birim CPU'ya gelen komutun anlamını yorumlayarak bir çeşit sözlük görevi görür. Komutun anlamına göre kontrol sinyalleri üretir.

### Aritmetik Mantık Birimi (Aritmetik Logic Unit)

Aritmetik Mantık Birimi (AMB) aritmetik ve mantık işlemlerini gerçekleştiren bir dijital devredir. AMB en basit işlemi gerçekleştiren mikro denetleyiciden, en karmaşık mikroişlemciye sahip bir bilgisayara kadar tüm işlemcilerin yapısıdır. Modern bilgisayarların içinde bulunan mikroişlemcilerin ve ekran kartlarının içinde çok karışık ve güclü AMB'ler bulunmaktadır. AMB kavramına ilk olarak 1945 yılında matematikçi John vonNeumann EDVAC adlı yeni bir bilgisayar üzerine bulgularını anlatan raporunda değinmiştir.

AMB'lerin çoğu aşağıdaki işlemleri gerçekleştirebilir.

- Toplama, çıkarma ve bazı durumlarda çarpma ve bölme işlemleri (çarpma ve bölme AMB'nin bir birimi olacağı gibi AMB'den bağımsız bir birimde olabilir. AMB'de ne kadar çok birim varsa yapısı da o kadar karmaşık olmaktadır.)
- Mantıksal işlemler: (VE, DEĞİL, VEYA, ÖZEL VEYA)
- Bit kaydırma işlemleri. Bir sayıyı belirtilen bit sayısı kadar sağa veya sola, işaret genişletilerek veya genişletilmeyerek, kaydirmak veya döndürmek. Kaydirmalar sayıyı 2 ile çarpma veya bölme olarak da düşünülebilir. Çarpma ve bölme işlemlerinde olduğu gibi bu işlemlerde AMB'den bağımsız birim tarafından üstlenilebilir.



Şekil 2.1: Aritmetik Mantık Birimi

### Kontrol Birimi ( Control Unit )

Kontrol birimi, sistemin tüm işleyişinden ve işlemin zamanında yapılmasıdan sorumludur. Kontrol birimi, bellekte program bölümünde bulunan komut kodunun alınıp getirilmesi, kodunun çözülmesi, ALU tarafından işlenmesi ve sonucun geri belleğe konulması için gerekli olan kontrol sinyalleri üretir.

### Merkezi İşlemci Biriminde İletişim Yolları

Mikro işlemcide işlenmesi gereken komutları taşıyan hatlar yanında, işlenecek verileri taşıyan hatlar ve kesme işlemlerini kontrol eden sinyalleri taşıyan hatlar bulunur. İşlenecek verileri işlemciye yollamak veya işlenen verileri uygun olan birimlere aktarmak için aynı hatlardan faydalanylir. Tüm bu yollara iletişim yolları adı verilir.

### Veri Yolu

Merkezi işlem biriminden bellek ve giriş / çıkış birimlerine veri göndermede ya da bu birimlerden işlemciye veri aktarmada kullanılan hatlar, veri yolu olarak isimlendirilir. Veri yolu genişliği, mikro işlemcinin yapısı, mikro işlemci kaydedici genişliği ve kullanılan kelime uzunluğu ile doğrudan ilişkilidir. 8-bitlik mikro işlemcilerde veri yolu 8 hattı içerirken 16-bitlik işlemcilerde 16 hattı içerir. Mikro işlemciye işlenmek üzere iletilen veriler veri yolu üzerinden iletildiği ya da mikro işlemcide işlenen veriler veri yolu üzerinden ilgili birimlere yollandığı için veri yolunda iki yönlü iletişim mümkün olmaktadır.

Bellekte bulunan ve CPU tarafından işlenmesi istenilen veriler, veri yolu üzerinden ilettilir (Şekil 1.11). Bellekteki verilerin hatlara yerleştirilmesinde veya hatlardan gelen verilerin CPU'ya aktarılmasında verileri kısa süre tutmak amacıyla kullanılan tamponlardan faydalanylir. Tampon olarak kaydediciler kullanılır. CPU'da işlenen verilerin harici elemanlara iletilesinde veya harici elemanlardan gelen verilerin CPU'ya gönderilmesinde ara birim olarak giriş/çıkış (G/Ç) birimi kullanılır. CPU ile G/Ç birimi arasında veri iletiminde veri yolundan faydalanylir. Veri yolu üzerinden G/Ç birimine gelen veriler, tamponlar kullanılarak veri yolu üzerinden klavye, monitör, yazıcı ve tarayıcı gibi birimlere gönderilir veya bu birimlerden gelen bilgiler CPU'ya aktarılır.

- Mikroişlemci tarafından hafızaya veya çıkış birimlerine veri göndermektede yada hafızadan ya da giriş birimlerinden hafızaya veri alınmadır

kullanılırlar.

- Veri taşırlar ve bu veriler komut ya da data olabilir.
- Veri yolunun genişliği bilgisayarın performansını çok etkiler. Aynı zamanda veri okuma yeteneğinin hızını artırır. Örneğin Intel'in mikroişlemcilerinde veri yolunun genişliği 8085 modellerde 8 bit, PENTIUM'larda ise 64 bittir.
- Birbirine paralel olan tellerden oluşan haberleşme kanalları olarak tanımlanabilir.
- Veri yolları iki yönlüdür.

### Adres Yolu

Verinin alınacağı (okunacağı) veya verinin gönderileceği (yazılacağı) adres bölgesini temsil eden bilgilerin taşınmasında kullanılan hatlar, adres yolu olarak isimlendirilir. Adres yolu, tek yönlüdür ve paralel iletişim sağlayacak yapıdır.

CPU'da işlenen verilerin bellekte saklanması veya diğer elemanlara gönderilmesi gerekebilir. Bu durumda, verinin saklanacağı veya gönderileceği yerin adresi, mikro işlemci içerisindeki PC yardımı ile adres yolu üzerine yerleştirilir. Yerleştirilen bilginin temsil ettiği adres bölgesi dâhilî bellekte olabileceği gibi harici bellekte de olabilir. Yerleştirilen bilginin kodu çözüülerek ilgili adres bölgesi bulunur ve bulunan adres bölgesindeki veri, veri yoluna konur. Yapılan bu işlemlerin düzgün ve kontrollü olarak gerçekleştirilmesinden zamanlama ve kontrol birimleri görevlidir.

Adres yoluna yerleştirilen bilgi, mikro işlemcinin kapasitesine ve adreslenebilecek bölge sayısına bağlı olarak değişir. Bir mikro işlemci tarafından adreslenebilecek maksimum bellek kapasitesi ' $2^n$ ' formülü ile hesaplanabilir. ' $n$ ' adres hattı sayısını gösterir.

Bu durumda;  $2^{16} = 65536$  Bayt = 64 KBayt adres bölgesi,  
 $2^{20} = 1048576$  Bayt = 1MBayt adres bölgesi ve  
 $2^{32} = 4$  GBayt adres bölgesi adreslenebilir.

Mikro işlemci veri yolu ve adres yolu farklı sayıda hattı içerebilir. Veri yolu 8 hattan oluşan bir mikro işlemcili sistemde, adres yolu 16 hattan (16 bit) oluşabilir. Günümüz mikro işlemcilerinde sürekli veri yolları artırılırken adres

yolları hattını büyük oranda korumaktadır. Adres yolları hatlarının fazla artmamasının sebebi, şu anda kullanılar adresleme kapasitesinin çok yüksek ve ileriye yönelik olmasındandır.

- Hedef ve kaynak verilerin adreslerini taşırlar.
- AdressBUS'ın genişliği sistemin maksimum hafıza kapasitesini belirler. Örneğin addressBUS'ı 16 bit olan bir mikroişlemcide max hafıza 64 kilobayttır. Adress yolunun genişliğinden kasıt aynı anda birden fazla işlemin bir arada yapılmasıdır. Örneğin address yolunun genişliği 8 bit olursa 256 adet adrese ulaşılabilir. İntel'in 8086 işlemcisinde bu genişlik 20 bitken, PENTİUM II'de 36 bite kadar gider.
- Adress yolu genişledikçe işlemci daha fazla RAM kullanır ve buda performansın artmasına neden olur.
- AdressBUS'lar tek yönlüdürler.
- İşlemcinin haberleşmeyi planladığı hafıza adresini belirlemede görev alır.

### Kontrol Yolu

Mikro işlemcili sistemde bulunan birimler arasındaki ilişkiyi düzenleyen sinyallerin iletilmesi amacıyla kullanılan hatlar kontrol yolu olarak adlandırılır. Her bir mikro işlemciye ait komut kümesi ve belirli amaçlar için kullanılan sinyallerin farklı olması sebebiyle her mikro işlemcide farklı sayıda hattı içeren kontrol yolu bulunabilir. Kontrol yolunda bulunan sinyaller üç farklı işlemi gerçekleştirmek için kullanılır:

**Sinyal seçimi:** Sistemde kullanılacak sinyallerin ve sinyallerin uygulanacağı yerin belirlenmesi işlemini gerçekleştiren sinyaller

**Yön tayini:** Sistemdeki verinin ne yöne gideceğini belirleyen sinyaller (okuma veya yazma)

**Zamanlama:** Yapılacak işlemlerin sırasını ve zamanlamasını belirleyen sinyaller

Kontrol yolunda bulunan hat sayısı, mikro işlemcinin bit sayısına bağlı değildir. Kontrol yolunu oluşturan hatların mikro işlemci içerisinde ağ şeklinde yayılması sebebiyle kontrol yolu terimi yerine kontrol hatları terimi

kullanılabilir. Mikro işlemcili sistemdeki birimlerin çalışması, kontrol hatları üzerinden iletilen tetikleme sinyalleri ile yönlendirilir. Mesela, bir bellekten veri okunacağı zaman, ilgili bellek entegresine aktif olmasını sağlayacak yetkilendirme (CS-Chipselect) sinyali ile birlikte, okuma işlemi için gerekli uygun R/W sinyalinin uygulanması gereklidir.

- Yol üzerindeki bir adresin hafıza adresi mi yoksa giriş çıkış birimlerinden biri mi olduğunu kontrol BUS yani kontrol yolu tespit eder.
- Okuma ve yazma sinyallerini sağlamak için kullanılır. CPU'nun input-output veya hafızaya bilgi göndermek mi yoksa onlardan bilgi almak mı istedığını belirler.
- Hafıza,giriş çıkış, yazma, okuma diye 4 çeşit kontrol yolu sinyali mevcuttur. İşlemci hangi adresi gösteriyorsa bu sinyallerden biri aktif olur.
- Mikroişlemcinin yaptığı işlemlerin birbirine karıştırmasını yine kontrol yolları engeller.
- Kontrol BUS tek yönlüdür.
- CPU'lar fetch (verilen talimatı gidip hafızadan almak) ve execute (alınan talimatı yerine getirmek) eylemlerini yerine getirebilmek için REGISTER, ALU, PROGRAM COUNTER ve KOMUT ÇÖZÜCÜ'ye sahiptir.

### **Yazmaçlar/Kaydediciler ( Registers )**

#### **Kaydediciler**

Kaydediciler, daha önce de bahsedildiği gibi genel ve özel amaçlı olmak üzere iki gruba ayrılır. Bunlardan başka programcıya gözükmeyen (ilgilendirmeyen) kaydediciler de vardır (IR, DAR, MAR ve MBR gibi). Genel amaçlılara 6502 işlemcisinde akümülatör, X indis ve Y indis kaydedicisi girmektedir. Özel amaçlılar ise PC, SP, bayraklar, DR gibi kaydediciler girmektedir (Şekil 1.5). Aşağıda bunlardan bazıları anlatılacaktır.

#### **Akümülatör**

Akümülatörler (ACC ya da A olarak da tanımlanabilir), bilgisayarın aritmetik ve mantık işlemleri sırasında depo görevi yapan önemli bir kaydedicidir. Eğer kaydediciler bir sistemde sekreterya olarak düşünülürse akümülatör bu sistemde baş sekreter olarak yerini alır. Ara değerlerin üzerinde tutulması, sisteme gelen verinin ilk alındığı yer, belleğe veya dış dünyaya gönderilecek verilerin tutulduğu yer olarak görev yapar. Bu yüzden, işlemcinin A kaydedicisini hedefleyen komutları çoktur. Bazı işlemcilerde B kaydedicisi de yardımcı akümülatör olarak kullanılır.

### **İndis Kaydedicileri**

X ve Y olarak tanımlanan indis kaydedicilerinin temelde üç görevi vardır. Hesaplamalarda ara değerlerin geçici tutulmasında, program döngülerinde ve zamanlama uygulamalarında bir sayıcı olarak ve bellekte depolanmış bir dizi verinin üzerinde bir indisçi olarak kullanılmaktadır. Bazı işlemcilerde sadece tek indisçi olabilir.

### **Program Sayıcı (PC)**

Mikro işlemcinin yürütmekte olduğu program komutlarının adres bilgisini tuttuğu özel amaçlı bir kaydedicidir. Bilindiği gibi bir programı oluşturan komutlar ve veriler normal bellekte saklıdır. Bilgisayarın çalışması sırasında hangi komutun hangi sırada kullanılacağı bilinmesi gereklidir. Bu görevi program sayıcı (PC) yerine getirir. Program sayıcının bit genişliği adres yolu genişliği kadardır. Eğer işlemcinin 16-bit adres hattı var ise  $PC=16$  bit, işlemcinin 32-bit adres hattı var ise  $PC=32$  bit büyüğünde olur. Bellekten alınan her komut kodundan sonra alınacak yeni komut kodunun adresi program sayıcıya otomatik olarak işlemci tarafından yüklenir. Komut çevrimi, PC'nin yeni adresi adres yoluna koyması ile başlar. Bunun ardından da ilgili kontrol sinyali gönderilir. Bellekten gelen her bilgiden sonra PC, kontrol devresinden aldığı işaretle uyarak adres satırını 1 arttırır. Böylece bilgilerin bellekten işlemciye düzenli bir şekilde gelmesi sağlanır.

### **Durum Kaydedicisi (Bayraklar)**

Durum kaydedicisi 8-bitlik bir kaydedicidir. Bu kaydedicinin her bir biti ayrı ayrı anlam ifade eder. Mikro işlemci içinde veya dışardan yapılan herhangi aritmetiksel, mantıksal veya kesmelerle ilgili işlemlerin sonucuna göre bu bitler değer değiştirir. Bir işlem sonucunda bu bitlerin aldığı değere göre program yön bulur. Programcı bu bitlerde oluşacak değerlere göre programa yön verebilir. Kabul edilen terminolojiye uyarak eğer bir bayrağa karşılık olan bit 1 ise söz konusu bayrak kuruldu, eğer bit 0 ise söz konusu bayrak silindi denir.

**Carry (elde bayrağı-C):** Elde / borç bayrağıdır. 8-bitlik bir işlem sonucunda dokuzuncu bit ortaya çıkıyorsa elde var demektir. Bu durumda C bayrağı mantıksal 1 olur. Bu bayrak biti programcı tarafından kurulur ya da silinebilir (CLC, SEC). Ayrıca bazı komutlar tarafından test edilebilir (BCC, BCS).

**Zero (sıfır bayrağı-Z):** Sıfır bayrağı, aritmetik ve mantık işlemi sonucunda kaydedici içeriği sıfır ise  $Z = 1$ 'e kurulur. Aksi durumda sıfırlanır ( $Z = 0$ ).

**Interrupt disable (kesme yetkisizleştirme bayrağı-I):** Mikro işlemci normal durumda komutları işlerken bir kesme (IRQ) geldiğinde bu kesme bu bayrak biti ile engellenebilir. Eğer bu bit komutlar (SEI) mantıksal 1 yapılrsa gelen kesmeler göz önüne alınmaz. Ancak bu bayrak mantıksal 0 olduğunda kesme dikkate alınır ve kesme hizmet yordamına dalınır. Yani bu bayrak biti, normal işleyiş sırasında bir kesme geldiğinde kabul edilip edilmeyeceğini belirler. Programcı bunu komutla yapar. NMI kesmesi bu bayrak için kullanılmaz.

**Decimal (ondalık bayrağı-D):** Bu bayrak 1 olduğunda aritmetik işlemler BCD modunda yapılrken 0 olduğunda ikilik modda yapılır. Bu işlem eğitim ve uygulama açısından programcıya büyük kolaylıklar sağlar.

**Overflow (taşma bayrağı-V):** Bu bayrak aritmetik işlemlerde, eğer işlem  $+127$  ile  $-128$  aralığını geçiyorsa bir taşma meydana gelir ve V bayrağı 1 olur. Diğer yandan yine benzer işlemlerde eğer pozitif bir sayı ile negatif bir sayı üzerinde işlem yapılıyorsa ve sonuç pozitif çıkması gerekiyorken negatif çıktıysa bu bayrak 1 olur. Taşma bayrağı işaretli sayılarla işlem yapılrken devreye girer.

**Negative (negatif bayrağı-N):** 8-bitlik bir işlemcide 7.bit MSB biti ola- rak bilinir. Eğer MSB biti bir işlem sonucunda 1 ise N bayrağı 1'e kurulur. Eğer MSB biti 0 ise kaydedicisindeki değer pozitif demektir ki N bayrağı 0

olur. Bayraklardaki 4. bit B (Break) olarak tanımlanmış olup program durdugunda otomatikman 1 olur. 5. bit ileride kullanılmak üzere boş (+5V) bırakılmıştır. Farklı mikro işlemcilerde birbirine benzer bayraklar olmasına rağmen faklı bayraklarda olabilir.

### Yığın İşaretçisi (SP)

RAM belleğin herhangi bir bölümü yığın olarak kullanılabilir. Yığın mikro işlemcinin kullandığı geçici bellek bölgesi olarak tanımlanır. Yığın işaretçisi, yığının adresini tutan özel amaçlı bir kaydedicidir. SP adres bilgisi göstereceği için 16-bit uzunluğundadır.

Bu kaydediciye programın başında yığının başlangıç adresi otomatik olarak atanır. İşlemci tarafından yığının başlangıç adresi SP'ye yüklenerek sonra artık belleğin bu bölgesi depo benzeri bir görev yürütür. Yığına veri girişi yapıldıkça yığın göstericisinin değeri de yapısına göre değişir.

Yığına her veri girişinde yığın göstericisinin değeri bir azalmakta, yığından her veri çekildiğin de ise yığın göstericisinin değeri otomatik olarak artmaktadır. Yığına gönderilen veri yığın göstericisinin işaret ettiği adresteki bellek hücresına yazılır. Yığından veri çekilirse yığın göstericisi bir önceki verinin adresine işaret edecek şekilde bir azalacaktır. Mikro işlemci işlediği ana programdan alt programa dallandığında veya bir kesme sinyali ile kesme hizmet programına dallandığında mevcut kaydedicilerin içeriklerini ve dönüş adresini saklayabilmek için otomatik olarak verileri ve adresleri yığına atar. Alt programdan veya kesme hizmet programından ana programa geri dönülmesi durumunda, ana programda kaldığı yerin adresini ve kaydedicilerdeki verileri kaybetmemiş olur.

#### 2.1.4 Mikroişleyicilerin Sınıflandırılması

Günümüzde tasarlanan ve üretilen işlemciler PC'lerde, cep telefonlarında, gömülü sistem uygulamaları gibi hayatımızın pek çok alanında kullandığımız elektronik aletlerde kullanılmaktadır. Tarihsel işlemci tasarım sürecinde teknolojinin hızla gelişmesiyle bazı tasarımlar güncellliğini kaybetmiştir, bununla

birlikte bazı tasarımların kullanımına ise günümüzde de devam edilmektedir. Bu kadar çeşitli alan için tasarlanan işlemciler elbette tek bir kriter göz önüne alınarak sınıflandırılamaz. İşlemci sınıflandırması birçok farklı duruma göre yapılabılır. Bunlar: paralelliğe, mimariye, belleklere, adresleme kiplerine göre yapılabilmekte.

### Paralelliğe göre sınıflandırma

Bu sınıflandırma paralel ve paralel olmayan şeklinde dir.

#### Paralel olmayan

**Single-cycle işlemciler:** Adından da anlaşılacağı gibi bu gruba dahil olan işlemcilerde komutlar eşit cycle sayısında işlenir. Örneğin türn komutlarının 1 ya da 2 saat çevriminde bitirilmesi gibi. İşlemcide kullanılacak saat hızı seçiminde en yavaş komut (criticalpath) dikkate alınarak seçim yapılmalıdır.

**Multi-cycle işlemciler:** Tasarımı singlecycle tasarımına göre daha zordur. Tasarım yapılırken komutlar daha küçük kısımlara ayrılırlar. Bu küçük kısımlar daha hızlı bir saat çevrimi ile işlenir. Örneğin bir komut 5 küçük kisma ayrılmış 5 cycle da işlenirken başka bir komut 2 cycle da tamamlanabilir. Performans olarak kısa komutların çokça kullanıldığı bir programda single-cycle işlemcilerden hızlıdır. Tasarım yapılırken daha fazla componente ihtiyaç duyulur.

#### Paralel

**Pipeline (iş hattı) işlemciler:** Çok tekrarlanan ve alt parçalara bölünen işlemlerde kullanılır. Genel yapısı yandaki resimde gösterildiği gibidir. Pipeline aritmetik iş hattı ve komut iş hattı( InstructionPipeline) olmak üzere iki kısımdan oluşur. Merkezi işlem birimleri komutları işlenirken belli alt işlemleri tekrar ederler. En basit iş hattı iki segmanlı olarak kurulabilir. Verimi artırmak için komut işlemi daha küçük parçalara ayrılabilir.

1. Komut al
2. Komut çöz
3. Efektif adres hesapla

4. Operand al
5. Komut yürüt
6. Sonucu yaz

**Superscalar işlemciler:** Superscalar işlemciler, her saat sinyalinde birçok komutu okuyarak kendi komut sıralamasına koyar. Aynı anda aynı fonksiyonel üniteler birden fazla komut tarafından kullanılamayacağından superscalar işlemcilerde fonksiyonel üniteler birden fazladır. Bunun dışında komutların program sırası dışında (out of order execution) işlenmesine olanak sağlayan yapılar ile paralellik artırılabilir. Bu işlem “reorderbuffer” ya da genişletilmiş register file ile sağlanabilir.

### Adresleme kiplerine göre sınıflandırma

1980'lerin başlarına kadar, tek dümdevre veya bir kart üzerinde tasarlanan bütün CPU'lar CISC(Complex Instruction Set Computer) tasarım felsefesini takip etti. CISC'in en temel özelliği, her mümkün durum için birçok komut içermesidir. Birçok komuta sahip bir CPU tasarlamak binlerce fazladan transistör kullanımının yanında, tasarımlı çok karmaşık yapmakta, zaman almakta ve pahalı olmaktadır. 1980'lerin başında yeni bir tasarım yöntemi olan RISC (Reduced Instruction Set Computer) sunuldu. Bu felsefenin savunucuların hareket noktası, CISC tipi CPU'larda bulunan komutların önemli bir kısmının çoğunlukla kullanılmamasıdır. Bu temel düşünüceden hareketle, RISC mimarilerinde CISC mimarilerine göre çok daha az komut bulunur. RISC yapısı CISC'e göre daha basittir. RISC programları CISC'e göre bir işlemi yapmada genellikle daha fazla komut kullanacağı için daha uzundur. 1980'lerden günümüze birçok ticari RISC mimarisi tabanlı mikroişlemci üretilmiş ve bilgisayarlarda kullanılmıştır. CISC mimarisinin klasik örnekleri 8086/8088'den başlayıp Pentium'larla devam eden x86 ailesi ve Motorola 680x0 ailesidir. PowerPC ve Sun SPARC yaygın olarak kullanılan RISC örnekleridir. 1980'lerin ortalarından 1990'ların başlarına kadar, birçok yeni, yüksek performanslı RISC (azaltılmış komut kümesi bilgisayarı (reduced instruction set computer)) mikroişlemciler üretildi. Bu mikroişlemciler, özel amaçlı makinelerde ve Unix iş platformlarında kullanıldı ve Intel-standart masaüstü hariç bütün rollerde evrensel bir hale geldi.

### CISC mimarisi

CISC mimarisinin karakteristik iki özelliğinden birisi, değişken uzunluktaki komutlar, diğer ise karmaşık komutlardır. Değişken ve karmaşık uzunluktaki komutlar bellek tasarrufu sağlar. Karmaşık komutlar İki ya da daha fazla komutu tek bir komut haline getirdikleri için hem bellekleri hem de programda yer olması gereken komut sayısından tasarruf sağlar. CISC yapısının en çok kullanıldığı alan sinyal işlemede kullanılan DSP işlemcileridir.

### RISC mimarisi

RISC mimarisi, CISC mimarili işlemcilerin kötü yanlarını piyasanın tepkisi ve ona bir alternatif olarak, işlemci mimari tasarımlarında söz sahibi olan IBM, Apple ve Motorola gibi firmalarca sistematik bir şekilde geliştirilmiştir. RISC yapısının güzelliği, basit yapılar içermesinden dolayı bu yapıların kullanılarak çok farklı şekilde kompleks yapıların çözülmüşindedir.

### Mimari Tabanlı Sınıflandırma

İşlemciler, komut set mimarileri temelinde sınıflandırılabilir. Önceki modülde açıklanan komut seti mimarisi, bize makinenin bir “programcının görünümünü” vermektedir.

Komut Seti Mimari Sınıfları:

CPU tarafından komutların ve verilerin saklamak için kullanılan mekanizma, ISA (Komut Seti Mimarisi), mikroişleyicilerini sınıflandırmak için kullanılabilir. Bu sınıflandırmaya dayanan üç tip makine vardır.

- Akümülatör Tabanlı Makineler
- Yiğin Tabanlı Makineler
- Genel Amaçlı Yazmaç (GPR) Makineleri

### Akümülatör Tabanlı Makineler

Akümülatör bazlı makineler, tek bir kaynak işlenmesini ve aynı zamanda gerçekleştirilen aritmetik veya mantıksal işlemlerin sonuçlarını saklamak için akümülatörler olarak adlandırılan özel yazmaçları kullanır. Böylece, toplayıcı yazmaçlar veri toplar (veya biriktirir). Akümülatör, işlenenlerden birini tuttuğundan, başka bir işlenenin adresini tutmak için bir tane daha yazmaç

gerekebilir. Akümülatör bir adres tutmak için kullanılmaz. Böylece akümülatör tabanlı makineler de 1-adresli makineler olarak adlandırılır. Akümülatör makineleri genellikle sadece bir tane olmak üzere çok az sayıda akümülatör kaydı kullanır. Bu makineler, kaynak işleneni ve işlemin sonucunu tutmak için bir yazmaç kullandıklarından hafızanın oldukça pahalı olduğu zamanlarda faydalıydı ancak şimdi hafızanın nispeten ucuz olduğu, bunların çok kullanışlı olmadığı düşünülmekte ve kullanımları, birçok işlenenin kullanıldığı ifadelerin hesaplanması için ciddi bir şekilde sınırlandırılmaktadır.

### **Yığın Tabanlı Makineler**

Yığın, ilk giren (LIFO) bir yapıda düzenlenmiş bir grup yazmaçtır. Böyle bir yapıda, ilk önce, push operasyonu ile saklanan işlenenlere, bir pop operasyonu yoluyla, sadece en son olarak erişilebilmektedir; İşlenenlere erişim sırası, depolama işleminin tersidir. Aritmetik ve mantıksal işlemler işlenenleri yığının en üstünden (Top-of-the-Stack) sefer ve sonuçları işlemin sonunda tekrar yığının en üstüne (TOS)'a aktarır. Yığın tabanlı makinelerde, işlenen adreslerin aritmetik veya mantıksal işlemler sırasında belirtilmesi gerekmez. Bu nedenle, bu makinelere 0 adresli makineler de denir. En bilinen örneklerinden biri Java Virtual Machine'dir. JVM yazılım tabanlı bir yığın makinesidir.

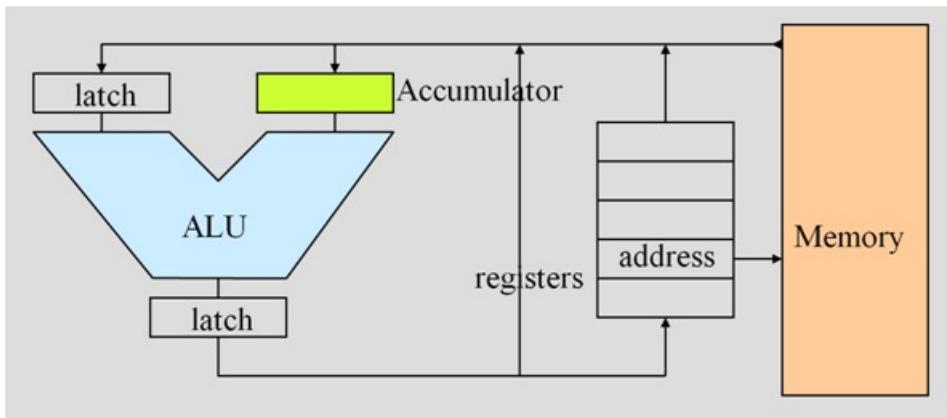
### **Genel Amaçlı Yazmaç Makineleri**

Genel amaçlı yazmaç makinelerinde, CPU içinde bir dizi yazmaç mevcuttur. Bu yazmaçların özel işlevleri yoktur ve çeşitli amaçlar için kullanılabilirler. Bir komutta kaydı tanımlamak için, bir yönerge kelimesinde az sayıda bit gereklidir. Örneğin, CPU'nun 64 kaydından birini tanımlamak için, komutta 6 bitlik bir alan gereklidir. CPU yazmaçları önbellekten daha hızlıdır. Yazmaçlar ayrıca derleyici tarafından diğer dahili depolama biçimlerine kıyasla daha kolay ve daha etkili bir şekilde kullanılır. Yazmaçlar değişkenleri tutmak için de kullanılabilir, böylece hafıza trafigini azaltır. Bu, yürütme hızını artırır ve kod boyutunu azaltır (bellekle karşılaşıldığında kod yazmaç adlarına daha az bit gerekir). Verilere ek olarak, yazmaçlar adresleri ve işaretçileri de (yani, bir adresin adresini) tutabilirler. Bu programcı için mevcut esnekliği artırır.

Genel amaçlı makinelerde bir dizi özel veya özel amaçlı yazmaç da mevcuttur, ancak bunların çoğu programcı için mevcut değildir. Saydam yazmaçlara örnek olarak yığın gösterici, program sayacı, bellek adres kaydı, bellek veri kaydı ve durum kodları (veya bayraklar) kaydı vb. Dahildir. Gerçekte,

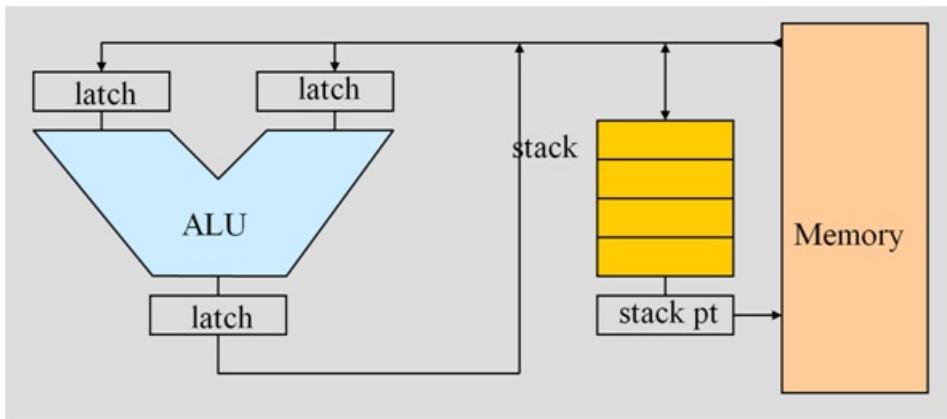
çoğu makinenin bu makine tiplerinin bir kombinasyonu olduğunu anlamalıyız. Akümülatör makineleri, CPU içindeki bir işlemin ara sonuçlarını saklayabildikleri için daha verimli olma avantajına sahiptir.

**Akümülatör** Bir işlenen (yazmaçta veya bellekte), akümülatör neredeyse her zaman dolaylı olarak kullanılır

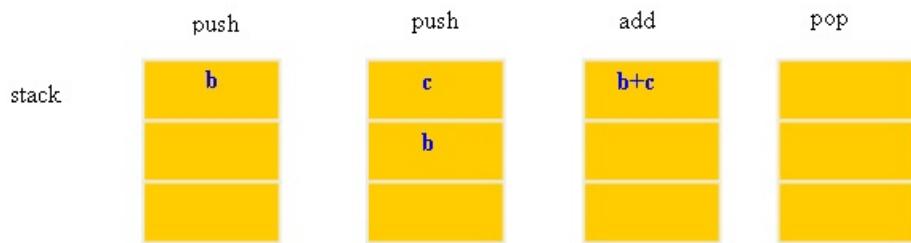


Şekil 2.2: Akümülatör

**Yığın** 0 işlenen: tüm işlenenler örtük (TOS)



Şekil 2.3: Yığın



Şekil 2.4: Yığın

**Yazmaç (load-store)**

3 işlenen: Tümü Yazmaçlarda Belleğe erişim yalnızca loads ve stores komutları üzerinden gerçekleşir (bellek(dolaylı) adresleme yöntemi)

**Yazmaç-Bellek**

2 işlenen: Bir Yazmaçta, Bir Bellekte

**Bellek-Bellek**

3 işlenen: Tümü Bellekte olabilir

(there are more varieties / combinations)

Let's look at the code for  $C = A + B$

Yığın Mimarisi	Akümülatör Mimarisi	Yazmaç- Bellek	Bellek- Bellek	Yazmaç (load-store)
Push A	Load A	Load r1,A	Add C,B,A	Load r1,A
Push B	Add B	Add r1,B		Load r2,B
Add	Store C	Store C,r1		Add r3,r1,r2
Pop C				Store C,r3

Şekil 2.5: Code for  $C = A + B$ **LOAD - STORE Makineleri**

Bu makineler aynı zamanda yazmaç-yazmaç makineleri olarak da adlandırılır.

maktadır. Genellikle  $1\frac{1}{2}$  adres komut formatını kullanırlar. Sadece load ve store komutları belleğe erişebilir. load komutu gerekli verileri hafızadan alır ve geçici olarak CPU yazmaçlarında saklar. Diğer komutlar bu verileri CPU yazmaçlarından kullanabilir. Daha sonra, sonuçlar store komutu ile belleğe geri saklanabilir. Çoğu RISC bilgisayarı bu makine kategorisine girer.

Yazmaç-Yazmaç komutları, toplam 3 işlenenden 0 bellek işlenenini kullanır. Böyle bir şemanın avantajları:

- Komutlar basit ve uzunluk olarak sabittir
- İlgili kod oluşturma modeli basittir
- Tüm komutlar yürütme için benzer sayıda saat döngüsü alır

Dezavantajları:

- Komut sayısı daha yüksektir; Belirli bir görevi tamamlamak için gerekli olan komutların sayısı, belleğin yükleme ve saklama işlemleri için ayrı komutlar gerekeceğinden daha fazladır.
- Komut boyutu sabit olduğundan, tüm alanları gerektirmeyen komutlar hafıza bitlerini boş'a harcar.

### **Yazmaç-Bellek Makineleri**

Yazmaç-Bellek makinelerinde, bazı işlenenler hafızadadır ve bazıları yazmaçtadır. Bu makineler tipik olarak, işlenenlerden birinin bir akümülatör veya genel amaçlı bir CPU kaydı olduğu 1 veya  $1\frac{1}{2}$  adres komut formatını kullanır.

Yazmaç-Bellek işlemleri, toplam iki işlenenden bir bellek işleneni kullanır. Bu komut formatının avantajları:

- Hafızada işlenenler, ilk önce ayrı bir load komutu ile yüklemeye gerek kalmadan erişilebilir.
- Yükleme işlenenlerinin ilk önce yazmaçlara olan ihtiyacının ortadan kaldırılması nedeniyle kodlama kolaydır

- Sabit bit sayısı başına daha fazla komut verildiği için, komut bit kullanımı nispeten daha iyidir.

Dezavantajları:

- İşlenenler, bir işlenenin iki işlevi (hem kaynak işleneni hem de hedef işlenen) olabileceği ve kaynak işlenenin imha edilebileceği için eşdeğer değildir.
- Bellek ve yazmaç için farklı boyutta kodlama, yazmaç sayısını sınırlayabilir
- Komut yürütme başına saat çevrimi sayısı, işlemciden uzaktaki işlemci getiricisine bağlı olarak, CPU yazmaçlarındaki işlenenlerle karşılaşıldığında yavaştır.

### **Bellek-Bellek Makineleri**

Bellek-Bellek makinelerinde, üç işlenen (2 kaynak işleneni ve bir hedef işlenen) bellekte bulunur. İşlenenlerden biri hem kaynak hem de hedef olarak kullanılıyorsa, 2-adres biçimini kullanılır. Aksi takdirde, bellek-bellek makineleri 3-adresli komutlar kullanır.

Avantajları:

- Bellek-Bellek komutları, kodlama israfının minimum olduğu en kompakt komut mimarisidir.
- İşlenenler doğrudan bellekten çağrıldıklarında veya bellekte saklandıklarında, geçici depolama için hiçbir CPU yazmacı boş harcanmaz.

Dezavantajları:

- Komut boyutu sabit değil; komut boyutlarındaki büyük değişiklik kod çözme kompleksini oluşturur
- Komut yürütme başına döngü, komuttan komuta değişiklik gösterir.

- Bellek erişimleri genellikle yavaştır, bu yüzden çok fazla referans performans düşüşüne neden olur.

## 2.2 Bellek

### 2.2.1 RAM - Random Access Memory (Rastgele Erişimli Bellek)

RAM bilgisayarın belleği olarak hizmet veren ve verileri geçici olarak saklayan bir donanım aracıdır. RAM, bir bilgisayardaki ana bellek birimidir ve HDD, SSD gibi hard disk türlerinden veya optik sürücü gibi diğer depolama türlerine oranla epey hızlı işlem gücüne sahiptir.

Rastgele erişimli bellek volatil olarak çalışır. Yani üzerine gelen güç kesildiğinde içerisindeki bilgiler kaybolur. Yukarıda da belirttiğimiz gibi verileri geçici olarak tutmasındaki nedenlerden biri de bundan ötürüdür. Bilgisayar yeniden başlatıldığında, işletim sistemi ve diğer dosyalar, genellikle bir HDD veya SSD'den RAM'e yeniden yüklenir.

RAM için kısa süreli hafıza birimi terimini kullanırsak, hard disk için ise uzun süreli hafıza birimidir diyebiliriz. Kısa süreli bellek, anında çalışmaya odaklanır, ancak herhangi bir zamanda yalnızca sınırlı sayıda verileri koruyabilir. Sınırlı sayıda verileri muhafaza ediyor olması ve anında çalışmaya odaklı olması onun hard disk ile karşılıklı bir ilişki içine girmesine sebep olur. RAM dolduğunda, bilgisayarın işlemcisi sürekli olarak RAM'deki eski verileri yeni verilerle takas etmek için hard diske gitmelidir. Bu işlem bilgisayarın çalışmasını yavaşlatır.

Ram için söylenen rastgele erişim terimi, herhangi bir hafıza adresine doğrudan erişilebilmesi ile alakalıdır. RAM, verilerin doğrudan belirli yerlere kaydedilip alınabilmesini sağlayacak şekilde düzenlenir ve kontrol edilir. Hard disk ve CD-ROM gibi diğer depolama türlerine de doğrudan veya rastgele erişilir, ancak rastgele erişim terimi bu diğer depolama türlerini tanımlamak için kullanılmaz.

## RAM Çalışma Mantığı

Ram'ın içinde 0 ve 1 leri bulunduran bölümler vardır. Bu bölümlerin herbirine hücre denir. İşlemci talimatı gönderir .Ram üzerindeki adreslerde ilgili bilgi bulunur ve adres ram kontrolörüne yollanır o da işlemciden gelen isteği koordine ederek belirli adrese yollar. Bu adres üzerindeki transistörler hücreleri açarak gereken kapasitördeki bilginin okunmasını sağlar.Belli bir voltaj değerinin üzerinde şarj olmuş kapasitör ikili sayı sisteminde 1 leri ve altında şarj olmuş kapasitör de 0 ları gösterir. Veriler böyle şekilde değerlendirilir.

## RAM Çeşitleri

### Dinamik RAM (DRAM)

Eğer RAM bellek kondansatör yapıya sahipse dinamik RAM olarak tanımlanır. Bu model kuşkusuz manyetik bellek modellerine göre çığır açan bir teknoloji ürünüdür. Ama en iyi RAM modeli değildir. Dinamik RAM bellek, kapasitör yapıda oldukları için devamlı olarak belirli bir akımda beslenmesi gereklidir. Bu sırada hafıza görevindeki kapasitörler şarj-deşarj olur. Şarj- deşarj durumundan dolayı hızda kayıp olur. DRAM ilk üretildiğinde FPM RAM ismiyle tanıtılmıştır. Ayrıca DRAM ekran kartında ve taşınabilir bilgisayar sistemlerine ait özel adlar ile de anılırlar.

### Statik RAM (SRAM)

Statik RAM modeli ise yarı iletken maddelerden oluşur. Yarı iletkenlerin anahtarlama özelliği sayesinde DRAM modelinin aksine RAM üzerine veri geldiğinde çalışır diğer durumlarda bellekler durağandır.(statiktir) SRAM, DRAM türüne göre daha hızlı çalışırlar. Fakat maliyetinin yüksek olmasından dolayı çok özel sistemlerde kullanılır. Bilgisayar anakart içerisinde sadece Cache bellek içerisinde SRAM kullanılır.

### FlashRAM (FRAM)

Pek çok kez sızıntı akım hakkında bazı şeyler duyarız. Mesela insanlar, elektronik ürünlerin fişini çekin çünkü onlar çalışmadığında bile akım çekiyor diye

söyler. İşte yukarıdaki duruma sebep olan ürün FRAM dir. Bu bellekler radio, video player, televizyon gibi aletlerin bazı özelliklerini kaybetmesini engellemek için kullanılır. Az bir enerji tüketimi ile verileri saklarlar.

### **VideoRAM (VRAM)**

Video ve ekran kartları için üretilmişlerdir. Gerçek anlamda DRAM den bir farkı yoktur. Fakat bilgisayarların karmaşıklaması ile birlikte işlemciye verilerin daha hızlı aktarılması için anakart dan ayrı olarak ekran kartına tümleşik RAM üretilmiştir.

### **RAMDAC (Dijital Analog Dönüştürücü RAM)**

VGA ekran modelleri sadece analog görüntüyü ekranda gösterebilir. Bundan dolayı ekran kartlarında RAMDAC denilen RAM kullanılmaktaydı. Bu modül yardımı ile dijital görüntü verileri analog verilere çevriliyordu. LCD ve LED ekranların çıkması ile bu sorun ortadan kalkmıştır.

### **Enhanced DRAM**

Geliştirilmiş DRAM'ler L2 cache hafızada kullanılır. 35 ns. DRAM içeresine 256 bayt 15 ns. SRAM eklenmesi suretiyle oluşturulmuştur. EDRAM aynı zamanda SRAM bölgeleri, verileri, yavaş olan DRAM bloklarından toplayabildiklerinden hız kazanır. Veri istendiğinde yavaş olan DRAM 128 bitlik bütün bir bloğu hızlı olan SRAM' gönderir.

### **SDRAM (Senkronize DRAM)**

İşlemcilerin hızlanması ile birlikte bu işlemcilerin maksimum seviyede işlem görebilmeleri için yüksek hızlı RAM'lere ihtiyaç duyulmuştur. SDRAM'le birlikte işlemci ve RAM birbirine aynı saat hızında kilitlenirler. Böylece işlemci ve RAM aynı saat hızında senkronize olarak çalışmaktadır. Bu nedenle de daha hızlı ve stabil çalışmaları sağlanmıştır.

### **DDR RAM (Double Data Rate)**

DDR RAM yüksek bant genişliği ve yüksek performans sunan hafıza sistemi spesifikasyonudur. Eski hafıza mimarilerinde, veri iki yönlü saat vuruşunun sadece bir anında veri aktarılıyordu. DDR sisteminde, saat vuruşunun iki yanında da veri aktarımı yapılarak, PC133 hafıza sisteminin hızı ikiye katlanarak 266MHz'a hızza ulaşılır. DDR teknolojisi, en çok ilgi gören 3D, Video ve Internet uygulamalarında yüksek hız sunmak için gerekli bant genişliğini sunarak, performansta yeni jenerasyon anlamına geliyor.

### **DDR2 RAM**

DDR2 (Double Data Rate), bir RAM türüdür, bilgisayarlarda ana bellek olarak kullanılır. Hızları 400mhz ile 1066 mhz arası değişir. DDR2, DDR bellek teknolojisinin bir ileri kuşağıdır. DDR2 bellek teknolojisi, daha fazla hız, yüksek bant genişlikleri, düşük güç tüketimi ve iyileştirilmiş ısı davranışını özelliklerini beraberinde getirmektedir.

### **DDR3 RAM**

DDR3 SDRAM, bir bilgisayarın veya başka bir sayısal elektronik aletin çalışıtı verileri yüksek hızlı biçimde saklamasında kullanılan bir RAM teknolojisidir. Sağladığı temel yarar, Giriş/çıkış (I/O) yolunu içerdigi bellek hücrelerinden 4 kat daha hızlı kullanabilmesi ve böylece yol hızını artırarak eski teknolojilerden birim zamanda daha fazla iş üretmesidir. Yalnız bunu başmanın maliyeti, daha fazla gecikme süresidir. DDR SDRAM'lerin 2,5 V ve DDR2 SDRAM'lerin 1,8 V'luk kaynak gerilimi gereksinimlerine karşın DDR3 SDRAM'ler 1,5V'luk gerilim gereksinimleriyle DDR2'lerden tüketimi vaadiyle ortaya çıktılar.

### **DDR4 RAM**

DDR4 SDRAM "Çift veri hızlı, 4. Nesil senkronize ve dinamik RAM" olarak geçiyor. Gerilimi azaltılmış, artan transfer oranları ve yüksek hızı ile çok

daha verimli bir hal almıştır. En son RAM 'ler yani DDR3'ler 2007 yılında çıktı ama geliştiriciler 2005 yılında DDR4 üzerinde çalışmalara başlamıştı. Samsung 2011 yılında ilk DDR4 belleğini imal etti ve bu teknoloji bilgisayar dünyasındaki en büyük gelişmelerden biridir.

### 2.2.2 ROM (Read Only Memory)

Rom bellek, Read Only Memory (sadece okunabilir bellek) olarak tanımlanır. ROM, sadece okunabilir bellekler için kullanılan genel bir ifadedir. İlk üretilen ROM sadece okunabilir özelliktedir. Daha sonra üretilen ROM çeşitleri üzerinde elektriksel yöntemlerle değişiklik yapılmaktadır. Bu tipteki hafıza birimleri elektrik kesildiğinde dahi bilgilerin saklanması gerektiği durumlarda kullanılmaktadır. Genel olarak dört gruba ayrılır.

#### ROM Çeşitleri

##### BIOS

Rom bellek çeşitlerine verilebilecek en güzel örnektir. BIOS cipi bilgisayar anakartında bilgisayarın açılış bilgilerini saklayan rom bellek türüdür. Bu cip anakart üzerindeki BIOS pili adı verilen saat pili yardımıyla beslenir.

#### PROM (Programmable read only memory- programlanabilir ROM)

Özel bir programlayıcı ve program yardımcı ile bilgilerin sadece bir kez yazılıa bildiği ROM bellek türüdür. Ucuz olduklarından dolayı tasarım örneklemeye işlemlerinde en iyi ürüne ulaşılana kadar kullanılırlar.

#### EPROM (Erasable programmable read only memory- silinebilir programlanabilir ROM )

Her zamanki gibi gelişen teknolojinin etkisi ile sadece yazılabilir ROM türü olan PROM'un yetersiz kalmasından dolayı EPROM üretilmiştir. Bu yonga

hem yazılabilir hemde silinebilir ROM bellektir. Fakat yeni bir program yüklemek için mevcut programın ultraviyole ışın yardımı ile silinmesi gereklidir. Bir çok kez yazılıp silinebilir olmasına karşın silmek ve programlamak için ayrı cihaza ihtiyaç vardır. Bundan dolayı maliyet oldukça artmaktadır. Üretici firmalar EEPROM ile PROM'un aksine kullanıcıya ROM bellek ile çalışma fırsatı sağlamışlardır.

**EEPROM (Erasable programmable read only memory- silinebilir programlanabilir ROM )**

EEPROM, EPROM yongasının eksik kalmış tüm yönlerini kapatan ROM bellek türüdür. Bu bellek yeni bir program yüklemek istendiğinde yüklü programı silmeden üzerine yeni bir program yazma fırsatı verir. Ayrıca içindeki programları silmek için ayrı bir cihaza ihtiyaç kalmamıştır. Bunun sebebi ise EEPROM'un elektriksel yöntemle programlanmasıdır.

**Flash Bellek**

Flash bellekler ROM ailesine verilebilecek en gelişmiş üründür. EEPROM da tek bir bayt büyülüüğünde olan değişiklik hızı Flash Bellekte blok blok yapılır. Bu bloklar ilk başlarda 512 bayt hızında iken şimdi bu hız daha da artmıştır. Flash Bellek, Rom bellek türündedir çünkü Rom bellek, Ram gibi elektrik kesintisinde bilgileri kaybetmez ve hard disk gibi diğer bellek türlerine göre de hızlı bir şekilde yazılıp silinebilir. Ayrıca tak çalıştır olarak da kullanılır.

RAM	ROM
Stands for Random Access Memory	Stands for Read Only Memory
It keeps operating system, application program, data in use	It contains programs used for booting up the computer and to run initial diagnostics.
Data in RAM can be accessed in any order and can be modified (changed or deleted)	ROM is fixed or data cannot be modified (read only memory)
RAM memory holds data temporarily while the program is being used	ROM is a fixed memory system and virtually indestructible
RAM is volatile (It loses data when power is turned off)	ROM is non-volatile (retains data even when power is off)
Higher units of space recommended up to GB (meant for running application programs)	Storage capacity not an issue as it is not used in running application programs
Cheaper	More expensive
RAM is fast	ROM memory is extremely fast compared to ROM
Types of RAM-dynamic RAM & static RAM	Types of ROM: PROM (programmable read-only memory) like CD ROM, EPROM, (Erasable Programmable Read-only Memory) EEPROM.(Electrically Erasable Programmable Read-Only Memory).
Example of RAM: RAM chips like 2GB, 4GB, 8GB etc of different companies like Corsair, Kingston etc.	Example of ROM: cartridge in video game consoles, computer BIOS.

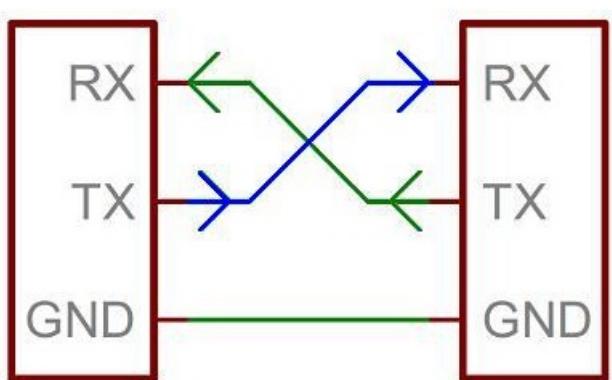
Şekil 2.6: RAM ve ROM arasındaki farklar

## 2.3 Çevre Birimleri

### 2.3.1 Seri İletişim

Seri veri iletimi, bir veri içindeki bitlerin aynı hat üzerinden art arda gönderilmesidir. Bilgisayar ağlarında kullanılan iletişim seri iletişimdir. Seri veri

iletiminde, bir kerede bir karakterin sadece bir biti iletilir. Alıcı makine, doğru haberleşme için karakter uzunluğunu, başla-bitir (start-stop) bitlerini ve iletim hızını bilmek zorundadır. Paralel veri iletiminde, bir karakterin tüm bitleri aynı anda iletildiği için başla-bitir bitlerine ihtiyaç yoktur. Dolayısıyla doğruluğu daha yüksektir. Seri veri yollarının bazı örnekleri arasında SPI,  $I^2C$ , DC-BUS, UNI / O ve 1-Wire bulunmaktadır.<sup>1 2</sup>



Şekil 2.7: Seri iletişim

### Senkron İletişim

Senkron iletişim alıcı ve vericinin eş zamanlı çalışması anlamına gelir. Yani alıcı ve verici aynı saat(clock) üzerinde olmalıdır. Önce gönderici taraf belirli bir karakter gönderir. Bu her iki tarafça bilinen iletişime başlama karakteridir. Alıcı taraf bu karakteri okursa iletişim kurulur. Verici bilgileri gönderir. Transfer işlemi veri bloku tamamlanana ya da alıcı verici arasındaki eşleme kayboluncaya kadar devam eder. Senkron veri iletişimin protokolünde bir çerçeveye içinde blok veri transferi yapılarak daha hızlı seri veri transferi sağlanır. Bu durumda çerçevenin biçimini tanımlayan özel kontrol karakterleri de veri ile beraber kullanılır.<sup>3 4 5</sup>

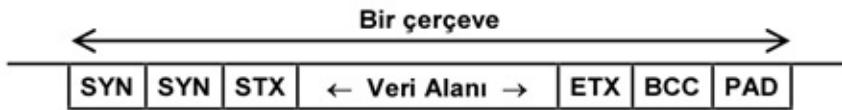
<sup>1</sup><https://otomasyonadair.com/2015/12/16/paralel-iletisim-seri-iletisim-nedir-ne-degildir/>

<sup>2</sup><https://arduinoblog.wordpress.com/2017/07/17/seri-iletisim/>

<sup>3</sup><https://otomasyonadair.com/2015/12/16/paralel-iletisim-seri-iletisim-nedir-ne-degildir/>

<sup>4</sup><http://herenkeskin.com/seri-haberlesme-protokollerleri-uart-spi-i2c/>

<sup>5</sup>[http://www.yildiz.edu.tr/~uzun/MS\\_PDF/PMS\\_15\\_VeriStandart.pdf](http://www.yildiz.edu.tr/~uzun/MS_PDF/PMS_15_VeriStandart.pdf)



Şekil 15-7 İki senkronlu seri veri protokolü kullanan senkron seri veri biçiminin bir çerçevesi

Tablo 15-1 İki senkron seri veri protokolünde kullanılan özel karakterler

Karakter	ASCII kodu	Açıklama
SYN	16	Senkron karakteri
PAD	FF	Çerçeve bloğunun sonu
DLE	10	Veri bağlantısından kaçış
ENQ	05	Araştırma
SOH	01	Başlığın başlangıcı
STX	02	Yazının (verinin) başlangıcı
ITB	0F	Ortadaki iletilen bloğun sonu
ETB	17	iletilen bloğun sonu
ETX	03	Yazının (verinin) sonu

Şekil 2.8: İki senkronlu seri veri protokolü

### Asenkron İletişim

Herhangi bir zamanda veri gönderilebilir. Veri gönderilmemiş zaman hat boşta kalır. Senkron seri iletişimden daha yavaştır. Her veri grubu ayrı olarak gönderilir. Gönderilen veri bir anda bir karakter olacak şekilde hatta bırakılır. Karakterin başına başlangıç ve sonunda hata sezmek için başka bir bit eklenir. Başlangıç için başla biti (0), veri iletişimini sonlandırmak için ise dur biti (1) kullanılır.<sup>6</sup> <sup>7</sup>

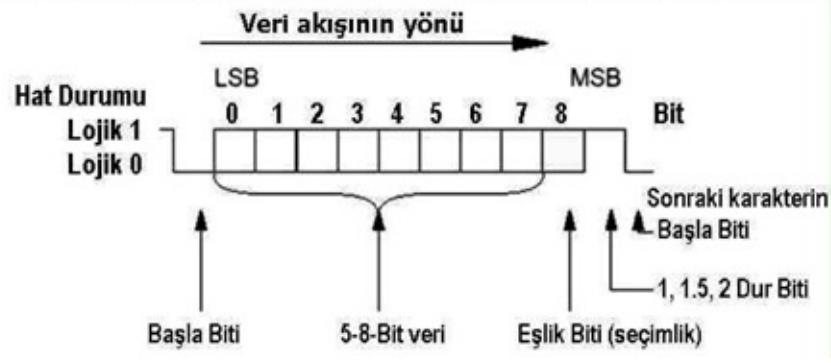
### I2C

Senkron haberleşme protokollerinden diğer bir tanesidir. Half-duplex (yarı eş zamanlı çift yönlü çalışabilen) olarak çalışır. Yani veri iletimi çift yönlü olur ancak aynı anda hem veri gönderip hem de alınamaz. Buna telsizleri örnek verebiliriz. Minimum bilgi alışverisi gerçekleştirilecek yerlerde tercih edilir. İletişim için bant genişliği oldukça düşüktür. SPI'da olduğu gibi master-slave

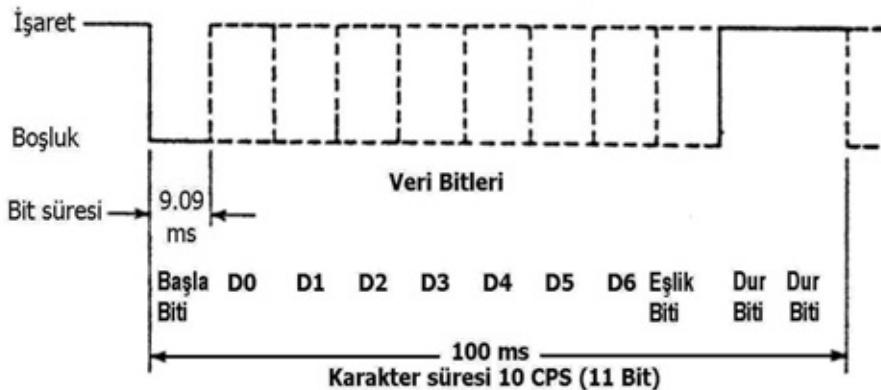
<sup>6</sup><https://otomasyonadair.com/2015/12/16/paralel-iletisim-seri-iletisim-nedir-ne-degildir/>

<sup>7</sup>[http://www.yildiz.edu.tr/~uzun/MS\\_PDF/PMS\\_15\\_VeriStandart.pdf](http://www.yildiz.edu.tr/~uzun/MS_PDF/PMS_15_VeriStandart.pdf)

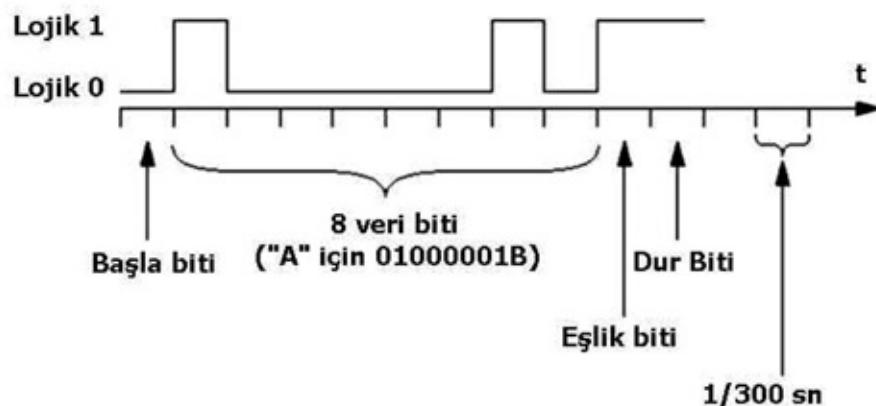
Asenkron seri veri iletişim protokolü, veri iletişiminin hızı, veri şeclinin çerçevesini kapsar.



Şekil 15-4 Asenkron protokol için seri veri biçimleri

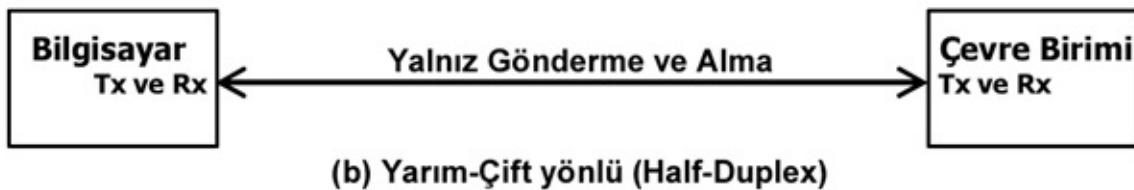


Şekil 15-5 110 baud, 7-bit veri, 1 eşlik, 2 dur biti çerçeve şeklindeki Asenkron seri veri biçimleri



Şekil 15-6 "A" karakterinin 8 veri biti, 1 dur biti, tek eşlik, ve 300 baud hızında gönderilmesi

Şekil 2.9: Asenkron iletişim



Şekil 2.10: I2C

yapısını kullanır.<sup>8</sup> <sup>9</sup>

## SPI

Senkron haberleşme protokollerinden bir tanesidir. Fully duplex (es zamanlı çift yönlü çalışabilen) olarak çalışabilir. Haberleşme gerçekleştirilecek cihazlar arasında master-slave ilişkisi vardır. Birden fazla slave cihazla haberleşme sağlanabilir. Senkron olarak çalıştığı için mutlaka bir clock sinyaline ihtiyaç duyulur. Kısa mesafeli iletimde kullanılır.<sup>10</sup> <sup>11</sup>

## Bitbang

Bir seri arabirim standarı taklit etmek için bir mikrodenetleyicinin genel amaçlı portlarını kullanan bir teknik. Tüm işlemciler seri olarak desteklenmez ve bu durumlarda seri iletişim kurmak için kendi kodunuzu tasarlamamanız gereklidir. Bu tür kodlamaya bit banging denir. Örneğin, PIC16C5x ve 16F5x cihazlarının herhangi bir seri desteği yoktur. Bu nedenle bir projeye seri cihazlar eklemek için iletişimi yönetmek için kod oluşturmanız gereklidir.

Avantajları:

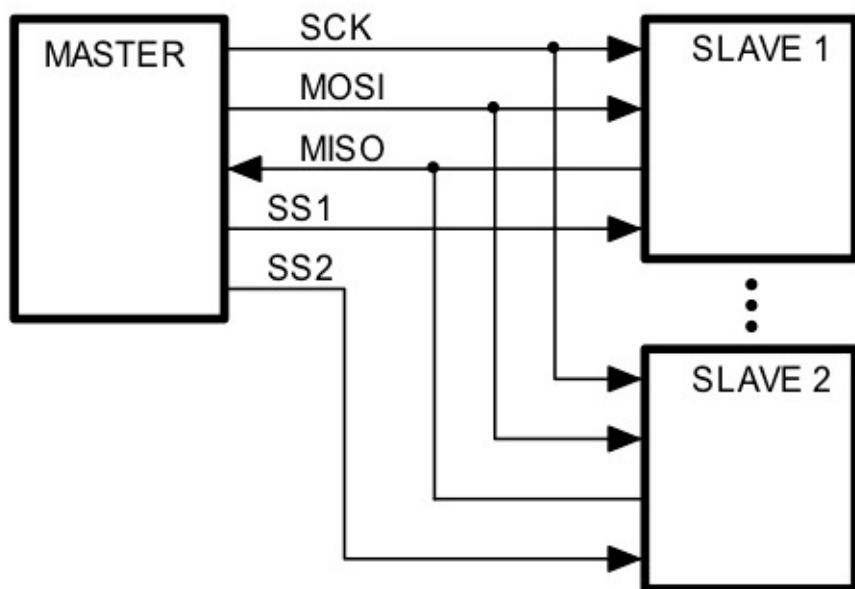
- Düşük maliyetli RISC işlemcileri kullanarak projeleri gerçekleştirebilirsiniz.

<sup>8</sup><http://herenkeskin.com/seri-haberlesme-protokoller-uart-spi-i2c/>

<sup>9</sup>[http://www.yildiz.edu.tr/~uzun/MS\\_PDF/PMS\\_15\\_VeriStandart.pdf](http://www.yildiz.edu.tr/~uzun/MS_PDF/PMS_15_VeriStandart.pdf)

<sup>10</sup><http://herenkeskin.com/seri-haberlesme-protokoller-uart-spi-i2c/>

<sup>11</sup>[http://www.yildiz.edu.tr/~uzun/MS\\_PDF/PMS\\_15\\_VeriStandart.pdf](http://www.yildiz.edu.tr/~uzun/MS_PDF/PMS_15_VeriStandart.pdf)



Şekil 2.11: SPI



Şekil 2.12: SPI

- İletişim sürecinin her yönü üzerinde tam kontrol sahibi olursunuz ve bu süreci ihtiyacı vardır.
- Farklı protokol ailelerinden cihazları karıştırabilirsiniz.
- İstediğiniz kadar çok veya birkaç cihaz komutunu uygulayabilirsiniz.

Dezavantajları:

- Tüm iletişim süreçlerini kodlamalısınız ve bu değerli bir bellek alıyor.
- Tüm zamanlama değerlendirmelerinden siz sorumlusunuz.

Bu yöntem, sinyal zamanamasının genellikle çok kritik olmadığı özellikle senkronizasyon protokollerinde (SPI, I2C) kullanılabilir.<sup>12</sup> <sup>13</sup>

## JTAG

JTAG (adını kodlayan Joint Test Action Group'dan sonra) tasarımları doğrulamak ve üretimden sonra baskılı devre kartlarını test etmek için bir endüstri standartıdır. JTAG standartları, satıcıya özgü özellikler sağlamak için uzmanlaşmış değişkenlere sahip birçok yarı iletken çip üreticisi tarafından genişletilmiştir.

### Tarihçe

1980'lerde çok katmanlı devre kartları ve kurşunsuz çerçeve entegre devreler (IC'ler) standart hale geldi ve probalar için mevcut olmayan IC'ler arasında bağlantılar yapıldı. Devre levhalarındaki üretim ve saha arızalarının büyük bir çoğunluğu, levhalardaki kötü lehim bağlantılarına, kart bağlantıları arasındaki kusurlara veya IC pedlerinden bağlantı kablolarına ve bağlantı kablolarının pim kurşun çerçevelerine bağlıydı. Ortak Test Eylem Grubu (JTAG) 1985 yılında bir IC pedinden diğerine bir pim çıkışı sağlamak için oluşturulmuştur, böylece bu hatalar keşfedilmiştir.

Endüstri standardı, IEEE Std olarak 1990 yılında bir IEEE standardı haline geldi. İlk kullanımdan yıllar sonra 1149.1-1990. Aynı yıl, Intel ilk işlemcilerini JTAG (80486) ile piyasaya sürdü ve bu da tüm üreticiler tarafından

<sup>12</sup><http://www.dnatechindia.com/Tutorial/8051-Tutorial/BIT-BANGING.html>

<sup>13</sup>[http://www.dwhoffman.com/bit\\_banging/bbi2c\\_preview.pdf](http://www.dwhoffman.com/bit_banging/bbi2c_preview.pdf)

daha hızlı bir endüstri benimsenmesini sağladı. 1994 yılında, sınır tarama tanımlama dilinin (BSDL) bir açıklamasını içeren bir ek eklenmiştir. PRESTOAD'dan SAMPLE kullanımını ve OBSERVE\_ONLY hücreleri için daha iyi bir uygulama kullanımını ayıran EXTEST için all-zeros kullanımı ile ilgili daha fazla iyileştirmeler yapılmış ve 2001 yılında piyasaya sürülmüştür. 1990 yılından bu yana, bu standart dünya çapında elektronik şirketleri tarafından benimsenmiştir. Sınır taraması büyük ölçüde JTAG ile eşanlımlıdır, ancak JTAG bu tür üretim uygulamalarının ötesinde temel kullanımlara sahiptir.

### Hata ayıklama

JTAG'ın erken uygulamaları, kurulu seviye testini hedeflese de, JTAG standartı, cihaz, kart ve sistem testi, tanımlama ve hata izolasyonu konularında yardımcı olmak için tasarlanmıştır. Günümüzde JTAG, entegre devrelerin alt bloklarına erişmenin birincil yolu olarak kullanılmakta, bu da diğer hata ayıklama yeteneğine sahip olmayan herhangi bir iletişim kanalına sahip olmayan gömülü sistemlerdeki hata ayıklama için önemli bir mekanizma haline gelmektedir. [Kaynak belirtilmeli] Çoğu sistemde JTAG tabanlı hata ayıklama CPU sıfırlama işleminden sonra ilk komuttan itibaren, herhangi bir şey kurulmadan önce çalışan önyükleme yazılımının geliştirilmesine yardımcı olur. Bir devre içi emülatör (veya daha doğru bir şekilde bir "JTAG adaptörü"), hedef CPU'nun içindeki yonga üstü hata ayıklama modüllerine erişim için taşıma mekanizması olarak JTAG'ı kullanır. Bu modüller, yazılım geliştiricilerinin gömülü bir sistemin yazılımını gerektiğiinde doğrudan makine talimat seviyesinde veya (daha tipik olarak) yüksek seviyeli dil kaynak kodu açısından hata yapmasına izin verir.

Sistem yazılımı hata ayıklama desteği, birçok yazılım geliştiricisinin JTAG'a ilgi duymasının ana sebebidir. PowerPC, MIPS, ARM, x86 gibi birçok silikon mimarisi, temel JTAG protokolü etrafında tüm yazılım hata ayıklama, talimat izleme ve veri izleme altyapısı inşa etti. Genellikle bireysel silikon satıcıları sadece bu uzantıların parçalarını uygular. Bazı örnekler ARM CoreSight ve Nexus'un yanı sıra Intel'in BTS (Branch Trace Storage), LBR (Son Şube Kayıt) ve IPT (Intel Processor Trace) uygulamalarıdır. NDA haricinde belgelenemeyen diğer birçok silikon satıcısına özgü uzanti var. JTAG standartının benimsenmesi, JTAG merkezli hata ayıklama ortamlarının, işlemciye özgü erken tasarımlardan uzaklaşmasına yardımcı oldu. İşlemciler normal olarak durdurulabilir, tek adım atılabilir veya serbestçe koşulabilir. Biri hem RAM'deki kod için (genellikle özel bir makine talimatı kullanarak) hem de ROM / flash'da kod kesme noktaları belirleyebilir. RAM'e toplu veri indirme

gibi veri kesme noktaları genellikle mevcuttur. Çoğu tasarımda “durma modu hata ayıklama” vardır, ancak bazıları hata ayıklayıcıların, ayıklanan çekirdeği durdurmaya gerek kalmadan kayıtlara ve veri yollarına erişmesine izin verir. Bazı araç takımları belirli bir alt programdan bir kayda ilk yedi erişimi görmezden gelmek üzere programlanmış bir mantık analizcisi gibi karmaşık donanım olaylarında hata ayıklayıcı (veya izleme) etkinliğini tetiklemek için ARM Gömülü İzleme Makrocell (ETM) modüllerini veya diğer mimarilerdeki eşdeğer uygulamaları kullanabilir.. .

Bazen FPGA geliştiricileri de hata ayıklama araçlarını geliştirmek için JTAG kullanırlar. Bir CPU içinde çalışan yazılımlarda hata ayıklamak için kullanılan aynı JTAG teknikleri, bir FPGA içindeki diğer dijital tasarım bloklarının hatalarını ayıklamaya yardımcı olabilir. Örneğin, özel JTAG talimatları, sınır tarama işlemlerine görünmeyen davranışların görünürüğünü sağlayarak, FPGA içindeki keyfi sinyal kümelerinden oluşturulan okuma yazmaçlarına izin vermek için sağlanabilir. Benzer şekilde, bu tür yazmaçların yazılması, başka türlü mevcut olmayan kontrol edilebilirliği sağlayabilir.

### Firmware saklamak

JTAG, cihaz programlayıcı donanımının, verileri dahili uçucu olmayan cihaz hafızasına (örn., CPLD’ler) aktarmasına izin verir. Bazı aygit programcılar, aygitin programlanması ve hata ayıklaması için iki yönlü bir amaca hizmet eder. FPGA’larda, geçici bellek cihazları, normalde geliştirme çalışmaları sırasında JTAG portu aracılığıyla programlanabilir. Ek olarak, dahili izleme yetenekleri (sıcaklık, voltaj ve akım) JTAG portu üzerinden erişilebilir.

JTAG programcılar, flaş belleğe yazılım ve veri yazmak için de kullanılır. Bu genellikle CPU’nun kullanılacağı gibi veri yolu erişimi kullanılarak yapılır ve bazen bir CPU tarafından işlenir, ancak diğer durumlarda bellek yongaları JTAG’ın kendileriyle iletişim kurar. Bazı modern hata ayıklama mimarileri, bir CPU’yu durdurmak ve devralmeye gerek kalmadan dahili ve harici bus master erişimi sağlar. En kötü durumda, sınır tarama tesisini kullanarak harici veriyolu sinyallerini sürmek genellikle mümkündür.

Pratik bir konu olarak, gömülü bir sistem geliştirirken, talimat deposunu taklit etmek, "hata ayıklama döngüsünü" (düzenleme, derleme, indirme, test etme ve hata ayıklama) gerçekleştirmenin en hızlı yoludur. [Kaynak belirtilmeli] Bunun nedeni devre içi olmasıdır Bir yönerge deposunu simüle eden emülatör, USB aracılığıyla, geliştirme sunucusundan çok hızlı bir şekilde güncellenebilir. Flash'a ürün yazılımı yüklemek için seri bir UART bağlantı

noktası ve önyükleyici kullanmak, bu hata ayıklama döngüsünü araçlar açısından oldukça yavaş ve muhtemelen pahalı hale getirir; JTAG aracılığıyla Flash'a (veya Flash yerine SRAM) ürün yazılımı yüklemek, bu aşırı uçlar arasında bir ara çözümüdür.

### Sınır tarama testi

JTAG sınır tarama teknolojisi, cihaz pinleri dahil, karmaşık bir entegre devrenin birçok mantıksal sinyaline erişim sağlar. Sinyaller, TAP üzerinden erişilebilen sınır tarama kaydında (BSR) gösterilir. Bu, test ve hata ayıklama için sinyallerin durumlarının kontrol edilmesinin yanı sıra test edilmesine izin verir. Bu nedenle, hem yazılım hem donanım (üretim) hataları bulunabilir ve bir işletim cihazı izlenebilir.

Yerleşik self-test (BIST) ile birleştirildiğinde, JTAG tarama zinciri, belirli statik hatalar (kısa devre, açılır ve mantık hataları) için bir IC'yi test etmek için düşük bir genel gider, gömülü çözüm sağlar. Tarama zinciri mekanizması genellikle olusabilecek zamanlama, sıcaklık veya diğer dinamik işletim hatalarını teşhis etmeye veya test etmeye yardımcı olmaz. Test senaryoları genellikle SVF veya onun ikili kardeşi XSVF gibi standartlaşmış formatlarda sağlanır ve üretim testlerinde kullanılır. Bitmiş tahtalarda bu tür testlerin gerçekleştirilebilmesi, günümüzün ürünlerinde Design For Test'in önemli bir parçasıdır ve ürünlerin müşterilere gönderilmesinden önce bulunabilecek hataların sayısını arttırr.

### Elektriksel özellikler

Bir JTAG arayüzü, bir çipe eklenen özel bir arabirimdir. JTAG versiyonuna bağlı olarak iki, dört veya beş pin eklenir. Dört ve beş pinli arayüzler, belirli koşulların sağlanması durumunda, bir tahtadaki çoklu yongaların JTAG hattlarının birlikte zincirleme yapılabilmesini sağlayacak şekilde tasarlanmıştır. İki pin arayüzü, bir yıldız topolojisine birden çok yonganın bağlanabilmesi için tasarlanmıştır. Her iki durumda da, bir test probunun sadece bir devre kartı üzerindeki tüm yongalara erişmek için tek bir "JTAG portuna" bağlanması gereklidir.

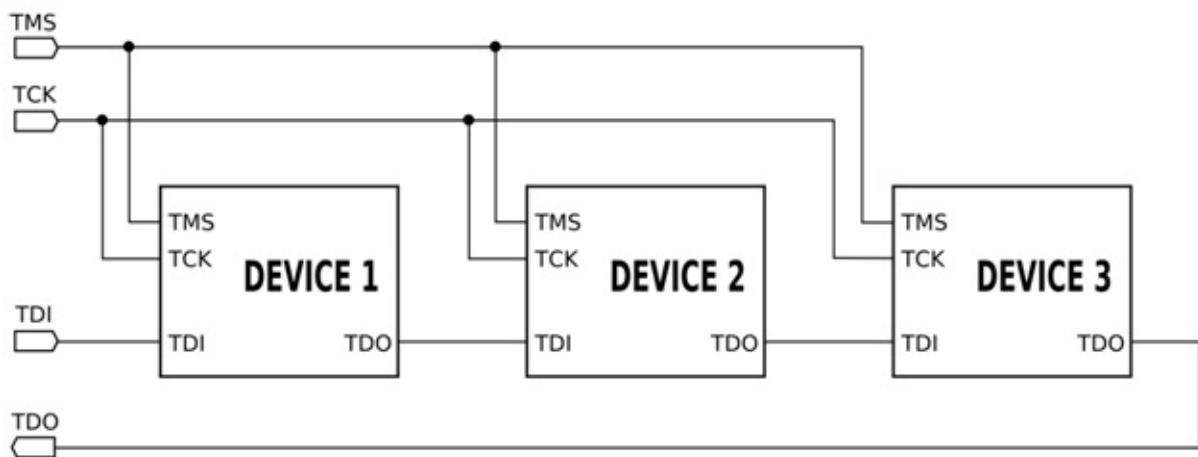
Konnektör pimleri şunlardır:

TDI (Test Verileri Girişi)

TDO (Test Verileri Çıkışı)

TCK (Test Saati)

TMS (Test Modu Seçimi) TRST (Test Sıfırlama) opsiyonel.



Şekil 2.13: JTAG zincir örneği

TRST pini, genellikle lojik olmayan, fakat bazen senkronize olan, test mantığına isteğe bağlı aktif-düşük sıfırlamadır. Pim mevcut değilse, test mantığı TCK ve TMS kullanılarak senkronizasyon durumuna senkronize olarak geçerek sıfırlanabilir. Test mantığını sıfırlamanın mutlaka başka bir şeyi sıfırlamayı gerektirmediğini unutmayın. Genellikle, hata ayıklanan çipin tamamını veya bir kısmını sıfırlayabilen bazı işlemci özel JTAG işlemleri vardır.

Tek bir veri hattı mevcut olduğundan, protokol seridir. Saat girişi TCK pinindendir. Bir bit veri, TDI'dan aktarılır ve TCK başına yükselen saat kenarı boyunca TDO'ya aktarılır. Farklı talimatlar yüklenebilir. Tipik IC'lere ilişkin talimatlar, çip ID, örnek giriş pimleri, sürücü (veya şamandıra) çıkış pimlerini okuyabilir, çip fonksiyonlarını manipüle edebilir veya baypas (çoklu çiplerin zincirlerini mantıksal olarak kısaltmak için TDI'yi boruya TDI).

Herhangi bir saat sinyalinde olduğu gibi, TDI'ya sunulan veriler, daha önce bazı çipe özgü Kurulum süreleri için geçerli olmalı ve ilgili (burada, artan) saat kenarından sonra bekleme süresine sahip olmalıdır. TDO verileri, TCK'nın düşme kenarından sonra bazı çipe özgü zamanlar için geçerlidir.

TCK'nın maksimum çalışma frekansı, zincirdeki tüm çipe bağlı olarak değişir (en düşük hız kullanılmalıdır), ancak tipik olarak 10-100 MHz'dir (bit başına 100-10 ns). Ayrıca TCK frekansları pano düzenine ve JTAG adaptör özelliklerine ve durumuna bağlıdır. Bir çipin 40 MHz JTAG saatı olabilir, ancak sadece JTAG olmayan işlemler için 200 MHz saat kullanıyorsa; Dü-

şük güç modundayken çok daha yavaş bir saat kullanması gerekebilir. Buna göre, bazı JTAG bağdaştırıcıları, bir RTCK (Dönüş TCK) sinyali kullanarak uyarlamalı bir saatlemeye sahiptir. Daha hızlı TCK freksansları, JTAG, bir programın çalıştırılabilir belleğini flash belleğe kaydederken olduğu gibi birçok veriyi aktarmak için kullanıldığında en faydalıdır.

Standartlaştırılmış bir JTAG durum makinesi aracılığıyla TMS adımlarındaki değişiklikler değişir. JTAG durum makinesi sıfırlanabilir, bir yönerge kaydına erişebilir veya yönerge kaydı tarafından seçilen verilere erişebilir.

JTAG platformları genellikle IEEE 1149.1 belirtimi tarafından tanımlanan avuç içine sinyaller ekler. Sistem Sıfırlama (SRST) sinyali oldukça yaygındır ve hata ayıklayıcıların JTAG desteğiyle sadece parçaları değil tüm sistemi sıfırlamasını sağlar. Bazen, ev sahibi tarafından veya JTAG aracılığıyla izlenen cihaz tarafından aktiviteyi tetiklemek için kullanılan olay sinyalleri vardır; veya belki de ek kontrol hatları.

Bazı tüketici ürünleri açık bir JTAG bağlantı noktası konnektörü sağlasa da, bağlantılar genellikle baskılı devre kartında geliştirme prototipinin ve / veya üretimin bir parçası olarak mevcuttur. İstisna edildiğinde, bu bağlantılar genellikle tersine mühendislik için en uygun araçları sağlar.<sup>14</sup>

### 2.3.2 Paralel İletişim

Paralel bir port, çevre birimleri bağlamak için bilgisayarlarda (kişisel ve diğer) bulunan bir arayüz türüdür. İsim, verilerin gönderilme şeklini ifade eder; Paralel portlar, tek seferde bitler gönderen seri arayüzlerin aksine, paralel iletişimde bir kerede birden fazla veri biti gönderir. Bunu yapmak için, paralel portlar kablolarında ve port konnektörlerinde birden fazla veri hattı gerektirir ve sadece bir veri hattı gerektiren güncel seri portlardan daha büyük olma eğilimindedir.

Paralel bağlantı noktaları, bir çok popüler bilgisayar çevre birimi bağlamak için kullanılabilir:

- Yazıcılar
- Tarayıcılar

---

<sup>14</sup><http://www.wiki-zero.net/index.php?q=aHR0cHM6Ly91bi53aWtpcGVkaWEub3JnL3dpa2kvS1RBw>

- CD yazıcıları
- Harici sabit diskler
- Iomega Zip çıkarılabilir sürücüler
- Ağ bağdaştırıcıları
- Teyp yedekleme sürücüleri

Birçok paralel bağlantı noktası türü vardır, ancak bu terim, 1970'lerden 2000'lere kadar çoğu kişisel bilgisayarda bulunan yazıcı bağlantı noktası veya Centronics bağlantı noktasıyla en yakından ilişkili hale gelmiştir. Uzun yıllar boyunca bir endüstri standartıydı ve nihayet 1990'ların sonunda Geliştirilmiş Paralel Bağlantı Noktası (EPP) ve Genişletilmiş Kapasite Portu (ECP) çift yönlü versiyonlarını tanımlayan IEEE 1284 olarak standart hale getirildi. Bugün, Ethernet ve Wi-Fi bağlantılı yazıcılar kullanarak ağ üzerinden yazdırmanın yanı sıra, Evrensel Seri Veri Yolu (USB) aygıtlarının yükselmesi nedeniyle paralel bağlantı noktası arabirimini neredeyse yok.

Paralel bağlantı noktası arabirimini, başlangıçta IBM PC uyumlu bilgisayarlarda Paralel Yazıcı Bağdaştırıcısı olarak biliniyordu. Öncelikle, IBM'in sekiz bit genişletilmiş ASCII karakter setini yazıcıya yazdırmak için kullanılan yazıcıları kullanmak üzere tasarlandı, ancak diğer çevre birimlerine uyum sağlamak için de kullanılabilir. Grafik yazıcılar, diğer cihazlarla birlikte, sistemle iletişim kurmak için tasarlanmıştır.

USB'nin gelişinden önce, paralel arabirim, yazıcılar dışındaki bazı çevre aygıtlarına erişmek için uyarlanmıştır. Paralel portun bir erken kullanımı, yazılım kopya koruması olarak bir uygulama yazılımı ile sağlanan donanım anahtarları olarak kullanılan dongle'lar içindi. Diğer kullanıcılar arasında CD okuyucular ve yazarlar, Zip sürücüleri, tarayıcılar, harici modemler, oyun kumandaları ve oyun çubukları gibi optik disk sürücüleri bulunmaktadır. En eski taşınabilir MP3 çalarların bazıları şarkılari cihaza aktarmak için paralel port bağlantısı gerektiriyordu. Adaptörler, paralel olarak SCSI cihazlarını çalıştırmak için mevcuttu. EPROM programlayıcıları ve donanım kontrolörleri gibi diğer cihazlar paralel port üzerinden bağlanabilir.

### Tarihsel kullanımlar

#### Arayüzler

1980'lerde ve 1990'larda PC'ye uyumlu çoğu sistemde, bir veya üç bağlantı noktası vardı;

- Mantıksal paralel bağlantı noktası 1: G / Ç bağlantı noktası 0x3BC, IRQ 7 (genellikle tek renkli grafik bağdaştırıcılarında)
- Mantıksal paralel bağlantı noktası 2: G / Ç bağlantı noktası 0x378, IRQ 7 (özel IO kartları veya anakart içine yerleşik bir denetleyici kullanarak)
- Mantıksal paralel bağlantı noktası 3: G / Ç bağlantı noktası 0x278, IRQ 5 (özel IO kartları veya anakartın içine yerleştirilmiş bir denetleyici kullanarak)

### Erişim

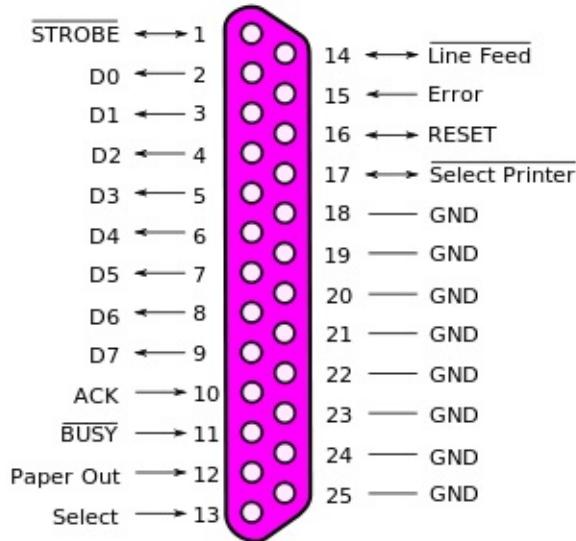
DOS tabanlı sistemler, LPT1, LPT2 veya LPT3 gibi aygit adları altında bulunan BIOS tarafından algılanan mantıksal paralel bağlantı noktalarını (sırasıyla, 1, 2 ve 3 mantıksal paralel bağlantı noktasına karşılık gelir) yapar. Bu adlar, Hat Yazdırma Terminali, Yerel Yazdırma Terminali veya Hat PrinTer gibi terimlerden türetilir. Benzer bir adlandırma kuralı, ITS, DEC sistemlerinin yanı sıra CP / M ve 86-DOS (LST) de kullanılmıştır. DOS'ta, paralel yazıcılara doğrudan komut satırından erişilebilir. Örneğin, "TYPE C: AUTOEXEC.BAT> LPT1:" komutu AUTOEXEC.BAT dosyasının içeriğini yazıcı bağlantı noktasına yönlendirir. LPT1 için bir diğer ad olarak bir PRN cihazı da mevcuttu. Bazı işletim sistemleri (Multiuser DOS gibi) bu sabit atamayı farklı yollarla değiştirebilir. Bazı DOS sürümleri, MODE tarafından sağlanan yerleşik sürücü uzantılarını kullanır veya kullanıcılar eşlemeyi dahili olarak CONFIG.SYS PRN=n yönergesini kullanarak (DR-DOS 7.02 ve üstü gibi) değiştirebilir. DR-DOS 7.02, alta yatan BIOS destekliyorsa, LPT4 için istege bağlı yerleşik destek sağlar.

Daha eski paralel yazıcı bağlantı noktalarında 8 bit veri yolu ve kontrol çıkışları için dört pin (Strobe, Linefeed, Initialize ve Select In) ve kontrol girişi için beş tane daha (ACK, Meşgul, Seç, Hata ve Kağıt Çıkışı) vardı. Veri aktarım hızı 150 kbit / s'dir. Daha yeni EPP'ler (Geliştirilmiş Paralel Bağlantı Noktaları) 8 bit veri yolu ve normal paralel yazıcı bağlantı noktasıyla aynı kontrol pinlerine sahiptir. Daha yeni bağlantı noktaları 2 MB / s'ye varan hızlara ulaşır.

Her bir girişin bir yazıcıyla kullanıldığından neler yaptığına daha yakından inceleyelim:

Tablo 2.1: Paralel port konnektörleri için pinoutlar

Pin Numarası (DB25)	Pin Numarası (36 pin)	Sinyal İsmi	Yönü	Register - bit	Inverted
1	1	Strobe	In/Out	Control-0	Yes
2	2	Data0	Out	Data-0	No
3	3	Data1	Out	Data-1	No
4	4	Data2	Out	Data-2	No
5	5	Data3	Out	Data-3	No
6	6	Data4	Out	Data-4	No
7	7	Data5	Out	Data-5	No
8	8	Data6	Out	Data-6	No
9	9	Data7	Out	Data-7	No
10	10	Ack	In	Status-6	No
11	11	Busy	In	Status-7	Yes
12	12	Paper-Out	In	Status-5	No
13	13	Select	In	Status-4	No
14	14	Linefeed	In/Out	Control-1	Yes
15	32	Error	In	Status-3	No
16	31	Reset	In/Out	Control-2	No
17	36	Select-Printer	In/Out	Control-3	Yes
18-25	19-30,33,17,16	Ground	-	-	-



Şekil 2.14: Paralel port konnektörü

1. Pin 1, strobe sinyalini taşır. 2,8 ve 5 volt arasında bir seviye korur, ancak bilgisayar bir bayt veri gönderdiğinde 0,5 voltun altına düşer. Bu voltaj düşüşü yazıcıya verilerin gönderileceğini bildirir.
2. Verileri taşımak için 2'den 9'a kadar pimler kullanılır. Bir bitin 1 değerine sahip olduğunu göstermek için doğru pin üzerinden 5 voltluk bir yük gönderilir. Pim üzerinde hiçbir ücret, 0 değerini gösterir. Bu, dijital bir bilgiyi gerçek zamanlı olarak bir analog kablo üzerinden iletmeyi basit fakat oldukça etkili bir yoludur.
3. Pin 10, onay sinyalini yazıcıdan bilgisayara gönderir. Pin 1 gibi, bir şarjı muhafaza eder ve bilgisayarın, verilerin alındığını bilmesini sağlamak için gerilimi 0,5 voltun altına düşürür.
4. Yazıcı meşgulse, Pin 11'i şarj eder. Ardından, bilgisayarın daha fazla veri almaya hazır olduğunu bilmesi için voltajı 0,5 voltun altına düşürür.
5. Yazıcı, Pin 12'de bir ücret göndererek kağıdın bittiğini bilerek bilgisayarı bilgilendirir.
6. Bilgisayar Pin 13'te bir ücret alıyorsa, cihazın çevrimiçi olduğunu bilir.

7. Bilgisayar 5 volt şarj kullanarak Pin 14 üzerinden yazıcıya bir otomatik besleme sinyali gönderir.
8. Yazıcının herhangi bir sorunu varsa, bilgisayarın bir hata olduğunu bildirmesi için voltağı Pin 15 üzerinde 0,5 volttan daha düşük bir değere düşürür.
9. Yeni bir yazdırma işi hazır olduğunda, bilgisayar yazıcıyı başlatmak için Pin 16'daki yükü düşürür.
10. Pin 17, bilgisayar tarafından yazıcıyı uzaktan almak için kullanılır. Bu, yazıcıya bir ücret göndererek ve yazıcıyı çevrimdışı istediğiniz sürece koruyarak gerçekleştirilir.
11. Pimler 18-25 topraklıdır ve düşük (0,5 volt'un altında) şarj için bir referans sinyali olarak kullanılır.

15 16

### 2.3.3 Network

#### Wi-Fi

Wi-Fi, IEEE 802.11 standartlarına dayalı cihazlarla kablosuz yerel alan ağı için bir teknolojidir. Wi-Fi, Wi-Fi Alliance'ın, birlikte çalışabilirlik sertifikasyonu testini başarıyla tamamlayan produklerde Wi-Fi Certified teriminin kullanımını kısıtlayan bir ticari markasıdır.

Wi-Fi teknolojisini kullanabilen cihazlar arasında kişisel bilgisayarlar, video oyun konsolları, akıllı telefonlar ve tabletler, dijital kameralar, akıllı TV'ler, dijital ses çalarları ve modern yazıcılar bulunur. Wi-Fi uyumlu cihazlar, WLAN ve kablosuz erişim noktası aracılığıyla İnternete bağlanabilir. Böyle bir erişim noktası (veya sıcak nokta), iç mekânda yaklaşık 20 metre (66 feet) menzile ve dışarıda daha geniş bir menzile sahiptir. Hotspot kapsama alanı, radyo dalgalarını engelleyen duvarlara sahip tek bir oda kadar küçük olabilir veya birden fazla çıkışan erişim noktası kullanılarak elde edilen kare kilometreye kadar daha büyük olabilir.

<sup>15</sup>[https://en.wikipedia.org/wiki/Parallel\\_port](https://en.wikipedia.org/wiki/Parallel_port)

<sup>16</sup><https://computer.howstuffworks.com/parallel-port1.htm>

Wi-Fi en çok 2.4 gigahertz (12 cm) UHF ve 5.8 gigahertz (5 cm) SHF ISM radyo bantlarını kullanır. Kablosuz ağ arabirim denetleyicisiyle kapsama alanındaki herkes ağa erişmeyi deneyebilir; Bu nedenle, Wi-Fi, kablolu ağlardan daha fazla saldırıyla (gizli dinleme olarak adlandırılır) karşı daha savunmasızdır. Wi-Fi Korumalı Erişim, Wi-Fi ağları arasında bilgi taşımak ve kişisel ve kurumsal ağlar için çözümler içeren bir teknoloji ailesidir. Wi-Fi Korumalı Erişim'in güvenlik özellikleri, güvenlik ortamı değişikçe daha güçlü korumalar ve yeni güvenlik uygulamaları içerecek şekilde sürekli olarak gelişmektedir.

### Wi-Fi Radyo Spektrumu

802.11b ve 802.11g, Birleşik Devletler'de Bölüm 15 Kurallar ve Yönetmelikler altında çalışan 2,4 GHz ISM bandını kullanır. Bu frekans bandı tercihi nedeniyle, 802.11b ve g ekipmanı zaman zaman mikrodalga fırınlardan, telsiz telefonlardan ve Bluetooth cihazlarından parazite maruz kalabilir.

Spektrum atamaları ve operasyonel sınırlamalar dünya çapında tutarlı değildir: Avustralya ve Avrupa, ABD'de 2,4 GHz bandı için izin verilen 11'in dışında iki kanal (12, 13) sağlarken, Japonya'da üç tane daha var (12-14). ABD ve diğer ülkelerde 802.11a ve 802.11g cihazları FCC Kural ve Yönetmelikleri'nin 15. Bölümünde izin verildiği şekilde lisanssız olarak işletilebilir.

Bir Wi-Fi sinyali, 2,4 GHz bandında beş kanalı kaplar. 2 ve 7 gibi beş veya daha fazla farklı iki kanal numarası çakışmaz. Bu nedenle, 1, 6 ve 11 kanallarının üst üste binmeye tek kanallar olduğu tekrarlanan atasözü, bu nedenle doğru değildir. Kuzey Amerika ve Birleşik Krallık'taki çakışan üç kanalın tek grubu olan 1, 6 ve 11 numaralı kanallar. Avrupa ve Japonya'da 802.11g ve 802.11n için 1, 5, 9 ve 13 numaralı kanalları kullanan önerilir. [Kaynak belirtilmeli]

802.11a, dünyanın çoğunda, bitişik kanalların üst üste geldiği 2.4 GHz ISM frekans bandından ziyade en az 23 örtüşmeyen kanal sunmakta olan 5 GHz U-NII bandını kullanmaktadır.

### Gömülü Sistemlerde Wi-Fi

Son birkaç yılda (özellikle 2007 itibariyle), gerçek zamanlı bir işletim sistemi içeren ve seri bağlantı noktası üzerinden haberleşebilen ve iletişim kuran herhangi bir aygıtın kablosuz olarak etkinleştirilmesini sağlayan yerleşik Wi-Fi modülleri mevcut hale gelmiştir. Bu basit izleme cihazlarının tasarımasına

izin verir. Bir örnek, evde bir hastayı izleyen portatif bir EKG cihazıdır. Bu Wi-Fi özellikli cihaz İnternet üzerinden iletişim kurabilir.

Bu Wi-Fi modülleri, OEM'ler tarafından tasarlanmakta, böylece uygulayıcılar, ürünleri için Wi-Fi bağlantısı sağlamak için yalnızca minimum Wi-Fi bilgisine ihtiyaç duymaktadır.

Haziran 2014'te, Texas Instruments, birleşik bir Wi-Fi MCU, SimpleLink CC3200 ile ilk ARM Cortex-M4 mikroişlemciyi tanıttı. Wi-Fi bağlantısı ile gömülü sistemleri mümkün kılar, tek-çipli cihazlar olarak inşa etmek mümkündür, bu da maliyetlerini ve minimum boyutlarını düşürür, bu da kablosuz ağ denetleyicileri ucuz olağan nesneler haline getirmeyi daha pratik hale getirir.<sup>17</sup>

### **Li-Fi**

Li-Fi, veri iletimi için ışık kullanan cihazlar arasında kablosuz iletişim için bir teknolojidir. Mevcut durumunda, görünür ışığın iletimi için sadece LED lambalar kullanılabilir. Terim ilk olarak 2011'de Edinburgh'daki TEDGlobal konuşması sırasında Harald Haas tarafından tanıtıldı. Teknik açıdan bakıldığında, Li-Fi, görünür ışık spektrumu, ultraviyole ve kızıl ötesi radyasyon üzerinden yüksek hızlarda veri iletebilen görünür bir ışık iletişim sistemidir.

Son kullanım açısından teknoloji Wi-Fi'ye benzer. En önemli teknik farklılık, Wi-Fi'nin veri iletmek için radyo frekansını kullanmasıdır. Verileri iletmek için ışık kullanmak, Li-Fi'nin daha yüksek bant genişliği boyunca çalışmak, elektromanyetik enterferansa (örn. Uçak kabinleri, hastaneler) duyarlı alanlarda çalışmak ve daha yüksek iletim hızları sunmak gibi çeşitli avantajlar sunmasını sağlar. Teknoloji, dünya genelinde çeşitli kuruluşlar tarafından aktif olarak geliştirilmektedir. 2351/5000 Bu optik kablosuz iletişim (OWC) teknolojisi, ağa bağlı, mobil, yüksek hızlı iletişimini Wi-Fi'ye benzer şekilde iletmek için bir ortam olarak ışık yayan diyonlardan (LED'ler) gelen ışığı kullanır.

Görünür ışık iletişimini (VLC), LED'lere giden akımı çok yüksek bir hızda kapatıp açarak, insan gözüyle fark edilmek için çok hızlı çalışır. Her ne kadar

<sup>17</sup><http://www.wiki-zero.net/index.php?q=aHR0cHM6Ly91bi53aWtpcGVkaWEub3JnL3dpa2kvV2ktRmk>

Li-Fi LED'leri veri iletmek üzere tutulacak olsa da, verileri taşımak için yeterli ışık yayarken, insan görünürlüğünün altına daraltılmış olabilirler. ışık dalgaları, çok daha kısa bir menzile sahip duvarlara nüfuz edemez, aynı zamanda Wi-Fi'ye göre hakelemeden daha güvenlidir. Li-Fi'nın bir sinyal iletmesi için doğrudan görüş hattı gerekli değildir; Duvarlardan yansyan ışık 70 Mbit / s'ye ulaşabilir.

Li-Fi, elektromanyetik interferansa neden olmadan uçak kabinleri, hastaneler ve nükleer santraller gibi elektromanyetik hassas alanlarda yararlı olma avantajına sahiptir. Hem Wi-Fi hem de Li-Fi, elektromanyetik spektrum üzerindeki veri iletilir, ancak Wi-Fi radyo dalgalarını kullanırken, Li-Fi görünür ışık, Ultraviyole ve Kızılıötesi kullanır.

Wi-Fi tam kapasiteye yakın olduğu için, Li-Fi'nin kapasite üzerinde neredeyse hiçbir sınırlaması yoktur. Görünür ışık spektrumu tüm radyo frekansı spektrumundan 10,000 kat daha büyütür. Araştırmacılar, 2013 yılında tipik hızlı geniş banttan çok daha hızlı olan 224 Gbit / s veri hızına ulaştı. Li-Fi'nin Wi-Fi'den on kat daha ucuz olması bekleniyor.

### **Li-Fi Standartları**

Wi-Fi gibi, Li-Fi kablosuz ve benzer 802.11 protokollerini kullanır, ancak daha büyük bant genişliğine sahip olan ultraviyole, kızılıötesi ve görünür ışık iletişimini kullanır (radyo frekansı dalgaları yerine).

VLC'nin bir kısmı IEEE 802 çalışma grubu tarafından kurulan iletişim protokollerinden sonra modellenmiştir. Ancak, IEEE 802.15.7 standarı güncel değildir: özellikle optik ortogonal frekans bölmeli çoğullama (O-OFDM) modülasyon yöntemlerinin kullanılmasıyla, optik kablosuz iletişim alanındaki en son teknolojik gelişmeleri dikkate almamaktadır. veri hızları, çoklu erişim ve enerji verimliliği için optimize edilmiştir. O-OFDM'in tanıtımı, optik kablosuz iletisimin standardizasyonu için yeni bir sürücünün gerekliliği anlamına gelir.

Bununla birlikte, IEEE 802.15.7 standarı, fiziksel katman (PHY) ve ortam erişim kontrolü (MAC) katmanını tanımlar. Standart, ses, video ve multimedya servislerini iletmek için yeterli veri hızları sunabilmektedir. Optik aktarım hareketliliği, altyapılarda yapay ışıklandırma ile uyumu ve ortam aydınlatması ile oluşabilecek parazitleri hesaba katar. MAC katmanı, bağlantıyı TCP / IP protokolüyle olduğu gibi diğer katmanlarla birlikte kullanmaya izin verir.

Standart, farklı oranlarda üç PHY katmanı tanımlar:

- PHY 1, açık hava uygulamaları için kurulmuş ve 11.67 kbit / s'den 267.6 kbit / s'ye kadar çalışmaktadır.
- PHY 2 katmanı, veri hızlarına 1.25 Mbit / s'den 96 Mbit / s'ye ulaşmayı sağlar.
- PHY 3, renk kayması anahtarlaması (CSK) adı verilen belirli bir modülasyon yöntemiyle birçok emisyon kaynağı için kullanılır. PHY III 12 Mbit / s'den 96 Mbit / s hızına ulaşabilir.
- PHY I ve PHY II için tanımlanın modülasyon formatları açık-kapalı anahtarlama (OOK) ve değişken darbe pozisyonu modülasyonudur (VPPM). PHY I ve PHY II tabakaları için kullanılan Manchester kodlaması, bir OOK sembolü "01" olan bir mantığı (0) ve bir OOK sembolü "10" olan bir mantığı (1) temsil eden, bir DC bileşenine sahip olan saatî içerir. DC bileşeni, mantıksal 0'ların uzun süreli çalışması durumunda hafif sönmeyi önler.

### Li-Fi Gömülü Kullanım Alanları

**Güvenlik:** Wi-Fi tarafından kullanılan radyo frekansı dalgalarının aksine, ışıklar duvarlara ve kapılara nüfuz edemez. Bu, onu daha güvenli hale getirir ve kimlerin ağa bağlanabileceğini kontrol etmeyi kolaylaştırır. Pencereler gibi saydam malzemeler kaplandığı sürece, bir Li-Fi kanalına erişim odanın içindeki cihazlarla sınırlıdır.

**Sualtı uygulama:** Uzaktan çalıştırılan su altı araçlarının (ROV) çoğu komutu iletmek için kablolar kullanır, ancak kabloların uzunluğu daha sonra ROV'lerin algılayabileceği alanı sınırlar. Bununla birlikte, bir ışık dalgası sudan geçerken, geri syinyalleri almak ve göndermek için araçlarda Li-Fi uygulanabilir.

Li-Fi'nin su altı uygulamalarında kullanılması teorik olarak mümkün olsa da, bunun faydası, mesafe ışığının suya nüfuz etmesi ile sınırlıdır. Önemli miktarda ışık 200 metreden daha fazla nüfuz etmez. 1000 metre geçmiş, ışık geçmez.

**Hastane:** Birçok tedavi şu anda birden fazla kişiyi içermektedir, Li-Fi sistemi hastaların bilgileri hakkında iletişimini iletmek için daha iyi bir sistem

olabilir. Daha yüksek hız sağlamanın yanı sıra, ışık dalgaları da tıp aletleri ve insan bedenleri üzerinde çok az etkiye sahiptir.

**Araçlar:** Karayolu güvenliğini artırmak için araçlar ön ve arka ışıklarla birbiriyle iletişim kurabiliyordu. Ayrıca sokak lambaları ve trafik sinyalleri de mevcut yol durumları hakkında bilgi sağlayabilir.

**Endüstriyel Otomasyon:** Endüstriyel alanlardaki verilerin nakledilmesi gereken her yerde Li-Fi, Endüstriyel Ethernet gibi kayma halkalarını, sürgülü kontakları ve kısa kabloları değiştirebilir. Otomasyon süreçleri için sıkılıkla ihtiyaç duyulan Li-Fi'nin gerçek zamanlı kabiliyeti nedeniyle, ortak endüstriyel Kablosuz LAN standartlarına da bir alternatififtir.<sup>18</sup>

## Bluetooth

Bluetooth, sabit mesafeden ve mobil cihazlardan kısa mesafelerdeki (ISM bandında 2,4 ila 2,485 GHz arasında kısa dalga boylu UHF radyo dalgaları kullanarak) veri alışverişi yapmak ve kişisel alan ağları (PAN) oluşturmak için bir kablosuz teknoloji standardıdır. Hollandalı elektrik mühendisi Jaap Haartsen tarafından icat edilen ve 1994 yılında telekom satıcısı Ericsson için çalışan şirket, orijinal olarak RS-232 veri kablolarına kablosuz bir alternatif olarak tasarlandı.

Bluetooth, telekomünikasyon, bilgi işlem, ağ iletişimini ve tüketici elektroniği alanlarında 30.000'den fazla üye şirkete sahip olan Bluetooth Özel İlgi Grubu (SIG) tarafından yönetilmektedir. IEEE, Bluetooth'u IEEE 802.15.1 olarak standartlaştırdı, ancak artık standardı koruyor. Bluetooth SIG, şartnamenin geliştirilmesini denetler, kalifikasyon programını yönetir ve ticari markaları korur. Bir üreticinin Bluetooth cihazı olarak pazarlamak için Bluetooth SIG standartlarını karşılaması gereklidir. Bireysel yeterlilik cihazlarına lisanslanmış olan teknoloji için bir patent ağı geçerlidir.

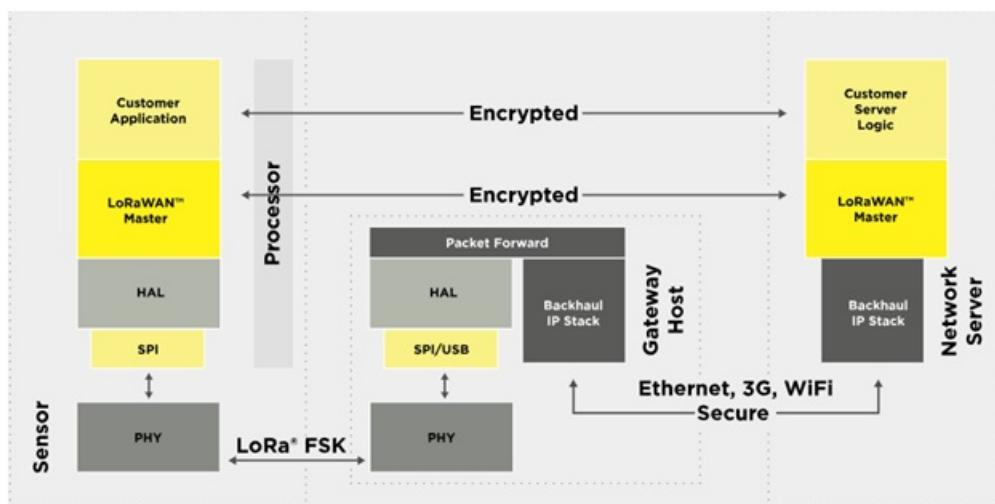
## Uygulanması

Bluetooth, 2402 ve 2480 MHz veya 2400 ve 2483,5 MHz frekanslarında, alt ucta 2 MHz genişliğinde ve üstte 3,5 MHz genişliğinde koruma bantları da-

<sup>18</sup><http://www.wiki-zero.net/index.php?q=aHR0cHM6Ly91bi53aWtpcGVkaWEub3JnL3dpa2kvTGktRmk>

hil olmak üzere çalışır. Bu, küresel olarak lisanssız (ancak düzenlenmemiş) endüstriyel, bilimsel ve tıbbi (ISM) 2,4 GHz kısa menzilli radyo frekans bandıdır. Bluetooth frekans atlamalı yayılma spektrumu adı verilen bir radyo teknolojisini kullanır. Bluetooth, iletilen verileri paketlere böler ve her bir paketi belirlenen 79 Bluetooth kanalından birine iletir. Her kanal 1 MHz'lik bir bant genişliğine sahiptir. Adaptif Frekans-Atlamalı (AFH) etkinken, genellikle saniyede 800 atlama gerçekleştirir. Bluetooth Düşük Enerji, 40 kanalı barındıran 2 MHz aralık kullanır.<sup>19</sup>

## LoRa



LoRa, uzun menzilli iletişim bağlantısını oluşturmak için kullanılan fiziksel katman veya kablosuz modülasyondur. Birçok eski kablosuz sistem, fiziksel katman olarak frekans kaydırmalı anahtarlama (FSK) modülasyonunu kullanır, çünkü düşük güçe ulaşmak için çok verimli bir modülasyondur. LoRa, FSK modülasyonu ile aynı düşük güç karakteristiklerini koruyan, ancak iletişim aralığını önemli ölçüde artıran chirp spektrum modülasyonuna dayanmaktadır. Chirp yayılma spektrumu, elde edilebilecek uzun iletişim mesafeleri ve girişime karşı dayanıklılık nedeniyle onlarca yıldır askeri ve

<sup>19</sup> <http://www.wiki-zero.net/index.php?q=aHR0cHM6Ly90ci53aWtpcGVkaWEub3JnL3dpa2kvQmx1ZXRvb3Ro>

uzay iletişiminde kullanılmıştır, ancak LoRa ticari kullanım için ilk düşük maliyetli uygulamadır.

### **LoRaWan**

LoRaWAN spesifikasyonu, bölgesel, ulusal veya küresel ağlarda 'işletmelerin' internete akü bağlantısını kablosuz olarak bağlamak ve iki yönlü gibi Nesnelerin İnterneti (IoT) gereksinimlerini hedeflemek için tasarlanmış bir düşük güç, geniş alan (LPWA) ağ protokolüdür.<sup>20</sup>

### **ZigBee**

ZigBee, kişisel alan ağları için kullanılan bir IEEE 802 standartına göre küçük, düşük güçlü dijital radyolar kullanılarak oluşturulan yüksek düzeyde iletişim protokollerinin özelleştirilmesidir.

Uygulamalarda, kablosuz ışık anahtarları, ev-içi-ekranlar ile elektrik sayaçları ve diğer tüketicinin ve endüstriyel ekipmanın düşük oranlarda veriyi kısa menzilli kablosuz transferi bulunmaktadır.

ZigBee düşük maliyetli, düşük güçlü wireless mesh ağ standardıdır. Düşük maliyetli teknoloji, yaygın kablosuz kontrolünü ve izleme uygulamalarında dağıtılmmasına olanak sağlar. Düşük güç kullanımı daha küçük pil ile daha uzun ömür sağlar. Mesh ağ, yüksek güvenilirlik ve daha kapsamlı olmayı sağlar.

ZigBee(ISM) endüstriyel, bilimsel ve tıbbi radyo bantlarında çalışır; Avrupa'da 868 Mhz, ABD ve Avustralya 915 Mhz ve dünya çapında 2.4 GHz. Veri İletim hızı 20 ile 900 kilobit/saniye arasında değişir.

ZigBee, ağ katmanı doğal olarak yıldız ve ağaç tipik ağları ve genel mesh ağları destekler. Her ağın yaradılışından sorumlu, parametre kontrol ve basit bakımını yapmak ile görevli bir koordinatör cihazı olması gereklidir. Yıldız ağları içinde koordinatör merkezi düğüm olmalıdır. Ağaçları ve örgütleri(mesh) iki ağ düzeyinde iletişimini uzatmak için ZigBee yönlendirici kullanımına izin verir.

<sup>20</sup><https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf>

ZigBee, düşük-ortam WPANs için fiziksel katman ve IEEE standarı 802.15.4(2003 sürümü) tanımlanan ortam erişim kontrolüne dayanıyor. Özelleştirme dört ana bileşeni ekleyerek standart tamamlamak için devam ediyor:

1. Ağ katmanı
2. Uygulama katmanı
3. ZigBee aygıt nesneleri(ZDos)
4. Özelleştirme lehine toplam entegrasyon için izin veren üretici tanımlı uygulama nesneleri.

Alt yapısını iki üst düzey ağ katmanları eklemenin yanı sıra en önemli gelişme ZDos'un getirilmesidir. Bunlar bir dizi görevden sorumludur; cihaz rollerini tutma, bir ağa katılma isteklerin yönetimi, aygıt keşfi ve güvenlik dahildir. ZigBee, powerline ağ destegine yönelik değildir. Ama en azından ara birim için akıllı ölçümleme ve akıllı cihaz olmasını amaçlar. Çünkü ZigBee düğümleri uykudan aktif moda geçiş süreleri 30 ms veya daha az, gecikme düşük ve cihazlar duyarlı,özellikle Bluetooth uyandırma gecikmeleri ile karşılaşıldığında (tipik olarak 3 saniye civarında), olabilir. Çünkü, ZigBee düğümlerinin çoğu zaman uyuması, ortalama güç tüketiminin az olması uzun pil ömrüne sahip olması ile sonuçlanır.

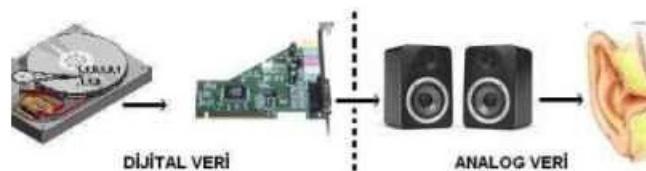
#### 2.3.4 DAC ve ADC

##### Dijital - Analog Dönüştürüçüler (DAC)

Bilgisayar ve dijital sistemler lojik değerler olan 1 ve 0 değerleri ile çalışır. İkililik sistemin basamakları olan bu değerler analog sistemler için anlamlı değildir. Ayrıca dijital değerlerin insanlar için daha anlamlı olan analog değerlere çevrilmesi gereklidir.

Örnek vermek gerekirse bilgisayarımızda sakladığımız MP3 formatındaki ses dosyaları dijital verilerden oluşur. Bu dosyaları dinlemek istediğimizde bilgisayar sistemimize bağlı olan bir ses kartına ve ona bağlı olan bir hoparlör sistemine gereksinim duyuyor. Örnekte ses kartının yaptığı işlem sabit sürücülerimizde saklı olan MP3 dosyasındaki dijital verileri (çok sayıda lojik 1

ve 0 değerleri) hoparlör üzerinde sese çevrilecek olan analog değerlere çevirmesidir. MP3 dosyasından faklı dizilimlerde gelen lojik değerler ses kartında farklı değerlerdeki gerilimlere çevrilecek ve bu gerilim değerleri ise hoparlör üzerinde farklı seslere dönüştürülerek kulağımıza ulaşacaktır.



Şekil 2.15: Dijital verinin ses kartı ile analog veriye çevrimi

### DAC

1 ve 0 gibi dijital bilgileri giriş olarak alan ve çıkışında giriş değerlerindeki değişime göre farklı değerlerde akım veya gerilim üreten devrelere veya entegrelere dijital analog çevriciler ve bu dönüştürme işlemine de dijitalden analoga çevirme işlemi adı verilir. Dijital analog çevriciler kısaca DAC olarak da adlandırılır.

Dijital analog çevriciler giriş olarak birden fazla dijital değeri alabilir. Dijital giriş değeri sayısı dijital analog çevricinin bağlı olduğu dijital devrenin çıkış sayısına eşittir. Bu konuyu daha iyi anlamak için giriş bölümünde verilen ses kartı örneğini tekrar inceleyelim. Ses kartı ana kart üzerinden bilgisayarın data yoluna bağlıdır. Dolayısı ile ses kartı analog sinyale dönüştürmek üzere kullanacağı dijital değerleri sistem data yolundan alır. Bilgisayarımızın data yolu 32 veya 64 bit olabilir. Dolayısı ile ses kartı MP3 dosyasından her seferde 32 veya 64 bitlik dijital veriyi alarak analog veriye çevirecektir. Sonuç olarak bu bit'lerin her birinin bağımsız olarak 1 veya 0 olması dönüşüm sonucunda elde edilecek analog sinyalin akım veya geriliminde değişime yol açar. Böylece biz de hoparlörden farklı tonlarda sesler duyuyoruz. Sonuç olarak girişe uygulanan dijital değerin bit dizilimindeki değişimim çıkıştaki analog sinyalin değerini belirler.

### Kavramlar

Dijital verilerin analog veriye çevrilmesinde analog çıkışın değerinin belirlenmesinde etkili olan bazı esaslar ve kavramlar vardır. Çevrim işleminin daha iyi anlaşılması için bu öncelikle bu kavramlar aşağıda açıklanmıştır.

**LSB**

Dijital değerlerin daha fazla anlam ifade etmesi için çok sayıda bitin bir arada kullanılması gereklidir. Örneğin bir bit ile sadece iki farklı (1 ve 0) durum ifade edilirken iki bit ile dört farklı durum ifade edilebilir (00, 01, 10 ve 11). Dijital devrelerinde daha fazla çıkış durumu ifade etmek için çok sayıda çıkış biti vermesi olası bir durumdur. Ancak bitlerin sayısı çoğalınca dijitalden analoga dönüşüm sırasında bir problem ortaya çıkmaktadır. Çok sayıda giriş biti alan bir DAC bunları çıkışa analog değer olarak aktarırken bitlerin ağırlıklarını (çıkış akım veya gerilimine etki oranını) neye göre belirleyecektir. Bu sorunun çözümü sayı sistemlerinin doğal yapısında çözümlenmiştir. Giriş bitleri peş peşe dizilerek bir ikilik sistemde rakam elde edilirse sağdan sola doğru basamakların değerleri de artmaktadır ve artış oranı sayı sisteminin taban değerine göre üstel şekilde belirlenmektedir. Dolayısı ile girişlerin sıralaması çıkışa etki oranını belirler. Binary(ikili) sayılar yazılırken en sağ-

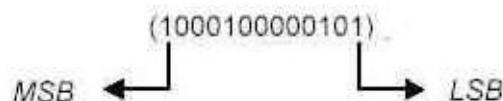
	n basamak	...	4.basamak	3.basamak	2.basamak	1.basamak
Üstel Değer	$2^{n-1}$	...	$2^3$	$2^2$	$2^1$	$2^0$
Ağırlık	$2^{n-1}$	...	8	4	2	1

Sekil 2.16: İkilik sayı sisteminde basamak değerleri

daki basamağa en düşük değerlikli bit LSB (Least Significant Bit-) olarak adlandırılır. Dönüşüm sırasında analog çıkış üzerindeki değer değişimine en az etkili olan dijital değerdir.

**MSB**

Benzer şekilde en soldaki basamağa en yüksek değerlikli bit MSB (Most Significant Bit) adı verilir. Dönüşüm sırasında analog çıkış üzerindeki değer değişimine en fazla etkili olan dijital değerdir.



Sekil 2.17: LSB ve MSB

**Tam Skala (Full scala)**

Dijital analog çeviricilerde giriş olarak kullanılan bit'lerin hepsinin 1 olması durumuna tam skala (Full scale ya da FS) denir. Giriş olarak verilen tüm bit'ler anlamlandırıldığı için çıkış voltajı veya akımı maksimum değerde olacaktır.

### Çözünürlük (Resolution)

Dijital analog çeviricilerin giriş değerlerindeki değişim gösterdiği minimum değişim çözünürlük (Resolution) ya da hassasiyet (sensitivity) denir. Çözünürlük değeri LSB olarak kabul edilen bit'in 1, diğer giriş bit'lerinin 0 olduğu durumda çıkış gerilimine eşittir. Giriş bit'lerinin değeri kademe kademe arttıkça çıkış voltajındaki artış çözünürlük kadar olacaktır. Çözünürlük değeri ne kadar küçükse giriş bitlerindeki değişimme karşılık gelen analog çıkış degerindeki artışlar o kadar az olacak ve hassasiyet artacaktır. Çözünürlük değeri iki değişkenle bağlıdır. Tam skalaaya karşılık gelen analog çıkış değeri ne kadar büyükse çözünürlük de o kadar büyük olur. Ayrıca giriş bitlerinin sayısı ne kadar fazla ise çözünürlük de artar. Burada dikkat edilecek nokta çözünürlüğün artması demek sayısal değerinin azalmasına gelmektedir. Çözünürlük değerinin matematiksel formülü şöyledir.

$$\text{Çözünürlük} = 1 / 2 \text{ Giriş bit sayısı} \quad \text{Çözünürlük Voltajı} = \text{Maksimum Çıkış Voltajı} * 1 / 2 \text{ Giriş bit sayısı}$$

Örnek: Maksimum çıkış voltajı 10V olabilen bir bir DAC devresinde 4 adet dijital giriş varsa çözünürlük nedir?

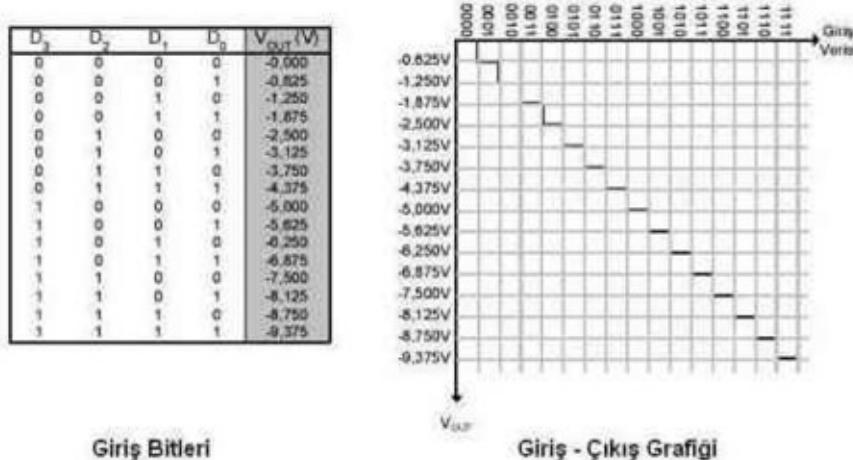
$$\text{Çözünürlük} = 1/2 \text{ Giriş bit sayısı} = 1/2^4 = 1/16 = 0,0625 \text{ başka bir deyişle \% 6,25'dir.} \quad \text{Çözünürlük Voltajı} = \text{Maksimum Çıkış Voltajı} * 1 / 2 \text{ Giriş bit sayısı} = 10 * 0,0625 = 0,625 \text{ V} \quad \text{Örnek deki DAC devresinin çıkışları 0,046 V katları şeklinde değişecektir.}$$

### Giriş-Cıkış İlişkisi

Giriş bit'lerindeki değişim çıkış voltajındaki değişim olarak gözlenmektedir. LSB'den MSB 'ye doğru bit'lerdeki ağırlık değeri artacağından çıkış voltajı üzerindeki etkisi de artacaktır. Birim artış çözünürlük değerine eşittir. Aşağıdaki şekilde yukarıda örnekde verilen DAC devresinin giriş-cıkış ilişkisi gösterilmiştir.

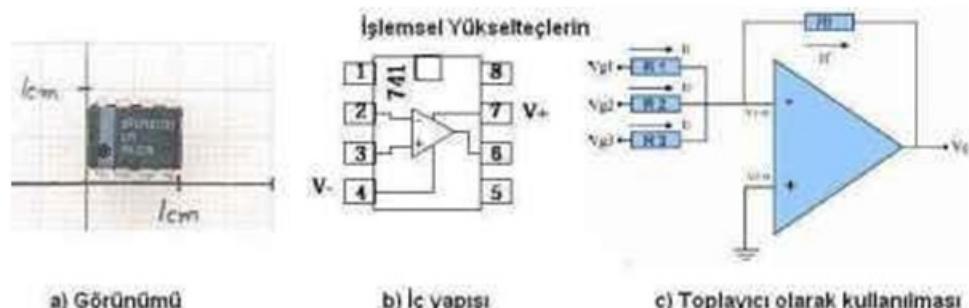
### Çalışma Prensibi

Dijital değerlerin analog değerlere dönüştürülmesinde kullanılan temel ele-



Şekil 2.18: Giriş-çıkış ilişkisi

man işlemel yükselteçlerdir. Dijital analog çevircilerin çalışma prensiblerini anlayabilmek için işlemel yükselteçlerin çalışması hakkında bilgi sahibi olmak gereklidir. İşlemel yükselteçler, girişine uygulanan gerilim değerini yine giriş ve çıkışına bağlanan dirençlerle belirlenen bir oranla çıkışa aktaran devre elemanıdır. Giriş değerinin çıkışa etki oranının belirlenebilmesi sayesinde girişi oluşturan dijital değerlerin çıkışa aktarılma oranı belirlenebilmektedir. İşlemel yükselteçler elektronik alanında çok farklı amaçlarla kullanılabilmektedir. DAC devrelerinde toplayıcı olarak kullanılabilme özellikleinden faydalananır. Giriş bitlerinin çıkışa etki oranı dirençler ile belirlenerek yükseltilmiş bir analog çıkış elde edilebilir. Şekil 2.19'da verilen toplayıcı dev-



Şekil 2.19: İşlemel yükselteç

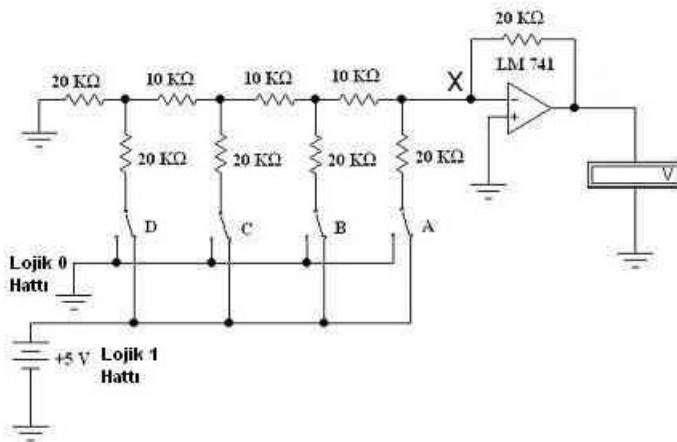
resinde Vg1, Vg2 ve Vg3 gerilimleri önlerine konulan dirençlerin büyüklüğü

ile ters orantılı olarak çıkışa aktarılırlar. Ayrıca giriş gerilimi çıkışa aktarılırken R1,R2 ve R3 dirençlerinin eş değeri ile Rf direncinin oranına göre yükseltilerek aktarılır.

### R-2R Merdiven Tip DAC

Bu devrede dirençlerin değerlerinin R-2R olarak sıralanması ve çıkış dalga şeklinin merdiven basamağı şeklinde artması sebebiyle bu tip çeviriciler R-2R merdiven tipi D/A çeviriçi adını alır. Şekil 2.20'de verilen R-2R merdiven tip DAC devresinde X ile gösterilmiş düğüme A,B,C ve D ile gösterilmiş dijital girişlerin etkileri farklıdır. Önünde çok direnç değeri olan dijital giriş X noktasına daha az akım ulaştıracaktır ve bunun sonucu olarak da çıkıştaki etkisi daha az olacaktır.

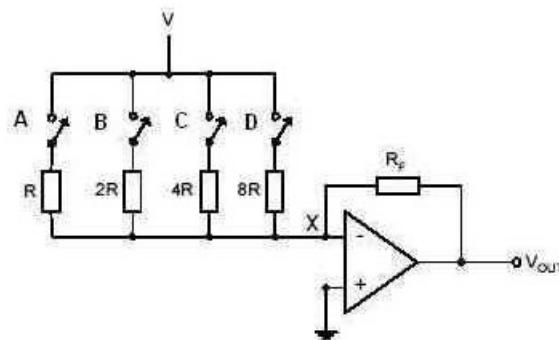
D en değeriksiz bit (LSB) olup devrenin çözünürlüğünü belirler. Referans geriliminin 16'da 1'i kadar çıkış etkiler. Her bir basamak değeri D'nin etkilediği değer kadar artar. A ise en değerlikli bit (MSB) olup çıkış tam skala değerinin yarısı olarak etki eder.



Şekil 2.20: R-2R merdiven tip DAC

### Ağırlık Dirençli Tip DAC

Bu devrede dirençlerin değerlerinin ağırlık dirençli olarak sıralanması dijital girişlerin önüne koyulan dirençlerin, dijital girişin çıkışa yansıtılma oranını ile ters orantılı bir şekilde belirlenmesinden kaynaklanır. Dirençler arasındaki oran belirlenirken 2'nin katları şeklinde gidilmesi gereklidir. Şekil 2.21 de



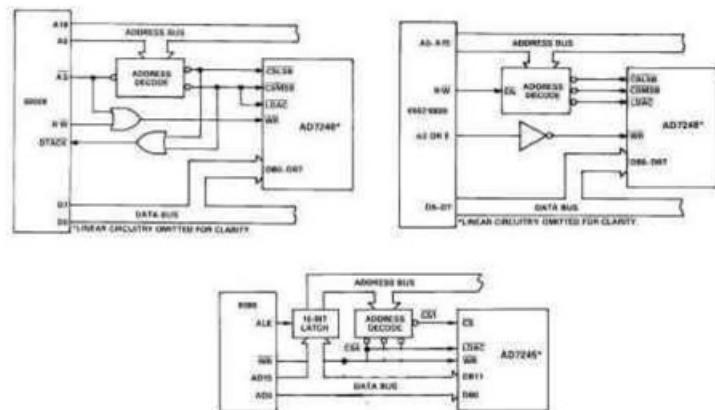
Şekil 2.21: Ağırlık dirençli tip DAC

verilen ağırlık dirençli tip DAC X ile gösterilmiş düğüme A,B,C ve D ile gösterilmiş dijital girişlerin etkileri farklıdır. Önünde yüksek direnç değeri olan dijital giriş X noktasına daha az akım ulaştıracaktır ve bunun sonucu olarak da çıkıştaki etkisi daha az olacaktır.

D en değeriksiz bit (LSB) olup devrenin çözünürlüğünü belirler. Referans geriliminin 16'da 1'i kadar çıkış etkiler. Her bir basamak değeri D'nin etkilediği değer kadar artar. A ise en değerlikli bit (MSB) olup çıkışa tam skala değerinin yarısı olarak etki eder.

### Mikroişlemci Uyumlu DAC'ler

Mikroişlemciler dijital veri ile çalışan elemanlardır. Mikroişlemcilerin çalışması sırasında data yolu adı verilen bir gurup iletken hat üzerinde sürekli dijital veri akışı olmaktadır. Bu veriyi mikroişlemci dışında bir analog devrede değerlendirmek istersek bir dijital analog dönüştürücü kullanmalıdır. Ancak yukarıda anlatılan devreler ve entegrelerin mikroişlemciler ile uyumlu çalışması mümkün değildir. Mikroişlemciler ürettiği kontrol sinyalleri sayesinde çevre birimlerinin çalışmalarını idare edebilirler. Bu nedenle mikroişlemciler ile uyumlu olan dijital analog dönüştürücülerin mikroişlemciden gelen data hatlarının yanında kontrol hatalarından da giriş kabul etmesi gerekmektedir. Bu amaçla üretilmiş dijital analog dönüştürücülere mikroişlemci uyumlu DAC adı verilir. Mikroişlemcilerin data ve kontrol hatlarının sayısı değişkendir. Bu nedenle mikroişlemci uyumlu DAC entegreleri belirli bir mikroişlemci ile uyumlu olacak şekilde üretilir. Bu uyumun anlamı çalışma hızı, data hattı sayısı denkliği ve kontrol sinyalleri ile DAC entegresinin kontrol edilebilmesi olarak özetlenebilir. Şekil 2.22'de 68808, 6502 ve 8086 mikroişlemcileri uyumlu DAC'leri ve bağlantıları gösterilmiştir.



Şekil 2.22: Mikroişlemci Uyumlu DAC'ler

### Analog - Dijital Dönüşürcüler (ADC)

Bilgisayar ve dijital sistemler lojik değerler olan 1 ve 0 ile çalışırlar. İkilik sistemin basamakları olan bu değerler analog sistemler için anlamlı değildir. Analog devreler geniş bir gerilim bandında çıkış verebilirler. Bu konuyu şu örnekle açıklayalım. Elektronik terazi veya termometre gibi cihazlar ortamdaki fiziksel değişikliği sensörleri ile algılar. Sensör bulunduran bir analog devre ortamındaki ölçümekte değişime çıkış gerilimindeki veya akımındaki değer değişimi ile tepki verir. Ancak bu değerler bir ölçü aleti kullanmıyorsak bizler için anlamlı değildir. Ölçülen sıcaklığın veya ağırlığın insanlar için anlamlı olan sayı sistemleri ile ifade edilmesi gereklidir. Örneğin 60 kg veya 35 derece gibi. Bu noktada da devreye giren A-D çevriciler sayesinde sensörlerden gelen analog sinyalleri önce ikilik sayı sisteminin rakamları ile ifade edilen dijital veriye çevrilir. Bu aşamadan sonra dijital devreler kodlayıcı ve display devrelerden geçerek insanlar için daha anlamlı olan onluk sayı sisteme çevrilebilir. Bu öğrenme faaliyeti ile bir analog devrenin çıkışını dijital bir devreye giriş olarak bağlayabilmek için kullanılan A-D çevricileri öğreneceksiniz.

### ADC

Basınç, sıcaklık veya ışık şiddeti gibi ortam değişikliklerini ölçen sensörler akım veya gerilim büyülüklerini çıkışlarında genellikle analog olarak verirler. Bilgisayar sistemleri ve diğer dijital devreler ise bu değerleri kullanamazlar.

Akım ve gerilim gibi analog sinyallerin dijital sinyallere dönüştürme işlemine yapan devrelere de analog-dijital çevirici kısaca ADC denir.

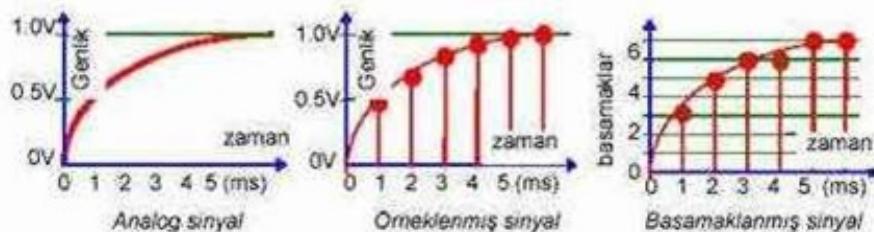
### Çalışma prensibi

Bir analog sinyal dijital sinyale çevrilirken belirlenen zaman dilimlerinde örnekleme yapılmalıdır. Bir referans gerilimi baz alınarak örneklenen her giriş gerilimine karşılık gelen bir dijital değer belirlenir. Analog işaretlerin dijital dönüşürtlmesi, örnekleme, basamaklama ve kodlama üzere üç aşamada yapılır. Analog sinyaller zaman ve genlik olarak sürekli sinyallerdir.



Şekil 2.23: ADC'lerin çalışma prensibi

Bunları dijitalleştirmek için önce belli aralıklarda örnekler alınması gereklidir. Örnekleme sıklığı uygun seçilmesi gereklidir. Alınan örnekler genlikleri her



Şekil 2.24: Analog sinyal örneklemesi ve basamaklanması

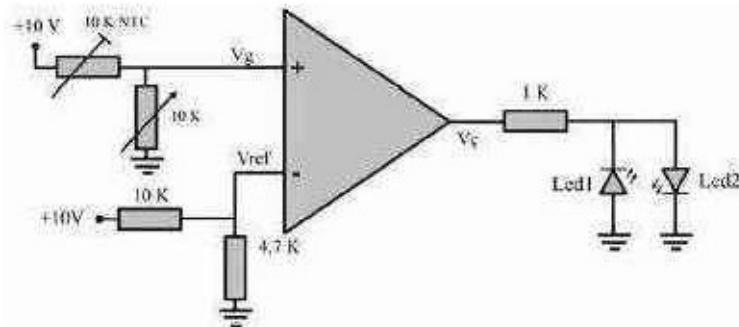
hangi bir değerde olabilir. Buna karşılık işaretin dijitalé çevrilebilmesi için kullanılacak seviye sayısının sınırlı olması gereklidir. Bu sayı, her bir örnek için kullanılacak kod uzunluğu ya da bit sayısı tarafından belirlenir.

Örnek olarak 8-bit'lik bir kodlama yapılacaksa  $2^8 = 256$  seviye, 3-bit'lik bir kodlama yapılacaksa sadece  $2^3 = 8$  seviye kullanılabilir. Seviye veya basamak sayısının artması dönüşüm kalitesini belirler. Daha iyi kalite için daha çok bit ve daha çok basamak kullanmak gereklidir.

Örneklemeye ile çevirmede karşılaşılan sorun belirli bir analog değer aralığına bir dijital değerin karşılık gelmesidir. Örneklem açıklaymak gerekirse 1.5 volt için 111 dijital çıkışını veren bir çevirici 1.7 volt için de aynı çıkış verebilir.

### Paralel Tip ADC

Analog büyülüklerin sayısal işaretlere dönüştürülmesinde kullanılan en kolay ve hızlı çevirici paralel tip ADC çeviricidir. Paralel tip ADC'lerde opampli karşılaştırıcı kullanılmaktadır. Opampli karşılaştırıcı devresinde opamp geri beslemesiz olarak kullanılır ve opamp girişlerinden biri referans olarak kullanılır. Diğer girişin referanstan büyük ya da küçük olmasına göre opamp çıkışı pozitif veya negatif bir değer alır. Opamlarda evren(-) giriş referans olarak kullanıldığında diğer girişe uygulanan gerilim referans gerilimden büyük olursa çıkış gerilimi pozitif olacaktır. Şekil 2.25'te verilen devre ile opampli

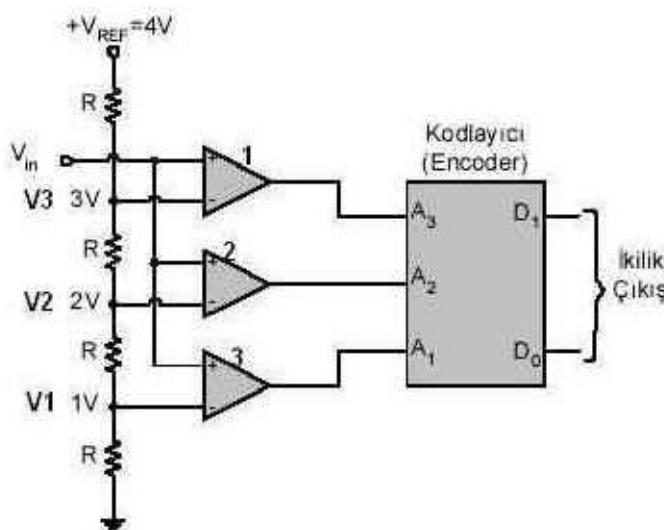


Şekil 2.25: Ortamın ısı değişikliğini opampli karşılaştırma yöntemi ile kontrol edilmesi

karşılaştırmanın nasıl yapıldığını daha iyi anlayalım. Devre öncelikle oda sıcaklığını referans alabilmek için 10k pot ile oynayarak Led1'in yanık Led2'nin sönübü konumda olamasını sağlayalım. Led1'in yanması için opamp çıkışının negatif olması gereklidir.  $V_g$  ile verilen giriş geriliminin  $V_{ref}$  ile verilen referans geriliminden küçük olması ile bu olay mümkün olacaktır. NTC ısı etkisine tutulursa direnci düşecektir (odanın ısınması durumunda). NTC'nin direncinin düşmesi sonucunda  $V_g$ 'nin değeri yükselecek ve  $V_{ref}$  ile belirtilen referans gerilimi aştığı anda opamp çıkışı pozitif olacaktır Led1 söñüp Led2 yanacaktır.

Şekil 2.25'te verilen devrenin çalışma mantığını kullanan çok sayıda karşılaştırıcı, opamp paralel olarak ve her birinde basamaklı olarak artan referans gerilimi kullanıldığında Paralel tip ADC elde edilir. Paralel tip ADC'de çevri-

lecek olan analog sinyal tüm karşılaştırıcı girişlerine aynı anda paralel olarak uygulanır. Karşılaştırıcıların diğer girişlerine ise referans gerilimi uygulanır. Şekil 2.4'te verilen paralel tip ADC de uygulanan referans gerilimini 4 V olduğunu düşünürsek V3 noktasında 3 V, V2 noktasında 2 V ve V1 noktasında 1 V bulunmaktadır. Uygulanan analog gerilime bağlı olarak karşılaştırıcıların çıkışları lojik 0 ya da 1 (pozitif olması lojik 1 durumudur) durumunu alır. Bu çıkışlar bir kodlayıcı devre ile ikili sayı sistemine çevrilerek dijital çıkışlar elde edilir. Şekil 2.26'da verilen devrede  $V_{in}$  gerilimi 2.25 volt olursa



Şekil 2.26: Paralel tip ADC

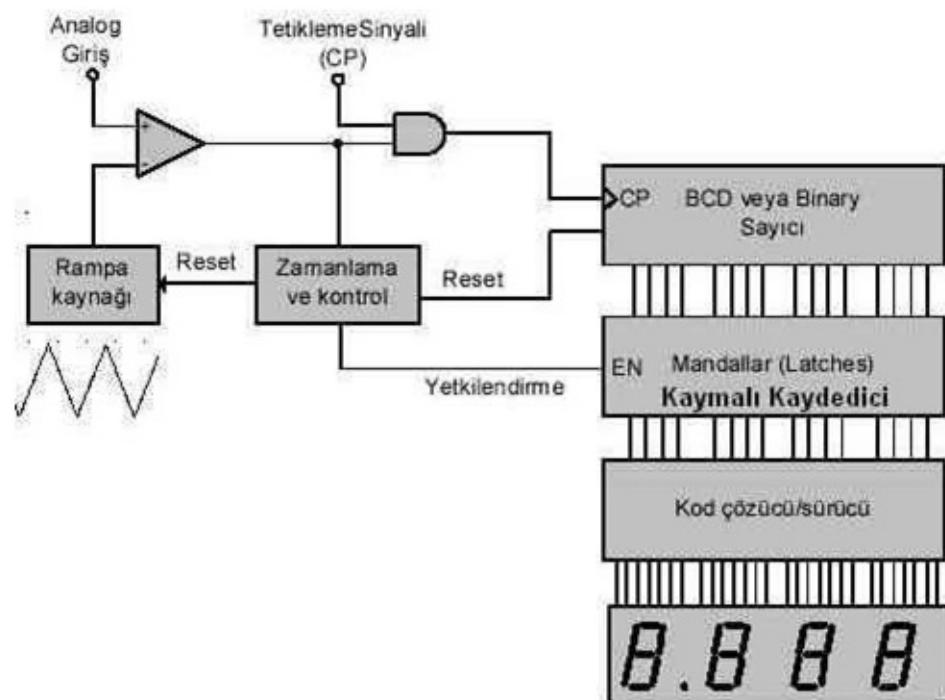
2 ve 3 numaralı opamplara uygulanan referans geriliminde büyük olacağından bu opampların çıkışı pozitif olacaktır. 2.25 volt gerilim 3 numaralı opampın çıkışını pozitif yapmaya yetmeyecektir. Sırası ile A3, A2, A1(011) lojik değerlerini alacaktır. Bu değerde kodlandığında (10) çıkışı elde edilecektir.

### Sayma Metotlu ADC

A-D çevirimde kullanılan bir diğer yöntem lineer rampa kaynağı, karşılaştırıcı ve sayıcılardan oluşmuş Sayma metotlu ADC çevircilerdir. Lineer rampa kaynağı, değişmeyen eğimli bir referans voltajının sağlanması için kullanılır.

Çevirimin başlangıcında sayıcı reset, rampa kaynağı çıkışı 0V yapılır. Karşılaştırıcının + girişine uygulanan analog giriş gerilimi, girişinden büyük olduğundan çıkış yükseye çekilecektir. Bu durumda rampa kaynağı tarafından

rampa üretilmeye başlanacak sonra VE kapısının çıkışında tetikleme sinyali görüleceğinden sayıcı sayma işlemine başlayacaktır. Bu işlem rampa kaynağı tarafında üretilen rampa geriliminin, analog giriş geriliminden büyük olmasına kadar devam edecektir. Böylece karşılaştıracı çıkış alçğa çekilecek, VE kapısının çıkışı lojik-0 olacak ve tetikleme sinyali gitmeyen sayıcı sayma işlemini bitirecektir. Kontrol devresi tarafından yetkilenen mandallar sayıcı verilerini saklayacaktır.<sup>21 22</sup>



Şekil 2.27: Sayma metotlu ADC

## 2.4 Enerji Kaynakları

Günümüzde belirli teknolojilerle tasarlanmış enerji kaynaklarını kullanarak bir gömülü sistem tasarlama makamları, kablolamadan önemli miktarda özgürlük sağlar. Kablosuz bağlantılardan güç kullanmak sistemi kablo fazlalığından kur-

<sup>21</sup>[www.elektrik.gen.tr](http://www.elektrik.gen.tr)

<sup>22</sup><http://www.wiki-zero.net/index.php?q=aHR0cHM6Ly91bi53aWtpcGVkaWEub3JnL3dpa2kvRGlnaXRhbC10by1hb>

tarır ve ekipmanı entegre etmek için muazzam bir esneklik sağlar. Ancak sistem tasarıımı için bazı hususlarda dikkat gerektirir. Böyle bir sistemi enerji hasatlama kaynağından çalıştırmak için güç maliyeti dikkatle düşünülmeli dir. Ayrıca girişlerin ve veri aktarımının, güneş panelleri gibi ölçülenebilir enerji kaynaklarının ve akü yedeğinin kontrolü ile bu durum yönetilebilir. Diğer bir konu ise, bataryadan sisteme güç sağlamaktır. Örneğin; bir Arduino kartının ihtiyaç duyduğu 7 ila 12 V'luk yüksek gerilimleri ve bir enerji hastalama kaynağında bulunan düşük voltaj ve akımın üstesinden gelmek için bazı gelişmiş güç yönetim cihazlarını kullanmak gerekebilir.

### Enerji Bütçesi

Tasarımın güç maliyeti tamamen uygulamaya bağlıdır. Örnek vermek gereklirse Arduino'da on dört pimin her biri 5 V'da 40 mA DC akımı destekler ve kablosuz bağlantı eklemeden geniş bir sensör yelpazesи için 2,8 W üzerinde potansiyel güç tüketimi sağlar. Bu, bir enerji hasatlama kaynağının desteklenmesi için zordur. Bununla birlikte, tüm hatlar her zaman aktif olmaya caktır. Aslında pek çoğu kullanılmayacaktır. Bu nedenle, kartın kullanım durumu, en yüksek güç gereksinimlerini belirlemek ve bunu enerji kaynağına ve pille eşleştirmek için hayatı önem taşır.

Güç gereksinimi, bazı yöntemler ile daha da azaltılabilir. Belki de saniyede bir kez okunan sensörler ve 5 saniyede bir aktif olan kablosuz bağlantı ile mevcut gereksinimler önemli ölçüde azalır ve bir enerji hasatlama kaynağı tarafından daha kolay bir şekilde karşılanabilir.

#### 2.4.1 Batarya

Batarya, iki ya da daha çok pil ya da akümülatörün iletkenler aracılığıyla birbirlerine bağlanmasıyla oluşan üreteçler grubudur. Üreteçler, seri, paralel ve karışık olmak üzere başlica üç biçimde birbirlerine bağlanır. Her üreticin eksik kutbunun, bir sonrakinin artı kutbuna, pozitif kutbun ise ötekinin negatif kutbuna bağlanması biçimindeki seri bağlamda, bataryanın elektromotor kuvveti, her üreticin elektromotor kuvvetlerinin toplamına eşittir.

Üreteçlerin benzer kutuplarından birbirlerine bağlanması biçimindeki paralel bağlama türü ise, elektromotor kuvvetleri aynı olan üreteçlerden oluşan bataryalarda uygulanır. Paralel bağlanan elemanlardan birinin elektromotor kuvveti ötekilerinininkine eşdeğer değilse, bazı elemanlar alıcı durumuna ge-

çer. Bu özellikten, akümülatörlerin elektrik üreteçleriyle (dinamo) doldurulmasında yararlanılır. Paralel bağlanmış baryaların elektromotor kuvveti, her üretecin elektromotor kuvvetine eşittir. Buna karşılık sistemin gücü, üreteçlerin güçlerinin toplamından oluşur.

Karışık bağlama ise, seri ve paralel bağlama türlerinin birlikte uygulanması biçimindedir. Karışık bağlamalı baryalardan, özel amaçlı işlemlerde yararlanılır. Baryalar, pil ve akümülatör gibi üreteçlerin yan sıra, sığaçların bağlanmasıyla oluşturulabilir. Bu sistemde, sığaç armatürlerinin seri ya da paralel bağlanmasıdan ve topraklamadan yararlanılır.

### **EnerChip(BC3105) Entegre Pili**



Şekil 2.28: EnerChip(BC3105) Entegre Pili

Gömülü sistemlerde kullanılan ve entegre pillere farklı bir boyut kazandıran EnerChip(BC3105) çipi, enerji depolama için yenilikçi bir yol sağlar. Bu yenilikçi çipin hızlı şarj süresi oldukça avantajlıdır.  $5 \mu Ah$ 'lık bir katı hal pili entegre eder ve geniş bir besleme aralığında şarj edilebilir. Pili entegre ederek, güç köprülemesi ve / veya ikincil güç gerektiren sistemler için yedek enerji depolama ve güç yönetimi sağlar. Tek bir EnerChip CC, paralel bağlanmış on adete kadar EnERChip'i şarj edebilir. Normal çalışma sırasında EnerChip CC, 2,5 V ile 5,5 V arasında çalışan bir dahili şarj pompası kullanarak kontrolü bir volajla kendini şarj eder. ENABLE pimi, mevcut tüketimi en aza indirmek için harici bir kontrol hattı kullanarak şarj pompasının aktivasyonunu ve devre dışı bırakılmasını sağlar. Güçün yönetimi, bir gömülü sistem için enerji hasatının kullanılmasında önemli bir anahtardır.. EnerChip'in güç yönetimi cihazı, yüksek verimli enerji dönüşümü için "MPPT"(Maximum Peak Power Tracking) algoritmalarını ve sistem yük empedansı eşleşmesini uygu-

lamak için bir enerji hasadı dönüştürücüsünü kullanır. Ayrıca, AVR mikro denetleyiciye kartta bir iletişim arayüzü sağlar ve gelen enerji ve depolama enerji seviyeleri için enerji durumu göstergeleri sağlar.

#### 2.4.2 Enerji Hasatlama

##### Supercap

Bir süper kondansatör (SC) (aynı zamanda bir supercap, ultracapacitor veya Goldcap olarak da adlandırılır), kapasitans değerleri, elektrolitik kondansatörler ve şarj edilebilir piller arasındaki boşluğu kapatılan diğer kondansatörlerden çok daha yüksek olan yüksek kapasiteli bir kondansatördür. Genellikle, birim hacim veya kütle başına elektrolitik kondansatörlere göre 10 ila 100 kat daha fazla enerji depolaralar, şarjları pillere göre çok daha hızlı kabul edebilir ve verebilirler. Şarj edilebilir pillere göre daha fazla şarj ve deşarj döngüsüne tahammül edebilirler.

Süper kondansatörler uzun süreli kompakt enerji depolama yerine hızlı şarj/deşarj döngüsüne ihtiyaç duyan uygulamalarda kullanılır: Otomobiller, otobüsler, trenler, vinçler ve asansörler, rejeneratif frenleme, kısa süreli enerji depolama veya patlama modu güç dağıtımına için kullanılırlar.

Sıradan kondansatörlerden farklı olarak, süper kondansatörler geleneksel katı dielektrik kullanmazlar, bununla birlikte, her ikisi de kondansatörün toplam kapasitansına katkıda bulunan elektrostatik çift katmanlı kapasitans ve elektrokimyasal pseudokapasitans kullanırlar. Süper kondansatörler, asimetrik elektrotlarla veya simetrik elektrotlar için, tasarım sırasında uygulanan bir potansiyel tasarımla polarize edilir.

##### Güneş Enerjisi Hasadı

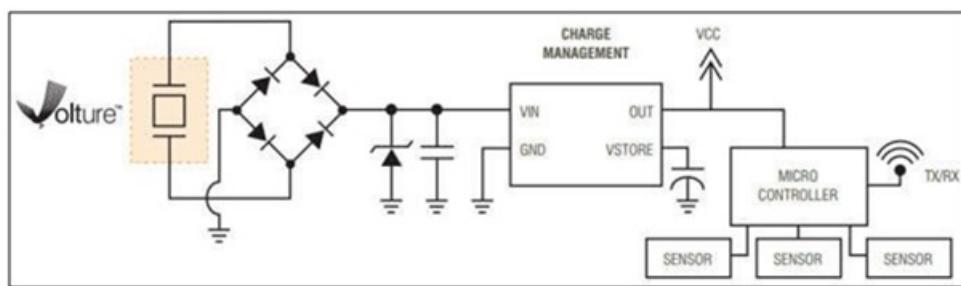
Küçük güneş pilleri, uydular, taşınabilir güç kaynakları, sokak lambaları, oyuncaklar, hesap makineleri ve daha fazlası gibi endüstriyel ve tüketici uygulamalarında kullanılır. Bunlar, ışığı elektrik enerjisine dönüştüren küçük bir fotovoltaik hücre kullanır. İç mekan uygulamaları için, ışık genellikle çok güçlü değildir ve tipik yoğunluk yaklaşık  $10\mu W/cm^2$ 'dir.

Bir iç mekan enerji toplama sisteminden gelen güç, güneş modülünün büyük-

lügüne ve ışığın yoğunluğuna veya spektral kompozisyonuna bağlıdır. Işığın aralıklı doğası nedeniyle, güneş pilleri genellikle uygulama için istikrarlı bir tedarik sağlamak için bir pil veya süperkondansatör şarj etmek için kullanılır.

### Kinetik Enerji Hasadı

Piezoelektrik dönüştürücüler, kinetik enerjiye titreşimlerden, hareketlerden ve ısı dalgalarından veya uçak kanatlarındaki ve diğer kaynaklardan gelen motor yatak gürültüsü gibi seslerden maruz kaldıklarında elektrik üretirler. Dönüştürücü, kinetik enerjiyi titreşimlerden bir AC çıkış voltajına dönüştürür, daha sonra ince bir film pilinde veya bir süper kondansatörde düzelttilir, düzenlenir ve depolanır.



Şekil 2.29: Midé Volture Piezoelektrik Enerji Hasat Devresi. (Görüntü kaynağı: Mouser)

Potansiyel kinetik enerji kaynakları, insanlar tarafından üretilen hareketi, akustik gürültüyü ve düşük frekanslı titreşimleri içerir. Bazı pratik örnekler:

**Bir pilsiz uzaktan kumanda ünitesi:** Güç, düğmeye basıldığında kullanılan güçten hasatlanır. Hasatlanan enerji, düşük güç devresine güç vermek ve kızılılolesi veya kablosuz radyo sinyalini iletmek için yeterlidir.

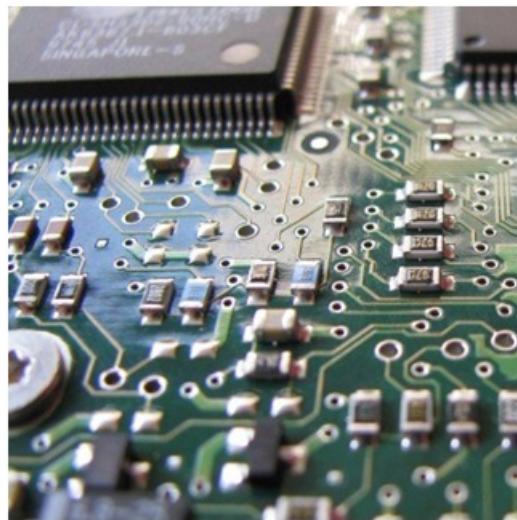
**Otomobil lastikleri için basınç sensörleri:** Piezoelektrik enerji toplama sensörleri, basıncı izledikleri ve sürücünün görmesi için bilgileri gösterge tablosuna ilettikleri otomobil lastığının içine yerleştirilir.

**Piezoelektrik yer dösemeleri:** Yerde yürüyen insanlardan gelen kinetik enerji; görüntü sistemleri, acil durum aydınlatması, gişe kapama kapıları ve daha fazlası gibi temel hizmetler için kullanılabilen elektrik enerjisine dönüştürülür.

## 2.5 Programlama

### 2.5.1 Gömülü Yazılım Geliştirme

Gömülü yazılım geliştirme, çoğu durumda, fiziksel dünya ile - donanım platformu ile yakın etkileşim gerektirir. "Çoğu durumda" diyoruz çünkü bireylerin yalnızca sistem için uygulama katmanı yazılımı üzerinde çalışmasını gerektiren çok büyük gömülü sistemler var. Bu uygulama geliştiricileri genellikle donanım ile herhangi bir etkileşime sahip değildir. Düzgün tasarlandığında, donanım aygıtı sürücülerini gerçek donanımdan soyutluyorlar, böylece uygulama düzeyinde bir geliştirici yazma yazılımı, bir dizinin ekrana nasıl çıkacağını bilmez.



Şekil 2.30: prog

### Gömülü Sistem Programcısı Olmak

#### Donanım Bilgisi

Gömülü yazılım geliştiricisi, tümleşik devreler, panolar ve otobüsler ile katı gömülü yazılımları (ayrıca bellenim olarak da bilinir) yazmak için kullanılan bağlı aygıtları yakından tanımalıdır. Gömülü geliştiriciler, şemalara dalmaktan, bir osiloskop probunu almaktan korkar ve neler olup bittigini öğrenmek için devre etrafında dolaşmaya başlar.

#### Verimli Kod

Gömülü sistemler genellikle, sistemin performans gereksinimlerini karşılayan

en az güçlü ve en maliyet etkin işlemci ile tasarlandığından, gömülü yazılım geliştiricileri her kod satırı sırasına sahip olmalıdır. Verimli kod yazabilme yeteneği, bir firmware geliştiricisi olarak sahip olmak için mükemmel bir kilitedir.

### **Güçlü Kod**

Gömülü sistemlerin çoğu durumda yıllarca çalışacağı yönünde beklentiler vardır. Bu, bir PC veya Mac için yazılmış yazılım uygulamaları için tipik bir gereklilik değildir. Şimdi istisnalar var. Ancak, ögle yemeğinizi uygun bir süre için ısıtmak için mikrodalga fırınımızı çıkarmanız gerekiyorsa, muhtemelen bu ürünü bir şirketten satın aldığınız son sefer olacaktır.

### **Minimum Kaynak Kullanımı**

Daha sağlam bir sistem yaratmanın aynı hatları boyunca, gömülü yazılım ile diğer yazılım türleri arasındaki diğer büyük bir farklılaştırıcı, kaynak kısıtlamalarıdır. Ürün yazılımı yazma kuralları, bir PC için yazılım yazma kurallarından farklıdır. Örneğin, bellek ayırma. Modern bir PC için bir uygulama, neredeyse sınırsız kaynaklara erişebileceğini kabul edebilir. Ancak gömülü bir sistemde, önceden plan yapmazsanız ve yazılımı doğru şekilde tasarlarsanız, bellek yetersiz kalır.

Gömülü bir yazılım geliştiricisinin kaynakları bellekten işlem gücüne yönetebilmesi gereklidir, böylece sistem belirtme göre çalışır ve böylece hatalar meydana gelmez. Örneğin, standart dinamik bellek ayırma işlevlerini kullanmak parçalanmaya neden olabilir ve sonunda sistem çalışmaz. Bu, gelen verileri saklamak için yeriniz olmadığı için yeniden başlatma gereklidir.

Sıklıkla, gömülü yazılımlarda, bir geliştirici başlangıçta sistem tarafından ihtiyaç duyulan tüm belleği tahsis edecektir. Bu, her zaman yapılamasa bile, dinamik bellek ayırma kullanmaktan daha güvenlidir.

### **Periferik Arayüzler**

En düşük seviyede, firmware çok özeldir, çünkü her bir bileşen veya devrenin, kendi faaliyetini gerçekleştirmek için kendi faaliyetine ve ayrıca kendi faaliyetini gerçekleştirme yöntemi vardır. Gömülü geliştiriciler, sistemindeki aygıtların tam denetimini sağlamak için farklı aygıtlarla veya çevre birimleriyle nasıl iletişim kuracaklarını bilmelidir. Dış çevre birimlerinden gelen uyarınlara tepki vermek, gömülü yazılım geliştirmenin büyük bir parçasıdır.

Örneğin, bir mikrodalga fırında, firmware harici analogdan dijital döntüşürücüde 8-bit bir kayıt okuyarak verileri bir sıcaklık sensöründen alabilir; Başka bir sistemde, veriler, tek bir tel üzerinden harici sensör devresine bağlanan bir seri veriyolu kontrol edilerek çıkarılabilir.

### **Yeniden Kullanılabilir Yazılım**

Kod portatifliği veya kod yeniden kullanımı — yazılımın donanım platformundan donanım platformuna taşınabilmesi için yazılım yazma - yeni projelere geçiş için çok yararlıdır. Bu her zaman yapılamaz. Dolayısıyla, bir sonraki projenizde daha önce bir sürücü geliştirdiğiniz bir LCD kullanılıyorsa, eski kodun içine düşebilir ve programda zaman kazanabilirsiniz.

### **Geliştirme Araçları**

Gömülü bir geliştirici olarak kariyeriniz boyunca kullanacağınız araçlar, şirketten şirketeye ve genellikle projeden projeye değişecektir. Bu, kariyerinizde devam ederken yeni araçları öğrenmeniz gerekeceği anlamına gelir. Tipik olarak, bu araçlar PC yazılımı geliştirmede kullanılan kadar güçlü veya kullanımı kolay değildir.

Karşılaşabileceğiniz hata ayıklama araçları, basit bir LED'den tam üflemeli bir in-circuit emülatörüne (ICE) kadar değişimdir. Bu, ürün yazılımı geliştiricisi ve kodunuzu hata ayıklama sorumlusu olarak çok becerikli olmanızı ve hata ayıklama ortamının bulunmadığı durumlarda arayabileceğiniz bir teknikler çantasına sahip olmanızı gerektirir.

### **Gömülü Programlama Dilleri**

Çoğu gömülü sistemdeki birkaç sabitten biri, C programlama dilinin kullanılmasıdır. Diğerlerinden daha fazla, C gömülü programcının dili haline geldi. Bu her zaman böyle olmayı ve sonsuza dek olmaya devam etmeyecektir. Ancak, şu anda, C gömülü dünyada bir standart için en yakın şeydir.

Başarılı yazılım geliştirme, belirli bir proje için en iyi dili seçmeye bağlı olduğundan, bir dilin hem 8 bit hem de 64 bit işlemciler için uygun olduğunu kanıtlaması şaşırtıcıdır; bayt, kilobayt ve megabayt bellek içeren sistemlerde; ve bir ya da bir düzine ya da daha fazla kişiye değişen ekipleri için.

Yine de bu, C'nin geliştirdiği projelerin tam aralığıdır.

C programlama dilinin birçok avantajı vardır. Öğrenmek için küçük ve oldukça basittir, derleyiciler bugün kullanılmakta olan hemen hemen her işlemci için mevcuttur ve çok deneyimli C programcıları vardır. Ayrıca, C programcılarının belirli bir işlemci mimarisinin detaylarından ziyade algoritmalar ve uygulamalara konsantre olmasına izin veren, işlemci bağımsızlığı yararına sahiptir. Bununla birlikte, bu avantajların birçoğu diğer üst düzey diller için de geçerlidir.

C nispeten "düşük seviyeli" bir dildir. Bu karakterizasyon pejorative değil; Bu sadece C'nin, çoğu bilgisayarın yaptığı aynı tür nesnelerle uğraştığı anlamına gelir. Bunlar, gerçek makineler tarafından uygulanan aritmetik ve mantıksal işlemler ile birleştirilebilir ve taşınabilir. Birkaç popüler üst düzey dil hemen hemen tüm işlemciler için kompakt, verimli kod üretiminde C ile rekabet edebilir. Ve bunlardan sadece C, programcıların altta yatan donanımla bu kadar kolay etkileşimde bulunmalarını sağlıyor.

Tabii ki, C yerleşik programcılar tarafından kullanılan tek dil değildir. En az dört başka dil - montaj, C ++, Forth ve Ada - daha ayrıntılı olarak bahsedilmeye değer.

İlk zamanlarda, gömülü yazılım sadece hedef işlemcinin montaj dilinde yazılmıştır. Bu, programcılara işlemcinin ve diğer donanımın tam kontrolünü sağladı, ancak bir fiyata. Montaj dilleri çok az dezavantajlara sahiptir, bunlardan en önemlisi daha yüksek yazılım geliştirme maliyetleri ve kod taşınabilirliği olmamasıdır. Ek olarak, yetenekli montaj programcılarını bulmak son yıllarda daha zor hale gelmiştir. Meclis şimdi, yüksek seviye dilde, genellikle yalnızca başlangıç sistemi kodu veya son derece verimli veya ultra kompakt olması gereken ya da başka herhangi bir şekilde yazılmaması gereken küçük kod parçaları için ek olarak kullanılmaktadır.

### 2.5.2 Programlamaya Giriş

#### İlk Gömülü Programı

Yazılan her programlama kitabı aynı örnekle başlıyor; "Merhaba, Dünya!" yazan bir program. Bunun gibi aşırı kullanılan bir örnek biraz sıkıcı görünebilir. Diğer şeylerin yanı sıra, bu örnek okuyucuların eldeki programlama ortamında hangi basit programların yazılabileceği konusundaki kolaylığı veya zorluğu hızlı bir şekilde değerlendirmelerine yardımcı olur. Bu anlamda, "Merhaba, Dünya!" programlama dilleri ve bilgisayar platformları kullanıcıları için yararlı bir kriter olarak hizmet vermektedir.

"Merhaba, Dünya!" programını düşünürsek, gömülü sistemler programcıların çalışması için en zorlu bilgisayar platformları arasındadır. Bazı gömülü sistemlerde, "Merhaba, Dünya!" programının uygulanması bile imkansız olabilir. Bunu destekleyebilen sistemlerde, metin dizgilerinin basılması genellikle bir başlangıçtan çok ileri düzeyde bir programlamadır.

Gömülü programcılar özgüvenli olmalıdır. Her yeni projeye, hiçbir şeyin işe yaramadığı varsayımla başlamalılar - güvenebilecekleri her şey onların programlama dilinin temel sözdizimi. Standart kütüphane rutinleri bile onlar için mevcut olmayabilir. Bunlar, diğer programcıların çoğunun aldığı printf ve memcpy gibi yardımcı işlevlerdir. Aslında, kütüphane rutinleri genellikle temel sözdizimi olarak C-dil standardının bir parçasıdır. Bununla birlikte, standardın kütüphane bölümünün tüm olası bilgisayar platformlarında desteklenmesi daha zordur ve gömülü sistemler için derleyiciler yapımcıları tarafından zaman zaman göz ardı edilmektedir.

"Merhaba, Dünya!" için popüler bir alternatif program LED yanıp sönen bir programdır. Bir ledi açıp kapatmak için kısa bir kod parçası gereklidir. Bu yüzden de programlama hataları için çok az yer içerir.

Mesela Arduino üzerinde gömülü led için bir blink programı aşağıdaki gibidir.

```
1 void setup() {
2     // Ledi output olarak tanımla
3     pinMode(LED_BUILTIN, OUTPUT);
4 }
5
6 void loop() {
7     digitalWrite(LED_BUILTIN, HIGH);      // Ledi ac
8     delay(1000);                      // 1 saniye bekle
```

```

9   digitalWrite(LED_BUILTIN, LOW);      // Ledi kapat
10  delay(1000);                      // 1 saniye bekle
11 }

```

## Sonsuz Döngü

Gömülü sistemler için geliştirilen programlar ile diğer bilgisayar platformları için yazılan programlar arasındaki en temel farklılıkların biri, gömülü programların neredeyse her zaman sonsuz bir döngüye sahip olmasıdır. Tipik olarak, bu döngü, Blinking LED programında olduğu gibi, programın işlevsellüğinin önemli bir bölümünü çevreler. Sonsuz döngü gereklidir, çünkü gömülü yazılımın işi asla sona ermez. Bu, ya dünya sona erene ya da sistem sıfırlanana kadar (hangisi önce olursa) çalıştırılmak üzere tasarlanmıştır.

Ek olarak, çoğu gömülü sistem sadece bir parça yazılım kullanır. Donanım önemli olmaksızın birlikte, sistem bu yazılım olmadan dijital saat veya cep telefonu veya mikrodalga fırın değildir. Yazılım çalışmayı durdurursa, donanım işe yaramaz hale gelir. Bu nedenle, gömülü bir programın işlevsel parçaları, her zaman sonsuza dek koşacaklarını garanti eden sonsuz bir döngü tarafından çevrelenir. Yanıp sönen LED programında sonsuz döngüyü unutsaydık, LED sadece bir kez durum değiştirirdi.

## Derleme, Bağlama ve Yerleştirme

Gömülü sistemlerin programlanması, daha önce yaptığınız programlamadan önemli ölçüde farklı değildir. Gerçekten değişmiş olan tek şey, hedef donanım platformunu anlamaz gerektiğidir. Ayrıca, her bir hedef donanım platformu benzersizdir - örneğin, bir seri arabirim üzerinden iletişim kurma yöntemi, işlemciye ve platformdan platforma değişebilir. Ne yazık ki, donanım platformları arasındaki bu benzersizlik, bir çok ek yazılım karmaşıklığını yol açmaktadır ve aynı zamanda yazılım geliştirme sürecinin her zamankinden daha fazla farkında olmanız için gerekenin nedeni de budur.

İnşa araçları, ürettikleri programla aynı sistem üzerinde çalışıklarında, sis-

tem hakkında birçok varsayılm yapabilirler. Bu, yerleşik yazılım geliştirme işleminde genellikle geçerli değildir; burada, oluşturma araçları, hedef donanım platformundan farklı olan bir ana bilgisayarda çalışır. Hedef platform iyi tanımlandığında yazılım geliştirme araçlarının otomatik olarak yapabileceği pek çok şey vardır. Bu otomasyon mümkündür çünkü araçlar, programınızın yüreteceği donanım ve işletim sisteminin özelliklerini kullanabilir. Örneğin, tüm programlarınız Windows çalıştırı IBM uyumlu PC'lerde yürütülecekse, derleyiciniz yazılım görünümünü otomatik hale getirebilir ve bu nedenle görünümünüzden gizleyebilirsiniz. Yazılım oluşturma sürecinin belirli yönleri. Gömülü yazılım geliştirme araçları ise, hedef platform hakkında nadiren varsayımlar yapabilir. Bunun yerine, kullanıcı, daha açık talimatlar vererek, kendi sistemindeki bazı bilgilerini araçlara sunmalıdır.

Bu şekilde kullanıldığında, "hedef platform" terimi, yalnızca donanımı değil, aynı zamanda yazılımınız için temel çalışma ortamı oluşturan işletim sistemi de kapsamaktadır. İşletim sistemi yoksa, bazen gömülü sistemde olduğu gibi, hedef platform sadece programınızın çalıştığı işlemcidir.

Gömülü yazılım oluşturma sürecinin her adımı, genel amaçlı bir bilgisayarda çalışan yazılım tarafından gerçekleştirilen bir dönüşümdür. Bu geliştirme bilgisayarını (genellikle bir PC veya Unix iş istasyonu) hedef gömülü sistemden ayırt etmek için, ana bilgisayar olarak adlandırılır. Derleyici, asembler, linker ve locator, gömülü sistemin kendisinden ziyade bir ana bilgisayarda çalışır. Yine de, bu araçlar çabalarını sadece hedef gömülü sistem üzerinde düzgün bir şekilde yüretecek bir çalıştırılabilir ikili görüntü oluşturmak için birleştirir.

Yeniden yüklenebilir programdan çalıştırılabilir ikili görüntüye dönüştürme işlemini gerçekleştiren araç, konumlandırıcı olarak adlandırılır. Yapım sürecinin en kolay adımı için sorumluluk alır. Aslında, bu adımdaki çalışmaların çoğunu, hedef panosundaki bellek hakkında konum bilgisine giriş olarak bilgi vererek kendiniz yapmanız gereklidir. Konumlandırıcı, bu bilgileri, yeniden yüklenebilir program içindeki kod ve veri bölümlerinin her birine fiziksel bellek adresleri atamak için kullanır. Daha sonra, hedefe yüklenebilecek bir ikili bellek görüntüsü içeren bir çıktı dosyası üretir.

### 2.5.3 BIOS

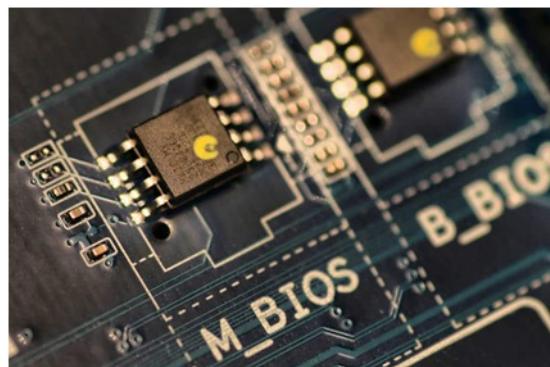
BIOS, Temel Giriş / Çıkış Sisteminin kısaltmasıdır ve ayrıca Sistem BIOS, ROM BIOS veya PC BIOS olarak da bilinir. IBM PC uyumlu bilgisayarlarda önyükleme (güç açma / başlatma) sırasında kullanılan bir ürün yazılımı türüdür. BIOS Firmware, PC'lerde yerleşiktir ve açıldığında çalıştırdıkları ilk yazılımdır. BIOS, tüm bilgisayar sistemlerinde, sistem bileşeni algılaması, bellek testi, CMOS ayarlarının periferik olarak algılanması ve doğrulanması gibi düşük seviye testleri gerçekleştirerek, normalde pil ile yedeklenen önemli bir bileşendir. BIOS, sorunları düzeltmek için kullanıcıyı BIOS ayarları ekranına girmeye zorlar. Gömülü bilgisayarlar için BIOS çok önemlidir, çünkü birçok gömülü proje aşırı çevre koşullarına çözüm olarak farklı düz panel ekran çözünürlükleri, özel sıçrama ekranları, yedekleme BIOS ayarları, hatta pil kullanamayan BIOS'lar ile ilgilenmek için özel bir BIOS kurulumu gereklidir.

#### BIOS'un Görevleri Nelerdir?

- RAM, ekran kartı gibi donanımları açılısta (BOOT) test eder.
- Donanımların voltaj düzeylerini ayarlar.
- Sistemi hard disk dışında bir sistemde çalıştırmak için başlangıç sürecini belirlemeye yarar.
- Donanımların frekans ayarlarını yapar.
- Kullanılmayan donanım parçalarını kapatmamıza olanak sağlar.

#### CMOS Nedir?

CMOS (Complementary Metal Oxide Semiconductor), BIOS çipinde bulunan, başlatma bilgilerini içeren ve çok düşük güç tüketimi yapan bir bellektir. Bilgisayarı başlatırken BIOS, tarih,



saat v.b. gibi bilgileri bu bellekten alır. GÜCÜNÜ de BIOS pili olarak adlandırdığımız pilden karşılar.

#### 2.5.4 Firmware

##### Tanımlar

Firmware, kalıcı hafızanın, program kodunun ve içinde saklanan verilerin birleşimidir. Firmware içeren cihazların tipik örnekleri, trafik ışıkları, tüketici cihazları, dijital saatler, bilgisayarlar, bilgisayar çevre birimleri, cep telefonları ve dijital kameralar gibi gömülü sistemlerdir. Bu cihazlarda bulunan Firmware, cihaz için kontrol programı sağlar.

Firmware şunları yapmalıdır:

- Bir sistem veri yoluna bağlı aygıtları bulması gerekiyor.
- Çalışma zamanı boyunca herhangi bir anda kaybolup görünecek cihazlarla uğraşmak zorunda.
- PC için yazılan tüm işletim sistemlerini ön yüklemek için hazırlanmalıdır.
- Bir dış uyaran oluşursa uyanması gereklidir.
- Arka ışığını, saatlerini ve sıcaklığı, sıcaklığa, güç kaynağına ve kullanımının durumuna (aktif veya etkin) göre ayarlaması gereklidir.

Ancak, gömülü sistemler ve IoT aygıtlarının PC BIOS'tan farklı benzersiz ürün yazılımı gereksinimleri vardır:

- Dış uyarlara (gerçek zamanlı davranış) cevap verme gerekliliği. Örnekler endüstriyel hassas makinelerdir. Bu makineler, çalışma sırasında belirtilen hata marjlarından herhangi bir sapmayı tolere edemezler.
- Yürütmeye sırasında kararlılık. Örnekler füzeler ve roketler. Muhtemelen bu yazılımları, fırlatma rampasından uçtuklarında farklı yazılım

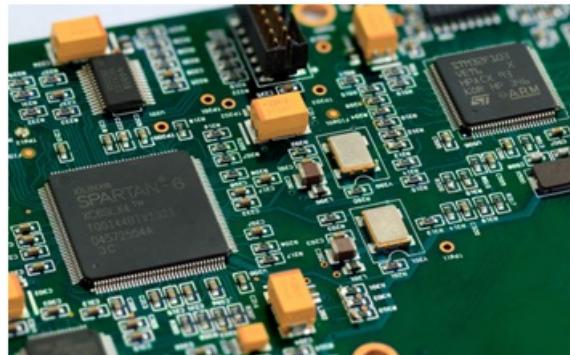
yolları ile görmek istemezsiniz; lansmanından sonra yazılımlarını her çalıştırıldıklarında farklı davranışırsa ne olacağını kim bilebilir? Yazılımın belirsiz olduğu durumlarda sonuçlar ölümcül olabilir.

- Sonuçların tahmin edilebilirliği. Örnekler endüstriyel makineler ve üretim robotlarıdır. Programlanmış davranışlarından sapmadan aynı sonucu tekrar tekrar vermeleri gereklidir; aksi halde üretilen ürünler kullanılamayabilir.
- Sabit fonksiyonlu kapalı sistem. Örnekler set üstü kutu ve GPS'dir. Bu şeylerin içinde çalışan yazılım, hatasız bir şekilde çok iyi bir işlev (veya birkaç işlev) yapıyor olmalıdır.
- Sınırlı genişleme ile kapalı sistem. Örnekler Mars Rover ve ev aletleridir. Bu şeyler teslim edildikten sonra (Dünyadan ayrılır veya eve gönderilir), içerisindeki yazılımlar sistemlere gelen ve giden bileşenlerle uğraşmak zorunda kalmamalıdır, çünkü kapalı bir sistem belirli noktalardan sonra herhangi bir genişleme yapamaz.
- Hızlı önyükleme süresi. Örnekler otomobilin, cep telefonlarının ve ev aletlerinin dikiz kameralarıdır. Bu şeylerin hızlı önyükleme yapması gerekiyor. Güvenlik nedenlerinden dolayı veya daha iyi bir kullanıcı deneyimi için. Bazı eski Blu-ray DVD oynatıcıların DVD'leri oynamaya hazır hale gelmesi uzun zaman allığında çok can sıkıcıydı çünkü tüketici elektroniki cihazlarının hazır olması için birkaç saniyeden fazla zaman alması gerekmiyordu (en azından bu, tüketicilerin alışkin olduğu şey değil).
- Küçük ayak izi. Örnekler giyilebilir cihazlar, sensörler ve sertifikalandırılması gereken yazılımlar. Büyük yazılımlar genellikle daha büyük depolama aygıtlarına ihtiyaç duyarlar ve hataya eğilimli olmaları ve sertifika vermeleri zordur.
- Güvenlik ve güvenilirlik. Örnekler satış noktası (POS) terminalleri ve araç içi bilgi-eglence sistemleridir (IVI). IVI'de, GPS, radyo ve telefon bağlantısını indirmek için sistemin DVD oynatıcısında bir yazılım çökmesi istemeyebilirsiniz ve kredi kartı bilgilerinizin bir mağazadaki ödeme sırasında çalınmasını istemezsiniz; Bu nedenle, kritik ve önemsiz unsurların izolasyonu, bazı uygulamalar için çok önemlidir.
- Sabit önyükleme hedefi. Örnekler Chromebook ve IVI. Bu cihazların aklında tek bir önyükleme hedefi vardır (ChromeOS veya gömülü Li-

nux); bu nedenle, bellenimin, herhangi bir işletim sistemini önyüklemeye hazır olan genel amaçlı bir önyükleme arabirimine sahip olma konusunda endişelenmesi gerekmekz. Doğrudan önyükleme arabirimi yalnızca kod alanını kaydedemez, aynı zamanda kısayollar alınabildiğinden hızlı bir şekilde önyükleme yapar.

Bunlar gömülü ürün yazılımı(firmware) için sadece ortak gereksinimlerdir. Gördüğünüz gibi, PC yazılımindan çok farklılar. Yine de, her gömülü sistem tasarımlı, önceki listeden gelebilecek ya da olmayabilen kendi gereksinimlerine sahip olabilir ya da bu özelliklerin katlarına sahip olabilir.

Bu gereksinimlere ek olarak, gömülü sistemler bilgisayarlardan daha uzun bir yaşam döngüsüne sahip olma eğilimindedir; Bu nedenle, kapsamlı ve derin özelleştirme maliyeti haklı olabilir.



### Gömülü Sistemler İçin Ürün Yazılımı(Firmware) Tasarlama ve Uygulama

**Gömülü sistemler için ürün yazılımı uygularken ne yapmam gereklidir?**

- Gömülü sistemler için kod yazmak, bir PC için kullanıcı kodunun yazılımasıyla aynı değildir (kişisel bilgisayar). Daha çok bir bilgisayar için bir sürücü yazmak gibi. TinyOS gibi özel bir gömülü işletim sistemi (OS) kullanılmazsa, yazılım mühendisi donanım kayıtlarının kendisinin ayarlanması gibi tüm temel şeylerin üstesinden gelmelidir. Kod minimalist, verimli, gerçek zamanlı, kararlı, okunması kolay vb. olmalıdır.

- Sisteminiz, genellikle motorlar, sensörler, iletişim yolları gibi gerçek donanıma bağlı olduğu anlamına gelir. Yazılım hatalarının dramatik ve pahali sonuçları olabileceği anlamına gelir. Pinler, yüksek akımlara ve elektronik parçaların zarar görmesine neden olan yazılımlarda kısa devre yapabilir. İzlenmesi gereken piller, düşük ve aşırı şarj edilebilir ve patlayabilir. Sensörler doğru kullanılmadığında yok edilebilir. Motorların redüktörleri uygun kontrol olmadan kırılabilir. Bu nedenle, sisteminizi önce seçilebilir akım sınırına sahip güç kaynakları kullanarak güvenli bir ortamda test edin ve mikro denetleyicinizin pinlerinin grişler ve çıkışlar olarak doğru şekilde yapılandırıldığından emin olun.
- Asla while döngülerinde beklemekle meşgul olmayın. Daima zamanlayıcı kullanın. Neden: Yoğun beklemeniz, işlemcinin yoğun bir durumda olmasını engeller. Bu süre zarfında yararlı bir şey yapamaz. Unutmayın: Gömülü sistemler gerçek zamanda çalışmalıdır. Genellikle, farklı işlemler arasında otomatik olarak geçiş yapan bir işletim sistemi için yazılım yazarken olduğu gibi zamanlayıcı yoktur.
- Kesintiler ve DMA (doğrudan hafıza erişimi) gibi tasarım stratejileri hakkında bilgi edinin. Kesmeler, bir giriş sinyalinin değiştirilmesi gibi bir koşulu aktif olarak beklemekten kaçınmanıza yardımcı olacaktır, ancak bunun yerine yalnızca değişiklik olduğunda tepki vermenize izin verecektir. DMA, işlemcınızı, bir verinin bir bellekten diğerine aktarılmasına hazır olmayan bekleyen döküm işinden kurtarır.
- Asla bir kesinti servisi rutinine çok fazla kod koymayın. Bu geri arama işlevleri sadece sinyal olarak orada ve mümkün olduğunda kısa olmalıdır. Bu nedenle, bir kesintiden sonra çok fazla iş yapmanız gerekiyorsa: bir bayrak ayarlayın ve işi ana döngüde yapın.
- Gömülü sistemlerin bilinmeyen zaman koşullarıyla yüz yüze olduğunun bilincinde olun. Örneğin. Bir kullanıcının tam olarak ne zaman ve hangi sırayla bir anahtar veya düğme kullanacağını veya bir UART arabiriminden bazı verileri alacağınızı asla bilemezsiniz. Bunun için hazırlıklı olun.
- Kodunuzu okunabilir ve tekrar kullanılabilir hale getirmek için kodlama standartlarına dikkat edin ve tutarlı olun.
- Mikrodenetleyiciler ve geliştirme ortamları genellikle hata ayıklama işlevleriyle birlikte gelir. Onları kullanın.

### Ürün Yazılımı(Firmware) Tasarımı

Gömülü sistemler için yazılım yazılımı deneme yanlış sorunu değil, düşünme ve kodlamadır. İyi bir uygulama stratejisi ve düzgün bir kurulum yazılımı olmadan kodunuz okunamaz, güncellenmesi ve hatalarının ayıklanması ve hatalar ve tasarım hataları ile dolu olması zor olacaktır. Kimse kodunuzu yeniden kullanmaya istekli olmayacağından emin olmalıdır. Ve bu sadece sizin firmwareinizin zaman kaybı olduğunu gösterir.

Bu yüzden dikkatlice cevaplamanız gereken şeyler:

- Firmwareinizin amacı nedir? Örneğin. Bir RS232 arabiriminin bir RS485 arabirimleriyle konuşmasına izin veren bir yazılım yazmak istiyorum.
- Firmwareinizin amacı ile ilgili karşılaşığınız zorluklar nelerdir? Örneğin. RS485, RS232 tam çift yönlü bir noktadan noktaya iletişim arabirim iken, yalnızca bir iletişim ortağının veri göndemesine izin verilen bir yarı çift yönlü noktadan çoklu iletişim arabirimidir. Yapmam gereken sorunlar:
  - Bir seferde sadece bir iletişim partnerinin veri göndemesi nasıl sağlanır?
  - RS232 üzerinden, veriyolu şu anda başka biri tarafından kullanıldığından RS485'e iletilemeyen verilerle ne yapmalıyım?
  - Deadlock nasıl engellenir - yani RS485 veri yolu sonsuza dek bloke edilir?
- Karşılaştığınız zorluklarla nasıl başa çıkmayı planlıyorsunuz? Hangi stratejileri takip etmeyi planlıyorsunuz? Örneğin:
  - Bir master ile tek bir iletişim partnerinin bir seferde veri gönderebileceği bir TDMA (zaman bölmeli çoklu erişim) protokolü uygulayacağım. Master, tek bir iletişim partnerinin, verileri soraarak bir kerede veri göndereceğini garanti edecektir. Hiçbir slave, istenmeden veri göndemesine izin verilmez.
  - İletilemeyen verileri saklayabilen bir halka tamponu uygulayacağım. Veri yolu boşaldığında, depolanan verileri göndererek arabalığı boşaltacağım.
  - Zaman aşımı durumunda RS485 veri yolunun tekrar kullanılmasını sağlayan bir zamanlayıcı kullanacağım. Eğer bir slave'in ce-

vabı zamanında alınmazsa, bu cihazda bir sorun olduğunu varsayıcağım ve host'a bir hata mesajı göndereceğim.

- Kullanacağınız firmware mimarisi için detaylı bir plan yapın. Hangi dosyalara ihtiyacım var? Mikrodenetleyicinin hangi çevreleri kullanılıyor? Bu çevreleri doğru şekilde nasıl kurarım? Hangi fonksiyonların uygulanması gerekiyor? Örneğin. Kullandığım her çevre için bir kaynak ve başlık dosyasına ihtiyacım var. Ayrıca, daha sık değiştirmem gerekecek genel konfigürasyonları sakladığım bir ana dosyaya ve bir yapılandırma dosyasına ihtiyacım var. Kodumu mümkün olduğunda tekrar kullanılabılır hale getirmek istedigim için, özel mikro denetleyiciye IO bağlantı noktalarının yapılandırması gibi özel yapılandırmalar yerleştirdiğimde özel bir başlık ve kaynak dosyası uygulayacağım. Ana dosyadaki ana init işlevinden çağrılan her bir periferik için başlatma işlevleri olacaktır. Sistem saatini doğru şekilde kurmak zorundayım. Tam 40MIPS hızını kullanmayı planlıyorum (saniye başına mega yönergesi). Bu yüzden PLL (faz kilitli döngü) ayarlamam gerekiyor. UART'lara (evrensel asenkron alıcı verici) ihtiyacım olacak. ADC'yi kullanmayı planlamıyorum. Böylece bunun kapalı olduğundan emin olacağım. Hata ayıklamak ve LED'lerle konuşmak için bazı GPIO'lara (genel amaçlı giriş çıkış) ihtiyacım olacak. Ve bunun gibi.

### **Firmware Uygulanması**

Belgeleriniz hazırlandıktan sonra uygulamaya başlayabilirsiniz. Unutmayın: önce düşünün ve sadece kodunuzu mikrodenetleyiciye yüklemeyin ve bazı şeyleri deneyin. Gömülü bir sistemle çalışıyorsunuz.

- Üstbilgi ve kaynak dosyalarınızın çerçevesini ayarlayın. Dosyalarınızı SVN sistemine gönderin.
- Mikrodenetleyicinizin saatini ve diğer temel yapılandırma kayıtlarını ayarlayın.
- main () ve main\_init () işlevlerini oluşturun.
- Mikrodenetleyicinizin pinlerini doğru şekilde ayarlayın. Tüm pinlerin doğru yapılandırıldığından emin olun. Girişler ve çıkışlar olarak. Pinlerinizle konuşmak için makrolar oluşturun.
- Bir zamanlayıcıya ihtiyacınız olacak. Bunu doğru şekilde ayarlayın.

- Bu zamanlayıcı ile bir GPIO portunu sürerek ve portu bir osiloskopla dikkatlice ölçerek zamanlayıcınızı ve saatinizi test etmek iyi bir fikir olacaktır. Dönemin doğru olduğundan ve titremenin olmadığından emin olun. Bu çalışıktan sonra dosyalarınızı SVN sistemine işlemek için iyi bir zamandır.
- Artık ekstra hata ayıklama olasılığına sahip olmak için bir UART kurmak isteyebilirsiniz. UART ile veri göndermeyi ve almayı deneyin.
- Tüm bu şeyler doğru bir şekilde çalışıktan sonra, belirli ürün yazılımınızı uygulamaya başlayabilirsiniz.



## Bölüm 3

# Geliştirme Ortamları

### 3.1 İşletim Sistemi Olmayan Geliştirme Ortamları

#### 3.1.1 Arduino IDE

Arduino IDE, arduino kitleri için geliştirdiği; komutların yazılmasına, derleme işleminin yapılmasına ve son olarak derlenen kodları doğrudan (Bilgisayarın USB portuna bağlı olan) Arduino kite yüklenmesine olanak sağlayan yazılım geliştirme platformudur. Açık kaynak Arduino Yazılımı (IDE), kodu yazmayı ve panoya yüklemeyi kolaylaştırır. Windows, Mac OS X ve Linux üzerinde çalışır. Ortam Java'da yazılmıştır.

**Arduino IDE programını bilgisayara indirme ve kurma:**

- IDE programı Arduino'nun resmi sitesinden ücretsiz olarak indirilebilir.
- <https://www.arduino.cc/en/Main/Software> sitesine girerek, Arduino IDE'nin en son sürümü için aşağıdaki sayfa görüntüsünün sağ tarafında bulunan linklerden (Windows, Mac OS veya Linux) uygun olanı seçilir.
- Bir sonraki yükleme sayfasında bağış yapmak isteyip istemediğimiz sorulmaktadır. Burada “JUST DOWNLOAD”(Sadece indir) ve “CONTRIBUTE & DOWNLOAD”(Katkıda bulun ve indir) seçeneklerinden

### Download the Arduino IDE



tercih ettiğiniz buton tıklanarak indirme işlemi başlatılır.

- İndirme tamamlandıktan sonra, indirilen dosya çift tıklanarak kurulumu yapılır. Program çalıştırıldığında Arduino kit, USB portuna bağlanarak algılanıp algılanmadığı(Ara yüz ekranında sağ alt köşede bağlı olduğu port adı görülmektedir.) test edilebilir.

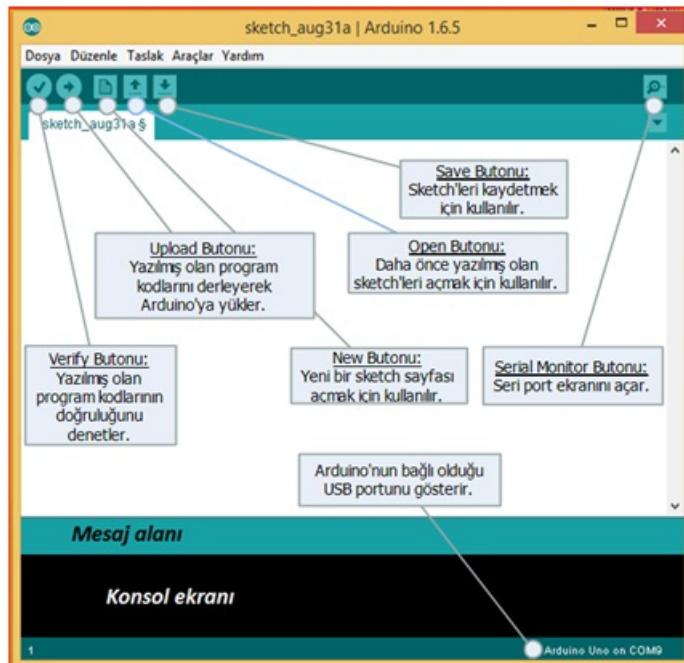
### Arduino IDE Kullanımı

Arduino IDE, kullanılarak yazılan programlara sketch adı verilir. Sketch'ler yani program kodları text editör olarak adlandırılan kısma yazılır. Alt taraflında ise işlemlerle ilgili mesajların alındığı bölüm bulunmaktadır. Programın ara yüzü üzerinde bulunan kısa yol butonlarının ve bildirim alanlarının açıklamaları şekilde verilmiştir.

#### IDE Programının menüleri:

##### Dosya/File menüsü:

- Yeni(New), bir sketch yazmak için sayfa açma,
- Açı(Open), daha önce yazılanları açarak ekrana getirme,
- Open Recent, en son üzerinde çalışılan sketch'ler ekrana getirilir,
- Kaydet(Save) ve Farklı Kaydet(Save AS) ile kaydetme işlemleri yapılabilir.
- Yazdır(Printer) ve Sayfa ayarları ile yazdırma işlemleri yapılabilir.
- Örnekler/examples butonu üzerine mouse getirildiğinde hazır birçok örnek uygulama listesi karşımıza çıkar.



#### Düzenle/Edit Menüsü:

- Son yapılan işlemi ileri(Redo)-geri(Undo) alma,
- Kes(Cut) – Kopyala(Copy) – Yapıştır (Paste),
- (Comment/Uncomment), imlecin bulunduğu satırı açıklama satırı yapma,
- Forum için kopyala, yazılan komutları forum sayfasında kullanılmak üzere hazırlar.
- HTML olarak kopyala, kodları HTML formatında biçimlendirilir.

#### Taslak/Sketch Menüsü:

- Kontrol et/Derle (Verify / Compiler), yazılan sketch'lerde hata olup olmadığını denetler ve aynı zamanda kodların derleme işlemini de yapar.
- Yükle(Load), yazılan sketch'lerin Arduino kite yüklenmesi sağlanır. Yükle butonu kullanıldığında önce derleme işlemini gerçekleştirir sonrasında derlenen kodları kite yükleme yapar.

- Programlayıcı kullanarak yükle, butonu Arduino'da dâhili olarak bulunan programlayıcıya ihtiyaç duymadan doğrudan Atmel marka mikrodenetleyici aracılığıyla yükleme işlemi yapılacaksa kullanılır.
- Include Library, butonu ile kütüphane dosyası eklenebilir. C, C++, Java gibi dillerde #include komutu kullanılarak yapılan işlemin aynısı bu buton ile yapılabilir.

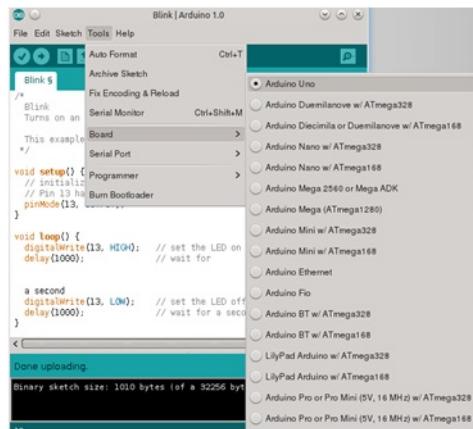
#### Araçlar/Tools Menüsü:

- Otomatik Düzenle (Auto Format) yazılan kodların görsel olarak daha düzenli hale getirir. Süslü parantezleri aynı hizaya getirir.
- Çalışmayı Arşivle (Archive Sketch), yazılan kodları .zip formatında arşivlenmesini sağlar. vSeri Port Ekranı, bazı uygulamalarda karttan bilgisayara veri gönderildiğinde bunların görüntülenmesini sağlayan ekranı açar.
- Kart(Board) butonu ile kullanılan Arduino kartı seçilir.
- Port(Serial Port), bilgisayara bağlı olan tüm seri port cihazlarını görüntüler. Birden fazla Arduino kit bağlanılmış ise bunlarda görüntülenir ve çalıştırılmak istenen Kart tercih edilir.
- Programlayıcı, Arduino için farklı programlayıcı kullanılacaksa buradan seçimi yapılr.
- Önyükleyici Yazdır(Burn Bootloader), Bootloader, Arduino için doğrudan USB aracılığıyla programın atılmasını sağlayan ve her Arduino içinde yüklü biçimde gelen bir programdır. Bu programın diğer Atmel mikrodenetleyicilere atılmasını sağlar.

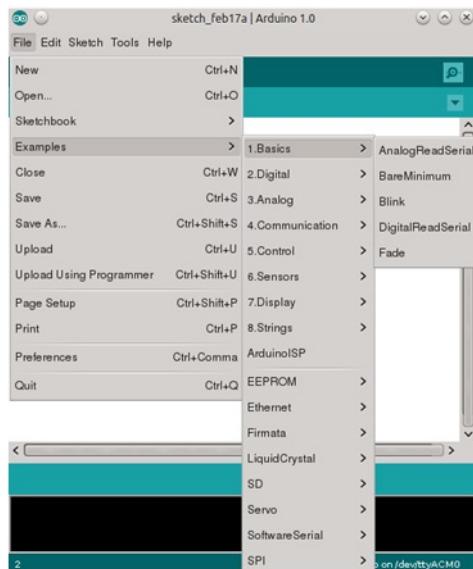
Arduino'ya program atmak için ilk olarak bilgisayarımıza bağladığımız Arduino'yu ve bağlantı portunu Arduino program menüsünden seçmemiz gerekiyor. Bunun için sırayla şu adımları izleyebiliriz:

1. İlk olarak programı açıyoruz.
2. Tools menüsü altında Serial Port seçeneğine gelip bağladığımız portun yanında tık işaretini yoksa tıklayıp işaretliyoruz. Genelde tek port gözükeceği için burda seçim hakkınız olmayacak. Birden çok seri port olduğu durumlarda Arduino'nuzu söküp takarak hangi port değişiyorsa onu seçmeniz gerektiğini anlayabilirsiniz.

3. Daha sonra kullandığımız Arduino modelini tanıtıyoruz. Onun için de su ekrandan elinizdeki Arduino modelini seçmeniz gerekiyor:



4. Bu seçenekler arasından da elimizdeki Arduino modelini seçtikten sonra, Arduino üzerinde bulunan örnek programlardan LED yakıp söndüren basit uygulamayı deneyebiliriz. Bu ekranda Blink seçeneğine tıklıyoruz ve önceden yazılmış test edilmiş basit bir kod karşımıza geliyor.



5. Bu ekranı da gördükten sonra, artık kodlarımız derlenmeye ve Arduino üzerinde çalıştırılmaya hazır demektir. Sadece derleyip, kodlarımızı

Arduino'ya atmak istemiyorsak en soldaki tık işaretini ile gösterilen Verify butonunu kullanabiliriz. Bu butona basıldığında kod derlenecek fakat Arduino'ya atılmayacaktır. Faydası ise kodta yazım vs. herhangi bir hata olduğu zaman derlenmeye engelleyecek veya uyarıya sebep olacak hataların önceden görülmemesidir. Upload yani gönderme buton kodumuzu derler ve Arduino'ya yükler. Bu işlemin ardından artık kodumuz çalışmaya hazırır. 13 numaralı pine genelde Arduino modellerinde bir LED bağlıdır. Bağlı değilse de, 330 Ohmluk bir direnç ile bir LED'i, LED'in + ucu 13 numaralı pine, direncin – ucu ise GND yazan uca bağlanacak şekilde ayarlayabilirsiniz.

### 3.1.2 Cypress

Cypress modern internet için yeni nesil, ön uç test aracıdır. Geliştiricilerin ve QA mühendislerinin, modern uygulamaları test ederken karşılaştıkları temel sorunları ele alıyoruz.

Cypress ile neler yapabiliriz ?

- Testler kurabiliriz.
- Testler yazabiliriz.
- Testleri çalıştırabiliriz
- Debug testleri hazırlayabiliriz.

Cypress kullanıcıları tipik olarak modern JavaScript çerçevelerini kullanarak web uygulamaları geliştiren geliştiriciler veya QA mühendisleridir.

Cypress her türlü testi yazmanızı sağlar. Uçtan uca testler, entegrasyon testleri, birim testleri. Cypress bir tarayıcıda çalışan her şeyi test edebilir. Cypress ücretsiz, açık kaynaklı, yerel olarak kurulmuş bir test aracıdır ve testlerini kaydetmek için bir hizmettir. Yüklemek veya yapılandırmak için sunucu, sürücü veya başka bağımlılık yoktur. Cypress ile yazılan testlerin okunması ve anlaşılması kolaydır. API, zaten aşina olduğumuz araçların üzerinde gelir.

Cypress, tarayıcınızın içeriği oluşturabileceği kadar hızlı çalışır. Uygulamalarınızı geliştirirken testleri gerçek zamanlı olarak izleyebilirsiniz. Okunabilir hata iletleri, hızlı bir şekilde hata ayıklamanıza yardımcı olur. Ayrıca bildiğiniz ve sevdığınız tüm geliştirici araçlarına da erişebilirsiniz.

### 3.1.3 ESP

ESP8266 Wifi Serial Transceiver Modül oldukça ekonomik ve kullanışlı bir Wifi modüldür. TCP/IP protokolünü desteklemektedir. ESP8266 üzerinde dahili anten bulunmaktadır. Bu sayede ortamdaki Wifi ağına rahatlıkla bağlanabilmekte, veri paketleri alıp gönderebilmektedir.

#### ESP8266 Geliştirme Kartı Özellikleri

- 802.11 b/g/n desteği
- Wi-Fi Direct (P2P) Desteği
- Dahili TCP/IP protokol yığını
- +19,5dBm çıkış gücü (802.11b modunda)
- Kaçak akım < 10uA
- Dahili düşük güç tüketimine sahip 32-bit'lik işlemci
- SDIO 1.1/2.0, SPI ve UART desteği
- STBC, 1x1 MIMO, 2x1 MIMO
- Uyanma ve veri paketi alma süresi < 2ms
- Stand-by durumunda güç tüketimi < 1mW

Esp32, Espressif firması tarafından üretilen bir önceki nesil olan ESP8266 modülüne göre oldukça gelişmiş bir yapıya sahiptir. Bu modül üzerine bluetooth modülü eklenerek, işlemci gücü ve bellek kapasitesi gibi bileşenler arttırlılmıştır. Genel özellikleri aşağıda görüldüğü gibidir. Detaylı bilgiyi üretici firmanın katalog bilgilerinden edinebilirsiniz.

Geliştirme ortamı olarak Espressif firması tarafından dağıtılan SDK (Software Development Kit-Yazılım Geliştirme Kiti) ve ya IDE (Integrated Development Environment- Tümleşik Geliştirme Ortamı) çözümlerinden birini kullanmak isterseniz karta ait tüm konfigürasyonları detaylı olarak kontrol edebileceğiniz bir ortam bulabilirsiniz ancak hem bu ortamların kurulum zorluğu hem de kullanımını için gerekli olan ileri seviye bilgilerden dolayı Arduino IDE kullanmanızı tavsiye ederim.

ESP32 Arduino IDE ile birlikte kullanmak için gerekli işlemelere <https://github.com/espressif/arduino-esp32> linkinden ulaşabilirsiniz. Kullanmakta olduğunuz işletim sisteme göre

DONANIM	Çevresel Birimler
Xtensa®Çift çekirdekli 32 bit LX6 mikroişlemcileri	12 kanallı 12 ADET SAR ADC
128 KB ROM	2x10 bit DA dönüştürücüler
QSPI Flaş / SRAM, 4 x 16'ya kadar MB	10x dokunmatik sensörler
416 KB SRAM	Sıcaklık sensörü
2 MHz - 40 MHz kristal osilatör	4sSPI, 2xI2S, 2xI2C, 2xUART
Kalibrasyonlu RTC için harici 32kHz osilatör	1 ana bilgisayar (SD / eMMC / SDIO) 1 slave (SDIO / SPI)
802.11 b / g / n / e / I 802.11 n (2.4Ghz), 150Mbps'ye kadar	Özel DMA ve IEEE 1588 desteği ile Ethernet MAC arabirimleri
Kablosuz internet Korumalı Kurulum (WPS)	CAN 2.0 401.176 (TX / RX)
Bluetooth 4.2 (BR / EDR / BLE)	Motor PWM LED PWM en fazla 16 kanal



(Windows, MAC OS, Linux) adımları uygulayarak kurulumu tamamlayabilirsiniz. Kurulum işlemi sorunsuz gerçekleştiyse Araçlar-Kart seçenekleri altında ESP32 DEV Module’ü seçerek deneyleri gerçekleştirebilirsınız.

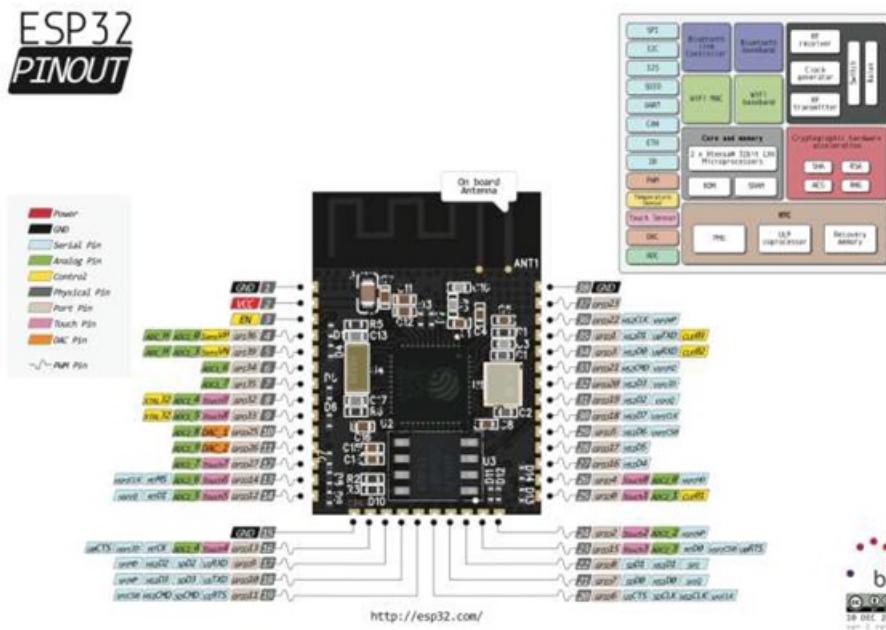
## 3.2 İşletim Sistemi Olan Geliştirme Ortamları

### 3.2.1 Linux

#### Gömülü Linux

##### Ayak izi üzeri özellikler

İşleme yetenekleri, bağlantı ve kullanıcı bekłentileri ilerledikçe, Linux'un özellikleri, karar verme aşamasında ayak izinden ağır basmaya başlayacak birçok gömülü sistem için çözüm. Bazı durumlarda, Linux değişkenleri bile gömülü sistemler için güvenilir seçenekler olarak görülmüyor.



Şekil 3.1: ESP32 Pin Yapısı

“Daha büyük bir ayak izi ekliyor, ancak size güvenilirlik sağlıyor” diyor Featherston. “Bu, birçok insan için daha küçük bir ayak izi olmaktan çok daha önemli görünüyor.”

10 yıl önce gömülü Linux atasözü oyun alanındaydı. Günümüzde ise resmi olarak üretime ulaştı.

### Yocto Projesi ile Gömülü Linux Dağıtımları Oluşturmak

#### Genel bakış

Yocto Projesi, donanım mimarisinden bağımsız olarak gömülü ürünler için özel Linux tabanlı sistemleri destekleyen şablonlar, araçlar ve yöntemler sağlayan açık kaynaklı bir işbirliği projesidir. Adı merak edenler için yocto terimi en küçük SI birimidir. Bir önek olarak, yocto  $10^{-24}$  gösterir.

Bu makale, Yocto Project'in endüstri standartı açık kaynak araçlarının, gömülü bir aygit için özelleştirilmiş bir Linux işletim sistemi oluşturmak ve iş-

letim sistemini QEMU kullanarak sanal bir makinede önyüklemek için adım adım ilerlemesini sağlar. Büyük donanım şirketleri ve işletim sistemleri satıcıları tarafından finanse edilen Linux Vakfı tarafından desteklenen bir açık kaynak projesi olan Yocto Projesi, Linux sistemleri oluşturmak için endüstri sınıfı araçlar, yöntemler ve meta veriler sağlar.

Yocto Projesinin iki ana bileşeni, OpenEmbedded projesi ile birlikte sürdürür: BitBake, yapı motoru ve yapıyı çalıştırma için kullanılan çekirdek tarifler olan OpenEmbedded-Core. Projenin tüm bileşenleri bir sonraki bölümde açıklanmıştır.

### **Yocto Projesi Tanımı**

Bazen bir “şemsiye” projesi olarak adlandırılan bir işbirliği projesi olarak Yocto Projesi, geliştirme sürecinin birçok farklı parçasını bünyesinde barındırıyor. Bu parçalar, genel Yocto Projesi kapsamında projeler olarak adlandırılır ve yapım araçları, tarifler, kütüphaneler, yardımcı programlar ve grafik kullanıcı arabirimleri (GUI’ler) olarak adlandırılan öğretim meta verileri oluştururlar. dar

Poky, Yocto Projesi için bir referans sistemidir. BitBake, OpenEmbedded-Core, bir board destek paketi (BSP) ve yapıya dahil edilen diğer paketler veya katmanları içerir. Poky adı, aynı zamanda, çok az (core-image-minimal) veya bir GUI (core-image-sato) içeren tam bir Linux sistemi olabilecek referans oluşturma sisteminin kullanılmasından kaynaklanan varsayılan Linux dağıtımına da gelir.

Projenin tamamı için bir uygulama sistemi olan Poky yapım sistemini düşünebilirsiniz - bu, işlemdeki sürecin çalışan bir örneğidir. Yocto Project’i indirdiğinizde, aslında burada açıklandığı gibi varsayılan sistemi oluşturmak için kullanabileceğiniz araçların, yardımcı programların, kitaplıkların, araç zincirinin ve meta verilerin bir örneğini yüklersiniz. Bu referans sistemi ve yarattığı referans dağılımı her ikisi de Poky olarak adlandırılır. Bunu, kendi dağıtımınızı oluşturmak için bir başlangıç noktası olarak da kullanabilirsiniz.

Tüm yapı sistemlerinin gerektirdiği bir öğe, bir takım zinciridir: belirli bir mimari için ikili yürütülebilir dosyalar oluşturmak için gerekli bir derleyici, derleyici, bağlayıcı ve diğer ikili yardımcı programlar. Poky, GNU Derleyici Koleksiyonunu (GCC) kullanır, ancak diğer araç takımlarını da belirtebilirsiniz. Poky, çapraz derleme olarak bilinen bir teknigi kullanır: bir mimaride bir toolchain kullanarak farklı bir mimari için ikili yürütülebilir dosyalar oluş-

turmak (örneğin, x86 tabanlı bir sistemde bir ARM dağılımı oluşturmak). Geliştiriciler genellikle, ana sistem sisteminin daha yüksek performansından yararlanmak için gömülü sistemler geliştirmede çapraz derleme kullanırlar.

### **Meta veri kümesi**

Meta veri kümesi, her katmanın altındaki katmanlara ayrı işlevler sağlayabilmesi için katmanlarda düzenlenmiştir. Temel katman, tüm yapılar için ortak ve gerekli olan tarifleri, sınıfları ve ilişkili işlevleri sağlayan OpenEmbedded-Core veya oe-core'dir. Daha sonra, oe-core üzerine yeni katmanlar ekleyerek yapıları özelleştirebilirsiniz.

OpenEmbedded-Core, Yocto Projesi ve OpenEmbedded projesi tarafından koordine edilmektedir. Yocto Projesi'ni OpenEmbedded'den ayıran bir katman, Poky dağıtım yapılandırmasını ve çekirdek BSP'leri sağlayan bir meta-yocto katmanıdır.

OpenEmbedded projesinin kendisi, (büyük ölçüde) değiştirilebilir tarifler ve Yocto Projesi'ne benzer hedefler, ancak farklı yönetim ve kapsamı olan ayrı bir açık kaynak projedir.

### **Kart destek paketleri**

Bir BSP, belirli bir tahta veya mimari için Linux oluşturmak için gerekli olan temel paketleri ve sürücülerini içerir. Bunlar genellikle tahtaları yapan donanım üreticileri tarafından korunur. BSP'ler, Linux işletim sistemi ile onu çalıştırınan donanım arasındaki arabirimdir. Sanal makineler için BSP'ler oluşturulmanın da mümkün olduğunu unutmayın.

### **BitBake**

BitBake bir yapı motorudur. Tarifler okur ve paketleri getirerek, bunları oluşturarak ve sonuçları önyüklenebilir görüntülerle birleştirerek izler. BitBake, Yocto Projesi ve OpenEmbedded projesi tarafından koordine edilmektedir.

### **Hob**

Gömülü Linux geliştirmeyi kolaylaştırmak için, Yocto Projesi grafiksel olarak çalışmak için birkaç farklı yöntem sunar. Projeye nispeten yeni bir ek olarak, BitBake ve inşa sürecine grafik bir ön uç sağlayan Hob denir. Her ikisi de, topluluk kullanıcı çalışmaları ile tamamlanmış sürekli bir gelişim içinde.

### Açık kaynaklı lisans uyumluluğu

Açık kaynak lisanslarına uymak, herhangi bir Linux geliştirme çabasının çok önemli bir parçasıdır. Yocto Projesinin bir amacı, uyumluluğu olabildiğince kolaylaştırmaktır. Yocto Proje araçlarını manifestolar oluşturmak ve hatta tüm kaynak depolarını oluşturmak ve belirli lisansları kullanan paketleri dışlamak için oluşturma işleminizi filtrelemek için kullanmak oldukça kolaydır. Proje, Yazılım Paketi Data Exchange® (SPDX™) spesifikasyonuyla ilgili olarak Açık Uyum Programında Linux Vakfı ile birlikte çalışmaktadır.

### EGLIBC

Gömülü GLIBC (EGLIBC), gömülü sistemler üzerinde iyi çalışacak şekilde tasarlanmış GNU C Kitaplığı'nın (GLIBC) bir çeşididir. EGLIBC'nin hedefleri arasında azaltılmış ayak izi, yapılandırılabilir bileşenler ve çapraz derleme ve çapraz test için daha iyi destek yer alır. EGLIBC Yocto Projesi şemsiyesi altındadır, ancak kendi yönetim yapısı içinde korunmaktadır.

### Uygulama Geliştirme Araç Seti

Uygulama Geliştirme Araç Takımı (ADT), sistem geliştiricilerinin Yocto Proje araçları kullanılarak oluşturdukları dağıtımlar için yazılım geliştirme kitleri (SDK'lar) sağlamasına ve uygulama geliştiricilerine bu sistem geliştiricileri tarafından sağlanan yazılım yığınlarına karşı geliştirmenin bir yolunu sağlar. ADT çapraz derleme araç zinciri, hata ayıklama ve profil oluşturma araçları ve QEMU öykünmesi ve destek komut dosyaları içerir. ADT, entegre geliştirme ortamları (IDE) ile çalışmak isteyenler için bir Eclipse eklentisi de içerir. Yocto Projesi şemsiyesi altındaki diğer araçlar

### Yocto Projesi afişi altındaki bazı diğer araçlar:

**Autobuilder:** Otomatik Kalite Güvencesi (QA) faaliyetlerine olanak tanıyan Yocto Proje araçlarının sürekli otomatik oluşturulmasını sağlar. **Cross-Prelink:** Çapraz derleme geliştirme ortamları için ön hazırlık sağlar, performansı artırır. **Pseudo:** Önyüklenebilir bir final görüntüsü oluşturmanın önemli bir parçası olan kök erişimini öykünür. **Swabber:** Çapraz derleme yapısının ana bilgisayar bileşenleri tarafından kirlendiği zamanı algılar. **Build Appliance:** Hob'i çalıştıran sanal bir makinedir, Linux dışı yapı kullananları Yocto Projesi sürecini ilk elden görmek için kullananlar sağlar. (Not: Yocto Projesi oluşturma araçları şu anda yalnızca Linux'ta desteklenmektedir.)

### Yönetim ve toplum

Herhangi bir açık kaynak projesi için hayatı önen taşıyan bir bileşen, bileşenlerini geliştiren ve destekleyen topluluktur. Yocto Projesi, kısmen donanım kuruluşları, işletim sistemleri ve araç sağlayıcıları ve elektronik üreticileri dahil olmak üzere kuruluşlar tarafından oluşturulmuş aktif bir topluluğa sahiptir, böylece gömülü gelişimin tüm yelpazesini yakalarlar. bu organizasyonlar.

Proje, tüm teknik kararları alan bir baş mimar ve bir dizi bakıcı ve teknik lider tarafından yönetilmektedir. Birçoğu normalde birbiriyile rekabet eden üye örgütler, adından da anlaşılacağı gibi, doğası tavsiye niteliğinde olan bir Danışma Kurulu oluşturmak için işbirliği yaparlar. Bu kurul, altyapı, savunuculuk ve sosyal yardım ve finans dahil olmak üzere projenin kaynaklarını yönetir.<sup>1</sup>

Muhtemelen bildığınız şey, Linux'un aslında gömülü sistemlerde kullanılmak üzere tasarlanmamış olmasıdır. Çoğu senaryoda, Linux'un determinizmi en iyi ihtimalle "yumuşak gerçek zamanlı" olarak nitelendirilebilir. Linux çekirdeğinin belirli bir işlemciye veya uygulamaya ayarlanması, geleneksel olarak, bir sistem oluşturmak ve çaprazlama yapmak önemli bir görev olmadığı için, geleneksel olarak şirket içi geliştirme uzmanlığı gerektirdi. Daha da önemlisi, bazı Linux uygulamaları bile birkaç megabayt RAM gerektirir. Ancak, paralel olarak, gömülü sistemler artık değişti. Günümüzde birçok gömülü sistem, grafik kullanıcı arabirimlerini (GUI) içerir ve gömülü web sunucularını dahil eder - geçmişte bunları tamamen gömülü sistemler olarak diskalifiye etmiş olabilir.

Bu işlevleri desteklemek ve Moore Yasası'ndan yardım almak için günümüzde, grafik işlem ünitelerini (GPU'lar), daha fazla RAM ve bellek yönetim birimlerini (MMU) birleştiren çok çekirdekli sistemlere (SoCs) dayalı gömülü sistemler bulmak yaygın değildir. Bu fenomenler, gömülü Linux'a tip, ulaşım, robot, dijital tabela ve iletişim gibi uygulamalarda bir hız vermiş ve biraz yol almıştır.

Fakat şimdi Linux'un gömülü hale geldiği noktaya varınca ki, bu biraz kolaylık sağladı, verim üzerinde duruyor, böylelikle mühendisler uygulamaları geliştirmek için daha fazla zaman harcıyor, altta yatan işletim sistemi ile daha az zaman harcıyor. Yine de açık kaynağın esnekliğini koruyorlar. Linux topluluğu ve donanım satıcıları, bu dengeyi Yocto Projesi gibi kendi başına (RYO) Linux dağıtım geliştirme ortamlarıyla zorlamaya devam ediyor.

---

<sup>1</sup><https://www.ibm.com/developerworks/library/l-yocto-linux/index.html>

“Dağıtım kavramı, paketlerin ön seçimidir; politikaların, uygulamaların ve çalışma biçimlerinin ön seçimi de denilebilir. DigiInternational’da ([www.digi.com](http://www.digi.com)) yazılım mühendisliği sorumlusu Alex Gonzalez, tüm bu şeylerin aslında dağıtımınız tarafından sabitlendiğini söylüyor. “Debian Linux gibi bir şeyle, ikili bir paket oluşturan bir dağıtım elde edersiniz ve kullanıcı olarak aldığınız tek şey, bir paket yönetim aracıyla yüklediğiniz bir dizi ikilidir.” der.

“Yocto ile, tüm bu paketleri kuruyorsunuz,” diye devam ediyor. "Bu ne demek, kaynak üzerinde tam kontrol sahibi olmanızdır. "

### **RYO Linux dağıtımlarına gömüldü**

Debian, Red Hat ve Yocto gibi dağıtımların tümü benzer açık kaynak bileşenlerine dayanıyor olsa da Yocto, yazılım mühendislerinin belirli bir yapıdaki tüm yapılandırma seçeneklerini seçmesine izin veriyor. Yocto dağılımindaki kaynak kodunun doğrudan kontrolü, geliştiricilerin yeni ikili dosyalar oluşturma, kaynak kodu alma, ardından ikili dağıtımın kendi versiyonunu değiştirme, yeniden paketleme, bütünlendirme ve sürdürme gibi yoğun bir süreçten geçmek zorunda olmadıkları anlamına da gelir. . Ayrıca, öncelikli olarak görülmeyen gömülü sistemlere uyarlanmış düzeltmeleri uygulamak için daha büyük Linux topluluğuna da güvenmek zorunda değiller.

Yocto'da özel dağıtımların olasılığı büyük ölçüde, yarı iletken üreticilerin Intel, NXP, Texas Instruments, Renesas, Xilinx, AMD ve diğerleri dahil olmak üzere projedeki doğrudan katılımı sayesinde yapılmıyor. Bu, Yocto yazılım eko-sistemine entegre edilen çok sayıda gömülü işlemci işlevselligi için koşullara yol açmış, bu da taşınabilirliği artırır ve kutudan daha yumuşak bir deneyim elde edilmesini sağlar.

Örneğin, Digi International, NXP i.MX6 işlemcisinin ConnectCore 6 modülündeki güvenli önyükleme özelliklerini etkinleştirmek için bir Yocto katmanı kullanır. Sonuç olarak, önyüklemeden JTAG güvenliğini sağlamak için eksiksiz bir güvenlik çerçevesi, uygulanması çok kolay olan yocto dosya sistemi olarak kullanılabilir.

### **Ubuntu**

#### **Ubuntu ile güvenli güncellemeler**

Ayrıca güvenlik hatları boyunca, uzak güncellemeler, Nesnelerin İnterneti (IoT) cihazları için önemli bir satış noktası haline gelmiştir. Uzak bellenim güncelleme özelliği, aygit üreticilerinin aygitların sahada konuşlandırılmışından sonra özellik ve yama güvenlik açıkları eklemesine izin verir, ancak işlemin kendisi kendi güvenlik sorunlarından oluşur. Örneğin, bir güncelleme kesintiye uğrarsa, ürün yazılımı dosyaları bozulabilir ve işletim sistemi yeniden başlatılamayabilir. Öte yandan, yeni bir firmware sürümü veya uygulaması güvenlik boşluklarını da tanıtabilir.

Bazı şirketler uzaktan güncelleme çözümleri sunarken ve diğerleri kendi geliştirmeyi tercih ederken, açık kaynak Ubuntu Çekirdek İşletim Sistemi bunların otomatik olarak gerçekleştirilemesine izin verir. Ubuntu Core'un eşsiz mimarisi, OS çekirdeğini, çekirdek Ubuntu işletim sistemini ve uygulamaları "enstantane" adı verilen izole paketlere böler. Bağlılı sistemlerde, enstantane güncellemeleri düzenli olarak kontrol edebilir ve bunları tek tek indirerek tüm sistem bileşenlerinin güncel olmasını sağlar ve en son yamaları içerir.

Otomatik uzak güncellemeler bazıları için rahatsız edici olsa da, Ubuntu Core güvenlik açısından iki kat avantaj sağlar. İlk olarak, işletim sistemi güncelleme işlemi sırasında Ubuntu Core, mevcut işletim sistemi değiştirilmeden önce doğrulanmış yeni çekirdek ve çekirdek işletim sistemi sürümlerinin tam kopyalarını indirir. Bu, bağlantının kesilmesi veya indirme işleminin kesintiye uğraması durumunda bile, sistemin sistemi bozmadan çalışmakta olan işletim sistemine geri dönmesini sağlar.

İkinci avantaj, bileşenlerin birbirinden yalıtlamasının, bir uygulamanın tehlikeye girmesi durumunda genel sistemin bütünlüğünü korumaya yardımcı olmasıdır.

"Ubuntu Core AppArmor'a öncülük etti ve bu da birçok uygulama hapsine izin veriyor" diyor Technologic Systems'de ([www.embeddedarm.com](http://www.embeddedarm.com)) gömülü mühendis Mark Featherston. "Bir uygulamanın kırılması durumunda, yalnızca tek bir uygulama olarak çalışan sistemle çok fazla iş yapmak çok zor olacak. Sistemin saldırısının etkililiğini sınırlar."

Technologic Systems'in TS-7970 tek kartlı bilgisayarı (SBC), zengin ağ desteği ve Endüstriyel Nesnelerin İnterneti (IIoT) ağ geçitleri gibi platformlar için güvenli yükseltme olanağı sağlayan Ubuntu Core'u çalıştırıyor

### **3.2.2 Android**

#### **Android Nedir?**

Android Linux kerneli üzerine inşa edilmiş bir işletim sistemidir. Google tarafından, telefonlar için bir işletim sistemi olarak üretilmiştir.

#### **Geliştirilmesi**

Symbian ve Windows Mobile'a rakip olarak üretilmiştir. İlk versiyonlarında dokunmatik ekran desteği yoktur. Kısa denilebilecek bir sürede en çok satan işletim sistemi olmuştur. Açık kaynak olarak, işletim sistemi kodu Google tarafından AOSP projesi kapsamında paylaşılmış, bu durum farklı üreticilerin bu kodu alıp üzerine istedikleri özellikleri ekleyebilmesine olanak sağlamıştır.

#### **Donanım**

Android işletim sistemi IOS'ta olduğu gibi belirli bir donanım kısıtlamasına sahip değildir. Bu durum yazılım güncellemelerini zorlaştırmıştır. Android, gerek açık kaynak, gerek özel sürücülerini destekler. Bu yolla Android farklı donanımlarda yaşanabilecek uyumsuzluk sorununu önlemiştir. Android RISC tabanlı işlemci mimarilerinde çalışmak üzere tasarlanmıştır. Genel olarak ARM işlemcileriyle çalışmaktadır. Android işletim sistemi akıllı cihazlardaki pek çok sensörü kullanabilecek biçimde tasarlanmıştır. Bunlara örnek olarak yakınlık sensörü, GPS sensörü, jiroskop, ivme ölçer ve ışık sensörü verilebilir. Android işletim sistemi, WiFi, Bluetooth ve NFC teknolojilerinin kullanımına olanak sağlar. Bunlara ek olarak, Android cihazlarla USB aracılıyla çalışan cihazların kullanımı mümkündür. Örnek olarak klavye, mouse ve joystick verilebilir.

#### **Yazılım**

Android işletim sistemi Linux kerneli üzerine kuruludur. Linux Foundation, bu sebepten dolayı Android'i bir Linux dağıtımı olarak görmektedir. Android yazılımı katmanlı mimariye sahiptir. İlk katmanda Linux kernel'i ve sürücüler bulunur. İkinci katmanda Android için özelleşmiş kütüphaneler(örnek olarak

libc, OpenGL, SQLite gösterilebilir) ve Android işleyişine olanak sağlayan bir Java sanal makinesi bulunmaktadır. Android'in GNU kütüphaneleri yerine özelleşmiş kütüphaneler kullanmasının nedeni, GNU kütüphanelerinin düşük frekanslı işlemciye sahip gömülü sistemlerde kullanılan kütüphaneler kadar verimli çalışmamasıdır. Bir üst katmanda, uygulama geliştirilmesine olanak sağlayan iskeletler(framework) bulunmaktadır. En üst katmanda ise kullanıcı tarafından kullanılacak uygulamalar yer alır. Android sistemi, sonradan eklenen bir özellik olarak, uygulamaların ve platformun güncellenmesi için Google Play Services modülüne sahiptir. Bu modül, uygulamaların işletim sistemi güncellemesi yapılmadan güncellenmesine olanak sağlar.

### Gömülü Sistemler Üzerindeki Etkisi

Android işletim sistemi gömülü sistem sayılabilen cihazlar üzerinde çalışmaktadır. Teknolojinin insan hayatıyla birleşmesine katkıda bulunmuştur. Buna ek olarak diğer gömülü sistemlerin kontrol ve takibi için kullanılmaktadır. Akıllı ev sistemleri ve sensör okuma gibi bir çok projede yönetici veya dinleyici araç olarak kullanılabilir. Bunlara imkan sağlayan ise donanım kısmında belirtilen bağlantı teknolojileridir.

#### 3.2.3 Windows

##### Windows Nedir?

Windows, Microsoft tarafından geliştirilmiş grafik arayüzüne sahip bir işletim sistemidir. Masaüstü bilgisayarlarda en yaygın sistem olmasına rağmen mobil platformdaki yarışı Android'in kazanması sonucu artık en çok satan işletim sistemi olmaktan çıkmıştır.

##### Geliştirilme Süreci

MS-DOS işletim sisteminin bir uzantısı olarak başlamıştır. İlk versiyonlarında bile yalnızca MS-DOS'a bir grafik kabuğu olmaktan fazlasını hedefleyerek kendi çalıştırılabilir dosya formatını ve cihaz sürücülerini içermiştir. İlk versiyonlarda bile birden fazla arayüz uygulamasının bir arada çalışmasına imkan sağlayan multitasking gözlemlenebilir. Windows ME versiyonun-

dan sonra Windows, DOS tabanlı bir işletim sistemi olmaktan çıkmıştır. Windows 9x, MS-DOS tabanına pek çok iyileştirme getirdi. Buna ek olarak dahili web tarayıcısının eklenmesi Windows 9x versiyonunda olmuştur. Windows NT'yle birlikte Windows işletim sisteminin ilk 32 bit versiyonu sunulmuş oldu. Windows XP, Windows NT mimarisini kullanarak, yeniden tasarlanmış arayüzü ve eski versiyonlara göre daha iyi performans sağlama-şıyla öne çıktı. Windows Vista arayüz, güvenlik açısından ve teknik açıdan pek çok geliştirmeye sahip olmasına rağmen uzun süren açılması ve perfor-mans sorunu dolayısıyla eleştirildi. Window işletim sistemi Windows 7,8,10 versiyonları olmak üzere kullanıcıya yönelik olarak devam etmiştir.

### **Gömülü Sistemler Üzerindeki Etkisi**

Windows işletim sistemi, kişisel bilgisayarlarda kullanılmak üzere geliştirilmesine rağmen, bir çok gömülü sistemde de kullanılmıştır. Örneğin Windows XP yayınlandığı zaman tabletlerde de kullanılabilir bir versiyona sahipti. Sunucularda kullanılmak üzere de Windows Server işletim sistemleri geliştirilmiştir. Bunlara ek olarak Windows Phone'larda kullanılan Windows Mobile versiyonu da Windows işletim sisteminin gömülü sistemlerde kullanımına örnektir. Bunun dışında Windows Embedded olarak sunulan oldukça az do-nanım gereksinimine sahip işletim sistemleri bulunmaktadır. Kaynak kodu kapalı olduğundan dolayı başka bir firma tarafından Linux kernelinin And-roid'e uyarlanması gibi bir durum Windows için geçerli değildir. Bu da ka-nımcı Windows'un özelleştirilmesini kısıtlayıp Windows'u gömülü sistemler için uygun bir platform olmaktan çıkarmıştır. Windows, gömülü sistemlerde kullanılacak işletim sistemleri sunmak dışında, oldukça yaygın olduğu için pek çok gömülü sistem geliştirme aracına hedef platform olmuştur.

# Bölüm 4

## Sensörler

### 4.1 Sensör Nedir? Ne İşe Yarar?

Arduino projelerinde ışık, sıcaklık, mesafe gibi fiziksel büyüklükleri elektrik sinyallerine dönüştürmek ve bu bilgileri işleyecek karar mekanizmaları kurabilmek için sensörleri kullanırız.

#### 4.1.1 Sensör Nasıl Çalışır? Çalışma Prensibi Nedir?

Sensör çeşitlerinin detayına girmeden önce bu sensörlerin temel olarak iki tipte var olduğunu bilmemiz gereklidir: Analog ve dijital.

Analog sensörler, algıladıkları fiziksel büyüklüğe orantılı olarak değişen bir akım veya gerilim çıktısı verirler. Bu tipte sensörleri dijital çalışan kontrol kartlarına bağlayabilmek için analog-dijital çeviriciler (ADC) kullanılır. Analog-dijital çeviriciler mikrokontrolcüler içerisinde de yer alacağı gibi (örn. Arduino analog giriş pinleri), sayısının veya hassasiyetinin yetmemesi durumunda harici olarak da bağlanabilirler. Popüler bir tek kart bilgisayar olan Raspberry Pi ise maalesef dahili olarak analog-dijital çeviriçiye sahip değildir. Dolayısıyla analog girişe ihtiyaç duyduğumuzda harici bir entegre kullanmamız gerekecektir.

Dijital sensörler ise genellikle I2C, SPI, OneWire vb bir haberleşme protokolü aracılığıyla bilgisayar (mikroişlemci) ile konuşurlar. Bunun yanı sıra, çoğu

analog sensör bir op-amp ile birlikte kullanılarak belirli bir seviye üzerinde lojik 1 (genellikle 5V veya 3.3V) çıkış verecek şekilde kullanılabilir. Böylelikle analog çıkışlı sensörler, Raspberry Pi gibi ADC'ye sahip olmayan kontrolcüler ile kullanılabilir.

## 4.2 Sensör Çeşitleri

### 4.2.1 Beslenme ihtiyaçlarına göre sensörler

**Aktif Sensörler:** Sinyallerini kendileri üretip bu sinyalin dış ortamla etkileşimlerini ölçen sensörlerdir.

**Pasif Sensörler:** Çevrelerinden aldıkları sinyalleri ölçen sensörlerdir.

### 4.2.2 Giriş büyüklüklerine göre sensörler

Sensörler, giriş büyüklüklerine göre altıya ayrırlar. Aşağıda bu sensör çeşitleri ve algılama özelliklerini inceleyebilirsiniz.

- Mekanik sensörler (Uzunluk, alan, miktar, kütlesel akış, kuvvet, tork, basınç, hız, ivme, pozisyon, ses dalga boyu ve yoğunluğu)
- Termal sensörler (İşı akışı ve sıcaklık)
- Elektriksel sensörler (Voltaj, akım, direnç, endüktans, kapasitans, di- elektrik katsayısı, polarizasyon, elektrik alanı, frekans)
- Manyetik sensörler (Alan yoğunluğu, akı yoğunluğu, manyetik moment, geçirgenlik)
- Işıma sensörleri (Yoğunluk, dalga boyu, polarizasyon, faz, yansıtma, gönderme)
- Kimyasal sensörler (Yoğunlaşma, içerik, oksidasyon/redaksiyon, reaksiyon hızı, pH miktarı)

Bu sensör çeşitleri kendi içlerinde de farklı şekillerde bulunurlar. Robotlarda ve sistemlerde en yaygın kullanılan sensörleri şu şekilde sıralayabiliyoruz.

- Mesafe sensörleri (Ultrasonik, PIR, Kapasitif, Endüktif, Kızılıötesi Optik...)

- Kuvvet/ağırlık/basınç sensörleri
- Eğim sensörleri (Flex, lineer/esnek Potansiyometre...)
- Manyetik sensörler (Hall effect, reed röle...)
- Sıcaklık/nem sensörleri (NTC,PTC...)
- Ses sensörleri (Dinamik/kapasitif/şeritli/kristal/karbon tozlu mikrofon)
- Işık/renk sensörleri (LDR, RGB, UV, fototransistör, fotodiyot...)

#### 4.2.3 Çıkış büyüklüklerine göre sensörler

Analog çıkışlara alternatif olan dijital çıkışlar bilgisayarlarla doğrudan iletişimde geçebilirler. Bu iletişimler kurulurken belli bazı protokoller kullanılır. Bunlardan seri iletişim protokollerine, aşağıda kısaca değinilmiştir.

**RS232C:** Bu protokol başta telefon veri iletişimini için tasarlanmıştır. Daha sonra birçok bilgisayar sistemi bunu kullanmaya başlamıştır. Sonuç olarak RS232 standart bir iletişim protokolü haline gelmiştir.

**RS4224A:** Bu protokol “differential ended” bir ara yüze sahiptir. Algılayıcı ve bilgisayar arasındaki iletişimde “twisted pair” kullanıldığından dış ekilerle etkileşimi azdır.

**RS485:** Standart 422A protokolü genişletilerek oluşturulmuş bir protokoldür. Bu protokol ile birlikte çalışabilen 32 adet alıcı-vericinin tek kabloyla veri iletişimini sağlanabilir. RS485 protokolü kablodaki iletişim problemlerini ortadan kaldırmaktadır.

### 4.3 Işık Sensörleri

Robot kontrolünde en yaygın kullanılan sensör ailesidir. LDR, fototransistör, fotodiyot, infared dedektörler gibi. Görünen veya görünmeyen ışık bilgilerini kullanan tüm sensörler bu kategoride yer almaktadır.

#### LDR Işık Sensörleri

LDR’ler(Light Dependent Resistor) ışık yoğunluğuna göre direnci değişen

devre elemanlarıdır. Kolay kullanımlı analog sensörlerdir. LDR'le (yaygın kullanımıyla fotoseller) genellikle kadmiyum sülfüidden (CdS) yapılmıştır.

### Fototransistörler-Fotodiyotlar

Bu ışık algılayan elektronik devre elemanlarını da ışık sensörü olarak kullanabiliriz. LDR'lere göre en büyük avantajları hızlı tepki süreleridir. Fototransistörler üzerine düşen ışık yoğunluğuna göre baz bacaklarına güç verirler; böylece ışıkla orantılı olarak emiter'den daha fazla akım çıkar. Fotodiyotlarda diyonterine düşen ışık belli bir değeri aşlığında diyonter iletme geçer. Fotodiyotlar da genellikle LED kılıfındadır.

## 4.4 Hareket Sensörleri

Hareket sensörü mekanik sensörler içerisinde incelenebilir. Hareket sensörlerinden en bilineni ise PIR sensörüdür.

**PIR Sensörü:** PIR ifadesinin açılımı, Passive InfraRed Sensor'dür. Sensörün çalışma mantığı ise, canlıların yaydığı pasif kızılötesi dalgaları algılamak ve bu kızılötesi dalgalarda meydana gelen değişime karşı bir değer üretmektir. Sensör, ortamda herhangi bir hareket algılandığında çıkışında 1 değerini üretir, hareketsiz durumlarda ise çıkışında 0 değerini üretir. Sensör üzerinde SX ve TX olmak üzere iki adet potansiyometre bulunmaktadır. SX potansiyometresi sensörün görme mesafesini 3 ile 5 metre arasında değiştirmektedir. TX potansiyometresi ise sensörün hareket algıladıkten sonra ne kadar süre çıkıştan lojik 1 vereceğini ayarlamaktadır.

## 4.5 Sıcaklık Sensörleri

**LM35 sıcaklık sensörü**, 0,5 derece hassasiyete sahip yarı iletken bir analog ısı sensöridür. Özellikle Arduino ile yapılmış sıcaklık uygulamalarında kullanılmaktadır. Isı ile motor kontrolü, lcd ekranı termometreye çevirme, sıcaklık kontrolü uygulamalarında sıkılıkla kullanılmaktadır. Teminin kolay olması ve ucuz bir sensör olması sebebiyle onunla yapılmış birçok uygulamayı bulabilirsiniz. LM35 sıcaklık sensörlerinin çıkış gerilimleri sıcaklık ile orantılı olarak değişir. Ölçüm aralığı -55 ile 150 derece arasındadır. 4-30 V aralığında bir besleme gerilimi ve 60 mikro A'den az akım ile 0,5 derece hassasiyet sağ-

lamaktadır. Her bir derece için çıkış volajı 10mV değerinde artar. Bu değer ile de, gerekli oran kurularak dış çevrenin sıcaklığı analogdan, dijitalde yani bizim sayısal olarak görebileceğimiz değerlere dönüştürülür.

## 4.6 Nem Sensörleri

### Toprak Nem Sensörü

Toprak Nemi Algılama Sensörü, toprağın içerisindeki nem miktarını veya ufak ölçekte bir sıvının seviyesini ölçmek için kullanabileceğiniz bir sensördür. Nem ölçer problr ölçüm yapılacak ortama batırılarak kullanılır. Toprağın veya içine batırılan sıvının meydana getirdiği dirençten dolayı, prob uçları arasında bir gerilim farkı oluşur. Bu gerilim farkının büyüklüğüne göre de nem miktarı ölçülebilir. Topraktaki nem oranı arttıkça iletkenliği de artmaktadır. Kart üzerinde yer alan trimpot sayesinde hassasiyet ayarı yapılmaktadır. Toprak Nemi Algılama Sensörü Arduino veya farklı mikrodenetleyiciler ile rahatlıkla kullanılabilir.

Çalışma Gerilimi: 3.3V-5V

Cıkış Gerilimi: 0-4.2V

Akım: 35 mA

Cıkış Türü: Dijital ve analog

Boyut: 30mm x 15mm

Ağırlık: 10g



## Bölüm 5

# Gömülü Sistemlerde Güvenlik

### 5.1 Giriş

Günümüzde farklı işler yapan gömülü sistem cihazları birbiri ile bağlantılı olmaya ve dahası bir ağa bağlanıp internete çıkabilme yetisini kazandı. Bu cihazları kullananlar neredeyse bütün internet uygulamalarını edinip, cihaz üzerinde çalıştırabiliyorlar ve bu cihazlara herkese açık olan internet ortamından veri gönderip alabiliyorlar. Bunun sonucunda bu verilerin güvenli bir şekilde iletilip alınması ve yetkisiz kullanıcılarca ulaşımının kısıtlanması bir gereksinim haline geliyor ve bu düzende kritik bir rol oynuyor.

Güvenli veri olarak nitelendirdiğimiz bu veriler farklı kategorilere ayrılıyor. Haliyle farklı kategoriler de farklı güvenlik önlemleri gerektiriyor. Bu kategoriyi ikiye ayıralım.

1. Kullanıcının kişisel verileri
2. Kullanıcı tarafından kısıtlanmış veriler

Kullanıcının kişisel verilerinin ifşa olmasını direkt olarak kullanıcının göreceği zarar olarak nitelendirilebiliriz. Örneğin kullanıcının internet bankacılık parolasının ele geçirilmesi. Kullanıcı tarafından kısıtlanmış verilerin güvenlik sorunları kullanıcının çok verilerin içeriğini ve aitliğini tehditiye sokmaktadır. Örneğin hakları alınmış bir multimedya dosyasının içeriğinin değiştirilip başkalarınca sunulması.

Bahsettiğimiz güvenli verinin sadece veri iletiminde değil, bu veri alındıktan sonra kişinin cihazındaki işlemlerinde de çeşitli güvenlik mekanizmaları bulunmalıdır. Bu cihazlardaki zafiyetlere örnek olarak, veriyi şifrelemeye ve deşifrelemeye yarayan gizli anahtara ulaşımın ne kadar kolay olup olmadığı verilebilir.

Gömülü sistemlerde de güvenli veri alışverişi ve bulunan gizli anahtarın yetkisiz kişilerce erişimi için çeşitli metotlar ve protokoller tanımlanmalıdır. Gömülü sistemlerdeki güvenlik ihtiyacı ikiye ayrılabilir.

1. Veri iletimindeki güvenlik
2. Cihaz içindeki güvenlik

## 5.2 Veri İletimindeki Güvenlik

Bir veri interneye çıktıgı andan itibaren birçok bilinmeyen noktadan geçer. Bu geçişsi sırasında bu veriyle ilgisi olmayan, ulaşım yetkisi olmayan bir kişi araya girse de, bu verinin içeriğini görememesi gereklidir. Bunun sağlanması kriptografik metotlarca şifreleme, deşifreleme, anahtar doğrulama, dijital imza ve dijital sertifika ile sağlanabilir.

### 5.2.1 Veri Şifreleme

Şifreleme veriyi bir anahtar kullanarak anlamsız bir veri dizisi haline getirmek ve elinde bu anahtar olmayan kimselerce verinin anlamsız kalmasını sağlaması işlemidir. Şifrelenen bu veriyi elinde sadece anahtar olan kişi (yani bu veriyi erişim yetkisi olan kişi) deşifreleyip içinde ne olduğunu görebilir. Bunun için açık kaynak olarak kullanılan DES, AES, 3DES algoritmaları kullanılabilir. Bu aşamada kullanılacak anahtarın, cihazlar arasında güvenli bir şekilde paylaşması önemli bir güvenlik sorunudur. Bu anahtarlar iletişimden önce cihaz içine gömülebilir ve değiştirilmek istediği zaman çevrimdışı bir şekilde değiştirilebilir. Diğer bir seçenek olarak çevrimiçi bir şekilde açık anahtar algoritmalarıyla (bir sonraki bölümde anlatılacak) anahtar değişimi yapılabilir.

### 5.2.2 Açık Anahtar Algoritması

Kendi ağınızda 100 tane cihaz olduğunu düşünen. Bu cihazlar arasındaki gizliliği sağlamak için kullanacağınız veri şifreleme metodu oldukça zor olacaktır. Bunun için açık anahtar algoritması kullanılacaktır.

Bu algoritmanın çalışması için her cihazın iki tane anahtara ihtiyacı olacaktır. Bu anahtarlardan biri açık, diğeri ise gizli anahtardır. Gizli anahtar hiçbir şekilde dışarıyla paylaşılmayacak olan anahtardır. Açık anahtar ise dışarı ile paylaşılabilen anahtar olacaktır. Bu algoritmanın zorluğu büyük kompozit çarpanlara ayrılmamasına dayanmaktadır. RSA algoritması olarak da bilinir. Kısaca şöyle anlatabiliriz:

Gizli anahtar ve açık anahtar ile yapılan şifrelemeler/deşifrelemeler birbirine dönüştürülebilir. Yani açık anahtarımla şifrelediğim veriyi, ben gizli anahtarımla deşifreleyebilirim ya da gizli anahtarımla şifrelediğim bir veriyi açık anahtarımla deşifreleyebilirim. Algoritmanın temel çalışma prensibi de budur.

Açık Anahtar:  $U_A$   
Gizli Anahtar:  $P_A$

Açık Anahtar:  $U_B$   
Gizli Anahtar:  $P_B$

Yukarıdaki senaryoya baktığımızda, A ve B cihazlarını görmekteyiz. Bu iki cihaz güvenli iletişim için ortak bir anahtarda karar kılmaya çalışıyorlar. A'nın gizli anahtarı  $P_A$ , B'nin gizli anahtarı ise  $P_B$ .  $U_A$  ve  $U_B$  sırasıyla A ve B'nin açık anahtarlarıdır. Burada Diffie-Hellman anahtar değişim algoritması Açık anahtar algoritmasıyla birleştirilip Diffie-Hellman algortimasının zayıfyeti olan kimlik doğrulama mekanizması oluşturulmuştur. Olay dizisi olarak:

1. C1 değişkeni A'nın gizli anahtarıyla imzalanmıştır. (Burada veri bütünlüğü ve kimlik doğrulama sağlanmıştır)
  - (a) İmzalanan değişken B'nin açık anahtarıyla şifrelenmiştir. (Burada veri gizliliği sağlanmıştır)
2. C2 değişkeni B'nin gizli anahtarıyla imzalanmıştır. (Burada veri bütünlüğü ve kimlik doğrulama sağlanmıştır)

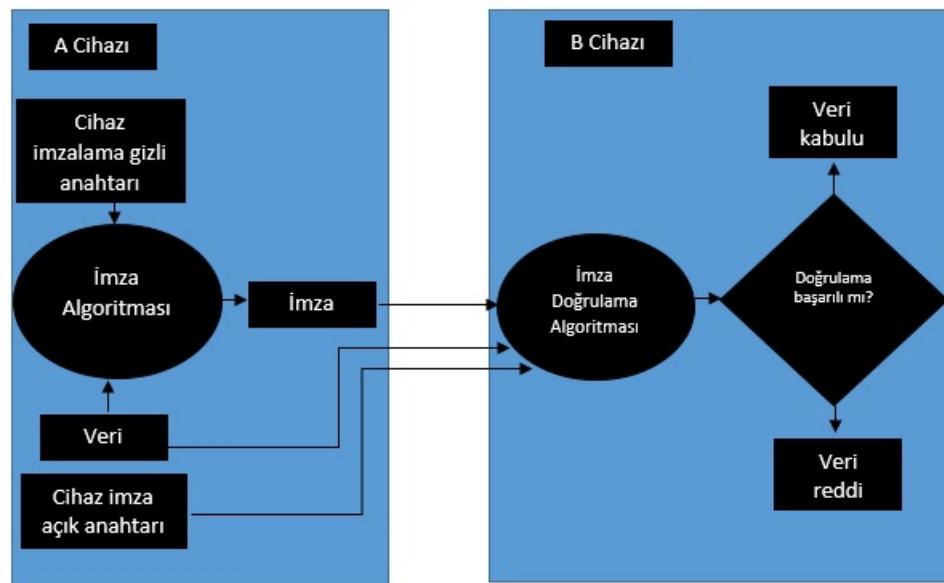
- (a) İmzalanan değişken A'nın açık anahtarıyla şifrelenmiştir. (Burada veri gizliliği sağlanmıştır)

Bunun ardından  $KA = \text{GenerateKey}(PA, UB, C2)$

$KB = \text{GenerateKey}(PB, UA, C1)$

Elde edilen KA ve KB,  $KA = KB = (PA, PB, C1, C2)$  şeklinde olacaktır. Bu anahtar değişiminde araya giren kişi hiçbir şekilde bir veri ele geçiremeyecektir.

### 5.2.3 Dijital İmza



Şekil 5.1: Yukarıda 1. ve 2. Maddelerde söylediğimiz imzalama olayı için söyle bir açıklama yapabiliriz.

### 5.2.4 Dijital Sertifika

Yukarıda anlattığımız gizli anahtar ve açık anahtar güvenlik protokolünü, veri şifrelemenin ağımızda 100 adet cihaz olduğunda zor olacağını söyleyerek anlatmıştık. Fakat şimdiki sorun cihazın internet ortamına çıktıgı zaman

doğuracak zorluklar olacak. Buradaki asıl sorun internette açık anahtar değişimini yapılrken, açık anahtarın acaba gerçek kullanıcıya ait olup olmadığıdır. Bunun için güvenilir bir merkezi yetki sistemi bulunmalıdır ki, bu sistemden elde edilen açık anahtarların istenilen kişinin açık anahtarına ait olduğundan emin olunabilinsin. Bu bahsettiğimiz yetki sistemi Certificate Authority (CA) olarak adlandırılır. CA tarafından imzalı bir açık anahtar olursa, buna güvenebilir ve bu anahtarı iletişimde kullanabiliriz. Bilinen sertifika formatlarından birisi X.509. Bunun içinde kullanıcının açık anahtarı dahil bir çok bilgi ve doğrulamayı sağlayacak bilgiler bulunur.

#### 5.2.5 Sertifika Hiyerarşisi

Dünya üzerinde birçok cihaz birbiriyle iletişim kurmakta. Bu yüzden sadece bir tane CA bulunması yetersiz olacaktır. Bu yüzden bir hiyerarşi oluşturulmuştur. En üstte yer alan Trusted Root CA, altında yer alan diğer güvenilir CA'lere verdiği izinlerle sertifika dağılımını sağlar.

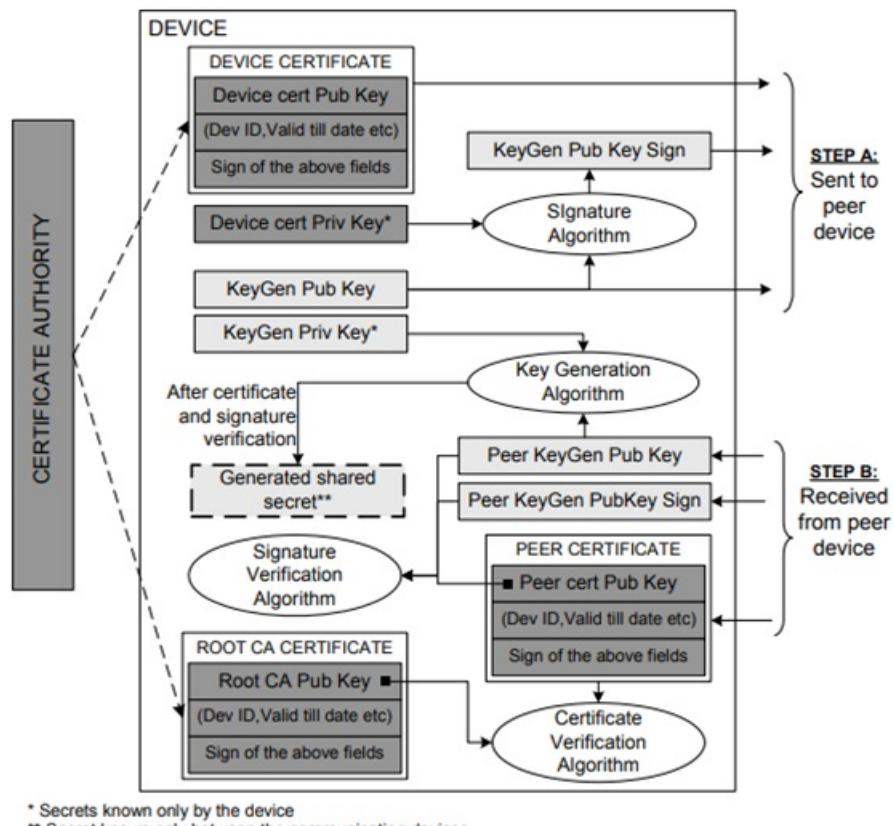
#### 5.2.6 Cihaz Üzerinde Anahtar Kabul Protokol Örneği

Burada cihaz internete bağlanmadan önce, CA'den sertifikasını alır. Veri imzalamak ve şifrelemek için elinde bulunan açık ve gizli anahtarları kullanır. Root sertifikası zaten cihaz içinde bulunmaktadır. İletişim kurulacak cihazın sertifikası bu Root sertifikasıyla doğrulanır ve geri kalan bütün bölüm 2.2 de anlatıldığı gibi devam eder.

### 5.3 Cihaz İçindeki Güvenlik

Veri iletimindeki güvenlik için 2. Bölümde anlatılan işlemlerin yapılacak işlem birimi kullanılacak olan cihazdır. Burada anlatılan gizli anahtarlar bu cihaz içinde tutulacak. Bu yüzden yazılım ve donanım seviyesindeki güvenlik protokollerı, yetkisiz kullanıcıların bu cihazda tutulan gizli anahtarlarla ulaşımını engellemek için oluşturulmalıdır. Örneğin cihaz içinde saklanacak Root CA sertifikasından bahsetmiştim. Eğer bu sertifikaya yetkisiz bir erişim ve değiştirme olursa bütün güvenlik önlemleri ve protokollerı boş gidecektir.

Cihazındaki güvenlik seviyesi, korunan içeriğin niteliğine bağlı olarak de-



Şekil 5.2: Cihaz Üzerinde Anahtar Kabul Protokol Örneği

sır. Cihaz güvenliği gereksinimi, konu kullanıcının kişisel dosyaları ve banka işlemleri gibi verileri olduğunda daha önemlidir. Ayrıca her cihazın kendine özel gizli bir anahtarının olması daha iyi olacaktır. Çünkü aynı ağdaki bir cihazın anahtarının ifşa olması diğer cihazların güvenliğini de tehdilkeye atacaktır.

### 5.3.1 Güvenli System On Chip

Bir metot olarak cihazın gizli anahtarını şifreleyip cihazın kalıcı belleğinde saklamak. Böylece cihazın bu kısmına ulaşan kimseler, gizli anahtarları ele geçirse bile, bu anahtarların şifreli hali olduğu için bir şey yapamayacaklardır.

Gizli anahtarı şifreleyen anahtarın nasıl saklanacağı ise yine güvenilir bir parti tarafından sağlanabilir. Bu da genelde cihazın üreticisidir. İkinci bir metot olarak gizli anahtarın güvenli ROM'da saklanması. Güvenli ROM'un donanım denetleyicisi(hardware controller), veriler ROM'dan alınmadan önce verileri deşifreler. Böylece yetkisiz erişimde gizli anahtarın elde edilme olasılığı oldukça düşük bir hal alır. Gizli anahtarları veya kriptografik ara değerleri tutan buffer, Güvenli SoC'un dahili RAM'ine gönderilir. Böylece gizli anahtarın Güvenli SoC dışındaki herhangi bir dış data bus'a verilmesi engellenir. Güvenli ROM uygulamalarındaki olası zafiyetler:

1. Güvenli ROM fiziksel olarak cihazdan çıkarılıp başka bir cihaza takıldıktan sonra, korunulan anahtarların çalışılır hale gelmesi.
2. Güvenli ROM ve RAM arasındaki bus'ların erişime açık halde olup anahtarın elde edilmesi.
3. Cihaz üzerinde çalışan yetkisiz bir uygulamanın kullandığı bir API ile Güvenli ROM'dan gizli anahtarı elde etmesi.

### 5.3.2 Dahili RAM ve Güvenli Process

Gizli anahtarın veya kriptografik ara değerlerin tutulduğu bufferin RAM'de saklandığı kısmı Güvenli Bellek Alanı olarak adlandırılalım. Cihaz işlemcisinde çalışan bütün processlerin de buraya erişimini kısıtlayalım. Sadece işletim sistemi tarafından görevlendirilen processler bu alanla ulaşabilsin.

Bunun sağlanması için işletim sistemi boot aşamasında MMU(memory management unit)'i yapılandırırken, Güvenli Bellek Alanına sadece Güvenli Process'lerin erişeceğine dair yetkilendirme yapmalıdır. Bu olurken, MMU yapılandırma kodunun yetkisiz erişimle değiştirilip Güvenli Bellek Alanına erişimin sağlanması da engellenmelidir. Bunun sağlanması sonraki bölüm olan Güvenli BootLoader(bir cihaz açıldığında işletim sistemini yükleyen bir program) ve kod imzalama kısmında anlatılacaktır. Burada işletim sisteminin herhangi yetkisiz bir process'e Güvenli Process olarak çalışma izni vermemesi gereklidir.

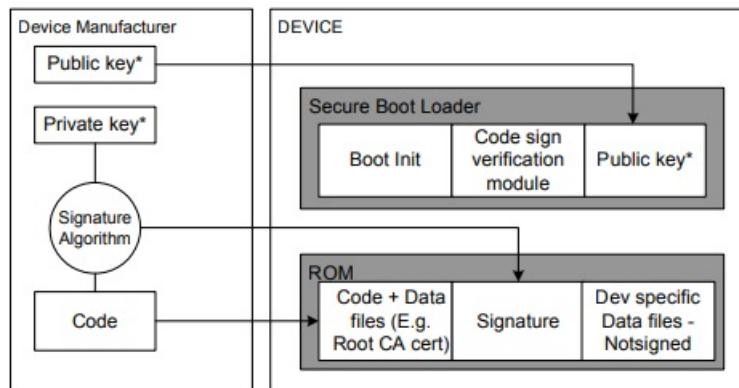
Güvenli Process tarafından gerçekleştirilem işlem genelde dışarı verilebilecek olan veridir. Açık anahtar veya şifrelenmiş veri. Bu verilerin başka bir cihaza iletilmesi veya üzerinde işlem yapılabilmesi için Güvenli Process'ten daha düşük yetkiye sahip processler kullanılacaktır. Örneğin: Güvenli Process bir

semaphore ile bir girdi almayı ve output üretmeyi beklemektedir. Girdinin bulunacağı buffer güvensiz bir bellek bölgesidir. Bunun sonucunda düşük yetkili bir process tarafından doldurulan buffer, Güvenli Process semaphore'una sinyal gönderip, çalışmasını sağlayabilir. Elde edilen çıktı yine düşük yetkili process tarafından okunup işlenebilir.

### 5.3.3 Güvenli BootLoader ve Kod İmzalama

Gizli anahtarlar, güvenilir bir donanım yapısı olsa bile kullanılan bir API ile ifşa edilebilir. Eğer cihaz üzerindeki firmware'ın değiştirilemeyeceğinden emin olunursa, yetkisiz bir kullanıcının kullandığı API ile de gizli anahtarın elde edilemeyeceğinden emin olunabilir. Çünkü firmware dahili RAM'in erişim izinlerinin belirlenmesinde rol alan yapılandırma kod bloklarını içerir. Bu yüzden firmware'ı değiştirmek üzere yapılan her işlemin etkisizleştirilmesi gereklidir. Güvenli BootLoader bunu sağlar.

İlk çalıştırılarda henüz firmware kodu belleğe yüklenmeden önce, Güvenli BootLoader, firmware'ın orijinal olup olmadığını kontrol eder.



Şekil 5.3: Güvenli BootLoader ve Kod İmzalama

Gizli ve açık anahtarlar cihaz üreticisi tarafından üretilir ve firmware kodunun imzalanıp doğrulanmasında kullanılır. Tabii bu sırada gizli anahtarın sadece üretici de olduğu unutulmamalıdır.

Güvenli BootLoader Güvenli SoC içindeki korunan ROM bölgesinde bulunur. Güvenli BootLoader'in yazmaya karşı korumalı bölgede tutulması, değiştiril-

mediginden emin olmak için gereklidir. Yukarıdaki şekilde de görüldüğü üzere Güvenli BootLoader; firmware kodunun imza doğrulama modülü, firmware kodunu doğrulamak için açık anahtara sahiptir.

Firmware kodu üreticinin gizli anahtarı kullanılarak imzalanmıştır. Boot aşamasında, Güvenli BootLoader yine üreticinin açık anahtarını kullanarak bu kodu doğrular. Üreticinin gizli anahtarının ifşa olması, cihazların da güvenliğini tehlikeye atacaktır.

Bazen firmwarelar güncellenemez (non-upgradable) olurlar. Bu durumda boot loader'ın güvenliğini oluşturmak daha kolay olacaktır. Çünkü firmware salt-okunur bölgeye yerleşip buradan çalışması sağlanacaktır ve yetkisiz değiştirilme söz konusu olmayacağından, tercih edilmez. Fakat güncellenemez firmware'lar birçok sınırlama getirdiğinden tercih edilmez.

Bazı cihaza özel dosyalar (şifrelenmiş gizli anahtarlar ve cihaz sertifikaları) değiştirildiğinde güvenli veri aktarımını tehlikeye düşürmeyeceği için, bu dosyaların imzalanması, herhangi bir yükseltmede aygitin çalışmamasında ek yükleme neden olacağından, tercih edilmez. Bu dosyalar cihazda imzalanmadan da saklanabilir.

#### 5.3.4 Şifreleme ve Deşifreleme Motoru

Birçok güvenli protokol tanımında bölüm 2.2 de anlatılan paylaşılan gizli anahtar, ana anahtar olarak kullanılır ve diğer başka alt anahtarlar da şifreleme ve deşifreleme işlemleri için kullanılırlar. Ana anahtar protocol olarak kullanıldığında, bu anahtarın ömrü günlerce veya aylarca olabilir. Bu veri iletim protokolünün tipine göre değişir. Fakat şifreleme/deşifrelemede kullanılacak alt anahtarların ömrü çok daha kısa olmalıdır. Saniyeler cinsinde. İşin güzel tarafı ise bu alt anahtarların güvenlik gereksinimleri, paylaşılan ana anahtar veya sertifikanın gizli anahtarları kadar kritik değil. Kullanılan alt anahtarlardan ana bir anahtar oluşturulamaz. Şifreleme ve deşifreleme motoru SoC'un içinde veya dışında yer alabilir. Bahsettiğimiz alt anahtarlar SoC içinde üretilir ve Güvenli SoC'un dış kısmında yer alan şifreleme deşifreleme motoruna gönderilir. Çoğu durumda bu motor SoC'un içinde yer alır. Çünkü anahtarları dışarıya vermek için kullanılacak bus'larla bu anahtarları ifşa etmemek için gereklidir.

### **5.3.5 Sistem Zamanı**

Bölüm 5.2.4'te bahsettiğimiz üzere dijital sertifikaların bir geçerlilik süresi olur. Bu periyot protokolden protokole değişir. SSL gibi protokollerde geçerlilik süresi bir kaç yıl veya onyılları bulabiliyor. Gömülü sistemlerde genelde bir arayüz olur ve kullanıcı bu arayızle sistem saatini ayarlar. Sertifikanın doğrulama aşamasında sistem zamanı oldukça önemlidir. Eğer sistem zamanı izinsiz/yetkisiz kişilerce değiştirilip, esas kullanıcı da bunu farketmezse süresi dolmuş bir sertifika cihaz tarafından kabul görülür ve kullanıcı bilgileri ifşa olabilir. Fakat çoğu durumda sistem zamanının değiştirilmesi sistem üzerinde kritik bir etki yaratmaz. Sertifikanın zaman aralığı senelerce olduğu için cihaz bu sertifikayı muhafaza edebilir. Fakat Root CA sertifikayı yenilediği zaman, cihazında bu sertifikaya ve aradaki diğer sertifikaları güncellemesi gerekektir.

## **5.4 Güvenli Veri İletimi için Kişisel Teknolojiler**

Kişiye özel bu teknolojiler cihazlar arasındaki güvenli veri iletişimini için bazı cihazlarda tanımlanmıştır. Bahsettiğimiz bu moduller 2. Bölümde anlatığımız şifreleme/deşifreleme, anahtar değişimi ve doğrulaması, dijital sertifika ve dijital imzanın bazlarını veya tamamını kapsayabilir.

Kişiye özel bu teknolojiler cihazlarda tanımlı olup cihazların iletişim kuracağı gizliliği saklar ve veri iletişimi için ek bir güvenlik sağlar. Bu yüzünden ki üretici bu teknolojiyi gizli tutmak zorundadır. Üretici, teknolojinin açığa çıkmaması için güvenlik yazılım modülünü cihaz içinde bir şekilde saklamak için metotlar bulmalıdır. Kişiye özel bu teknolojinin kodları (binary) genelde açık metin yani olduğu şekilde cihazda saklanır. Fakat böyle olması durumunda birileri bu koda erişip tersine mühendislikle güvenli yazılım modülünü çözüzebilir ve sonucunda üreticinin teknolojisi de ifşa olmuş olur. Bu yüzden bu binary kod cihaz üzerinde erişilemeyecek şekilde ya da sadece üreticinin bildiği bir anahtar ile şifrelenmiş şekilde SoC'de tutulmalıdır. Tabi yine bu senaryo ekstra bir performans gerektirecektir. Çünkü eğer bu kod şifrelenmiş şekilde Güvenli SoC'da tutulursa cihaz boot olurken bu kodun deşifrelenmesini sağlayan bir işlem gerekli olacaktır.

Kişiye özel bu teknolojiler cihazlarda ikiye ayrılabilir.

## *5.5. CİHAZ ÜRETİMİ SIRASINDA SERTİFİKA VE ANAHTAR İDARESİ*

1. Cihaza özel
2. Bütün cihazlar için geçerli

Cihaza özel durumlarda güvenli veri transferi sadece aynı üreticiye ait olan cihazlar arasında gerçekleşecektir. Örnek olarak Apple’ın iPod’u verilebilir. Apple kişiye özel teknolojisini kendi ürettiği cihazlar veya uygulamalarda uygular.

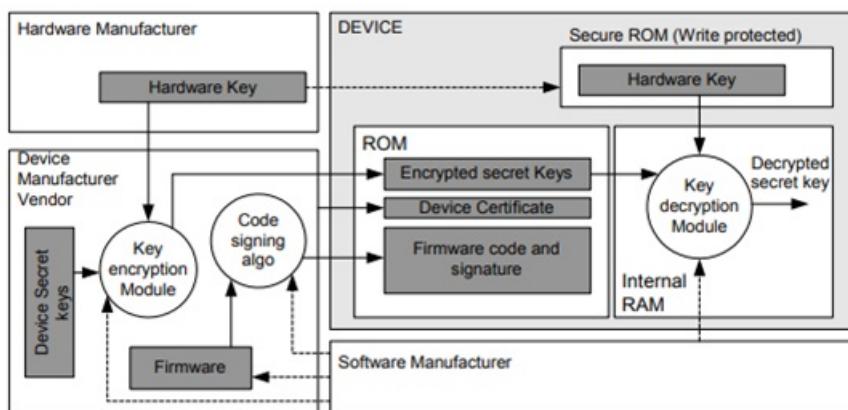
Bütün cihazlar için geçerli bir teknolojide ise bu durum değişir. Çünkü farklı üreticilerin cihazlarıyla da güvenli iletişim kurması için cihazın ekstra güvenlik protokollerine ihtiyacı olmasının gerekliliği var.

### **5.5 Cihaz Üretimi Sırasında Sertifika ve Anahtar İdaresi**

Çoğu zaman gömülü bir sistemin farklı yazılım ya da donanım bileşenleri farklı satıcılarından temin edilebilir. Donanım satıcısı donanım bileşenini sağlarken, bu bileşen için gerekli driver’ın sağlanması da yazılım satıcısı tarafından sağlanır. Genelde üreticiler gelir elde etmek için yazılım ve donanım kısımları birleştirip ürünü öyle oluştururlar. Her cihaz için bulunacak gizli anahtar cihaz üretim halindeyken cihaza entegre edilmelidir. Üreticiler genelde üretikleri bu gizli anahtarları korumak için başka üreticilerle paylaşmaktan kaçınırlar. Donanım ve yazılım satıcılarının destekleriyle, üretici gizli anahtarı cihaz içinde, satıcılarla vermeden saklayabilir. Gizli anahtarın idaresinde kullanılabilecek iki metod:

1. Donanım satıcıları donanımı salt okunur Güvenli ROM olarak sağlar, bu ROM önceden rastgele bir sayı kullanılarak programlanmış olup bir veya birden fazla cihaz için kullanılabilir. Kullanılan bu rastgele sayı, cihazın gizli anahtarını şifrelemek için kullanılan cihaz ana anahtarı olarak kullanılabilir. Donanım anahtarını kullanacak deşifreleme metodu firmware’da bulunmalıdır.
2. Donanım satıcısı, üreticiye programlanabilir Güvenli ROM sağlayabilir, böylece üretici cihazın gizli anahtarıyla cihazı programlayabilir.

Her cihazın cihaz sertifikası farklı olur. Çünkü her cihaz için şifrelenmiş gizli anahtar farklıdır. Cihaz üreticisi cihaz sertifikasını veya şifrelenmiş ci-



Şekil 5.4: Cihaz Üretimi Sırasında Sertifika ve Anahtar İdaresi

haz anahtarını yazılım satıcısının belirttiği gibi ROM'daki konumuna yükler. Böylece sertifikayı kullanacak olan yazılım bileşeni bu sertifikayı işlemek için ilgili konumdan alır ve işler.

Root CA sertifikası iletişim kurmak için aynı protokolü kullanan her cihaz için özgündür. Bu yüzden Root CA sertifikası 3. Bölümde anlatıldığı gibi code-signed bir şekilde firmware kodunda bulunur.

## 5.6 Sonuç

Cihaz içinde saklanan gizli anahtarlar gibi saklanan güvenli verileri korumak için mevcut güvenlik önlemleri henüz kusuruz değildir. Güvenli anahtara fiziksel bir saldırı yapıldığında gizli anahtarın sıfırlanması için bir donanım mekanizması gerekebilir. Gizli anahtar ve diğer verileri korumak için cihazda uygulanan güvenlik önlemleri ne kadar fazla olursa, cihaz o kadar maliyetli olacaktır. Böylece, cihazda uygulanan donanım güvenlik önlemleri, uygulama maliyeti ile korunan verinin maliyeti arasında bir ticarettir. Gizli anahtarları korumak ve cihaz içindeki verileri korumak için uygun maliyetli ancak güvenilir bir yönteme ulaşmak, özellikle kopya korumalı dijital içeriklerin içerik sağlayıcısı için güvenlik gerektiren içeriğin sahibine bir nimet olacaktır.

## Bölüm 6

# Örnek Geliştirme Ortamları

### 6.1 Arduino

#### What is Arduino?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable,

3D printing, and embedded environments.

### **Why Arduino?**

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package.

### **History**

The Arduino project started at the Interaction Design Institute Ivrea (IDII) in Ivrea, Italy. At that time, the students used a BASIC Stamp microcontroller at a cost of \$100, a considerable expense for many students. In 2003 Hernando Barragán created the development platform Wiring as a Master's thesis project at IDII, under the supervision of Massimo Banzi and Casey Reas, who are known for work on the Processing language. The project goal was to create simple, low cost tools for creating digital projects by non-engineers. The Wiring platform consisted of a printed circuit board (PCB) with an ATmega168 microcontroller, an IDE based on Processing and library functions to easily program the microcontroller.[4] In 2003, Massimo Banzi, with David Mellis, another IDII student, and David Cuartielles, added support for the cheaper ATmega8 microcontroller to Wiring. But instead of continuing the work on Wiring, they forked the project and renamed it Arduino.

The initial Arduino core team consisted of Massimo Banzi, David Cuartielles,

Tom Igoe, Gianluca Martino, and David Mellis, but Barragán was not invited to participate. Following the completion of the Wiring platform, lighter and less expensive versions were distributed in the open-source community. Adafruit Industries, a New York City supplier of Arduino boards, parts, and assemblies, estimated in mid-2011 that over 300,000 official Arduinos had been commercially produced, and in 2013 that 700,000 official boards were in users' hands. In October 2016, Federico Musto, Arduino's former CEO, secured a 50% ownership of the company. In April 2017, Wired reported that Musto had "fabricated his academic record.... On his company's website, personal LinkedIn accounts, and even on Italian business documents, Musto was until recently listed as holding a PhD from the Massachusetts Institute of Technology. In some cases, his biography also claimed an MBA from New York University." Wired reported that neither University had any record of Musto's attendance, and Musto later admitted in an interview with Wired that he had never earned those degrees.

Around that same time, Massimo Banzi announced that the Arduino Foundation would be "a new beginning for Arduino." But a year later, the Foundation still hasn't been established, and the state of the project remains unclear. The controversy surrounding Musto continued when, in July 2017, he reportedly pulled many Open source licenses, schematics, and code from the Arduino website, prompting scrutiny and outcry. In October 2017, Arduino announced its partnership with ARM Holdings (ARM). The announcement said, in part, "ARM recognized independence as a core value of Arduino ... without any lock-in with the ARM architecture." Arduino intends to continue to work with all technology vendors and architectures

## IDE

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Ardu-

ino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

### **Writing Sketches**

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

### **Sketchbook**

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

### **Tabs, Multiple Files, and Compilation**

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

### **Uploading**

Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241 (for an Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or /dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx , /dev/ttyUSBx or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the Sketch menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets)

## 6.2 TI LaunchPad

### What is LaunchPad?

LaunchPads are microcontroller development kits from Texas Instruments. They come in a variety of flavors to address your various project needs. Microcontrollers are little chips that enable intelligence in everyday objects. These tiny computers can be integrated into anything and can enable inexpensive, anytime, anywhere computing! From your electric toothbrush to your vehicle's key fob, a microcontroller is the 'brain' programmed by code. A microcontroller is also referred to as an MCU or uC.

But that's not all. Microcontrollers enable new inventions every day. They

can be the building blocks of your next big idea. Connect them with wireless modules, lights, motors, and more to create new or better solutions for today's problems.

Texas Instruments has a range of 16-bit and 32-bit MCUs, each with a unique LaunchPad™ Development Kit to help users evaluate the various functionalities of that device.

BoosterPacks are available from Texas Instruments, from third parties, and from the community. They include functions such as capacitive touch, wireless communication, sensor readings, LED lighting control, and more. BoosterPacks are available in 20- and 40-pin variants, and multiple BoosterPacks can plug into your LaunchPad to enhance the functionality of your design.

Energia is more than a Soviet-era rocket! Energia is an open-source electronics prototyping platform started by Robert Wessels based on the Wiring framework. Energia is built by the community and includes an integrated development environment (IDE) based on Processing that allows users to get started easily. LaunchPad and Energia together enable interactive objects to control lights, motors and other outputs. Let Energia take your projects to new heights with its minimal learning curve and simple functions.

CCS Cloud is TI's web-based IDE that enables developers to get up and running quickly without the need for a lengthy install process. It allows you to create, edit, build, and run CCS and Energia projects on your LaunchPad. CCS Cloud is an efficient way to get started on your LaunchPad. However, just like Energia, some advanced debugging features are not supported. Enter CCS, TI's Eclipse-based IDE on the desktop. It has powerful features such as true debugging for pausing your code, reading register values, and more. Energia and CCS Cloud projects can be imported into CCS to gain access to its additional features, so transitioning is easy. CCS is an extremely powerful development environment that can help developers extract the most features and functionality out of the microcontroller.

### **Energia IDE**

Energia is an intuitive IDE that is based on the popular & easy-to-use processing ([processing.org](http://processing.org)) IDE. In addition to being a simple IDE, Energia is also supported by a robust framework of intuitive APIs that is based on Wiring. The Energia IDE also supports in-line C, assembly & Driver Library based code. Energia is supported in Windows, Mac & Linux.

Energia is an open-source electronics prototyping platform started by Robert Wessels in January of 2012 with the goal to bring the Wiring and Arduino framework to the Texas Instruments MSP430 based LaunchPad. The Energia IDE is cross platform and supported on Mac OS, Windows, and Linux. Energia uses the mspgcc compiler by Peter Bigot and is based on the Wiring and Arduino framework. Energia includes an integrated development environment (IDE) that is based on Processing. Energia is also a portable framework/abstraction layer that can be used in other popular IDEs. Utilize a web browser based environment with CCS Cloud at dev.ti.com. Community maintained Energia plug-ins and integrations are available for Xcode, Visual Studio, and Code Composer Studio.

The foundation of Energia and Arduino is the Wiring framework that is developed by Hernando Barragan. The framework is thoughtfully created with designers and artists in mind to encourage a community where both beginners and experts from around the world share ideas, knowledge and their collective experience. The Energia team adopts the philosophy of learning by doing and strives to make it easy to work directly with the hardware. Professional engineers, entrepreneurs, makers, and students can all benefit from the ease of use Energia brings to the microcontroller.

Energia started out to bring the Wiring and Arduino framework to the Texas Instruments MSP430 LaunchPad. Texas Instruments offers a MSP430, MSP432, TM4C, C2000, and CC3200 LaunchPad. The LaunchPad is a low-cost microcontroller board that is made by Texas Instruments. The latest release of Energia supports the majority of the LaunchPad product offerings. Additional community kits from RedBearLab are also supported. Together with Energia, LaunchPad can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. LaunchPad projects can be stand-alone (only run on the Target Board, i.e. your LaunchPad), or they can communicate with software running on your computer (Host PC). You can also add wireless modules to enable communication over various types of RF including Wi-Fi, NFC, Bluetooth, Zigbee, cellular, and more.

## 6.3 Cypress

### Cypress Softune IDE

SOFTUNE is an integrated development environment that was designed to respond to the various demands of program developers and pursues ease of use. Program development requires repeated editing, make/build, and debugging operations. SOFTUNE is designed to perform such repetitive processes smoothly and efficiently.

### **Structure of SOFTUNE**

#### **Workbench**

- Unification of manager section and debugger section.
- Errors that are found can be fixed on the spot, and the result can be debugged immediately. Three types of debugger functions are supported that need to be used at various different stages of the development cycle.
  - Simulator
  - Emulator
  - Monitor debugger

#### **Language tools**

- **C/C++ compiler:** When a source file represented in C or C++ language is described, the C/C++ compiler generates an assembler source file which is expressed in assembly language.
- **Assembler:** This Assembler assembles the source programs described with an assembly language.
- **Linkage kit:** The linkage kit consists of a linker that is used to connect object modules, a librarian that is used to control object modules and a converter that converts to object type in order to write information on a ROM.

#### **Support CPU**

- FR family
- F2MC-16 family
- F2MC-8FX family

### PSoC Creator IDE

PSoC Creator is an Integrated Design Environment (IDE) that enables concurrent hardware and firmware editing, compiling and debugging of PSoC and FM0+ systems. Applications are created using schematic capture and over 150 pre-verified, production-ready peripheral Components. Components are analog and digital peripherals represented by a symbol that users drag-and-drop into their designs and configure to suit a broad array of application requirements. Each Component in the rich mixed-signal Cypress Component Catalog is configured with a customizer dialog and includes a full set of dynamically generated API libraries. After configuring all the peripherals, firmware can be written, compiled, and debugged within PSoC Creator or exported to leading 3rd party IDEs such as IAR Embedded Workbench®, Arm® Microcontroller Development Kit, and Eclipse™.

PSoC and FM0+ Systems are energy optimized beyond a typical MCU because PSoC Creator optimizes designs to enable only the required functionality. Users can even create custom Components using state machine diagrams or Verilog to further optimize hardware and energy usage. PSoC Creator is a free Windows-based IDE that includes:

- Hardware design with complete schematic capture and easy-to-use wiring tool
- Over 150 pre-verified, production-ready Components
- Full communications library including I2C, USB, UART, SPI, CAN, LIN, and Bluetooth Low Energy
- Digital peripherals with powerful graphical configuration tools
- Extensive analog signal chain support with amplifiers, filters, ADC and DAC
- Dynamically generated API libraries
- Free C source code compiler with no code size limitations
- Integrated source editor with inline diagnostics, auto-complete and code snippets
- Built-in debugger



# Bölüm 7

# Projeler

## 7.1 Veritabanı, Web Servisleri ve Gömülü Sistemleri ile oluşturulmuş Kart Okuma Sistemi

### Amaç

Gömülü sistemlerle bir kapının giriş çıkışını kişilere özel yetkilendiren ve bunun kaydını tutan bir sistem

### İşlem

İlk olarak hangi malzemeleri ve cihazları kullanılacağına karar verildi. Bunlar:

- Arduino Uno R3
- Raspberry Pi Zero Wireless
- ESP8266
- RFID
- Servo Motor
- Direnç(ler)
- LED

Bu sistemin nasıl çalıştığını gerçekten anlamak için MySQL, PHP, HTTP ve LAN'ın nasıl çalıştığı hakkında bilgi sahibi olmak önerilir.

Sistemde ESP8266 Arduino'dan aldığı kart verilerini HTTP \_ GET metodu şeklinde Raspberry Pi üzerinde çalışan sunucuya yönlendirir. Sunucu üzerinde çalışan web servisi gelen GET metodu içindeki veriyi, veritabanında çeşitli işlemlerden geçirdikten sonra kişinin yetkisi olup olmadığını kaydını tutar ve buna göre servo motorun çalışması sağlanır. Ayrıca bu kayıtlar basit bir HTML tablosunda yetkili kişiye sunulur.

### Gelelim kodlama kısmına...

RFID modülü kart okumak için kullanılacağından hazır kütüphanesi ve bu module SPI iletişimini sağlayacağı için SPI kütüphanesi programa dahil edilmiştir. Yerel ağda prototip sağlandığı için benim DHCP'm tarafından Raspberry Pi'ye atanın IP adresi 192.168.43.45'tir.

```

1 #include <RFID.h>
2 #include <SPI.h>
3 #include <Servo.h>
4 #define ssid "wifi"
5 #define password "wifiPassword"
6 #define IP "192.168.43.45"

12 void setup() {
13   Serial.begin(115200);
14   pinMode(7, OUTPUT);
15   digitalWrite(7, LOW);
16   sg.attach(3);
17   SPI.begin();
18   lrt720.init();
19   Serial.println("AT");
20   delay(3000);
21
22   if(Serial.find("OK")){
23     Serial.println("AT+CWMODE=1");
24     delay(2000);
25     String connectionCommand = String("AT+CWJAP=\"" + ssid + "\",\"", + password + "\"\"");
26     Serial.println(connectionCommand);
27     delay(5000);
28   }
29   digitalWrite(7, HIGH);
30 }
```

ESP8266 115200 Baudrate'te iletişim kurduğu için Arduino'nun UART Seri

haberleşme baudrate’ı de 115200 olarak ayarlandı. ESP8266, AT komutlarıyla kontrol edilebilir. 19. Satırda, bir ‘AT’ komutu yollanıyor. Eğer buna ‘OK’ cevabı dönerse anlaşılıyor ki, ESP8266 ile haberleşme başlamıştır. ESP8266 burada bizim yerel ağımızda çalışacağı için ‘AT+CWMODE=1’, ESP’yi Station modu’na getiriyor. Yani artık ESP kablosuz ağlara bağlanabilecektir. 25. Satırda ESP’nin bağlanması ve en üstte ‘define’ ile belirttiğimiz ağa bağlanması sağlanıyor. 7. pin LED’i kontrol etmektedir. Kart okuma süresi 5 saniye olarak ayarlanmıştır (değiştirelibir bir değerdir). Yani sistem tekrar kart okunmaya hazır olduğunda LED yanacak ve kullanıcıya bunu bildirecek.

Mikroişleyicimiz olan Arduino burada herhangi bir kart okunmasını bekliyor. RFID’nin tanımlayabileceği bir kart okutulduğunda LED sönecek ve işlem başlayacaktır. 37 ve 46. Satır arasındaki işlemler kartın ID’sini elde etmek için RFID kütüphanesinin kullanım şeklidir. Bu işlem sonucunda kartın ID’si elde edilmiştir. Elde edilen kart sunucusyla iletişim kurması için ‘sendHttpRequest()’ metoduna gönderilmiştir.

```

33 void loop() {
34   flag = false;
35   String card_id = "";
36   sg.write(0);
37   digitalWrite(7, HIGH);
38   if(lrt720.isCard()){
39     digitalWrite(7, LOW);
40     if(lrt720.readCardSerial()){
41       card_id = String(lrt720.serNum[0]) + ".";
42       card_id += String(lrt720.serNum[1]) + ".";
43       card_id += String(lrt720.serNum[2]) + ".";
44       card_id += String(lrt720.serNum[3]) + ".";
45       card_id += String(lrt720.serNum[4]);
46     }
47     Serial.println(card_id);
48     Serial.println(String("Flag is") + flag);
49     if(flag){
50       sg.write(180);
51     }
52
53     sendHttpRequest(card_id);
54     delay(5000);
55   }
56   lrt720.halt();
57 }
```

```

59 void sendHttpRequest(String card_id){
60     Serial.println(String("AT+CIPSTART=\"TCP\",\"") + IP + "\",80");
61     delay(1000);
62
63     if(Serial.find("Error")){
64         Serial.println("AT+CIPSTART Error");
65         return;
66     }
67     String commandToSend = "GET /embedded/request.php?rfid=" + card_id + "\r\n\r\n";
68     Serial.print("AT+CIPSEND=");
69     Serial.println(commandToSend.length() + 2);
70     delay(3000);
71
72     if(Serial.find(">")){
73         Serial.print(commandToSend);
74         Serial.print("\r\n\r\n");
75         String response = "";
76         char res;
77         while(Serial.available() > 0){
78             res = Serial.read();
79             response += res;
80         }
81         Serial.println(response);
82     }
83     else{
84         Serial.println("AT+CIPCLOSE");
85     }
86 }

```

60. satırda ESP8266'nın ‘AT+CIPSTART’ komutuyla ‘IP’ diye belirtilen Raspberry Pi üzerinde çalışan HTTP (Port 80) (Apache) sunucuya TCP iletişimini başlatması sağlanıyor. 63 – 65. Satırlarda gerekli hata kontrolü yapılıyor. Raspberry Pi sunucu üzerinde (başka işlemlerde olduğu için) Bu projeye için oluşturulan ..//embedded/ dizini altında işlem yapıldı. Burada “..//request.php?rfid=” kısmına yazılan kart ID’sidir. AT+CIPSEND komutuyla da bu metot ESP8266 üzerinden sunucuya GET методу olarak gönderiliyor. Ardından AT+CIPCLOSE komutuyla oluşturulan TCP bağlantısı kapatılıyor.

### **Veritabanı**

Veritabanı Raspberry Pi üzerinde çalışan MySQL'dir. Bu proje için iki tablo kullanılmıştır. E/R diagramı:

Access -> User

Access	
access_date	
access_time	
card_id (FK)	
access_id (PK)	

User	
user_id (PK)	
name	
surname	
card_id	
access_right	

PK: primary key

FK: foreign key

User tablosu:

```
MariaDB [embedded]> select * from User;
+-----+-----+-----+-----+
| name | surname | card_id | access_right |
+-----+-----+-----+-----+
| Harun | Artuner | 16.178.244.162.244 | 1 |
| Serhat | Yilmaz | 222.154.124.96.88 | 1 |
| Embedded | Systems | 71.124.37.217.199 | 0 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Access tablosu:

```
MariaDB [embedded]> select * from Access;
+-----+-----+-----+-----+
| access_date | access_time | card_id | access_id |
+-----+-----+-----+-----+
| 09/05/2018 | 10.03.35 | 222.154.124.96.88 | 3 |
| 09/05/2018 | 10.04.13 | 16.178.244.162.244 | 4 |
| 09/05/2018 | 10.04.26 | 71.124.37.217.199 | 6 |
| 09/05/2018 | 10.04.55 | 16.178.244.162.244 | 7 |
| 09/05/2018 | 10.05.47 | 16.178.244.162.244 | 9 |
| 09/05/2018 | 10.06.20 | 71.124.37.217.199 | 11 |
| 09/05/2018 | 10.07.25 | 222.154.124.96.88 | 15 |
| 09/05/2018 | 10.12.19 | 222.154.124.96.88 | 17 |
| 25/05/2018 | 03.21.41 | 16.178.244.162.244 | 18 |
| 25/05/2018 | 03.22.26 | 16.178.244.162.244 | 20 |
| 25/05/2018 | 03.22.49 | 71.124.37.217.199 | 22 |
| 25/05/2018 | 03.24.21 | 16.178.244.162.244 | 24 |
+-----+-----+-----+-----+
12 rows in set (0.01 sec)
```

Daha anlaşılır olması için: Gelen GET istekleri bu şekilde veritabanında tu-

```
MariaDB [embedded]> select * from User natural join Access;
+-----+-----+-----+-----+-----+-----+-----+
| card_id | name   | surname | access_right | access_date | access_time | access_id |
+-----+-----+-----+-----+-----+-----+-----+
| 16.178.244.162.244 | Harun   | Artuner |          1 | 09/05/2018 | 10.04.13    |      4 |
| 16.178.244.162.244 | Harun   | Artuner |          1 | 09/05/2018 | 10.04.55    |      7 |
| 16.178.244.162.244 | Harun   | Artuner |          1 | 09/05/2018 | 10.05.47    |      9 |
| 16.178.244.162.244 | Harun   | Artuner |          1 | 25/05/2018 | 03.21.41    |     18 |
| 16.178.244.162.244 | Harun   | Artuner |          1 | 25/05/2018 | 03.22.26    |     20 |
| 16.178.244.162.244 | Harun   | Artuner |          1 | 25/05/2018 | 03.24.21    |     24 |
| 222.154.124.96.88  | Serhat  | Yilmaz  |          1 | 09/05/2018 | 10.03.35    |      3 |
| 222.154.124.96.88  | Serhat  | Yilmaz  |          1 | 09/05/2018 | 10.07.25    |     15 |
| 222.154.124.96.88  | Serhat  | Yilmaz  |          1 | 09/05/2018 | 10.12.19    |     17 |
| 71.124.37.217.199  | Embedded | Systems |          0 | 09/05/2018 | 10.04.26    |      6 |
| 71.124.37.217.199  | Embedded | Systems |          0 | 09/05/2018 | 10.06.20    |     11 |
| 71.124.37.217.199  | Embedded | Systems |          0 | 25/05/2018 | 03.22.49    |     22 |
+-----+-----+-----+-----+-----+-----+-----+
12 rows in set (0.00 sec)
```

tulmaktadır.

```
pi@raspberrypi:/var/www/html/embedded/includes % cat test.txt
:222.154.124.96.88:09/05/2018:10.03.35
Serhat Yilmaz:16.178.244.162.244:09/05/2018:10.04.13
Harun Artuner:71.124.37.217.199:09/05/2018:10.04.26
Embedded Systems:16.178.244.162.244:09/05/2018:10.04.55
Harun Artuner:16.178.244.162.244:09/05/2018:10.05.47
Harun Artuner:71.124.37.217.199:09/05/2018:10.06.20
Embedded Systems:230.45.172.24.119:09/05/2018:10.06.46
:222.154.124.96.88:09/05/2018:10.07.25
Serhat Yilmaz:222.154.124.96.88:09/05/2018:10.12.19
Serhat Yilmaz:16.178.244.162.244:125/05/2018:03.21.41
Harun Artuner:16.178.244.162.244:125/05/2018:03.22.26
Harun Artuner:71.124.37.217.199:25/05/2018:03.22.49
Embedded Systems:16.178.244.162.244:25/05/2018:03.24.21
pi@raspberrypi:/var/www/html/embedded/includes % cat write.php
<?php
fclose($fh);
if(!empty($rfid)){
  $myfile = fopen(__DIR__."/test.txt", "a") or die("Unable to open file!");
  fwrite($myfile, $name.":". $rfid. ":" . $date. ":" . $hour. "\n");
  fclose($myfile);
}
mysqli_query($conn, "insert into Access (access_date, access_time, card_id) values ('".$date."','".$hour."','".$rfid."')");
if($isExist == TRUE){
  $message = exec("/var/www/html/embedded/includes/deneme.py 2>1");
  print_r($message);
}

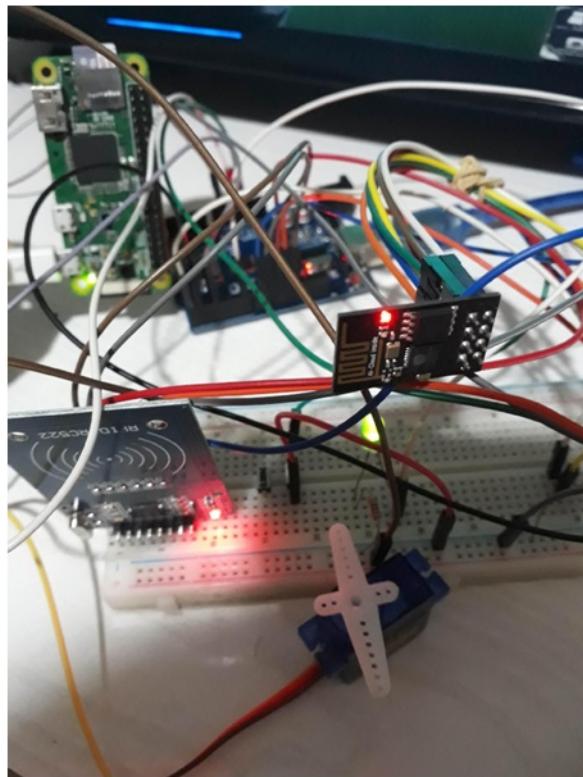
?>
pi@raspberrypi:/var/www/html/embedded/includes %
```

Gelen istekler veritabanına gönderilmeden önce sunucu üzerinde bir metin belgesi üzerine kayıt ediliyor. Ardından veritabanıyla işlem yapılıyor. Gerekli kontroller sonucu bir script çalıştırılıp yetkili veya yetkisiz olma durumları Arduino'ya iletiliyor. HTML kaydı şeklinde en son hali veritabanından alınan verilerce görüntüleniyor.

## 7.1. KART OKUMA SİSTEMİ

179

Name	Date	Hour	Access
Serhat Yılmaz	09/05/2018	10:03:35	Access Granted
Harun Arıtuner	09/05/2018	10:04:13	Access Granted
Embedded Systems	09/05/2018	10:04:26	Access Denied
Harun Arıtuner	09/05/2018	10:04:55	Access Granted
Harun Arıtuner	09/05/2018	10:05:47	Access Granted
Embedded Systems	09/05/2018	10:06:20	Access Denied
Unknown	09/05/2018	10:06:46	Access Denied
Serhat Yılmaz	09/05/2018	10:07:25	Access Granted
Serhat Yılmaz	09/05/2018	10:12:19	Access Granted
Harun Arıtuner	25/05/2018	03:21:41	Access Granted
Harun Arıtuner	25/05/2018	03:22:26	Access Granted
Embedded Systems	25/05/2018	03:22:49	Access Denied
Harun Arıtuner	25/05/2018	03:24:21	Access Granted



## 7.2 Gömülü Sistemleri ile oluşturulan Ev Otomasyonu Sistemi

### Amaç

Gömülü sistemleri içinde barındıran uzaktan erişimle kontrol edilebilecek prototip bir ev otomasyon sistemi tasarlamak.

### İşlem

İlk olarak hangi malzemeleri ve cihazları kullanacağına karar verildi. Bunlar:

- Arduino Uno R3
- ESP8266
- 5110 LCD
- Servo Motor
- LM35
- Direnç(ler)
- LED

Bu sistemin nasıl çalıştığını gerçekten anlamak için HTTP ve TCP protokolünün, LAN'ın nasıl çalıştığı hakkında bilgi sahibi olmak önerilir.

Sistemde ESP8266'nın kablosuz ağlara bağlanabilme özelliği ve üzerinde basit bir sunucu kurarak bu sunucu üzerinde mikrokontrolcüyü kontrol etme işlemi söz konusudur. ESP8266, Arduino tarafından programlanacak ve kullanıcı girdileriyle tekrar Arduino tarafından kontrol edilen LED ve Servo Motor yönlendirilecektir. 5110 LCD üzerinden ise sistemin ne kadardır aktif olduğu bir sayıcıyla kullanıcıya sunulacaktır.

### Gelelim kodlama kısmına..

Servo Motor ve LCD5110 kullanılacağından ilk olarak `<Servo.h>` ve `<LCD5110_Basic.h>` programa dahil edilmiştir. Bütün işlem yerel ağ üzerinden kontrol edileceğinden (prototip olarak), kullanılacak yerel ağın SSID ve parolası başta belirtilmiştir. ESP8266'dan girdi alındıktan sonra kontrol edilecek GPIO pinleri burada tanımlanmıştır. İleri kısımlarda daha net görülecektir. 'extern' olarak

```
#include <Servo.h>
#include <LCD5110_Basic.h>

#define ssid "wifi"
#define pass "wifiPassword"

#define light 7
#define curtain 5
```

tanımlanan değişkenler LCD5110 kütüphanesinden geleceği için yansıtılacak yazı tiplerinin biçimini belirtmiştir. Alt kısmında da myGLCD adında Arduino'nun hangi pinlerini kullanacağına dair hazır kütüphaneden bir nesne oluşturulmuştur. Ardından da servo motor kontrolü için bir nesne oluşturulmuştur.

```
extern uint8_t SmallFont[];
extern uint8_t MediumNumbers[];

LCD5110 myGLCD(8 ,9, 10, 11, 12);
Servo sg;
```

setup() kısmında hangi pinlerin giriş mi çıkış mı olarak kullanılacağı belirlenmiştir. LCD5110 aktifleştirilmiştir. 30. satırdan itibaren: ESP8266 115200 Baudrate'te iletişim kurduğu için Arduino'nun UART Seri haberleşme baudrate'i de 115200 olarak ayarlandı. ESP8266, AT komutlarıyla kontrol edilebilir. 31. Satırda, bir 'AT' komutu yollanıyor. Eğer buna 'OK' cevabı dönerse anlaşılıyor ki, ESP8266 ile haberleşme başlamıştır. ESP8266 burada bizim yerel ağımızda çalışacağı için 'AT+CWMODE=1', ESP'yi Station modu'na getiriyor. Yani artık ESP kablosuz ağlara bağlanabilecektir. 36. Satırda ESP'nin bağlanması ve en üstte 'define' ile belirttiğimiz ağa bağlanması sağlanıyor. 41. Satırda 'AT+CIPMUX=1' komutuyla, birden fazla TCP bağlantısının mümkün olacağı söyleniyor. 43. Satırda ESP, sunucu olarak kullanılacağından server modu 1 olarak ayarlanıyor ve port numarasına 80 verilerek kurulacak sunucunun HTTP protokolüyle çalışacağı bildiriliyor.

Buradaki işlemlerde LCD5110'da gösterilecek sayıcı işlemler, gerçekleştiriliyor. LM35 sıcaklık sensörünün çıkış bacagında okuduğumuz her 10 mV,  $1^{\circ}\text{C}$  sıcaklığa denk geldiğinden;  $10 / 1,0742$  bize yaklaşık olarak 9,31 değerini veriyor. Yani bu demek oluyor ki, analog girişten ölçüduğumuz değeri 9,31 ile çarparsak, elde ettiğimiz sonuç sensörün verdiği çıkış geriliminin santigrat

```

19 void setup() {
20     analogReference(INTERNAL);
21
22     pinMode(light, OUTPUT);
23     pinMode(curtain, OUTPUT);
24
25     sg.attach(3);
26     sg.write(0);
27
28     myGLCD.InitLCD();
29
30     Serial.begin(115200);
31     Serial.println("AT");
32     delay(3000);
33     if(Serial.find("OK")){
34         Serial.println("AT+CWMODE=1");
35         delay(2000);
36         String connectionCommand = String("AT+CWJAP=\"") + ssid + "\", \"\" + pass + "\"";
37         Serial.println(connectionCommand);
38         delay(2000);
39     }
40
41     Serial.println("AT+CIPMUX=1");
42     delay(2000);
43     Serial.println("AT+CIPSERVER=1,80");
44     delay(1000);
45 }
```

cinsinden karşılığını elde ediyoruz.<sup>1</sup>

Eğer yerellığımızdaki herhangi bir tarayıcıdan, ESP'ye atanan IP adresini girersek ESP8266 seri portundan içinde ‘IPD’ içeren bir cevap yolluyor. Arduino'da bunu görürse oluşturulan ESP'ye html değişkeni içindeki metini tekrar ESP'ye yolluyor. Fakat bunu ‘AT+CIPSEND’ komutuyla gönderiyor. Bu komutla aslında bir TCP verisi yollanıyor. Bu veri ESP tarafından sunucu tarafına aktarılıp, tarayıcı ile açıldığı zaman anlamlı bir HTML düzeni oluşturmazı sağlanıyor. ‘AT+CIPCLOSE’ komutuna geçmeden önce: process() metodunu incelemekte fayda var. Bu metot yine Arduino'nun seri portunu kontrol ediyor. İletişim asenkron olduğu için kullanıcının ne zaman istek yapacağı bilinmiyor. Bu yüzden bu metot mikrokontrolcude sürekli çalışıyor. Eğer seri portta bir metin oluştuysa bu metin derlenip bir değişken(response) içinde saklanıyor. Gelen isteğin neyle ilgili olduğunun anlaşılması için if-else kontrolünden geçiyor. Ardından AT+CIPCLOSE komutuyla

---

<sup>1</sup><http://maker.robotistan.com/>

```

void loop() {

    //*****
    myLCD.setFont(MediumNumbers);
    if(minute < 10){
        myLCD.print("0", 0, 16);
        myLCD.printNumI(minute, 12, 16);
    }
    else{
        myLCD.printNumI(minute, 0, 16);
    }

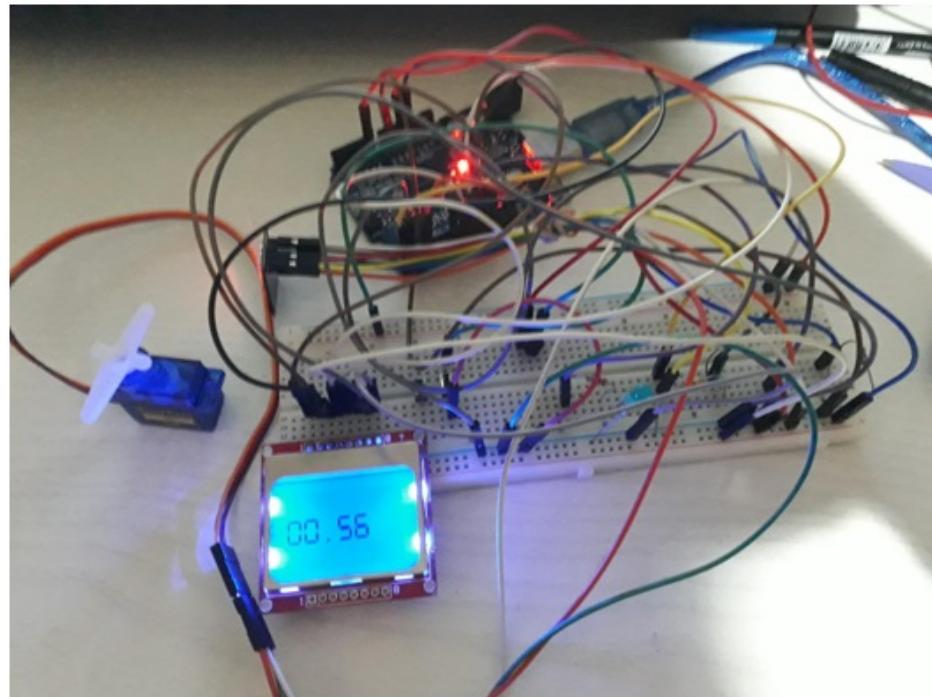
    myLCD.print(".", 24, 16);
    if(sec < 10){
        myLCD.print("0", 36, 16);
        myLCD.printNumI(sec++, 48, 16);
    }
    else{
        myLCD.printNumI(sec++, 36, 16);
    }
    if(sec == 60){
        sec = 0;
        minute++;
    }

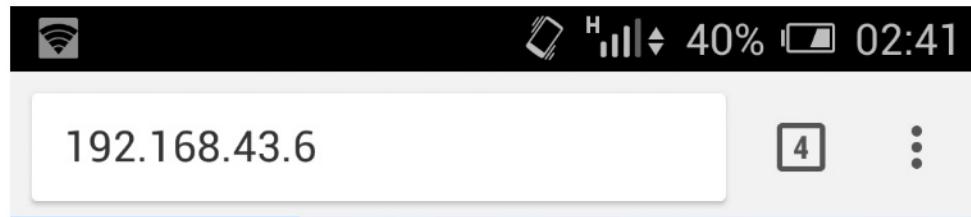
    if(Serial.available() > 0 ){
        if(Serial.find("+IPD")){
            String html = "<html><head><title>Remote Home Automation</title></head><body><div><h1>HOME AUTOMATION SYSTEM</h1></div><div><ul><a href='?light=off'><button type='button'>OFF</button></a></li><li><h2>Temperature</h2><p id='temp'>" ;
            html += "</p></li>";
            html += "</div><script type='text/javascript'>var x =";
            html += tempreatureC;
            html += "document.getElementById('temp').innerHTML = x;</script>";
            html += "</body></html>";
            String cipSend = "AT+CIPSEND=0,";
            cipSend += html.length();
            Serial.println(cipSend);
            delay(500);
            Serial.println(html);
            delay(500);
            process();
            Serial.println("AT+CIPCLOSE=0");
        }
    }
}

```

kurulan TCP bağlantısı kapatılıyor.

```
100 void process(){
101     String response = "";
102     char character;
103     while(Serial.available() > 0){
104         character = Serial.read();
105         response += character;
106     }
107
108     if(response.indexOf(":GET /?light=on") > 1){
109         digitalWrite(light, HIGH);
110     }
111     if(response.indexOf(":GET /?light=off") > 1){
112         digitalWrite(light, LOW);
113     }
114
115     if(response.indexOf(":GET /?curtain=on") > 1){
116         sg.write(180);
117     }
118
119     if(response.indexOf(":GET /?curtain=off") > 1){
120         sg.write(0);
121     }
}
```





## HOME AUTOMATION SYSTEM

- Lights

ON

OFF

- Curtains

ON

OFF

- Temperature

19.65

### 7.3 Seri haberleşme (I2C)

İki adet Arduino Uno kullanılacaktır. Arduino Uno'lardan birisi master birisi de slave görevinde bulunacaktır.

Master görevindeki Arduino, slave görevindeki Arduino'ya bağlı LED'leri kontrol edecek ve slave görevindeki Arduino'dan veri alacak. Slave görevindeki Arduino, master görevindeki Arduino'dan gelen veriyi yorumlayacak. Gelen veriye göre de LED'leri kontrol edecek ve diğer Arduino'ya veri yollayacak.

Bu uygulamayı yapmak için ihtiyacımız olan malzemeler:

- 2 x Arduino
- 2 x 4.7K ohm direnç
- 1 x LED
- 1 x 220 ohm direnç
- 1 x Breadboard

**Proje Videosu:**

<https://drive.google.com/file/d/1TEaw7GZAQDQ1Q2mXeRqIzUBF7j609u4s/view?usp=sharing>

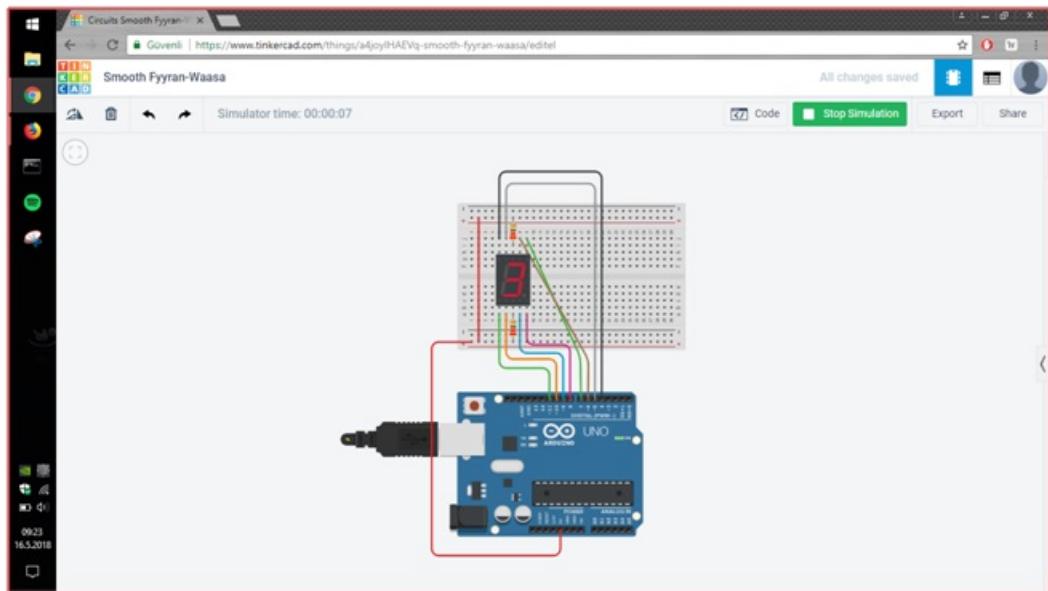
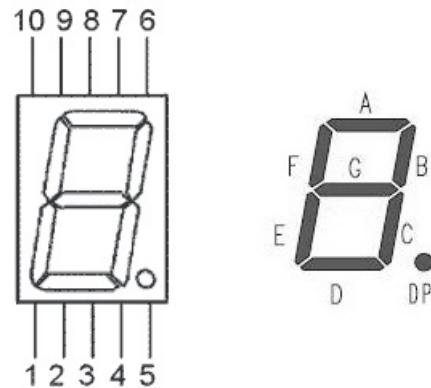
### 7.4 7 Segmentli Gösterge

**Proje Videosu:**

[https://drive.google.com/open?id=1xb5c9rt\\_oFE-3THhuIN7oJZabsxYF2Dx](https://drive.google.com/open?id=1xb5c9rt_oFE-3THhuIN7oJZabsxYF2Dx)

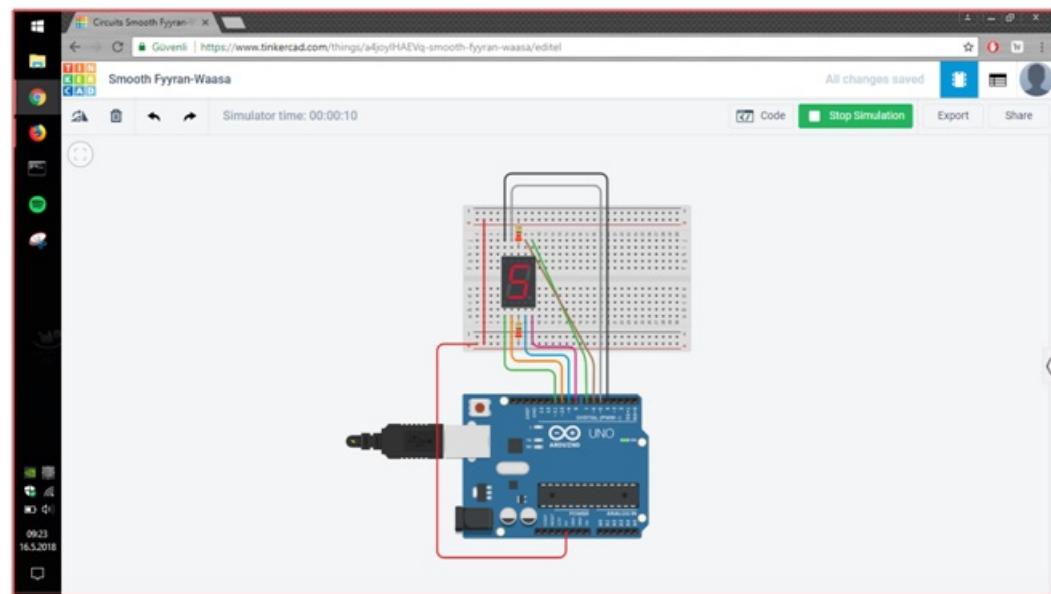
7 segmentli göstergeler ortak anotlu veya ortak katotlu olmak üzere ikiye ayrılıyorlar. Ortak anot dedığımız ortak pinlerin VCC, ortak katot dedığımız ise ortak pinlerin GND olmasıdır. Biz bu projemizde ortak anot kullandık ve segmentte 2 sn gecikme ile 0-9 aralığındaki sayıları yazdirdık (Şekil 1, Şekil 2, Şekil 3).

**Devre Şeması:**

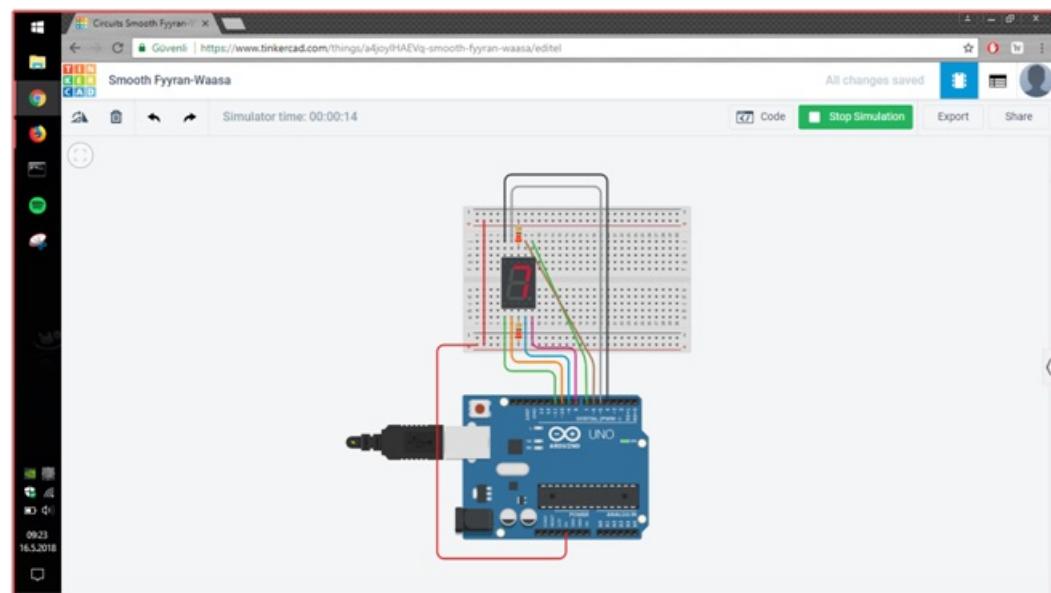


Şekil 7.1: Şekil 1

**Kaynak Kodu:**



Şekil 7.2: Şekil 2



Şekil 7.3: Şekil 3

The image shows two separate windows of the Arduino IDE. Both windows have a title bar '7segmentDisplay | Arduino 1.8.5' and a menu bar with File, Edit, Sketch, Tools, and Help. The top window displays the following code:

```
1 int a = 6;
2 int b = 7;
3 int c = 9;
4 int d = 10;
5 int e = 11;
6 int f = 5;
7 int g = 4;
8
9 void setup()
10 {
11   pinMode(a, OUTPUT);
12   pinMode(b, OUTPUT);
13   pinMode(c, OUTPUT);
14   pinMode(d, OUTPUT);
15   pinMode(e, OUTPUT);
16   pinMode(f, OUTPUT);
17   pinMode(g, OUTPUT);
18 }
19
20 void loop()
```

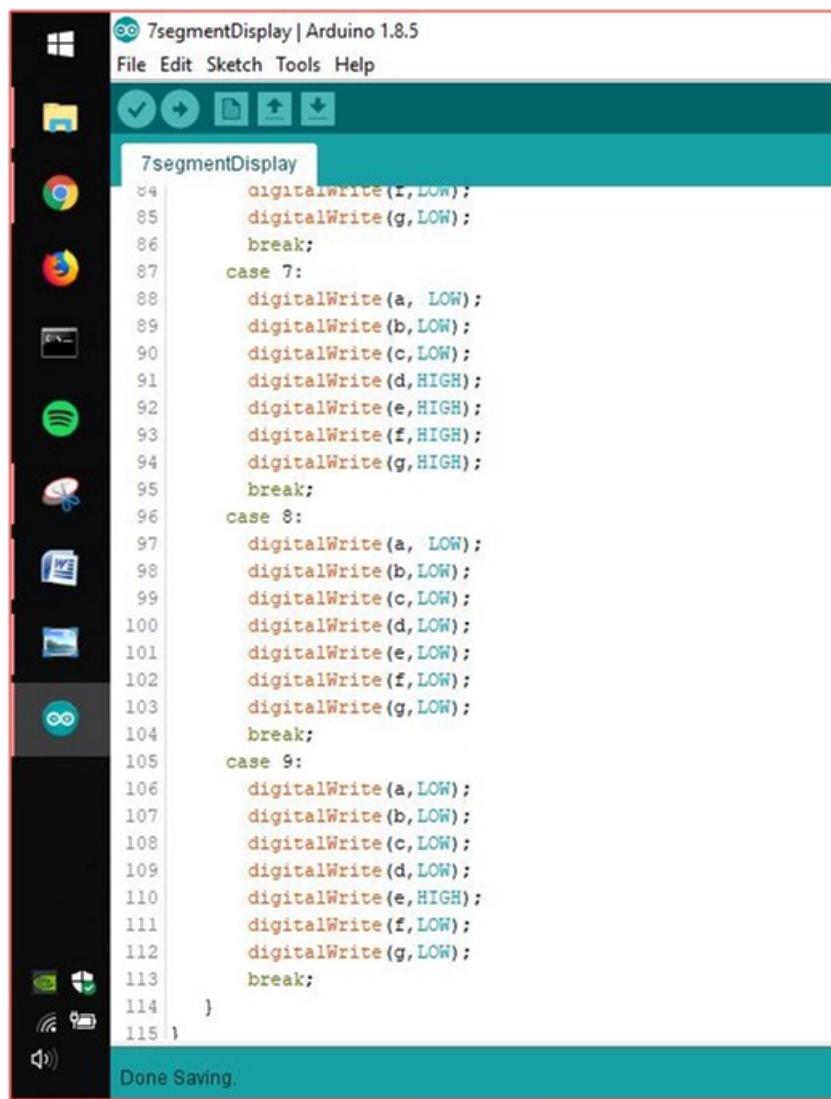
The bottom window displays the following code:

```
17
20 void print_number(int number)
21 {
22   switch(number)
23   {
24     case 0 :
25       digitalWrite(a,LOW);
26       digitalWrite(b,LOW);
27       digitalWrite(c,LOW);
28       digitalWrite(d,LOW);
29       digitalWrite(e,LOW);
30       digitalWrite(f,LOW);
31       digitalWrite(g,HIGH);
32       break;
33     case 1:
34       digitalWrite(a,HIGH);
35       digitalWrite(b,LOW);
36       digitalWrite(c,LOW);
37       digitalWrite(d,HIGH);
38       digitalWrite(e,HIGH);
39       digitalWrite(f,HIGH);
40       digitalWrite(g,HIGH);
41       break;
42     case 2:
43       digitalWrite(a, LOW);
44       digitalWrite(b, LOW);
45       digitalWrite(c, HIGH);
46       digitalWrite(d, LOW);
47       digitalWrite(e, LOW);
48       digitalWrite(f, HIGH);
49       digitalWrite(g, LOW);
50       break;
```

The screenshot shows the Arduino IDE interface. The title bar reads "7segmentDisplay | Arduino 1.8.5". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for upload, download, and other functions. The main area displays the code for a 7-segment display. The code uses a switch statement to control seven pins (a through g) based on the digit value. The sketch is saved, as indicated by the message "Done Saving." at the bottom.

```
7segmentDisplay | Arduino 1.8.5
File Edit Sketch Tools Help
7segmentDisplay

51     case 3:
52         digitalWrite(a, LOW);
53         digitalWrite(b,LOW);
54         digitalWrite(c,LOW);
55         digitalWrite(d,LOW);
56         digitalWrite(e,HIGH);
57         digitalWrite(f,HIGH);
58         digitalWrite(g,LOW);
59         break;
60     case 4:
61         digitalWrite(a, HIGH);
62         digitalWrite(b,LOW);
63         digitalWrite(c,LOW);
64         digitalWrite(d,HIGH);
65         digitalWrite(e,HIGH);
66         digitalWrite(f,LOW);
67         digitalWrite(g,LOW);
68         break;
69     case 5:
70         digitalWrite(a, LOW);
71         digitalWrite(b,HIGH);
72         digitalWrite(c,LOW);
73         digitalWrite(d,LOW);
74         digitalWrite(e,HIGH);
75         digitalWrite(f,LOW);
76         digitalWrite(g,LOW);
77         break;
78     case 6:
79         digitalWrite(a, LOW);
80         digitalWrite(b,HIGH);
81         digitalWrite(c,LOW);
```



The screenshot shows the Arduino IDE interface. The title bar reads "7segmentDisplay | Arduino 1.8.5". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for file operations. The main area displays the code for a 7-segment display. The code uses digital pins r through g to control segments. It includes cases for digits 0 through 9, with break statements at the end of each case. The code ends with a closing brace. The status bar at the bottom says "Done Saving".

```
7segmentDisplay
File Edit Sketch Tools Help
7segmentDisplay
84     digitalWrite(r,LOW);
85     digitalWrite(g,LOW);
86     break;
87     case 7:
88         digitalWrite(a, LOW);
89         digitalWrite(b,LOW);
90         digitalWrite(c,LOW);
91         digitalWrite(d,HIGH);
92         digitalWrite(e,HIGH);
93         digitalWrite(f,HIGH);
94         digitalWrite(g,HIGH);
95         break;
96     case 8:
97         digitalWrite(a, LOW);
98         digitalWrite(b,LOW);
99         digitalWrite(c,LOW);
100        digitalWrite(d,LOW);
101        digitalWrite(e,LOW);
102        digitalWrite(f,LOW);
103        digitalWrite(g,LOW);
104        break;
105    case 9:
106        digitalWrite(a,LOW);
107        digitalWrite(b,LOW);
108        digitalWrite(c,LOW);
109        digitalWrite(d,LOW);
110        digitalWrite(e,HIGH);
111        digitalWrite(f,LOW);
112        digitalWrite(g,LOW);
113        break;
114    }
115 }
```

```
116
117 void loop(){
118   for(int i=0; i<=9;i++){
119     print_number(i);
120     delay(2000);
121   }
122 }
```

Done Saving.

## 7.5 Engelden Kaçan Robot

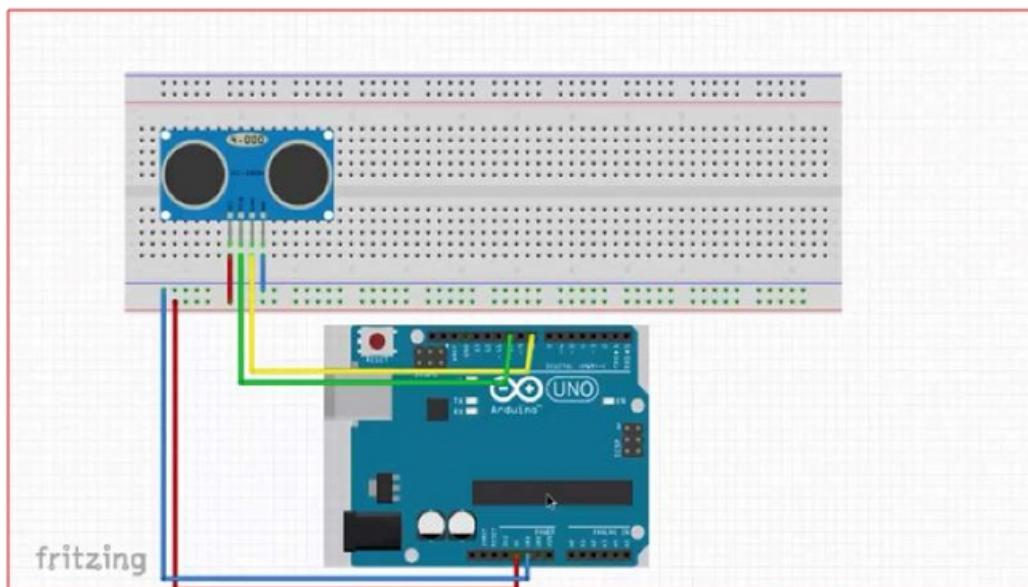
### Proje Videosu:

<https://drive.google.com/open?id=1-k208XYiDhhJcsjLTY9rNN9xvUGc1aig>

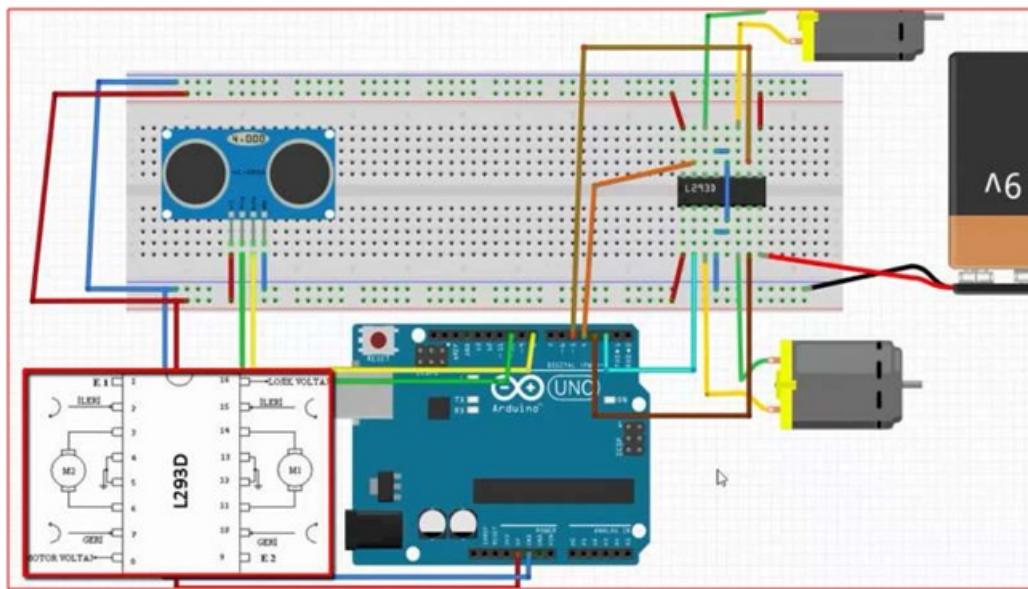
Bu projede, bir engeli önlemek için en basit yolu tasarladık. İlk olarak, armatürün GND pinini kartın toprak hattına ve 5 volt + tarafına bağladık. Vcc kaidesini 5 volt GND köprüsüne toprak hattına bağladık. Kalan trig ve eko mandalları, armatür için istediğimiz 2 pime bağlanır. Motor sürücüsünü sağ ve sol motorlar için yatay olarak bağladık, böylece kartın ortasına kısa devre olmadı. Daha sonra sol bacağı ve doğru motoru bağladık. Ardından yer işaretlerini gerekli pinlere bağladık. Daha sonra motorların ve bacakların bağlantılarını yaptık.

### Devre Şeması

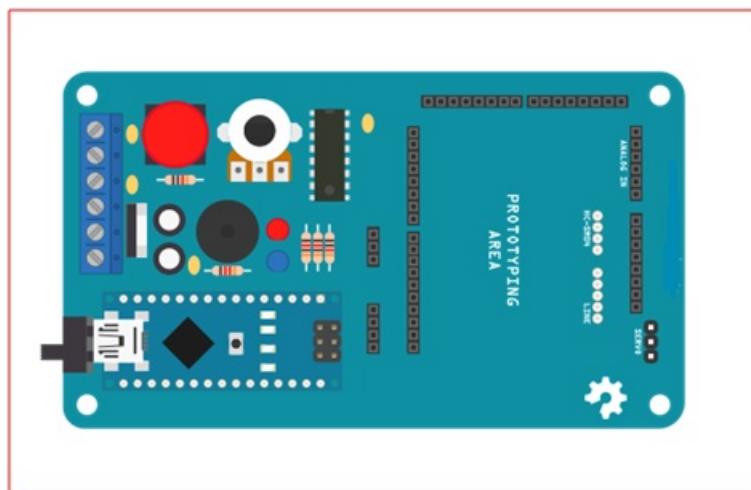
Parçaları tek tek test ettikten sonra gerçek robot sahnesine geçtik ve daha iyi bir görünüm elde etmek için robot üzerinde Arduino nano kullandık. Aşağıdaki kartı kullandık ve prototip alanında Arduino Nano ile breadboard koyduk.



Şekil 7.4: Devre Şeması



Şekil 7.5: Devre Şeması

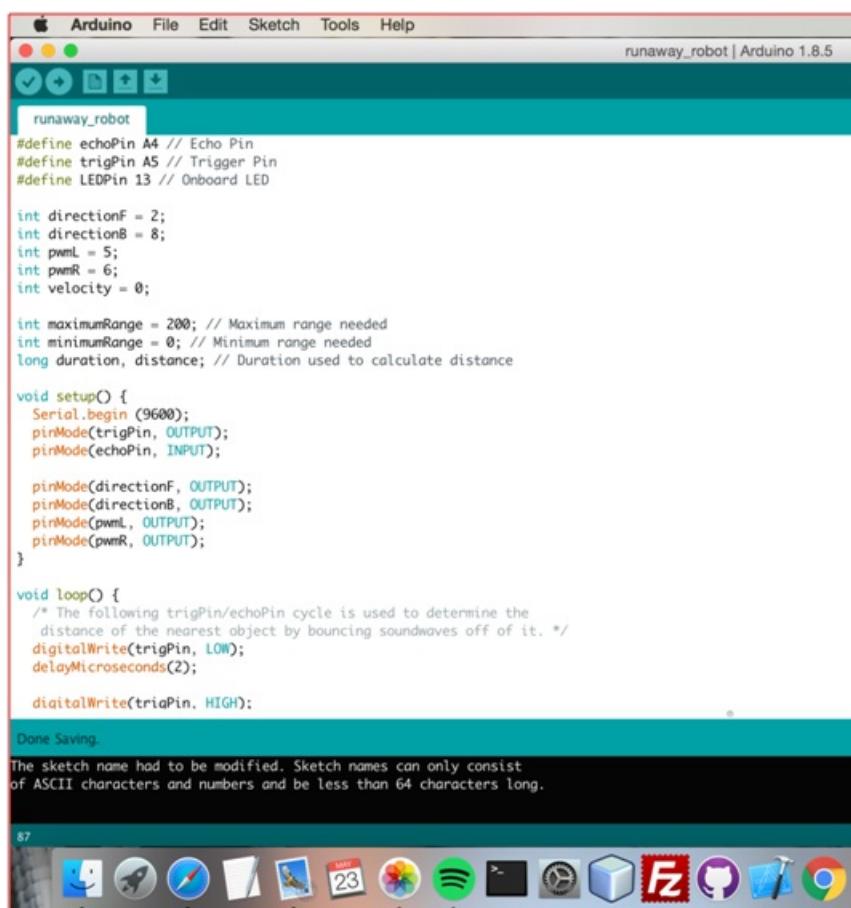


### Arduino Nano bağlantıları:

- Buton——>A0
- Buzzer——> D3
- Potansiyometre——> A6
- Mavi led———> D13
- Kırmızı led———>D7
- IR detektörü———>D4
- Motor——>D9
- HC-SR04——>A5(trig),A4(echo)

Çalışma mantığı sinyal işleme ile ilgilidir. Elektronik parça sadece ultrasonik sensörden veri almak ve motorları aktive etmek içindir. Sinyal işleme bölümünde yazılım olarak, ultrasonik sensörden gelen veriler döngü içinde gelir ve gerekli motor eylemleri gerçekleştirilir.

## Setup



The screenshot shows the Arduino IDE interface with the sketch named "runaway\_robot". The code defines pins for echo and trigger, initializes variables for direction, PWM, and velocity, and sets up serial communication. The setup function configures pins and starts serial communication. The loop function sends a pulse to the trigger pin, waits for the echo, and calculates distance. A message at the bottom indicates that the sketch name must be modified to meet naming conventions.

```
#define echoPin A4 // Echo Pin
#define trigPin A5 // Trigger Pin
#define LEDPin 13 // Onboard LED

int directionF = 2;
int directionB = 8;
int pwmL = 5;
int pwmR = 6;
int velocity = 0;

int maximumRange = 200; // Maximum range needed
int minimumRange = 0; // Minimum range needed
long duration, distance; // Duration used to calculate distance

void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  pinMode(directionF, OUTPUT);
  pinMode(directionB, OUTPUT);
  pinMode(pwmL, OUTPUT);
  pinMode(pwmR, OUTPUT);
}

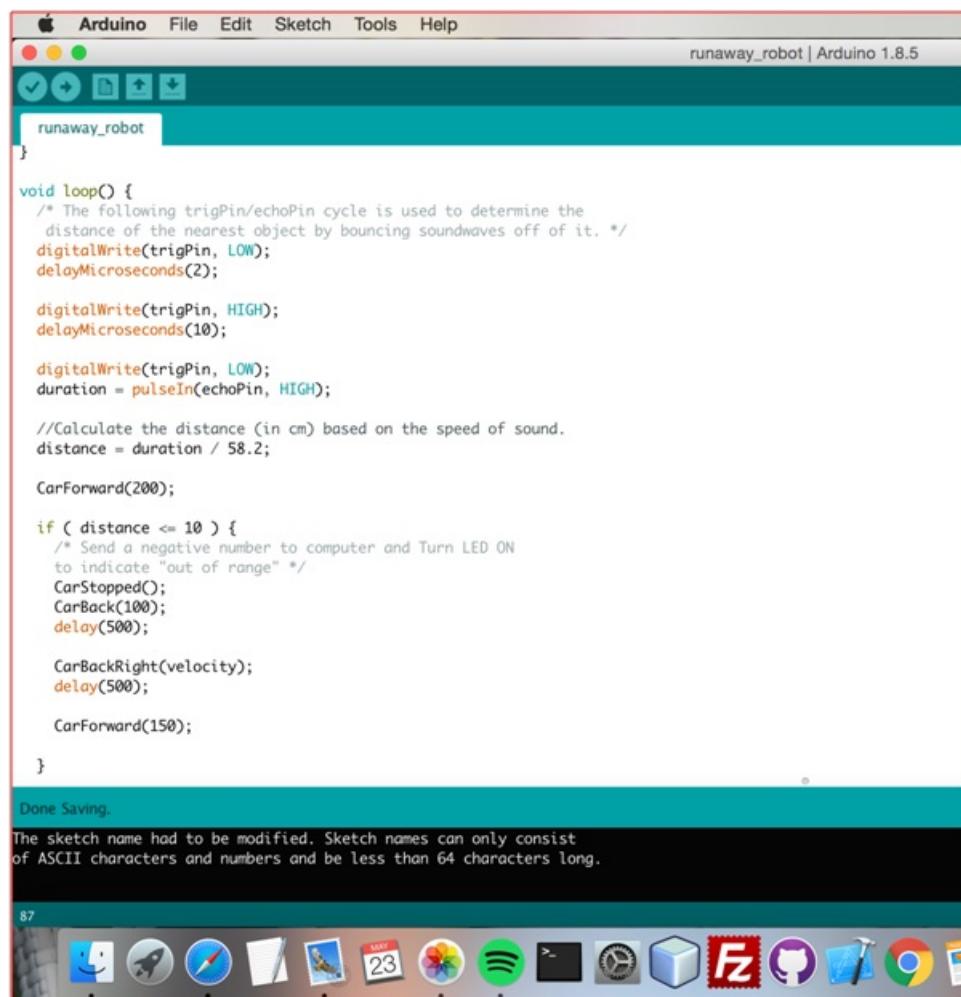
void loop() {
  /* The following trigPin/echoPin cycle is used to determine the
   * distance of the nearest object by bouncing soundwaves off of it. */
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
```

Done Saving.  
The sketch name had to be modified. Sketch names can only consist  
of ASCII characters and numbers and be less than 64 characters long.

Sekil 7.6: Setup

## Loop



The screenshot shows the Arduino IDE interface on a Mac OS X desktop. The window title is "runaway\_robot | Arduino 1.8.5". The code editor displays the following C++ code:

```
void loop() {
    /* The following trigPin/echoPin cycle is used to determine the
       distance of the nearest object by bouncing soundwaves off of it. */
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);

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

    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);

    //Calculate the distance (in cm) based on the speed of sound.
    distance = duration / 58.2;

    CarForward(200);

    if ( distance <= 10 ) {
        /* Send a negative number to computer and Turn LED ON
           to indicate "out of range" */
        CarStopped();
        CarBack(100);
        delay(500);

        CarBackRight(velocity);
        delay(500);

        CarForward(150);
    }
}
```

The status bar at the bottom of the IDE window shows the message: "Done Saving." and "The sketch name had to be modified. Sketch names can only consist of ASCII characters and numbers and be less than 64 characters long." Below the IDE window, the Mac OS X Dock is visible, showing various application icons.

Sekil 7.7: Loop

### Fonksiyon tanımları



The screenshot shows the Arduino IDE interface on a Mac OS X desktop. The window title is "runaway\_robot". The code editor contains several function definitions:

```
}
void CarForward(int x) {
    digitalWrite(directionF, HIGH);
    digitalWrite(directionB, LOW);
    analogWrite(pwmL, x);
    analogWrite(pwmR, x);
}
void CarStopped() {
    digitalWrite(directionF, LOW);
    digitalWrite(directionB, LOW);
    analogWrite(pwmL, 0);
    analogWrite(pwmR, 0);
}
void CarBackRight(int x) {
    digitalWrite(directionF, LOW);
    digitalWrite(directionB, HIGH);
    analogWrite(pwmL, x);
    analogWrite(pwmR, x - 50);
    delay(500);
}
void CarBack(int x) {
    digitalWrite(directionF, LOW);
    digitalWrite(directionB, HIGH);
    analogWrite(pwmL, x);
    analogWrite(pwmR, x);
}

Done Saving.
```

The status bar at the bottom displays the message: "The sketch name had to be modified. Sketch names can only consist of ASCII characters and numbers and be less than 64 characters long." The desktop background shows a blurred image of a building.

Şekil 7.8: Fonksiyon tanımları

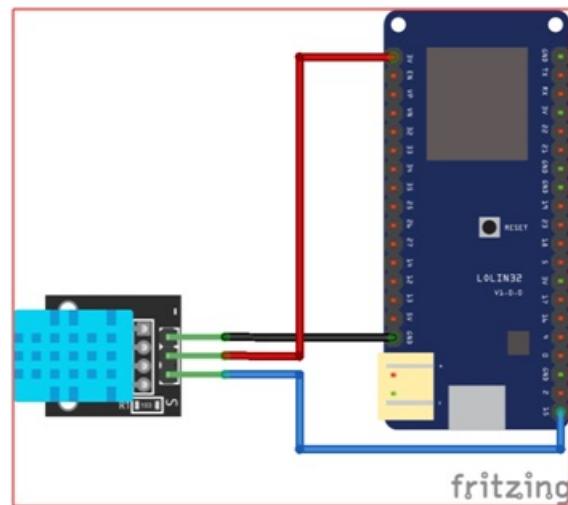
## 7.6 ESP32 WiFi-Bluetooth Modül

### Proje Videosu:

<https://drive.google.com/open?id=1OWCHN9tqQd6JmRVtuZvz6X9703LAhnjX>  
[https://drive.google.com/open?id=11P96M\\_bjj08bbmH25thhZL41NCvKSCQm](https://drive.google.com/open?id=11P96M_bjj08bbmH25thhZL41NCvKSCQm)

ESP32 en güncel ESP-WROOM-32 modülü ile güçlendirilmiş ve breadboardla kolayca ayarlanabilen, az yer kaplayan, minimal bir sistem geliştirme kartıdır. ESP-WROOM-32 etrafında inşa edilen bu minimal sistem geliştirme kartı, ESP32 tarafından sunulan zengin çevre birim seti, Wi-Fi ve Bluetooth radio çözümleri ile optimal performansı elde etmektedir. Bu projemizde ESP32 kartını kullanarak WiFi ağına bağlanıp sensörümüzden gelen veriyi ağ üzerinden, ücretsiz IoT bulut hizmeti sağlayan ThingSpeak.com adresine göndereceğiz. Thinspeak: internete bağlı cihazlar için gerçek zamanlı veri platformu olan web sitesi.

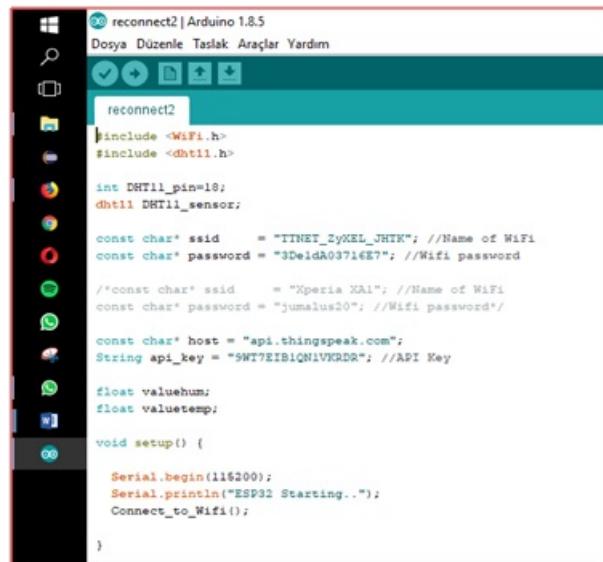
### Devre Şeması:



Şekil 7.9: Devre Şeması

### ESP32 Ardiuno IDE Kullanarak Programlama

Kodumuzu yazarken başlangıç olarak kullandığımız sensörün tanıtılmasını gerçekleştiriyoruz. ESP32 üzerinden hangi pin aracılığı ile bilgi alacağımızı



```

reconnect2 | Arduino 1.8.5
Dosya Düzenle Taslak Araçlar Yardım
reconnect2
#include <WiFi.h>
#include <dht11.h>

int DHT11_pin=10;
DHT11 DHT11_sensor;

const char* ssid      = "TTNET_ZyKEL_JHTEK"; //Name of WiFi
const char* password = "3DeidA0371e87"; //Wifi password

/*const char* ssid      = "Xperia XA1"; //Name of WiFi
const char* password = "jumalus20"; //Wifi password*/

const char* host = "api.thingspeak.com";
String api_key = "9WT7EIBIQLN1VKRDR"; //API Key

float valuehum;
float valuetemp;

void setup() {
    Serial.begin(115200);
    Serial.println("ESP32 Starting..");
    Connect_to_Wifi();
}


```

belirtiyoruz. Daha sonra ESP32 modülünün bağlanacağı WiFi ağının SSID ve şifre bilgilerini değişken olarak tanımlıyoruz. Daha sonra program çalışırken bu bilgileri WiFi ağına bağlanma fonksiyonuna parametre olarak göndereceğiz.

Setup bölümünde serial ekranımızı başlatıyor ve ESP32 modülünün internete bağlanması sağlıyoruz.

Aşağıda Connect to Wifi fonksiyonunu görüyorsunuz. Bu fonksiyonun içinde WiFi kütüphanesinin bize sağlamış olduğu WiFi.begin() fonksiyonunu çağırıyoruz ve bu fonksiyona başlangıçta tanımladığımız, bağlanmak istediğimiz WiFi ağının bilgilerini parametre olarak gönderiyoruz. Sonuç olarak ESP32 modülü bu fonksiyon sayesinde WiFi ağına bağlanmış oluyor.



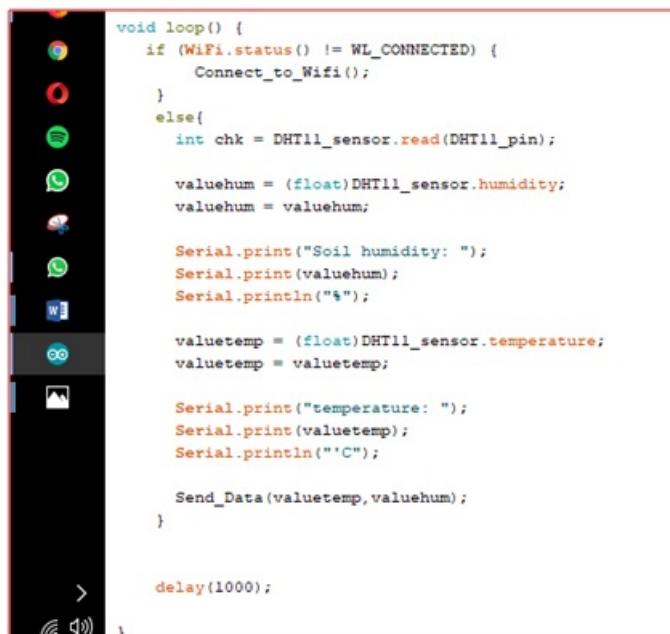
```

void Connect_to_Wifi(){

    while (WiFi.status() != WL_CONNECTED){
        WiFi.begin(ssid, password);
        Serial.println("ESP32 Connecting..");
        delay(3000);
    }
}

```

Aşağıda programın loop bölümünü görmektesiniz. Bu kısımda öncelikle ESP32 modülünün sürekli olarak WiFi ağına bağlı kalmasını istiyoruz. Bunun için bir kontrol bloğu kullanıyoruz. Modül bağlı değilse tekrardan WiFi ağına bağlanma fonksiyonunu çağırıyoruz. Bu bağlantı modemin kapanması, internetin kesilmesi, elektrikliğin kesilmesi vb. durumlarda gerçekleşebilir. Programımız bu gibi durumlarda dışardan müdahaleye gerek duymadan tekrar bağlantıyı gerçekleştiriyor.



```

void setup() {
  // put your setup code here, to run once:
}

void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    Connect_to_Wifi();
  }
  else{
    int chk = DHT11_sensor.read(DHT11_pin);

    valuehum = (float)DHT11_sensor.humidity;
    valuehum = valuehum;

    Serial.print("Soil humidity: ");
    Serial.print(valuehum);
    Serial.println("%");

    valuetemp = (float)DHT11_sensor.temperature;
    valuetemp = valuetemp;

    Serial.print("temperature: ");
    Serial.print(valuetemp);
    Serial.println("C");

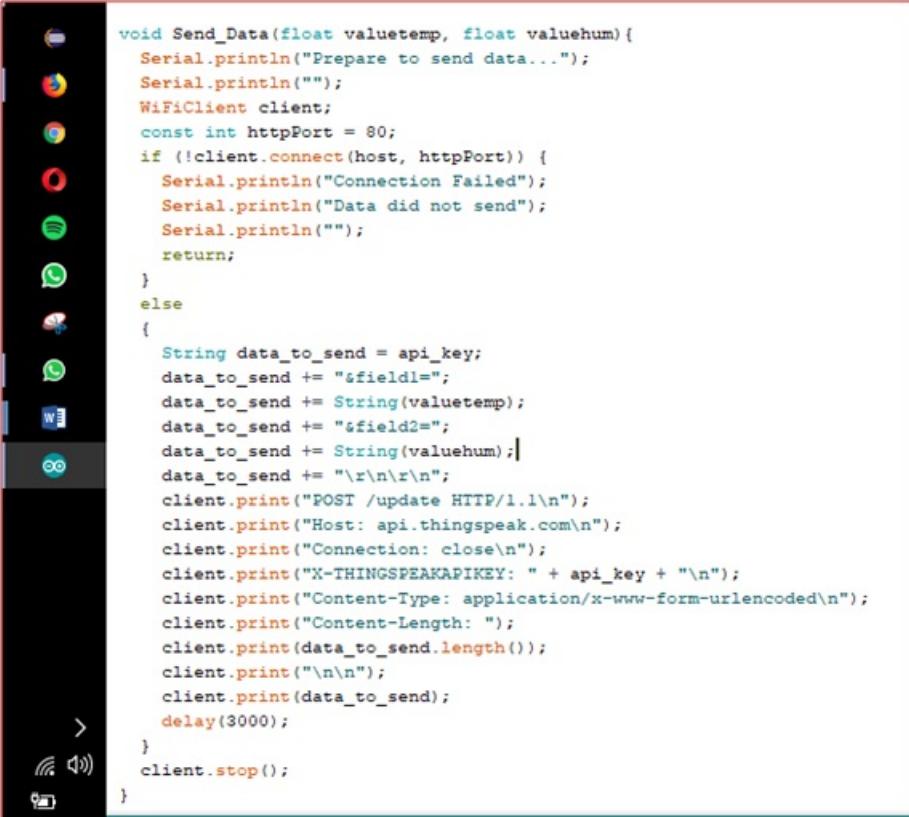
    Send_Data(valuetemp,valuehum);
  }

  delay(1000);
}

```

Daha sonra sensörden veri okuyoruz ve bu veriyi hem serial ekrana yazdırıp hem de kullandığımız ThingSpeak adresine gönderiyoruz. Veri gönderme işlemini Send\_Data() fonksiyonu ile gerçekleştiriyoruz.

Bu fonksiyonda sensörden aldığımız verileri ThingSpeak adresine gönderme işlemini gerçekleştiriyoruz. Sitenin kullanıcı hesabımıza vermiş olduğu API KEY'i kullanarak bu işlemi gerçekleştiriyoruz. Bu sayede ThingSpeak adresi gönderdiğimiz verileri bizim adımıza kayıt altında tutuyor. ThingSpeak adresi bize göndermiş olduğumuz verilerin hem json dosyası halinde hem de grafik olarak kayıtlarını tutmaktadır. Aşağıda örnek bir ThingSpeak grafiği görebilirsiniz.



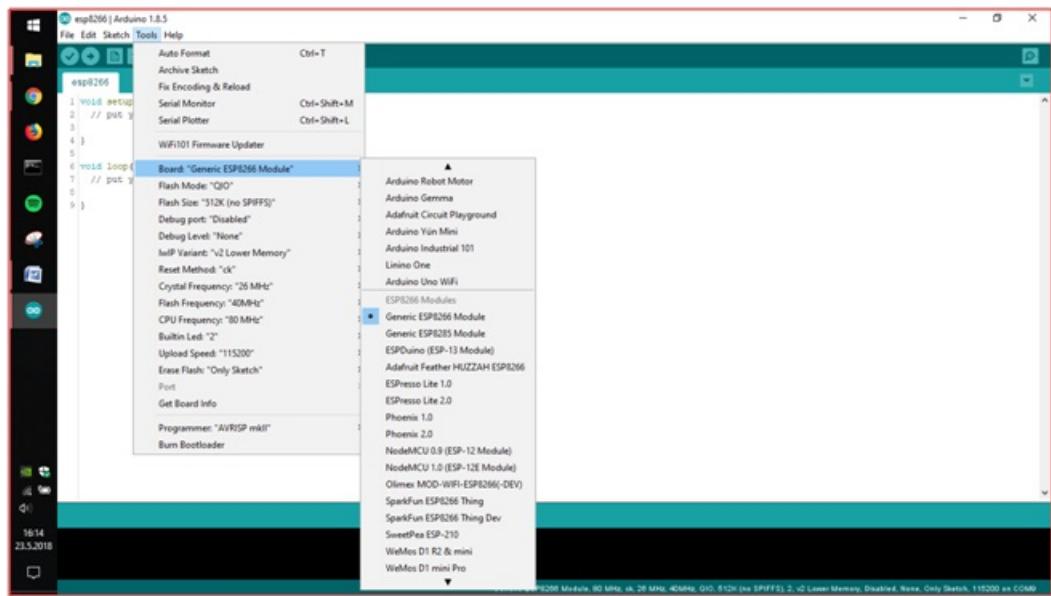
```
void Send_Data(float valuetemp, float valuehum){  
    Serial.println("Prepare to send data...");  
    Serial.println("");  
    WiFiClient client;  
    const int httpPort = 80;  
    if (!client.connect(host, httpPort)) {  
        Serial.println("Connection Failed");  
        Serial.println("Data did not send");  
        Serial.println("");  
        return;  
    }  
    else  
    {  
        String data_to_send = api_key;  
        data_to_send += "&field1=";  
        data_to_send += String(valuetemp);  
        data_to_send += "&field2=";  
        data_to_send += String(valuehum);  
        data_to_send += "\r\n\r\n";  
        client.print("POST /update HTTP/1.1\r\n");  
        client.print("Host: api.thingspeak.com\r\n");  
        client.print("Connection: close\r\n");  
        client.print("X-THINGSPEAKAPIKEY: " + api_key + "\r\n");  
        client.print("Content-Type: application/x-www-form-urlencoded\r\n");  
        client.print("Content-Length: " + data_to_send.length());  
        client.print("\r\n");  
        client.print(data_to_send);  
        delay(3000);  
    }  
    client.stop();  
}
```



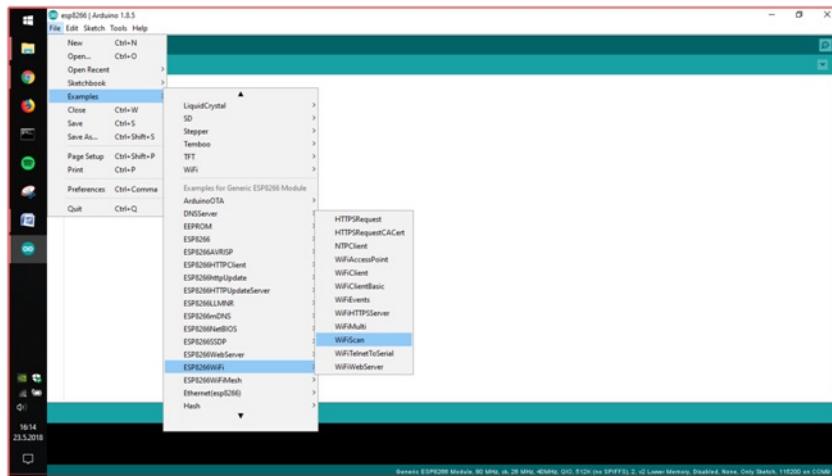
Şekil 7.10: ThingSpeak grafiği

## 7.7 ESP8266 (ESP-01) WiFi Modülü

Projemizde Arduino'ya bağladığımız ESP-01 modülünü etraftaki Wifi ağlarını taramak için kullandık. Öncelikle Arduino IDE'mize ESP8266 kütüphanelerinin ve örneklerinin tanımlarını indirip kurduk ve "Board" sekmesinden ESP8266'yi seçtik (Şekil 1). Sonrasında, ESP-01 modülünü Arduino'ya bağlarken 2 adet 3.3V regülatör kullandık, çünkü modül 3.3V uyumlu fakat Arduino'nun Tx(1. pin) ve Rx(0. pin) pinlerinden 5V çıkış sağlıyor. Bu regülatörler sayesinde bağlantıyı başarılı bir şekilde sağladık. Örneklerin içinde gelen WifiScan kodunu kullanarak projemizi tamamladık (Şekil 2).

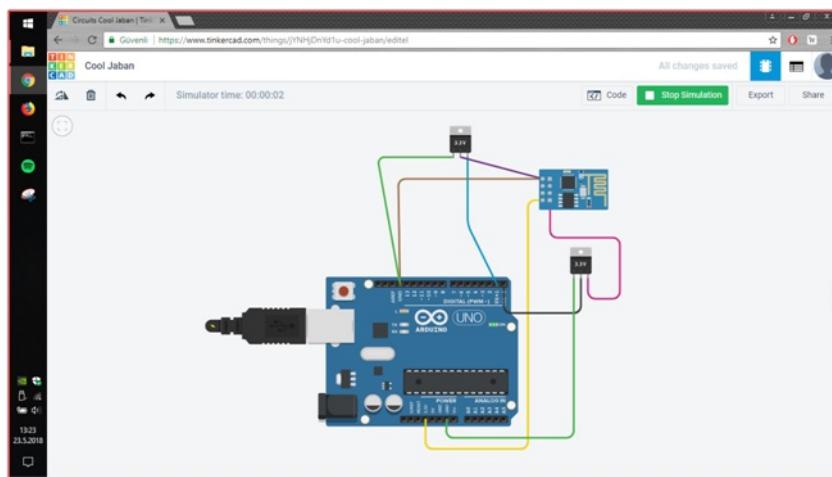


Şekil 7.11: Board Olarak ESP8266 Seçiliyor

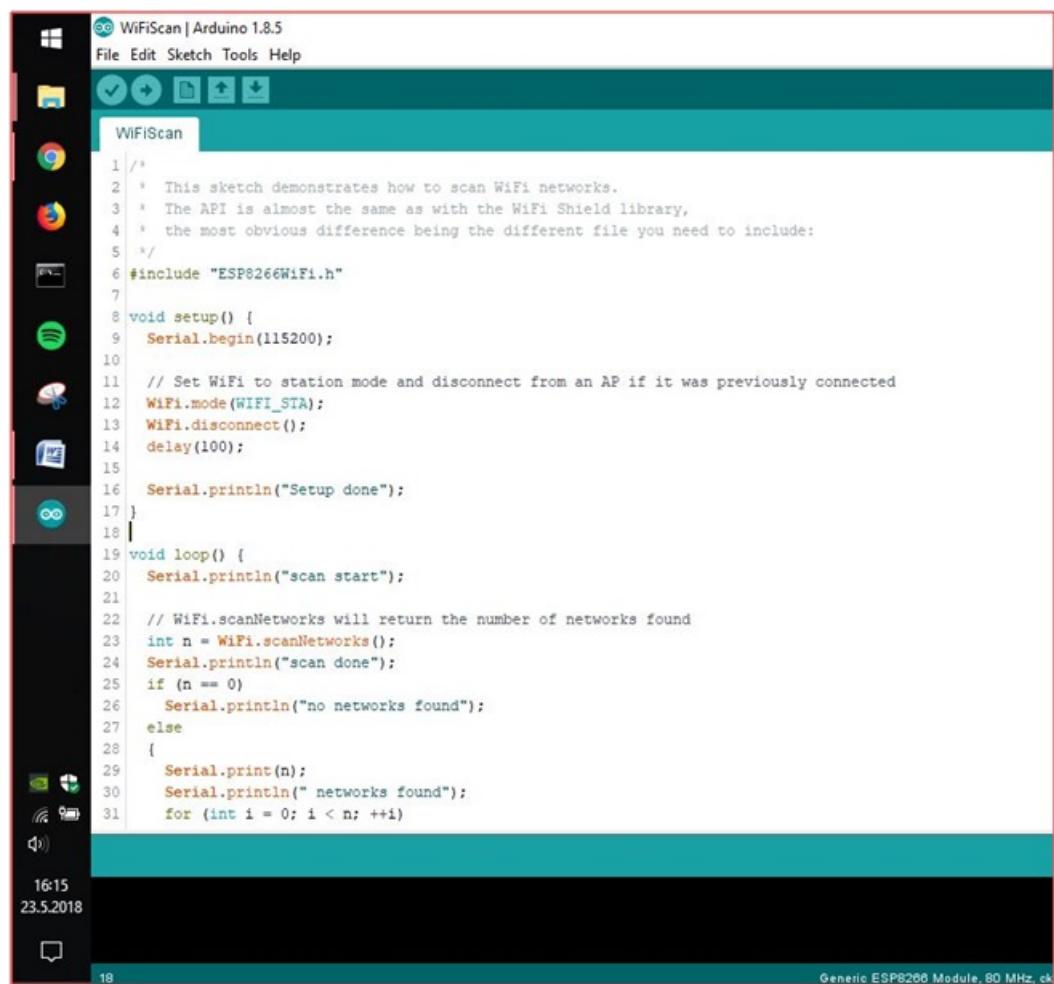


Şekil 7.12: Örneklerin İçerisinden WifiScan Seçiliyor

### Devre Şeması:



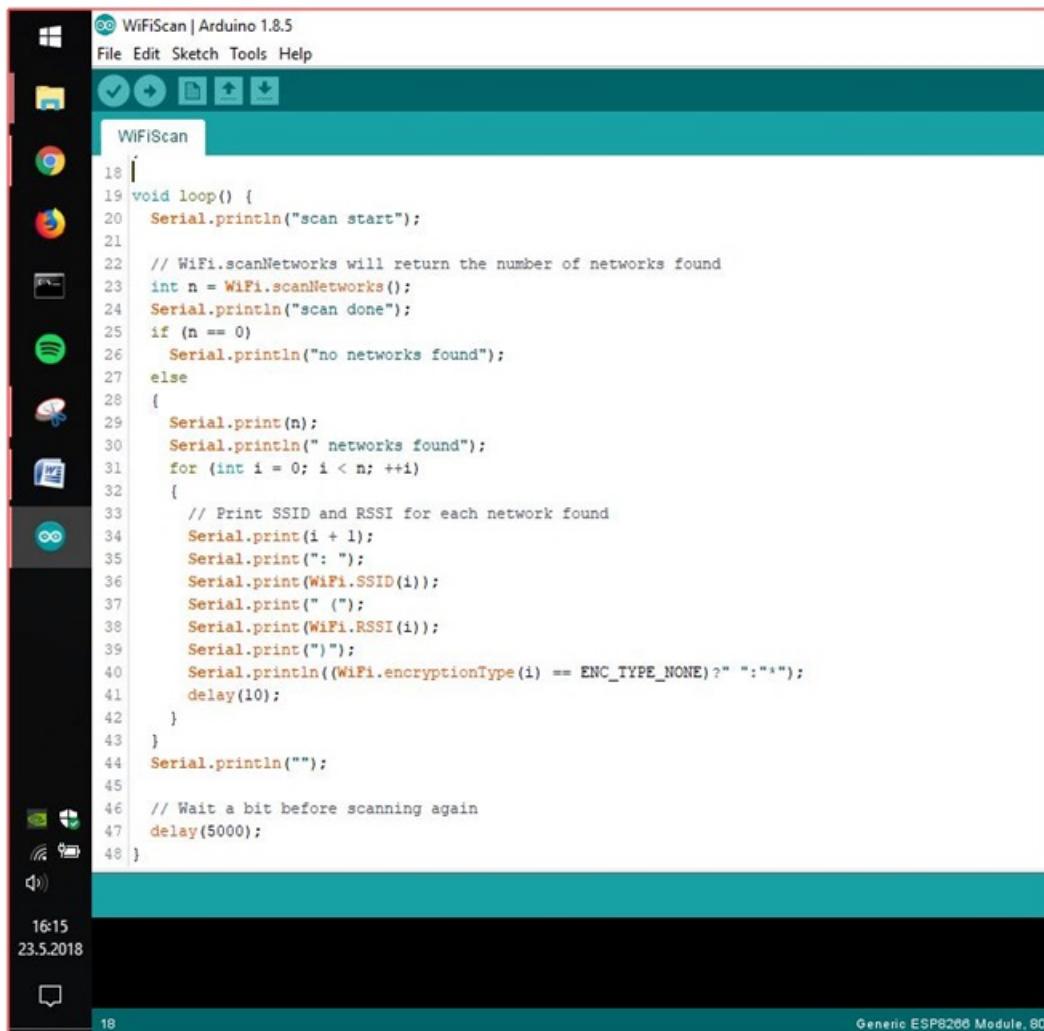
Şekil 7.13: Arduino ve ESP-01 Bağlantı Şeması

**Kaynak Kodu:**

The screenshot shows the Arduino IDE interface with the title bar "WiFiScan | Arduino 1.8.5". The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar has icons for Open, Save, Print, and Upload. The code editor contains the following C++ code for a WiFi scan:

```
1 // This sketch demonstrates how to scan WiFi networks.
2 // The API is almost the same as with the WiFi Shield library,
3 // the most obvious difference being the different file you need to include:
4 //
5 #include "ESP8266WiFi.h"
6
7 void setup() {
8     Serial.begin(115200);
9
10    // Set WiFi to station mode and disconnect from an AP if it was previously connected
11    WiFi.mode(WIFI_STA);
12    WiFi.disconnect();
13    delay(100);
14
15    Serial.println("Setup done");
16 }
17
18
19 void loop() {
20     Serial.println("scan start");
21
22     // WiFi.scanNetworks will return the number of networks found
23     int n = WiFi.scanNetworks();
24     Serial.println("scan done");
25     if (n == 0)
26         Serial.println("no networks found");
27     else
28     {
29         Serial.print(n);
30         Serial.println(" networks found");
31         for (int i = 0; i < n; ++i)
32             Serial.println(i);
33     }
34 }
```

The status bar at the bottom shows "16:15 23.5.2018" and "Generic ESP8266 Module, 80 MHz, ck".



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** WiFiScan | Arduino 1.8.5
- Menu Bar:** File Edit Sketch Tools Help
- Sketch Name:** WiFiScan
- Code Content:**

```
18 |
19 void loop() {
20   Serial.println("scan start");
21
22   // WiFi.scanNetworks will return the number of networks found
23   int n = WiFi.scanNetworks();
24   Serial.println("scan done");
25   if (n == 0)
26     Serial.println("no networks found");
27   else
28   {
29     Serial.print(n);
30     Serial.println(" networks found");
31     for (int i = 0; i < n; ++i)
32     {
33       // Print SSID and RSSI for each network found
34       Serial.print(i + 1);
35       Serial.print(": ");
36       Serial.print(WiFi.SSID(i));
37       Serial.print(" (");
38       Serial.print(WiFi.RSSI(i));
39       Serial.print(")");
40       Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE) ? " :\"");
41       delay(10);
42     }
43   }
44   Serial.println("");
45
46   // Wait a bit before scanning again
47   delay(5000);
48 }
```

- Bottom Status Bar:** 18 Generic ESP8266 Module, 80

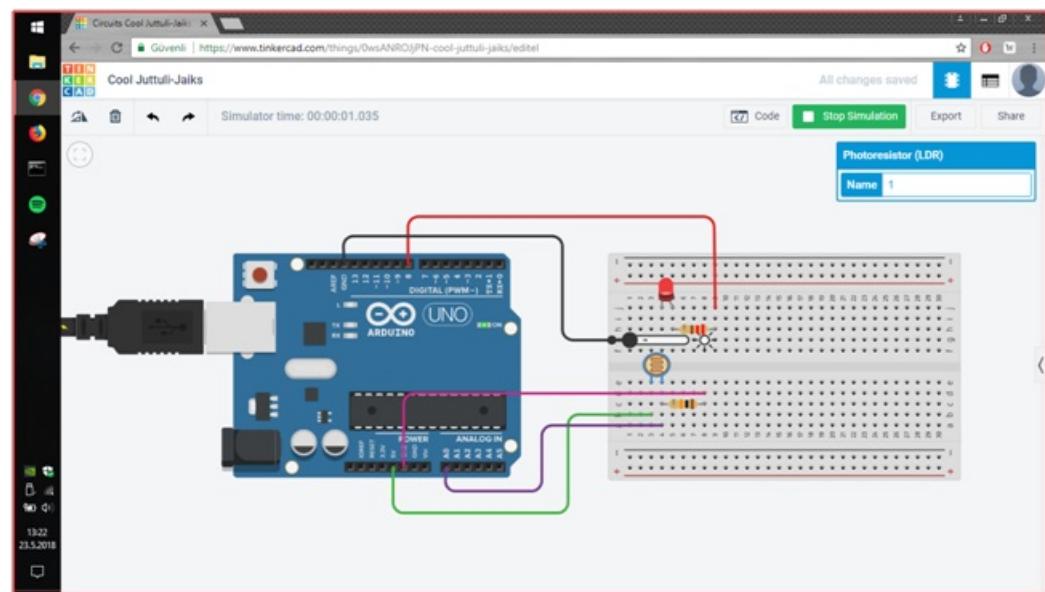
## 7.8 LDR Işık Sensörü

### Proje Videosu:

<https://drive.google.com/open?id=1gRIPuFp2vTujuVF08qABC1vdt0J05Gaq>

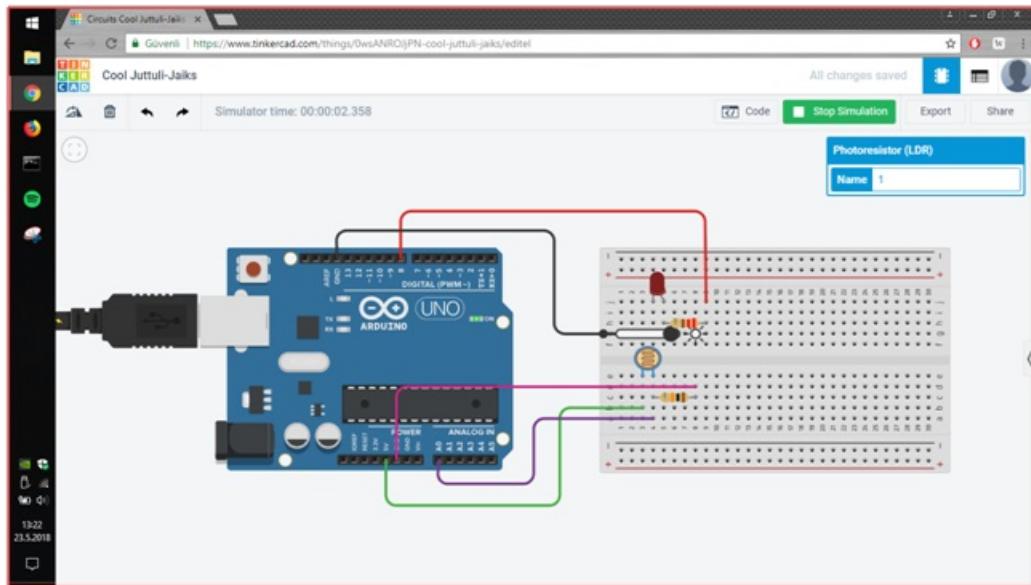
LDR ışık sensörü en basit optik sensör olarak adlandırılabilir. Çalışma prensibi şöyledir: Ortamdağı ışığın şiddetine göre direnç değeri artar ya da azalır ve bu değer analog olarak gönderilir. Ortam ışığı yüksek ise direnç düşer, ortam ışığı düşük ise direnç artar. Bu projemizde Arduino'ya bağladığımız LDR ışık sensörünün döndürdüğü analog değere bağlı olarak led ışık kaynağımızı yakıp söndürüyoruz. Eğer ortam karanlık sayılabilcek seviyedeyse led yanıyor (Şekil 1); aydınlatı sayılabilcek seviyedeyse led sönüyor (Şekil 2).

### Devre Şeması:



Şekil 7.14: Ortam Karanlıksa Led Yanıyor

### Kaynak Kodu:



Şekil 7.15: Ortam Aydınlıksa Led Sönüyor

```
isiksensor | Arduino 1.8.5
File Edit Sketch Tools Help
isiksensor
1 int LDR_Pin = A0; //analog pin 0
2
3 int lightVal;
4
5 void setup(){
6   Serial.begin(9600);
7   pinMode(8, OUTPUT);
8 }
9
10 void loop(){
11   lightVal = analogRead(LDR_Pin);
12   if(lightVal<500){
13     digitalWrite(8, HIGH);
14   }else{
15     digitalWrite(8, LOW);
16   }
17 }
18 }
```

## 7.9 Joystick Modülü

### Proje Videosu:

<https://drive.google.com/open?id=1hu7bWxkzWqkinmf2-1WCYadq3asLxaT6>

Joystick modülünün bağlantı için 5 pini bulunmaktadır. Bu pinlerden VRx yatay eksenindeki sinyalleri, VRy dikey eksendeki sinyalleri, SW pini ise joystick tıklama buton pin değerini okumayı sağlar. Joystick modülü yatay ve dikey eksende 0 ile 1023 arasında analog değerler üretir.

Joystick modülünü Arduino'ya bağlamak için;

- Gnd pinin arduino Gnd pinine,
- Vcc pinin arduino 5V pinine,
- VRx pinin arduino A0-A5 analog pinlerinden birine,
- VRy pinin arduino A0-A5 analog pinlerinden birine,
- SW pinin arduino dijital pinlerinden birine bağlanmalıdır.

Biz bu projede joystick modülünden dönen analog değerleri işleyerek VRx ve VRy değerlerine göre joystickin o an hangi yönde olduğunu hesaplayarak o yöne uygun led ışıklarının yanmasını sağladık. Bu led ışıklarını kuzey, güney, doğu ve batı olmak üzere 4 ana yöne koyduk.

## 7.10 LM35 Sıcaklık Sensörü ve 16x2 LCD

### Proje Videosu:

<https://drive.google.com/open?id=1GsUBL10a0yjKWxRiQB7yeWrkIs4i8iwF>

LM35 serisi sıcaklık sensörleri, çıkış olarak sıcaklık ile doğru orantılı bir voltaj verirler.  $60 \mu A$  gibi düşük bir akım çeker ve bu nedenle, çalışırken kendi içinde oluşan ısı düşüktür. Yani verdiği voltaj ölçülürken hata payı küçük olur. Projemizde kullandığımız bir diğer cihaz da 16x2 LCD ekrandır. Bu ekrana yazı yazabilmek için kullanacağımız karakterler daha önce Arduino geliştiricileri tarafından tanımlanmıştır. Tanımlanmış karakterleri kullanabilmek için öncelikle “LiquidCrystal.h” kütüphanesini projemize eklemeliyiz. Sonrasında LCD’ye bağlanan Arduino pinleri Şekil 1’deki gibi belirtilmelidir. Setup fonksiyonu içerisinde LCD türünü de belirttikten sonra LCD ekran kullanıma hazırdır.

```
1 #include <LiquidCrystal.h>
2
3 LiquidCrystal lcd(12,11,5,4,3,2);
4
```

Şekil 7.16: LCD Pin Tanımlaması

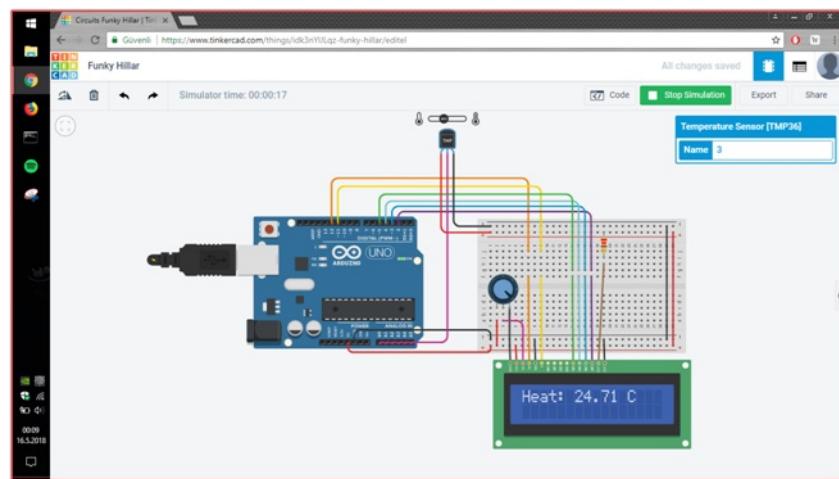
### Önemli LCD Fonksiyonları:

- `lcd.begin(sutun_sayısı, satır_sayısı)`: LCD ekranın tanınması için setup fonksiyonu içerisinde kullanılır. LCD kurulumu için fonksiyona sütun ve satır sayısı eklenmelidir.
- `lcd.print("Hello World!")`: LCD ekranı yazı yazdırma için kullanılır.
- `lcd.setCursor(sütun_sayısı, satır_sayısı)`: LCD ekran üzerinde imlecin yerini ayarlamak için kullanılır. Sütun ve satır sayıları 0’dan başmaktadır. Örneğin alt satırda inmek için fonksiyon içerisinde (0,1) yazılmalıdır. Böylece imleç, 0. sütun ve 1. satırda gidecektir. İmlecin yeri ayarlandıktan sonra yazma işlemi, imlecin bulunduğu yerden başlar.
- `lcd.clear()`: LCD ekranında yazan her şeyi siler ve imleci en başa alır.

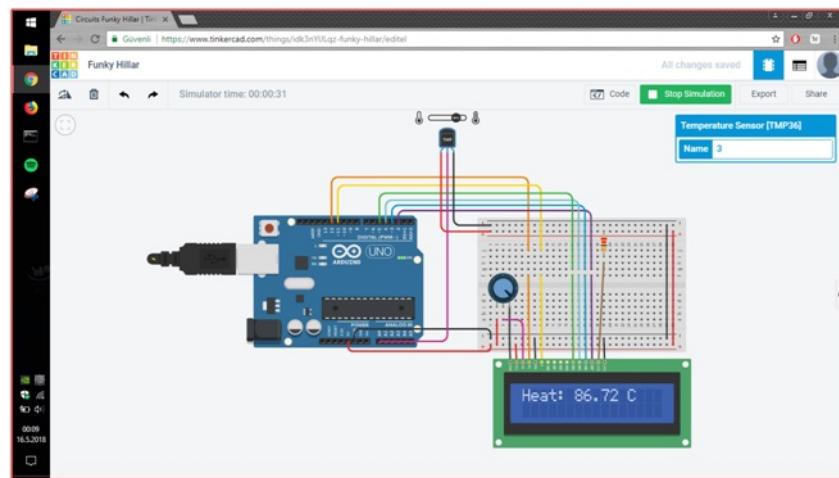
Projemizde yukarıda tanımladığımız cihazları ve Arduino Uno R3 kullanarak ortam sıcaklığını ölçen bir elektronik termometre yaptık. Bu termometre,

LM35 sensöründen gelen analog sinyali işleyerek çıkan sonucu LCD ekrana basıyor (Şekil 2, Şekil 3, Şekil 4).

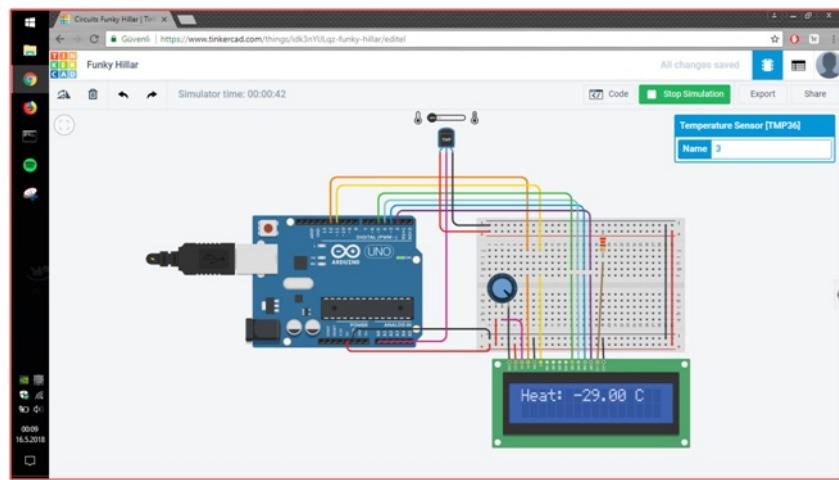
#### Devre Şeması:



Şekil 7.17: Ayarlanan Sıcaklık =  $25^{\circ}\text{C}$



Şekil 7.18: Ayarlanan Sıcaklık =  $86^{\circ}\text{C}$



Şekil 7.19: Ayarlanan Sıcaklık =  $-30^{\circ}\text{C}$

```
sicaklik | Arduino 1.8.5
File Edit Sketch Tools Help
sicaklik
1 #include <LiquidCrystal.h>
2
3 LiquidCrystal lcd(12,11,5,4,3,2);
4
5 const int heat_pin = A0;
6
7 void setup(){
8   lcd.begin(16,2);
9 }
10
11 void loop (){
12   lcd.clear();
13   lcd.print("Heat: ");
14   lcd.print(0.48828125*analogRead(heat_pin));
15   lcd.print(" C");
16   delay(1000);
17 }
18 }
```

Şekil 7.20: Kaynak kodu

## 7.11 Arduino Boy Ölçer

Sabit bir yükseliğe koyduğum mesafe sensörü sayesinde , bu sensörün altına geçen kişilerin boyu ölçülmektedir.

### Ultrasonic Sensor HC-SR04

#### Açıklama

HC-SR04 ultrasonik sensör, yarasalar gibi bir nesneye olan mesafeyi belirlemek için sonar kullanır. Kullanımı kolay bir pakette, yüksek doğruluk ve kararlı okumalarla mükemmel temassız alan tespiti sunar. 2 cm'den 400 cm'ye veya 1 "den 13 feet'e. Çalışması güneş ışığından etkilenmez veya keskin telemetreler gibi siyah materyaller (kumaş gibi akustik olarak yumuşak materyallerin tespit edilmesi zor olabilir). Ultrasonik verici ve alıcı modülü ile birlikte geliyor. Sinyalin iletilmesi ve alınması arasındaki süre, bir nesneye olan mesafeyi bilmemizi sağlar. Bu mümkündür çünkü sesin havadaki hızını biliyoruz.

#### Nasıl Çalışır?

Ultrasonik sensör bir nesneye olan mesafeyi belirlemek için sonar kullanır. İşte olan:

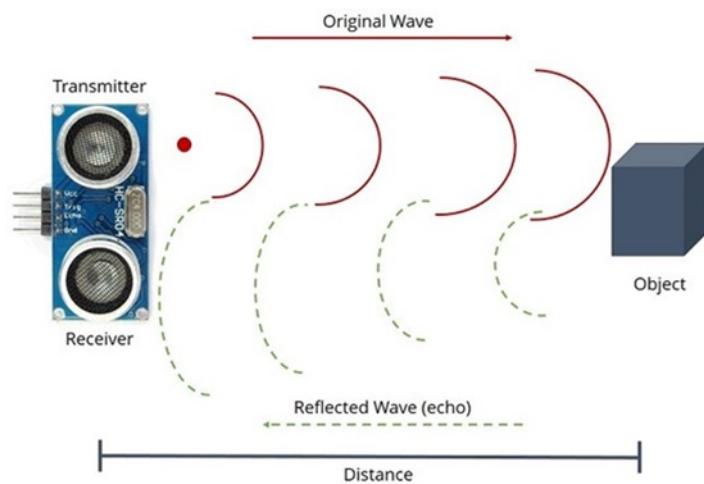
1. Verici (trig pin) bir sinyal gönderir: yüksek frekanslı bir ses
2. Sinyal bir nesne bulduğunda, yansıtılır ve
3. Verici (eko pin) bunu alır.

#### Kullandığım Parçalar

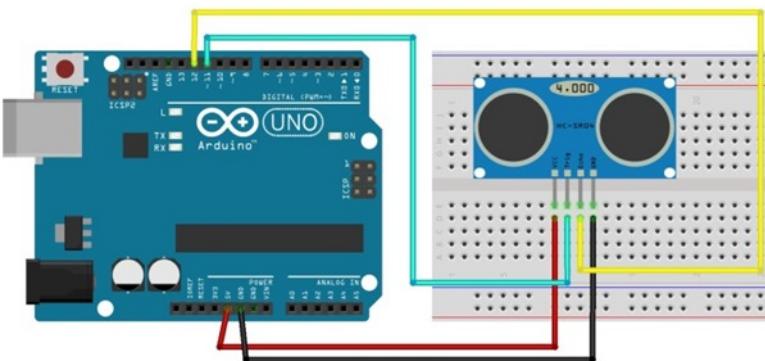
- Arduino UNO
- Ultrasonic Sensor (HC-SR04)
- Breadboard
- Jumper wires
- Bir adet Red LED
- Bir adet 560 ohm direnç

#### Şema

HC-SR04 Ultrasonik Modülü 4 pimli, Toprak, VCC, Trig ve Echo'ya sahiptir.

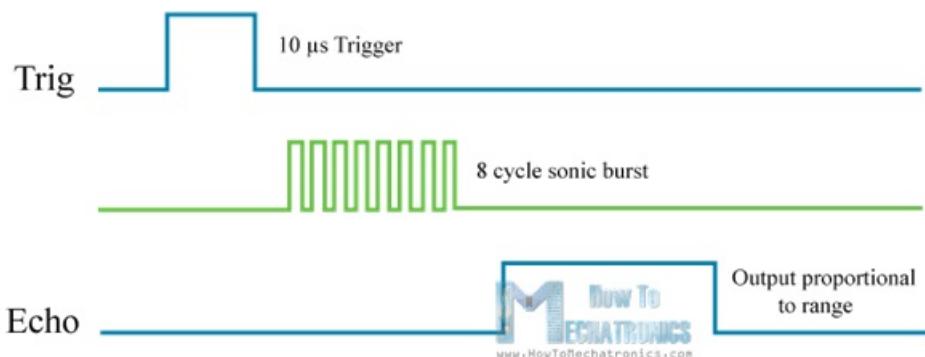


Modülün Zemini ve VCC pinlerinin, Arduino Board’undaki Ground ve 5 volt pinlerine ve Arduino Board üzerindeki herhangi bir Digital I / O pinine sırasıyla trig ve echo pinlerine bağlanması gereklidir.

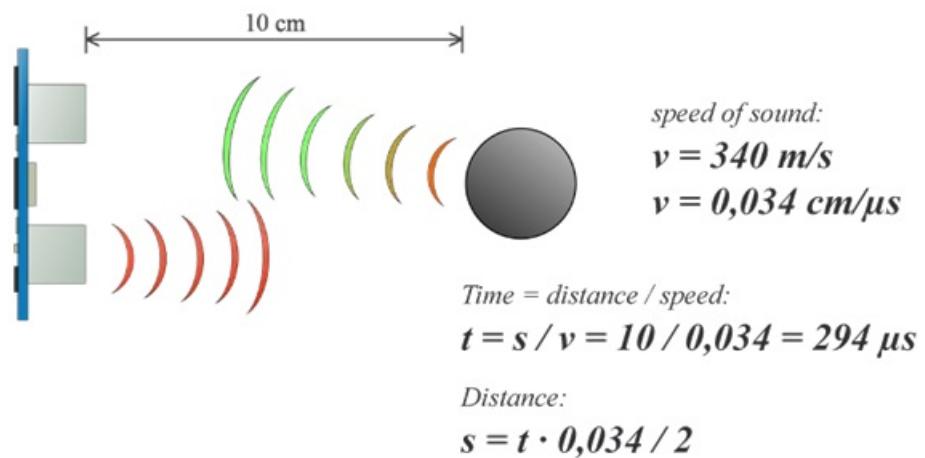


Ultrason üretmek için,  $10\mu s$  için Trig’ı Yüksek Durumda ayarmanız gereklidir. Bu, hız sesinde hareket edecek ve Echo pininde alınacak olan bir 8 döngü ses patlaması gönderecektir. Yankı pimi, ses dalgasının katettiği mikro saniye cinsinden süreyi verecektir.

Örneğin, eğer nesne sensörden 10 cm uzakta ise ve ses hızı  $340m/s$  veya  $0,034cm/\mu s$  ise, ses dalgasının yaklaşık 294 saniye sürmesi gerekecektir. Fakat Echo pininden alacağınız şey bu sayıdan iki kat daha fazla olacaktır çünkü ses dalgasının ileriye doğru hareket etmesi ve geriye doğru sıçraması gereklidir.



Böylece, mesafeyi cm olarak almak için, alınan seyahat süresi değerini eko ığnesinden 0,034 ile çarpmalı ve 2'ye bölmeliyiz. <sup>2</sup> <sup>3</sup> <sup>4</sup>

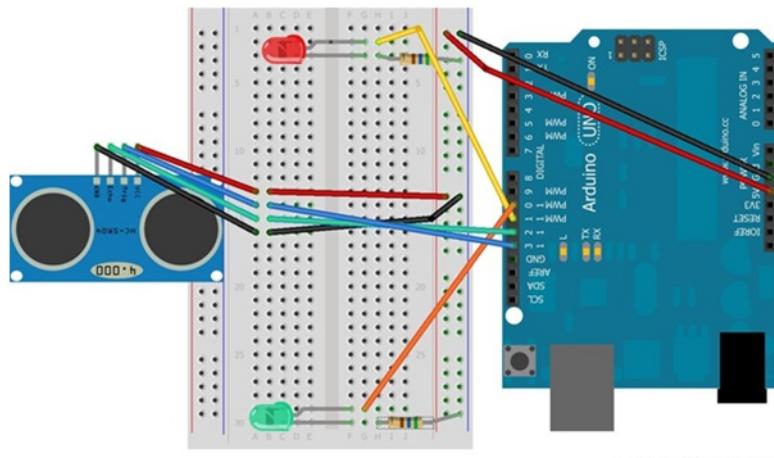


<sup>2</sup><https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/>

<sup>3</sup><https://howtomechatronics.com/tutorials/arduino/>

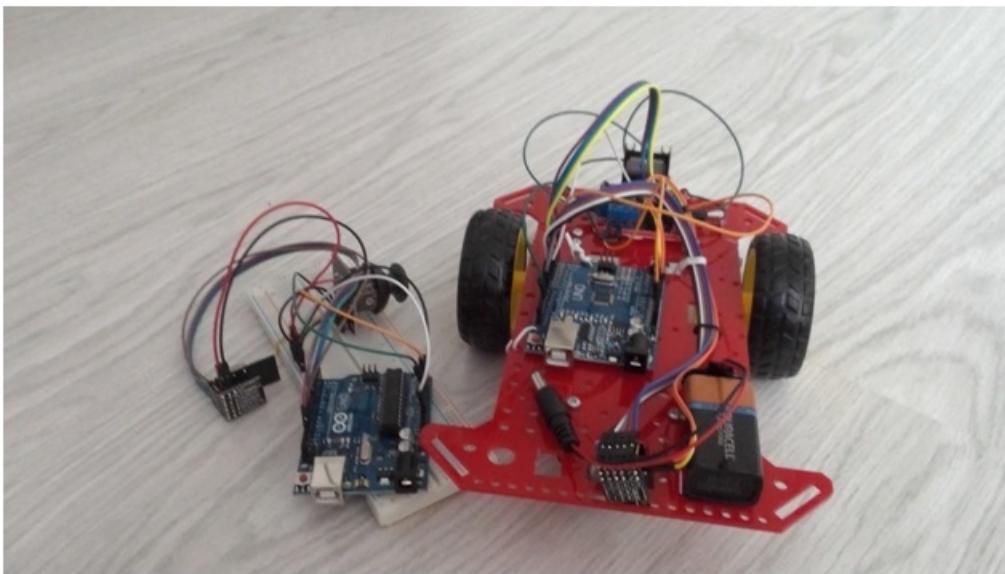
ultrasonic-sensor-hc-sr04/

<sup>4</sup><http://www.instructables.com/id/Simple-Arduino-and-HC-SR04-Example/>





## 7.12 Arduino Uzaktan Joystick Kontrollü Araba



Projede, Arduino'nun kablosuz iletişiminin bir yöntemi olan NRF24L01+ modülünü inceleyeceğim. Bu 2,4 GHz bandını kullanarak 2 yönlü iletişim sağlayan ucuz bir modüldür.

### Malzemeler

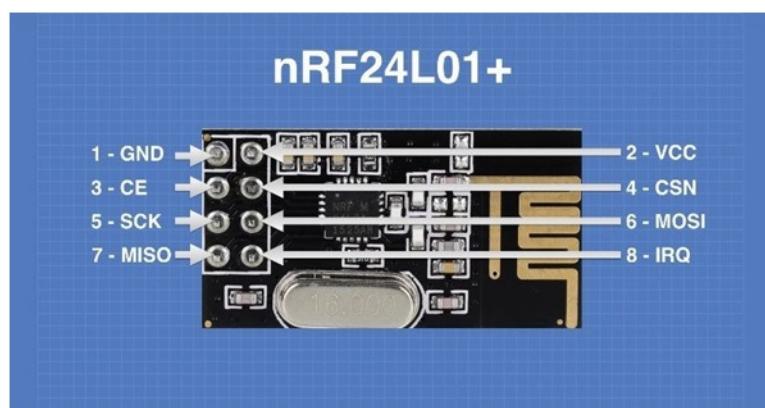
- 2x Arduino UNO
- 2x NRF24L01 2.4GHz wireless modülü
- 1x L298N Motor Sürücü
- 2x Gear Motor
- 1x Joystick
- 2x 9v Pil
- 1x Jumper Seti
- 1x Araba İskeleti

### NRF24L01 Bağlantıları

NRF24L01'in, dış dünyayla arabirim oluşturan 8 pinli bir konnektörü vardır. Bu bağlayıcı, NRF24L01 modüllerinin iki stili arasında yaygındır. NRF24L01, 1.9 ila 3.9 volt beslemeyle çalıştırılmasına rağmen, mantık pimleri 5 volt toleranslıdır, bu nedenle doğrudan bir Arduino veya diğer 5 voltlu bir mantıksal mikro denetleyici ile kullanılabilirler.

#### Modül Bağlantıları

NRF24L01 modülüne olan bağlantılar aşağıdaki gibidir:



#### NRF24L01 Adaptör Modülü

NRF24L01 Adaptör Modülü, NRF24L01 ile çalışmayı basitleştiren çok ucuz bir prototip karttır.

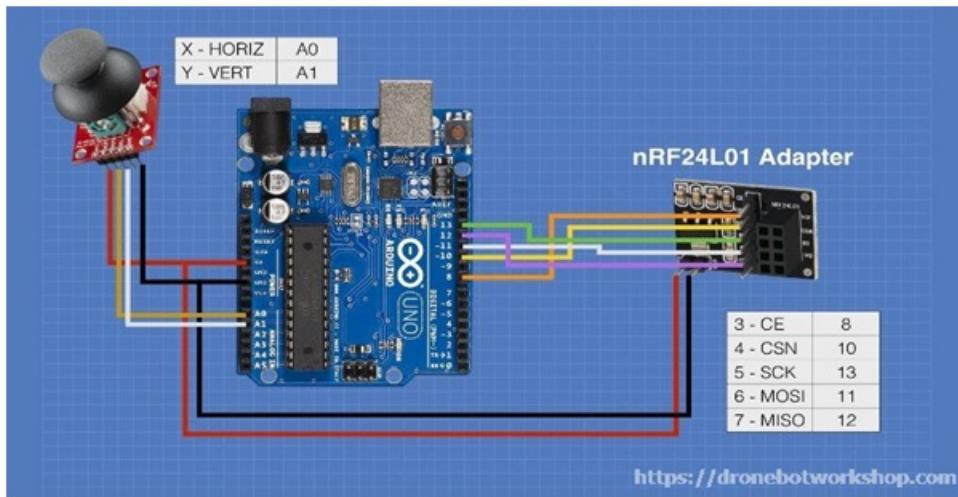
Adaptör modülünde bir NRF24L01 takabilmeniz için 8 pinli dişi konnektör bulunur; bu modül entegre veya harici anten ile birlikte kullanılabilir. Ayrıca SPI ve Interrupt bağlantıları için 6 pinli erkek konnektör ve güç girişi için 2 pinli bir konnektör vardır.

Adaptör modülü kendi 3,3 voltluq voltaj regülatörüne ve bir dizi filtre kondansatörüne sahiptir, böylece 5 voltluq bir güç kaynağı ile güç sağlayabilirsiniz.

#### Gerekli Bütün Bağlantılar

İşte bir Arduino Uno için yapmanız gereken bağlantılar:

Bağlantıları yaptıktan sonra yazdığım kodları alıcı ve vericiye yükledim ve so-

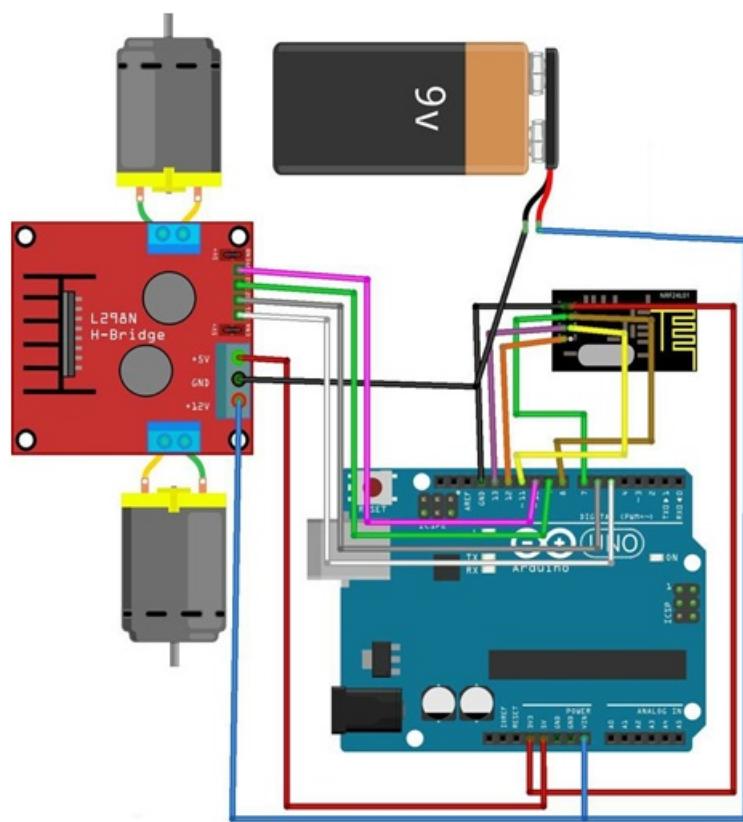


Şekil 7.21: Gerekli bağlantılar

runsuz bir şekilde çalışmalarını sağlamak için gerekli güncellemeleri yaptım. Sonuç olarak sistemin çalışmasını sağladıkten sonra çıkan sonuç aşağıdaki linktedir.

**Proje Videosu:**

[https://drive.google.com/file/d/1NUCct0SBj9i6mzDbxzC2nGsw\\_5SKxP\\_Y/view?usp=sharing](https://drive.google.com/file/d/1NUCct0SBj9i6mzDbxzC2nGsw_5SKxP_Y/view?usp=sharing)



Sekil 7.22: Gerekli bağlantılar

## 7.13 Piyano

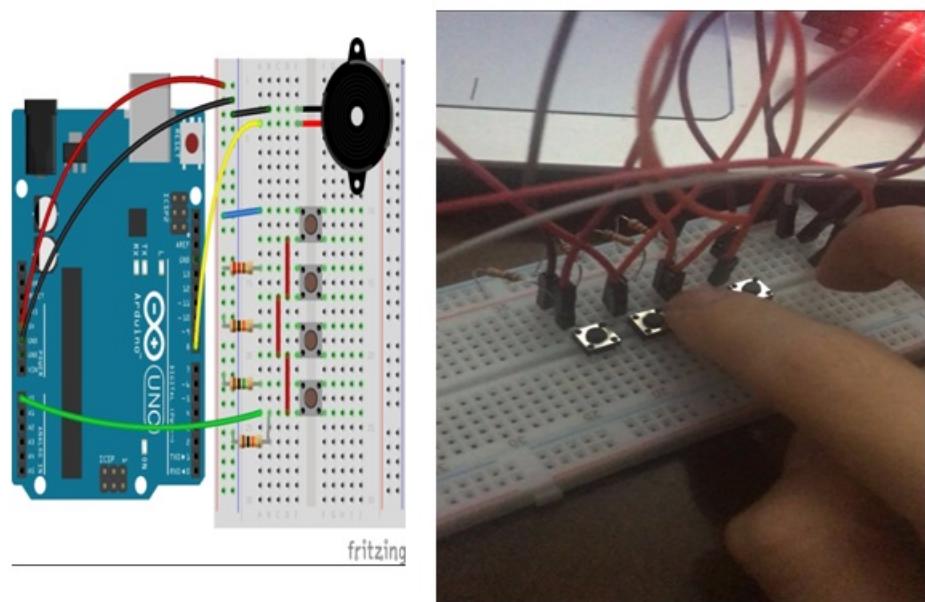
### Proje Videosu:

<https://drive.google.com/file/d/1zCC1fS19A78gDWpkFH513z-mphdz6DZR/view>

### Malzemeler

- 1 Arduino UNO
- 1 Breadboard
- 1 buzzer
- 4 Switch
- 1 x 220-ohm resistor
- 2 x 10-kilohm resistor
- 1-megohm resistor
- 9 Jumper

Resistor ladder olarak adlandırılan yeni bir direnç kablolama yöntemi vardır. Bu, analog girişi kullanarak bir dizi anahtarı okumak için bir yoldur. Dijital girişi kullanmaktan daha yararlıdır. Analog bir pine paralel bağlanan birkaç anahtar bağlayacağız (yani A0). Anahtarların çoğu, bir direnç aracılığıyla güce bağlanır. Nasıl çalışır: Her düğmeye bastığımızda, giriş pinine farklı bir voltaj seviyesi geçecektir. Aynı anda iki düğmeye basarsak, paralel olarak iki dirençleri temel alan benzersiz bir giriş elde ederiz (bu yüzde onları paralel olarak bağlamamız gereklidir)



Sekil 7.23: Piyano

## 7.14 Anroid Telefon ile Lamba Kontrolü

### Proje Videosu:

<https://drive.google.com/file/d/1ygB6NN6ta9PYse5gopz1Eb9QFmE0ohva/view>

Anahtar ve android telefondan odamızın lambasını rôle ve vavien kullanarak kontrol edeceğiz. Role ve vavien kullanmamızın sebebi hem anahtardan lambamızı açıp kapatırken hemde telefondan bu işlemi gerçekleştirmemizi sağlayacak. Vavien iki yollu bir elektrik anahtarıdır. Kapalı veya Açık konumda iken A bacağını B veya C bacağına bağlar. Vavien anahtarlar karşılıklı çalışır bir odada 2 anahtar ile bir lamba kontrol etmek istediklerinde vavien anahtar kullanılabilir. Rôle üzerinden akım geçtiği zaman çalışan elektromanyetik bir devre elemanıdır. Üzerinden akım geçtiği zaman ortak ucu A veya B bacağına bağlar. Vavien ve Rôle birbirine benzeyen yapılardır birbirinden ayıran özellikleri vavien fiziksel bir güçle role ise elektromanyetik güçle açık kapalı işlevini görür. Projemiz için duvarda bulunan anahtarımızı Vavien anahtar ile değiştirmemiz gerekmektedir.

- Arduino uno
- Vavien Anahtar
- Tekli 5V Rôle Kartı
- Lamba
- Bluetooth Modul (hc-05 veya hc-06)
- Kablo

### ARDUINO' MUZA BLUETOOTH MODULÜNÜN BAĞLANTISINI YAPALIM

- RXT —> TXD
- TXD —> RXT
- GND —> GND
- VCC —> 5V

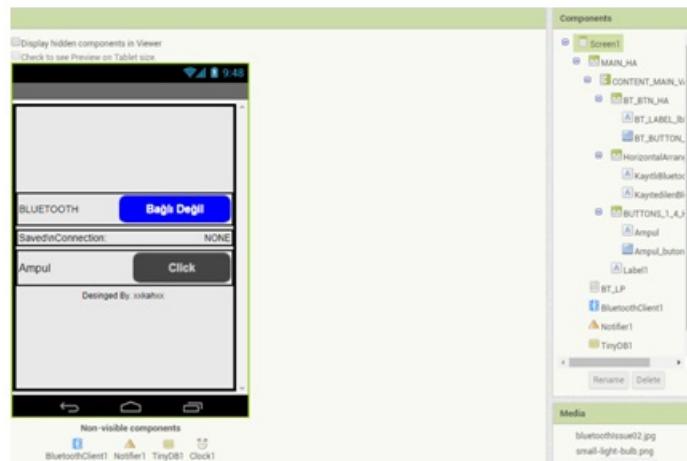
ARDINDAN RÖLE MODÜLÜNÜN ARDUİNO İLE BAĞLANTISINI YAPALIM

- VCC —> 5V
- GND —> GND
- İN —> Digital Pin 9

VE SON OLARAK RÖLE VE VAVİEN ANAHTAR BAĞLANTISINI YAPALIM

- Duvardan gelen kablomuzun birini Röle Modülümüzün C girişine bağlıyoruz.
- Duvardan gelen 2. kablomuzu Vavien anahtaramızın C girişine bağıyoruz.
- Vavien anahtaramızın A çıkışını Rölemizin NC girişine Bağlıyoruz
- Vavien anahtaramızın B çıkışını Rölemizin NO girişine Bağlıyoruz

### **ANDROİD PROGRAM İÇİN APP İNVERTOR 2 ADLI UYGULAMA**



Şekil 7.24: App Inventor adlı uygulama



Sekil 7.25: Anroid Telefon ile Lamba Kontrolü

## 7.15 Akıllı Kablosuz Zar

### Kullanılan parçalar

- MPU-6050 Gyro
- NRF24L01+
- Arduino Nano V3
- Pil

Zar tarafı değişir ise sayı'yı ve ID'yi sunucuya aktarır, Sunucudan komutları alır ve komutlara göre gereken işlemleri uygular

#### Örnek komutlar:

1. ID değiştirmeye komutu, seçilen zar'a yeni ID numarasını kayıtlar.
2. Radyo kanalı değiştiren komut, seçilen zar diğer radyo kanala geçmesini sağlar.

#### Çalışma algoritması:

Radyo modülü sürekli dinleme modunda çalışır ve herhangi bir komut geldiğinde komut içindeki ID'yi kendi ID'si ile karşılaştırır. Eğer ID'ler aynı ise – komutu uygular, farklı ise – hiç bir şey yapmayacak.

Zar tarafı değişmişse radyo modülü dinleme modunu kapatır ve yazma moduna geçer. Sunucuya kendi ID ile taraf bilgileri ilettikten sonra yazma modunu kapatır ve dinleme moduna geçer.

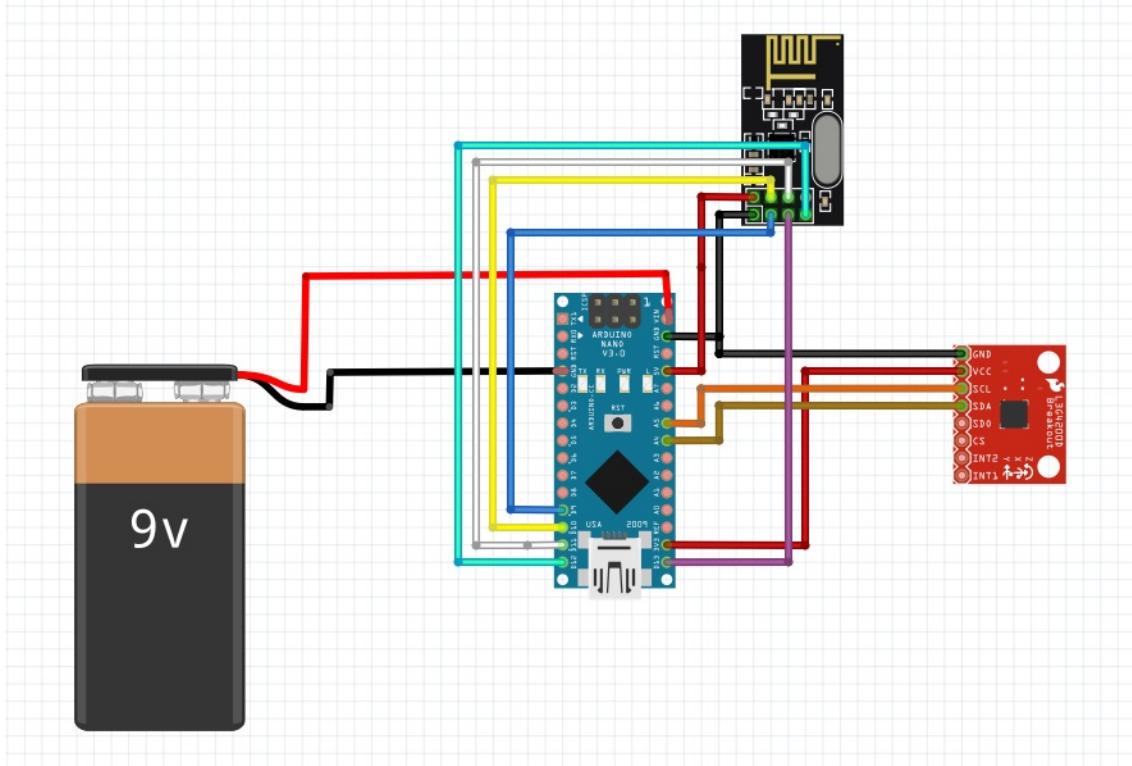
#### Sunucu

### Kullanılan parçalar Arduino Nano V3 NRF24L01+

Zarlar dan gelen mesajları elde edip istenen formatta kullanıcıya iletir. Seri port ile kullanıcıdan komutları alır ve gereken işlemleri seçilen zar'lara uygular.

#### Çalışma algoritması:

Radyo modülü sürekli dinleme modunda çalışır ve herhangi bir mesaj geldiğinde kullanıcıya güzelce iletir.



Şekil 7.26: Zar devre şeması

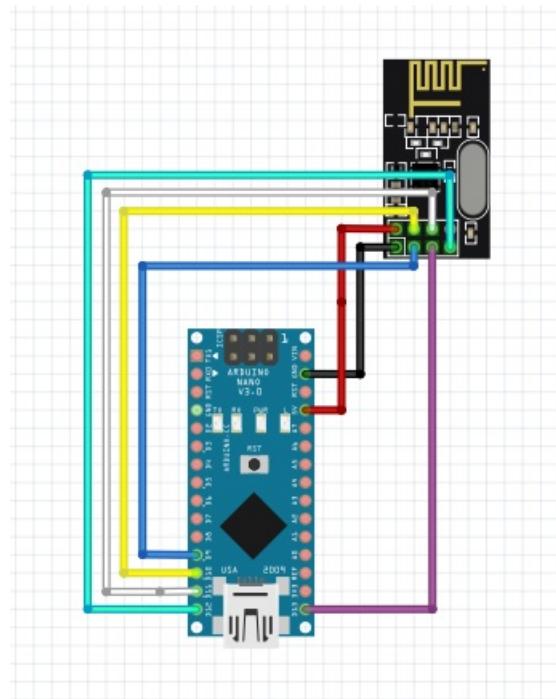
Kullanıcı herhangi komut girdiyse radyo modülü dinleme modunu kapatır ve yazma moduna geçer. Zarlar'a girilen komutu ilettikten sonra yazma modunu kapatır ve dinleme moduna geçer.

Kod

Zar Kodu

```
1 #include <Servo.h>
2 #include "I2Cdev.h"
3 #include "MPU6050_6Axis_MotionApps20.h"
4 #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
5 #include "Wire.h"
6 #endif

7 #include <SPI.h>
8 #include "RF24.h"
9 #include <nRF24L01.h>
```



Şekil 7.27: Sunucu devre şeması

```
11  uint64_t pipe = 0xF0F1F2F3F4LL;
13  short channel = 120;
14  RF24 radio(9, 10); // CE, CSN
15
16  typedef struct {
17      long cubeCode;
18      int side;
19  } CUBEINFO;
20  CUBEINFO cubeInfo;
21
22  typedef struct {
23      byte opCode;
24      long targetCubeCode;
25      uint64_t pipe;
26      short channel;
27      long cubeCode;
28  } COMMAND;
29  COMMAND commands;
30
31
```

```

33   MPU6050 gyro_mpu;
34   // MPU control/status vars
35   bool gyro_dmpReady = false; // set true if DMP init was
36   // successful
37   uint16_t gyro_packetSize; // expected DMP packet size (
38   // default is 42 bytes)
39   uint16_t gyro_fifoCount; // count of all bytes currently
40   // in FIFO
41   uint8_t gyro_fifoBuffer[64]; // FIFO storage buffer
42   Quaternion gyro_q; // [w, x, y, z]
43   gyro_quaternion container
44   VectorFloat gyro_gravity;
45   float gyro_euler[3]; // [psi, theta, phi] Euler
46   // angle container Z Y X
47
48   bool upsidedown = false;
49   short oldSide = 0;
50   short side = 0;
51   boolean sideChanged = false;
52   short treshold = 30;
53   long lastSideChangeTime = 0;
54   long sideChangeTolerance = 700;
55
56   bool debug = false;
57
58   const short AXIS_Z = 0;
59   const short AXIS_Y = 1;
60   const short AXIS_X = 2;
61
62   void setupGyro() {
63     // join I2C bus (I2Cdev library doesn't do this automatically)
64     #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
65       Wire.begin();
66       TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
67     #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
68       Fastwire::setup(400, true);
69     #endif
70
71     gyro_mpu.initialize();
72     gyro_mpu.testConnection();
73     // supply your own gyro offsets here, scaled for min
74     // sensitivity
75     gyro_mpu.setXAccelOffset(-967);
76     gyro_mpu.setYAccelOffset(-502);
77     gyro_mpu.setZAccelOffset(1252);
78     gyro_mpu.setXGyroOffset(142);
79     gyro_mpu.setYGyroOffset(82);
80     gyro_mpu.setZGyroOffset(22);

```

```

75 // make sure it worked (returns 0 if so)
76 if (gyro_mpu.dmpInitialize() == 0) {
77     gyro_mpu.setDMPEnabled(true);
78     gyro_dmpReady = true;
79     gyro_packetSize = gyro_mpu.dmpGetFIFOPacketSize();
80 } else {
81     Serial.print(F("DMP Initialization failed"));
82 }
83 }

84 void setupRadioWrite() {
85
86     radio.begin();
87     radio.setChannel(channel);
88
89     radio.setDataRate(RF24_250KBPS);
90     radio.setPALevel(RF24_PA_MAX);
91
92     radio.openWritingPipe(pipe);
93 }
94

95 void setupRadioRead() {
96     radio.begin();
97     radio.setChannel(channel);
98
99
100    radio.setDataRate(RF24_250KBPS);
101
102    radio.setPALevel(RF24_PA_MAX);
103
104    radio.openReadingPipe(1, pipe);
105    radio.startListening();
106
107 }

108 void gyroMainLoop() {
109     if (!gyro_dmpReady) return;
110     gyro_mpu.resetFIFO();
111     gyro_fifoCount = gyro_mpu.getFIFOCount();
112     int i = 0;
113     while (gyro_fifoCount < gyro_packetSize) {
114         i++;
115         if (i > 10000) {
116             //Wire.detach();
117             setupGyro();
118             return;
119         }
120         gyro_fifoCount = gyro_mpu.getFIFOCount();
121     }
122     gyro_mpu.getFIFOBytes(gyro_fifoBuffer, gyro_packetSize);
123 }
```

```
125     gyro_fifoCount == gyro_packetSize;
126     gyro_mpu.dmpGetQuaternion(&gyro_q, gyro_fifoBuffer);
127     gyro_mpu.dmpGetGravity(&gyro_gravity, &gyro_q);
128     gyro_mpu.dmpGetYawPitchRoll(gyro_euler, &gyro_q, &gyro_gravity
129         );
130     //gyro_mpu.dmpGetEuler(gyro_euler, &gyro_q);
131     gyro_euler[0] = gyro_euler[0] * 180/M_PI;
132     gyro_euler[1] = gyro_euler[1] * 180/M_PI;
133     gyro_euler[2] = gyro_euler[2] * 180/M_PI;
134     upsidedown = (gyro_gravity.z > 0)? false : true;
135     if(debug){
136         Serial.print("euler\t");
137         Serial.print(gyro_euler[AXIS_X]);
138         Serial.print("\t");
139         Serial.print(gyro_euler[AXIS_Y]);
140         Serial.print("\t");
141         Serial.print(gyro_euler[AXIS_Z]);
142         Serial.print("\t");
143         Serial.print(gyro_gravity.z);
144         Serial.print("\t");
145         Serial.print(side);
146         Serial.print("\t");
147     }
148
149     float deg(float source, float target){
150         return abs(source - target);
151     }
152
153     void broadcastSideChange(){
154         cubeInfo.side = side;
155         radio.stopListening();
156         setupRadioWrite();
157         if(radio.write(&cubeInfo, sizeof(cubeInfo))){
158             Serial.println("TRANSMIT OK");
159         } else{
160             Serial.println("FAILED");
161         }
162         Serial.print("side    ");
163         Serial.println(side);
164         setupRadioRead();
165     }
166
167     void onSideChange(){
168         sideChanged = true;
169         lastSideChangeTime = millis();
170     }
171 }
```

```

173     void setSide( short newSide){
174         oldSide = side;
175         side = newSide;
176         if( oldSide != side){ // side change event
177             onSideChange();
178         }
179     }
180
181     void checkup(){
182         if( deg(gyro_euler[AXIS_X], 0) < threshold && deg(gyro_euler[
183             AXIS_Y], 0) < threshold && !upsidedown){
184             setSide(1);
185         } else if( deg(gyro_euler[AXIS_X], -90) < threshold && deg(
186             gyro_euler[AXIS_Y], 0) < threshold){
187             setSide(2);
188         } else if( deg(gyro_euler[AXIS_X], 90) < threshold && deg(
189             gyro_euler[AXIS_Y], 0) < threshold){
190             setSide(3);
191         } else if( deg(gyro_euler[AXIS_X], 0) < threshold && deg(
192             gyro_euler[AXIS_Y], 90) < threshold){
193             setSide(4);
194         } else if( deg(gyro_euler[AXIS_X], 0) < threshold && deg(
195             gyro_euler[AXIS_Y], -90) < threshold){
196             setSide(5);
197         } else if( deg(gyro_euler[AXIS_X], 0) < threshold && deg(
198             gyro_euler[AXIS_Y], 0) < threshold && upsidedown){
199             setSide(6);
200         }
201     }
202
203     void setup() {
204         Serial.begin(9600);
205         setupGyro();
206         setupRadioRead();
207         cubeInfo(cubeCode = 1234567890;
208
209         debug = false;
210     }
211
212     void execCommand(){
213         if( cubeInfo(cubeCode != commands.targetCubeCode){
214             return;
215         }
216
217         if( commands.opCode == 1){
218             cubeInfo(cubeCode = commands(cubeCode;
219             Serial.println("cube code changed");
220         } else if( commands.opCode == 2){
221             channel = commands.channel;
222         }
223     }

```

```

215   radio.stopListening();
216   setupRadioRead();
217   Serial.println("cube channel changed");
218 }
219 }

220 void loop() {
221   gyroMainLoop();
222   checkup();

223   if (sideChanged && (millis() - lastSideChangeTime) >
224       sideChangeTolerance){
225     sideChanged = false;
226     broadcastSideChange();
227   }

228   if (radio.available()){
229     radio.read(&commands, sizeof(commands));
230     Serial.print("opCode: ");
231     Serial.print(commands.opCode);
232     Serial.print(" target: ");
233     Serial.print(commands.targetCubeCode);
234     //Serial.print(" pipe: ");
235     //Serial.print(commands.pipe);
236     Serial.print(" channel: ");
237     Serial.print(commands.channel);
238     Serial.print(" cube: ");
239     Serial.println(commands(cubeCode));
240     execCommand();
241   }
242   delay(1);
243 }
244 }
```

### Sunucu Kodu

```

1 #include <SPI.h>
2 #include "RF24.h"
3 //#include "DigitalIO.h"
4 #include <nRF24L01.h>
5
6 short channel = 120;
7 const uint64_t pipe = 0xF0F1F2F3F4LL;
8
9 String args = "";
10
11 RF24 radio(9, 10); // CE, CSN
```

```
13     typedef struct {
14         long cubeCode;
15         int side;
16     }INFO;
17     INFO myCMD;
18
19     typedef struct {
20         byte opcode;
21         long targetCubeCode;
22         uint64_t pipe;
23         short channel;
24         long cubeCode;
25     }COMMAND;
26     COMMAND commands;
27
28
29     void setupRadioWrite() {
30         radio.begin();
31
32         radio.setChannel(channel);
33
34         radio.setDataRate(RF24_250KBPS);
35
36         radio.setPALevel(RF24_PA_MAX);
37
38         radio.openWritingPipe(pipe);
39     }
40
41
42     void setupRadioRead() {
43         radio.begin();
44         radio.setChannel(channel);
45         radio.setDataRate(RF24_250KBPS);
46         radio.setPALevel(RF24_PA_MAX);
47         radio.openReadingPipe(1, pipe);
48         radio.startListening();
49     }
50
51     boolean sendMessage() {
52         boolean result = false;
53         radio.stopListening();
54         setupRadioWrite();
55         if (radio.write(&commands, sizeof(commands))) {
56             Serial.println("TRANSMIT OK");
57             result = true;
58         } else {
59             Serial.println("FAILED");
```

```
61    }
62    setupRadioRead();
63    return result;
64 }

65 void setup() {
66     Serial.begin(9600);
67     commands.targetCubeCode = 1234567890;
68     commands.opcode = 1;
69     commands(cubeCode = 987650;
70     setupRadioRead();
71     Serial.println("Receiver init ok");
72 }

73

74

75 String getValue(String data, char separator, int index)
76 {
77     int found = 0;
78     int strIndex[] = {0, -1};
79     int maxIndex = data.length() - 1;

80     for (int i=0; i<=maxIndex && found<=index; i++){
81         if (data.charAt(i)==separator || i==maxIndex){
82             found++;
83             strIndex[0] = strIndex[1]+1;
84             strIndex[1] = (i == maxIndex) ? i+1 : i;
85         }
86     }
87 }

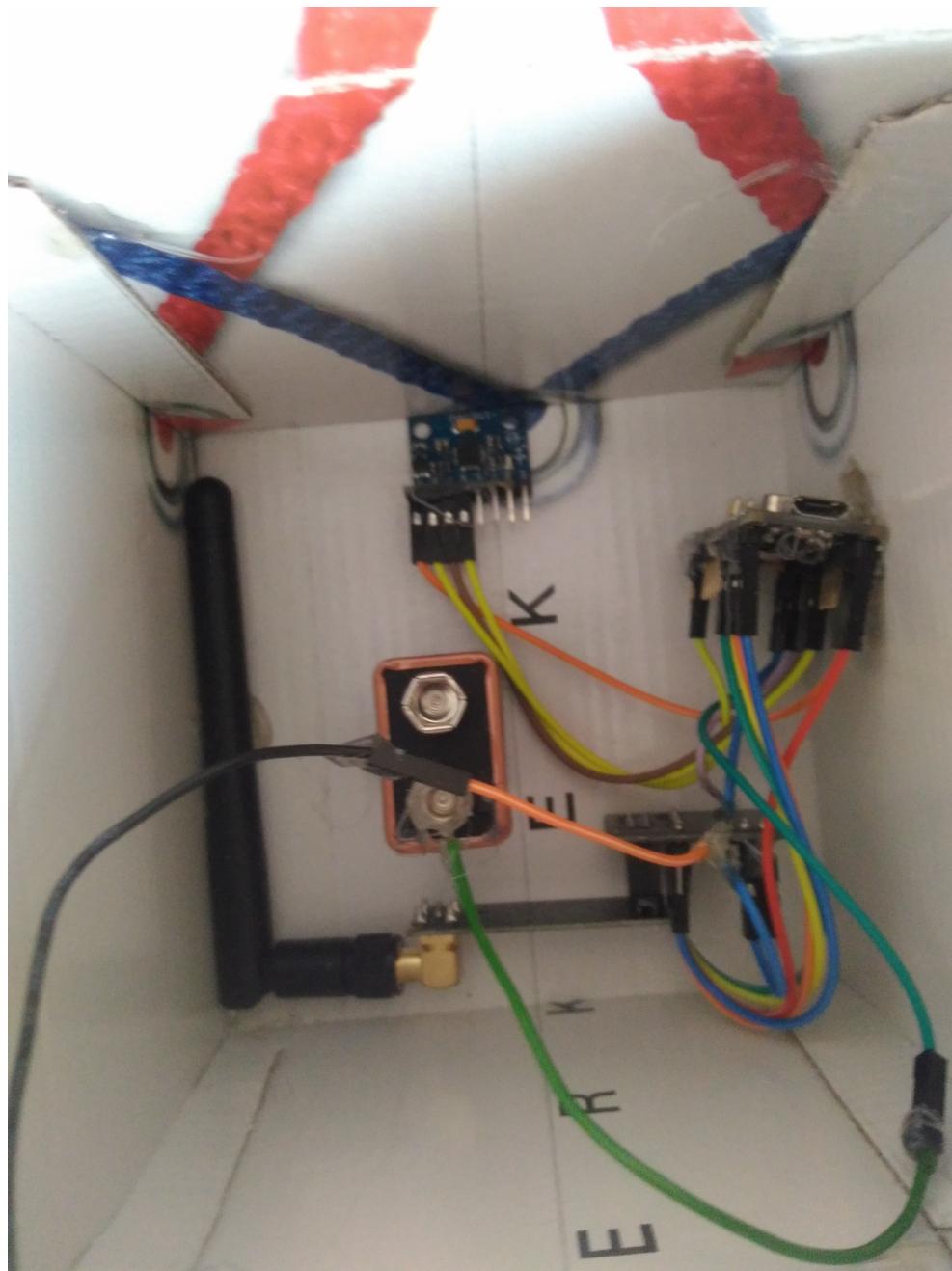
88     return found>index ? data.substring(strIndex[0], strIndex[1])
89     : "";
90 }

91 void loop(){
92     /*if (state && digitalRead(5)){
93         state = false;
94         sendMessage();
95     }
96     */
97     if (Serial.available() != 0){
98         args = Serial.readString();
99         if (getValue(args, ' ', 0) == "rename"){
100            commands.opcode = 1;
101            commands.targetCubeCode = atol(getValue(args, ' ', 1).c_str());
102            ;
103            commands(cubeCode = atol(getValue(args, ' ', 2).c_str());
104            sendMessage();
105        }
106        if (getValue(args, ' ', 0) == "newchannel"){
107            commands.opcode = 2;
```

```
109     commands.targetCubeCode = atol(getValue(args, 1, 1).c_str());
110     ;
111     commands.channel = atoi(getValue(args, 1, 2).c_str());
112     if(sendMessage()){
113         channel = commands.channel;
114         radio.stopListening();
115         setupRadioRead();
116     }
117     if (radio.available()){
118         radio.read(&myCMD, sizeof(myCMD));
119         Serial.print("data: ");
120         Serial.print(myCMD.side);
121         Serial.print("    cube: ");
122         Serial.println(myCMD(cubeCode));
123     }
124 }
```



Sekil 7.28: Akilli kablosuz zar



Şekil 7.29: Akıllı kablosuz zar

## 7.16 LCD Ekran ve Butonlarla Şifre Girişi

### Gerekli malzeme

- Arduino Uno
- 16x2 LCD Ekran
- 4x3 Buton Matrisi
- LCD ekranın karşılığını belirlemek için bir direnç
- Hangi satır ve sütüna basıldığı bulmak için 4 farklı değerde direnç
- Pull-down resistor olarak kullanılacak 3 direnç

Giriş Kurulan sistem, LCD ekran ve buton matrisi yardımıyla şifre girişine olanak sağlar.

### Kaynak Kodu

```
#include <LiquidCrystal.h>
2 int a0 = 0, a1 = 0, a2 = 0;
4 int tolerance = 3;
5 String passwordEntered = "";
6 String passwordRequired = "1997";
7 constint rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
8 LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

10 char pad [3][4] = {{'1','4','7','*'}, {'2','5','8','0'}, {'3','6',
11     , '9', '#'}};
12 int values [] = {930,956,1010,1020};
13 char key = 0;

14 void setup()
{
16 pinMode(A0, INPUT);
17 pinMode(A1, INPUT);
18 pinMode(A2, INPUT);

20 lcd.begin(16, 2);
21 lcd.print("Enter password:");
22 lcd.setCursor(0, 1);
23 }
24 voidloop() {
```

```

26 a0 = analogRead(A0);
27 a1 = analogRead(A1);
28 a2 = analogRead(A2);
29 if(a0 >900){ // Keys 1,4,7,*
30 key = pad[0][getKeyRow(a0)];
31 } elseif(a1 >900){ // Keys, 2,5,8,0
32 key = pad[1][getKeyRow(a1)];
33 } elseif(a2 >900){ // Keys 3,6,9,#
34 key = pad[2][getKeyRow(a2)];
35 } else {
36 if(key != 0){
37 if(key != '#'){
38 lcd.print(key);
39 passwordEntered += key;
40 } else {
41 lcd.clear();
42 if(passwordEntered.equals(passwordRequired)){
43 lcd.print("Access granted!");
44 } else {
45 lcd.print("Access denied!");
46 }
47 delay(1500);
48 lcd.clear();
49 lcd.print("Enter password:");
50 lcd.setCursor(0, 1);
51 passwordEntered = "";
52 }
53 key = 0;
54 }
55 }
56 delay(100);
57 }
58 // Returns true if x is around y by given tolerance value
59 boolean around(int x, int y){
60   return (x >= y - tolerance) && x <= (y + tolerance);
61 }

63 int getKeyRow(int value){
64   int i = 0;
65   for(; i <4; i++){
66     if(around(value, values[i])){
67       break;
68     }
69   }
70   return i;
71 }
72 }
```

İlk kısımda a0, a1, a2 olarak analoglardan okunacak değeri tutacak değişkenler tanımlanmış, okumada meydana gelebilecek hataları kapsayacak bir tolerans değeri verilmiş, şifre değişkenleri tanımlanmış ve son olarak LCD ekran için gerekli pin değerleri ve obje ilklenmiştir.

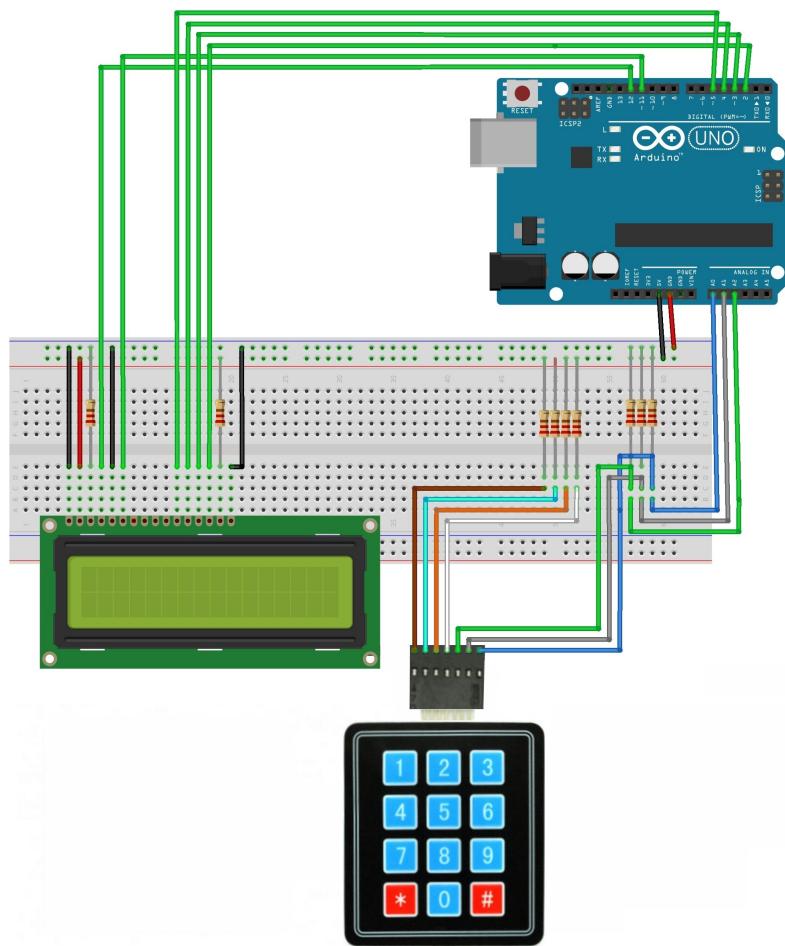
setup() kısmında A0, A1 ve A2 analog pinleri input olarak işaretlenmiş, lcd kütüphanesi yardımıyla tanımlanmış olan lcd objesi gerekli metodlarla ilklenmiştir.

loop() kısmında önce analogRead() fonksiyonuyla analog okuma yapılır. Bu aşamadan sonra basılan butonun hangi sütunda olduğu kontrol edilir. getKeyRow() fonksiyonuyla basılan butonun satır sayısı da tespit edilir basılan buton key değişkenine kaydedilir. Buton bırakıldıktan sonra kodun else kısmı devam eder. Bu kısımda if(key != 0) kontrolüyle daha önceden tuşa basılıp basılmadığı anlaşılır. Eğer tuşa basılmışsa, kullanıcının şifre giriş işlemeye devam mı ettiği, yoksa bu işlemi tamamlayıp sonuç mu beklediği basılan tuşun # olup olmaması kontrolüyle bulunur. Eğer kullanıcı şifre giriş işlemeye devam ediyorsa bastığı tuş LCD ekrana yazdırılır ve passwordEntered değişkenine eklenir. Eğer kullanıcı sonuç bekliyorsa passwordEntered değişkeninde tutulan şifrenin önceden belirlenmiş şifreyle aynı olup olmadığı kontrol edilir ve buna uygun bir mesaj LCD ekranda gösterilir. Basılan tuş # olsa da olmasa da key = 0; komutuyla en son basılan tuş sıfırlanır.

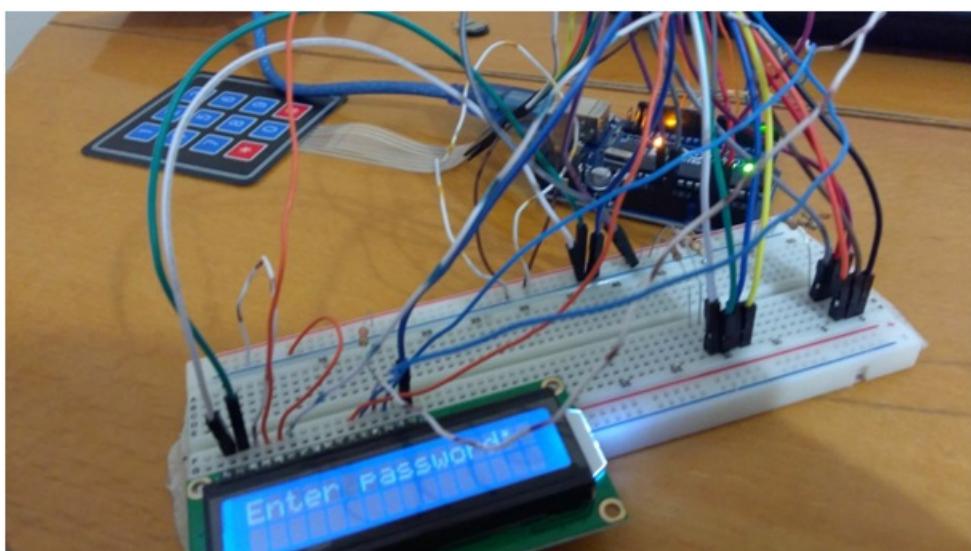
LCD bağlantısı ile ilgili detaylar <https://www.arduino.cc/en/Tutorial>HelloWorld> adresinde bulunabilir. Keypad'den değer okumak için analog pin kullanılmasının nedeni dijital pinlerden tasarruf etmektir. Çalışma prensibi şu biçimdedir:

Eğer hiç bir tuşa basılmiyorsa hiç bir satır ve sütun birleşmez, yüksek empedans okumamak için kullanılan pull-down dirençleri sayesinde sıfır okunur.

Eğer bir tuşa basılıyorsa basılan satır ve sütun birleşir. Basılan sütuna bağlı analog pinden gelen değer okunur. Her satır için farklı direnç değeri kullanıldığından yalnızca 3 analog pinle hangi sayıya basıldığı tespit edilebilir.



Sekil 7.30: Devre şeması



Sekil 7.31: LCD Ekran ve Butonlarla Şifre Girişи

## 7.17 Morse Alfabe Çeviricisi

### Gerekli malzeme

- Arduino Uno
- 2x 4 bacaklı buton
- 2x Gösterge LED
- Piezo Buzzer
- Pull-down resistor olarak kullanılacak 2 direnç

### Giriş

Kurulan sistem iki moda sahiptir. Giriş modundayken, butona basılma durumuna göre morse alfabetesiyle girilen harfleri algılar ve bilgisayara gönderir. Çıkış modunda ise bilgisayardan aldığı yazıyı buzzer yardımıyla morse alfabesinde çalar.

### Kaynak Kodu

```

1   bool isPressed = false, isReleased = false, gotLetter = false
2   , isModeChanged = true;
3   longint pressedTime = 0, releasedTime = 0;
4   int buttonPin = 8, buzzerPin = 2, ledYellowPin = 3,
5   ledRedPin = 4, modeButtonPin = 5, mode;
6   String str = "";
7   int RECEIVE_MODE = 0, SEND_MODE = 1;
8   String alphabet[36] = { /*A*/".-", /*B*/"-..", /*C*/"-.-.", ...
9   . . . };
10  voidsetup() {
11    pinMode(buttonPin, INPUT);
12    pinMode(ledYellowPin, OUTPUT);
13    pinMode(ledRedPin, OUTPUT);
14    pinMode(modeButtonPin, INPUT);
15    Serial.begin(9600);
16    mode = SEND_MODE;
17  }
18  voidloop() {
19    if(digitalRead(modeButtonPin) == HIGH){
20      isModeChanged = false;
21    } else {
22      if(!isModeChanged){
23        mode = (mode + 1) % 2;
24        playDot();
25      }
26    }
27  }
28  voidplayDot() {
29    digitalWrite(buzzerPin, HIGH);
30    delay(pressedTime);
31    digitalWrite(buzzerPin, LOW);
32    delay(releasedTime);
33  }
34  voidplayDash() {
35    digitalWrite(buzzerPin, HIGH);
36    delay(pressedTime * 3);
37    digitalWrite(buzzerPin, LOW);
38    delay(releasedTime);
39  }
40  voidplayLetter(String letter) {
41    if(letter == ".") {
42      playDot();
43    } else if(letter == "-") {
44      playDash();
45    }
46  }
47  voidplayWord() {
48    if(alphabet[mode] == ".") {
49      playDot();
50    } else if(alphabet[mode] == "-") {
51      playDash();
52    }
53  }
54  voidplayMessage(String message) {
55    for(int i = 0; i < message.length(); i++) {
56      playLetter(message[i]);
57    }
58  }
59  voidplayMode() {
60    if(mode == RECEIVE_MODE) {
61      playWord();
62    } else if(mode == SEND_MODE) {
63      playMessage("SEND MODE");
64    }
65  }
66  voidplayError() {
67    playMessage("ERROR");
68  }
69  voidplayHelp() {
70    playMessage("HELP");
71  }
72  voidplayAbout() {
73    playMessage("ABOUT");
74  }
75  voidplayVersion() {
76    playMessage("VERSION");
77  }
78  voidplayExit() {
79    playMessage("EXIT");
80  }
81  voidplayUnknown() {
82    playMessage("UNKNOWN");
83  }
84  voidplayUnknownChar() {
85    playMessage("UNKNOWN CHAR");
86  }
87  voidplayUnknownMode() {
88    playMessage("UNKNOWN MODE");
89  }
90  voidplayUnknownWord() {
91    playMessage("UNKNOWN WORD");
92  }
93  voidplayUnknownLetter() {
94    playMessage("UNKNOWN LETTER");
95  }
96  voidplayUnknownDash() {
97    playMessage("UNKNOWN DASH");
98  }
99  voidplayUnknownDot() {
100   playMessage("UNKNOWN DOT");
101 }
102 
```

```
22 playDash();
23 isModeChanged = true;
24 }
25 }
26 if (mode == SEND_MODE){
27   digitalWrite(ledYellowPin, HIGH);
28   digitalWrite(ledRedPin, LOW);
29   int buttonState = digitalRead(buttonPin);
30   if (buttonState == HIGH){
31     if (!isPressed){
32       gotLetter = false;
33       pressedTime = millis();
34       isPressed = true;
35     }
36     isReleased = false;
37   } else {
38     if (!isReleased){
39       releasedTime = millis();
40       isReleased = true;
41       if (millis() - pressedTime >120){ // If button is pressed for
42         str += "-";                         // more than 200 ms
43       } elseif (millis() - pressedTime >40){ // If button is pressed
44         str += ".";                         // for more than 100 ms
45       } else {
46         if (!gotLetter && (millis() - releasedTime >400)){
47           for (int j = 0; j <36; j++){
48             if (alphabet[j] == str){
49               char c;
50               if (j <26){
51                 c = 'A' + j;
52               } else {
53                 c = '0' + j - 26;
54               }
55               Serial.print(c);
56               break;
57             }
58           }
59           str = "";
60           gotLetter = true;
61         }
62       }
63       isPressed = false;
64     }
65   } elseif (mode == RECEIVE_MODE){
66   digitalWrite(ledYellowPin, LOW);
67   digitalWrite(ledRedPin, HIGH);
```

```

70  if (Serial.available() >0) {
71    char c = Serial.read();
72    if(c != ' '){
73      c = toUpperCase(c);
74      if(c >= 'A'&& c <= 'Z'){
75        c -= 'A';
76      } elseif(c >= '0'&& c <= '9'){
77        c -= '0';
78      }
79      String s = alphabet[c];
80      for(int i = 0; i < s.length(); i++){
81        if(s.charAt(i) == '.'){
82          playDot();
83        } else {
84          playDash();
85        }
86      }
87    }
88  }
89}
90}
91}
92voidplayDot(){
93  tone(buzzerPin, 500, 80);
94  delay(80);
95}
96voidplayDash(){
97  tone(buzzerPin, 1000, 240);
98  delay(240);
99}

```

İlk kısımda eleman pinleri, morse alfabetesindeki harfleri İngiliz alfabetesine eşleştirilen bir alfabe arrayi ve program akışında kullanılacak değişkenler(gotLetter, pressedTime, vb.) tanımlanmıştır.

setup() kısmında gerekli pinler INPUT ya da OUTPUT olarak işaretlenip Serial.begin(9600) komutuyla bağlantı başlatılmıştır.

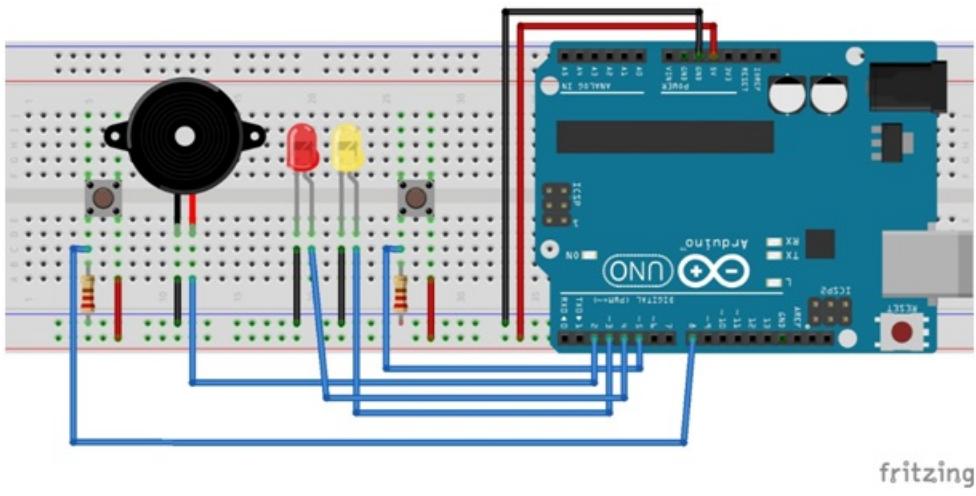
loop() kısmında ilk yapılan iş modun değişim değişmediğini butonu okuyarak kontrol etmektedir. Eğer mod değiştiyse kullanıcı buzzer'dan çalınan melodiyile haberدار edilir. Ondan sonra, şu anki modun hangisi olduğuna göre yapılacak eylemler farklılaşır.

### Giriş Modu

Yapılan ilk iş sarı LED'i yakıp diğerini söndürmektir. Ondan sonraki eylem giriş butonunu okumaktır. Bu aşamadan sonra butona ne kadar süreyle basılıp bırakıldığına göre girilen harf belirlenecek ve bu harf bilgisayarda ekranda göstermek üzere gönderilecektir.

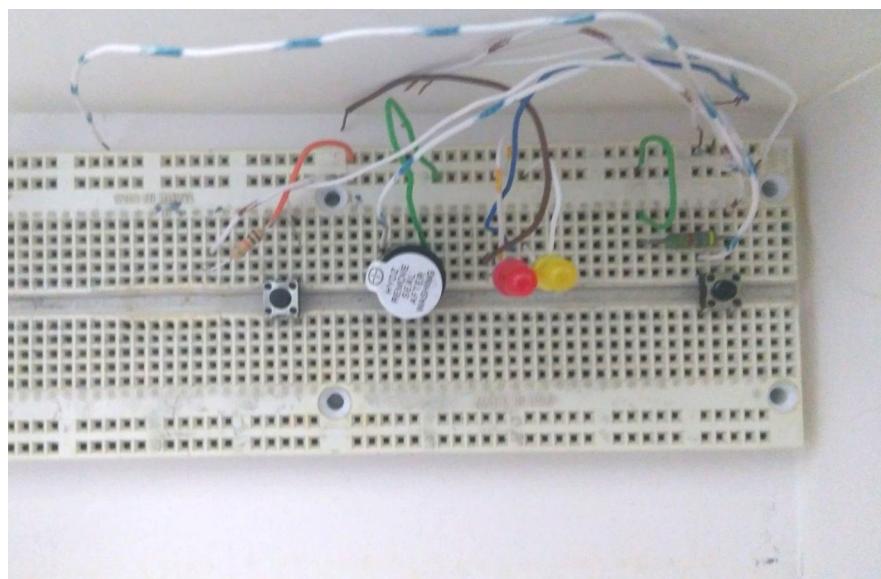
### Cıkış Modu

Yapılan ilk iş kırmızı LED'i yakıp diğerini söndürmektir. Ondan sonra, Serial sınıfını kullanarak bilgisayardan girilen bir karakter okur. Alfabe dizisi, playDot() ve playDash() fonksiyonları yardımıyla okunan bu harfi Morse alfabetesine uygun olarak buzzer'da çalar.



Şekil 7.32: Devre Şeması

Kaynak kodunda belirtilen pinlere gerekli elemanlar bağlanmıştır. Butonlar için diğer projelerde olduğu gibi pull-down dirençleri kullanılmıştır.



Şekil 7.33: Morse Alfabe Çeviricisi

## 7.18 Tilt Mouse

### Gerekli malzeme

- Arduino Uno
- ADXL345 ivmeölçer
- 2x 10k ohm direnç
- Pull-down resistor olarak kullanılacak 2 direnç

### Giriş

Kurulan sistem, bir ivmeölçer iki buton ve java programı yardımıyla, modülü eğerek(tilt) mouse kontrolü sağlamaktadır.

### Kaynak Kodu – Arduino

```
1 #include <Wire.h>

3 #define DEVICE (0x53)      //ADXL345 device address
# define TO_READ (6)        //num of bytes we are going to read
                           //each time (two bytes for each axis)
5 byte buff[TO_READ] ;     //6 bytes buffer for saving data read
                           //from the device
7 char str[512];           //string buffer to
                           //transform data before sending it to the serial port
9 int lButtonPin = 2;
int rButtonPin = 3;
void setup()
{
11   pinMode(lButtonPin, INPUT);
13   pinMode(rButtonPin, INPUT);
14   Wire.begin();           // join i2c bus (address optional for
                           // master)
15   Serial.begin(9600);    // start serial for output
16   //Turning on the ADXL345
17   writeTo(DEVICE, 0x2D, 0);
18   writeTo(DEVICE, 0x2D, 16);
19   writeTo(DEVICE, 0x2D, 8);
20 }
21
22 void loop()
23 {
24   int regAddress = 0x32;  //first axis-acceleration-data
                           //register on the ADXL345
```

```

25    int x, y, z;

27    readFrom(DEVICE, regAddress, TO_READ, buff); //read the
        acceleration data from the ADXL345

29    //each axis reading comes in 10 bit resolution, ie 2 bytes.
        Least Significant Byte first!!
30    //thus we are converting both bytes in to one int
31    x = (((int)buff[1]) <<8) | buff[0];
32    y = (((int)buff[3]) <<8) | buff[2];
33    z = (((int)buff[5]) <<8) | buff[4];

35    //we send the x y z values as a string to the serial port
36    sprintf(str, "%d %d %d", x, y, z);

37

39    if(digitalRead(lButtonPin) == HIGH){
40        strcat(str, ":l[p]");
41    } else {
42        strcat(str, ":l[r]");
43    }

45    if(digitalRead(rButtonPin) == HIGH){
46        strcat(str, ":r[p]");
47    } else {
48        strcat(str, ":r[r]");
49    }

51    Serial.println(str);

53    //It appears that delay is needed in order not to clog the
        port
54    delay(15);
55}

57    //————— Functions
58    //Writes val to address register on device
59    voidwriteTo(int device, byte address, byte val) {
60        Wire.beginTransmission(device); //start transmission to device
61        Wire.write(address);          // send register address
62        Wire.write(val);             // send value to write
63        Wire.endTransmission();      //end transmission
64    }

65    //reads num bytes starting from address register on device in
        to buff array
66    voidreadFrom(int device, byte address, int num, byte buff[]) {
67        Wire.beginTransmission(device); //start transmission to device
68        Wire.write(address);          //sends address to read from
69    }

```

```

71   Wire.endTransmission(); //end transmission
72
73   Wire.beginTransmission(device); //start transmission to device
74   Wire.requestFrom(device, num); // request 6 bytes from
75   device
76
77   int i = 0;
78   while(Wire.available()) //device may send less than
79   requested (abnormal)
80   {
81     buff[i] = Wire.read(); // receive a byte
82     i++;
83   }
84   Wire.endTransmission(); //end transmission
85 }
```

Arduino kısmında yapılan tek iş, sensörden okunan veriyi Serial sınıfı yardımıyla bilgisayara iletmektir. Sensör ve mikrodenetleyici arasındaki iletişim için i2c kullanılmıştır. Bu, Wire.h kütüphanesiyle mümkün olmuştur.

### Kaynak Kodu – Java

```

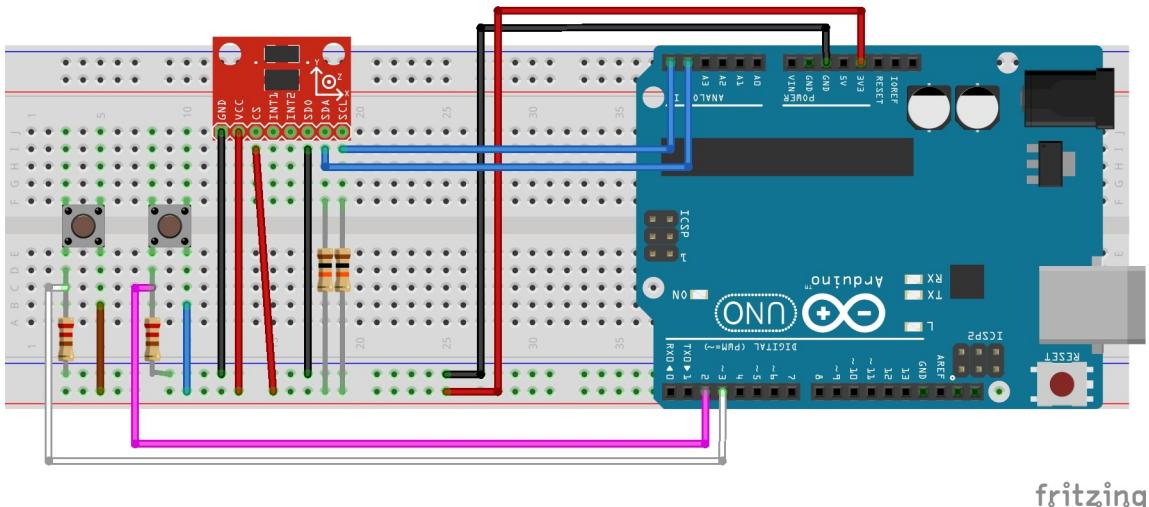
2   public class TiltMouse {
3     private SerialPort comPort;
4     private String portDescriptor;
5     private int baudRate;
6     private Robot robot;
7     private float x, y, z;
8     private double sensitivity = 0.05;
9     private Scanner scanner;
10    private boolean stop;
11
12    public TiltMouse(String portDescriptor, int baudRate) throws
13      AWTException {
14      this.portDescriptor = portDescriptor;
15      this.baudRate = baudRate;
16      this.robot = new Robot();
17      this.x = 0;
18      this.y = 0;
19      this.z = 0;
20      this.stop = false;
21
22      comPort = SerialPort.getCommPort(portDescriptor);
23      comPort.setBaudRate(baudRate);
24      comPort.setComPortTimeouts(SerialPort.TIMEOUT_SCANNER, 100, 0)
25    ;}
```

```

24
25     }
26
27     public void start() {
28         comPort.openPort();
29         scanner = new Scanner(comPort.getInputStream());
30         while(scanner.hasNext() && !stop) {
31             String readString = scanner.nextLine();
32             String[] splitted = readString.split(" ");
33             String[] splittedLR = null;
34             if(splitted.length<3) {
35                 continue;
36             }
37             try {
38                 x = (float) -(Integer.parseInt(splitted[0]));
39                 y = (float) -(Integer.parseInt(splitted[1]));
40                 splittedLR = splitted[2].split(":");
41                 z = (float) -(Integer.parseInt(splittedLR[0]));
42             } catch(NumberFormatException e) {
43                 continue;
44             }
45
46             PointerInfo a = MouseInfo.getPointerInfo();
47             Point b = a.getLocation();
48             int mousex = (int) b.getX();
49             int mousey = (int) b.getY();
50
51             if(splittedLR.length == 3) {
52                 if(splittedLR[2].equals("r[p]")) {
53                     robot.mousePress(MouseEvent.BUTTON3_DOWN_MASK);
54                     robot.mouseRelease(MouseEvent.BUTTON3_DOWN_MASK);
55                 } elseif(splittedLR[2].equals("r[r]")) {
56
57                 }
58                 if(splittedLR[1].equals("l[p]")) {
59                     robot.mousePress(MouseEvent.BUTTON1_DOWN_MASK);
60                 } elseif(splittedLR[1].equals("l[r]")) {
61                     robot.mouseRelease(MouseEvent.BUTTON1_DOWN_MASK);
62                 }
63
64             robot.mouseMove(mousex + (int)(y*sensitivity), mousey + (int)(
65                 x*sensitivity));
66             scanner.close();
67             comPort.closePort();
68         }
69         public double getSensitivity() {
70             return sensitivity;
71         }
72     }
73
74     public void stop() {
75         stop = true;
76     }
77
78     public void setSensitivity(double sensitivity) {
79         this.sensitivity = sensitivity;
80     }
81
82     public void setComPort(CommPort comPort) {
83         this.comPort = comPort;
84     }
85
86     public void setStop(boolean stop) {
87         this.stop = stop;
88     }
89
90     public void setScanner(Scanner scanner) {
91         this.scanner = scanner;
92     }
93
94     public void setX(float x) {
95         this.x = x;
96     }
97
98     public void setY(float y) {
99         this.y = y;
100    }
101
102    public void setZ(float z) {
103        this.z = z;
104    }
105
106    public void setSensitivty(double sensitivty) {
107        this.sensitivty = sensitivty;
108    }
109
110    public void setRobot(Robot robot) {
111        this.robot = robot;
112    }
113
114    public void setMouseInfo(PointerInfo mouseInfo) {
115        this.mouseInfo = mouseInfo;
116    }
117
118    public void setComPort(CommPort comPort) {
119        this.comPort = comPort;
120    }
121
122    public void setScanner(Scanner scanner) {
123        this.scanner = scanner;
124    }
125
126    public void setStop(boolean stop) {
127        this.stop = stop;
128    }
129
130    public void setX(float x) {
131        this.x = x;
132    }
133
134    public void setY(float y) {
135        this.y = y;
136    }
137
138    public void setZ(float z) {
139        this.z = z;
140    }
141
142    public void setSensitivty(double sensitivty) {
143        this.sensitivty = sensitivty;
144    }
145
146    public void setRobot(Robot robot) {
147        this.robot = robot;
148    }
149
150    public void setMouseInfo(PointerInfo mouseInfo) {
151        this.mouseInfo = mouseInfo;
152    }
153
154    public void setComPort(CommPort comPort) {
155        this.comPort = comPort;
156    }
157
158    public void setScanner(Scanner scanner) {
159        this.scanner = scanner;
160    }
161
162    public void setStop(boolean stop) {
163        this.stop = stop;
164    }
165
166    public void setX(float x) {
167        this.x = x;
168    }
169
170    public void setY(float y) {
171        this.y = y;
172    }
173
174    public void setZ(float z) {
175        this.z = z;
176    }
177
178    public void setSensitivty(double sensitivty) {
179        this.sensitivty = sensitivty;
180    }
181
182    public void setRobot(Robot robot) {
183        this.robot = robot;
184    }
185
186    public void setMouseInfo(PointerInfo mouseInfo) {
187        this.mouseInfo = mouseInfo;
188    }
189
190    public void setComPort(CommPort comPort) {
191        this.comPort = comPort;
192    }
193
194    public void setScanner(Scanner scanner) {
195        this.scanner = scanner;
196    }
197
198    public void setStop(boolean stop) {
199        this.stop = stop;
200    }
201
202    public void setX(float x) {
203        this.x = x;
204    }
205
206    public void setY(float y) {
207        this.y = y;
208    }
209
210    public void setZ(float z) {
211        this.z = z;
212    }
213
214    public void setSensitivty(double sensitivty) {
215        this.sensitivty = sensitivty;
216    }
217
218    public void setRobot(Robot robot) {
219        this.robot = robot;
220    }
221
222    public void setMouseInfo(PointerInfo mouseInfo) {
223        this.mouseInfo = mouseInfo;
224    }
225
226    public void setComPort(CommPort comPort) {
227        this.comPort = comPort;
228    }
229
230    public void setScanner(Scanner scanner) {
231        this.scanner = scanner;
232    }
233
234    public void setStop(boolean stop) {
235        this.stop = stop;
236    }
237
238    public void setX(float x) {
239        this.x = x;
240    }
241
242    public void setY(float y) {
243        this.y = y;
244    }
245
246    public void setZ(float z) {
247        this.z = z;
248    }
249
250    public void setSensitivty(double sensitivty) {
251        this.sensitivty = sensitivty;
252    }
253
254    public void setRobot(Robot robot) {
255        this.robot = robot;
256    }
257
258    public void setMouseInfo(PointerInfo mouseInfo) {
259        this.mouseInfo = mouseInfo;
260    }
261
262    public void setComPort(CommPort comPort) {
263        this.comPort = comPort;
264    }
265
266    public void setScanner(Scanner scanner) {
267        this.scanner = scanner;
268    }
269
270    public void setStop(boolean stop) {
271        this.stop = stop;
272    }
273
274    public void setX(float x) {
275        this.x = x;
276    }
277
278    public void setY(float y) {
279        this.y = y;
280    }
281
282    public void setZ(float z) {
283        this.z = z;
284    }
285
286    public void setSensitivty(double sensitivty) {
287        this.sensitivty = sensitivty;
288    }
289
290    public void setRobot(Robot robot) {
291        this.robot = robot;
292    }
293
294    public void setMouseInfo(PointerInfo mouseInfo) {
295        this.mouseInfo = mouseInfo;
296    }
297
298    public void setComPort(CommPort comPort) {
299        this.comPort = comPort;
300    }
301
302    public void setScanner(Scanner scanner) {
303        this.scanner = scanner;
304    }
305
306    public void setStop(boolean stop) {
307        this.stop = stop;
308    }
309
310    public void setX(float x) {
311        this.x = x;
312    }
313
314    public void setY(float y) {
315        this.y = y;
316    }
317
318    public void setZ(float z) {
319        this.z = z;
320    }
321
322    public void setSensitivty(double sensitivty) {
323        this.sensitivty = sensitivty;
324    }
325
326    public void setRobot(Robot robot) {
327        this.robot = robot;
328    }
329
330    public void setMouseInfo(PointerInfo mouseInfo) {
331        this.mouseInfo = mouseInfo;
332    }
333
334    public void setComPort(CommPort comPort) {
335        this.comPort = comPort;
336    }
337
338    public void setScanner(Scanner scanner) {
339        this.scanner = scanner;
340    }
341
342    public void setStop(boolean stop) {
343        this.stop = stop;
344    }
345
346    public void setX(float x) {
347        this.x = x;
348    }
349
350    public void setY(float y) {
351        this.y = y;
352    }
353
354    public void setZ(float z) {
355        this.z = z;
356    }
357
358    public void setSensitivty(double sensitivty) {
359        this.sensitivty = sensitivty;
360    }
361
362    public void setRobot(Robot robot) {
363        this.robot = robot;
364    }
365
366    public void setMouseInfo(PointerInfo mouseInfo) {
367        this.mouseInfo = mouseInfo;
368    }
369
370    public void setComPort(CommPort comPort) {
371        this.comPort = comPort;
372    }
373
374    public void setScanner(Scanner scanner) {
375        this.scanner = scanner;
376    }
377
378    public void setStop(boolean stop) {
379        this.stop = stop;
380    }
381
382    public void setX(float x) {
383        this.x = x;
384    }
385
386    public void setY(float y) {
387        this.y = y;
388    }
389
390    public void setZ(float z) {
391        this.z = z;
392    }
393
394    public void setSensitivty(double sensitivty) {
395        this.sensitivty = sensitivty;
396    }
397
398    public void setRobot(Robot robot) {
399        this.robot = robot;
400    }
401
402    public void setMouseInfo(PointerInfo mouseInfo) {
403        this.mouseInfo = mouseInfo;
404    }
405
406    public void setComPort(CommPort comPort) {
407        this.comPort = comPort;
408    }
409
410    public void setScanner(Scanner scanner) {
411        this.scanner = scanner;
412    }
413
414    public void setStop(boolean stop) {
415        this.stop = stop;
416    }
417
418    public void setX(float x) {
419        this.x = x;
420    }
421
422    public void setY(float y) {
423        this.y = y;
424    }
425
426    public void setZ(float z) {
427        this.z = z;
428    }
429
430    public void setSensitivty(double sensitivty) {
431        this.sensitivty = sensitivty;
432    }
433
434    public void setRobot(Robot robot) {
435        this.robot = robot;
436    }
437
438    public void setMouseInfo(PointerInfo mouseInfo) {
439        this.mouseInfo = mouseInfo;
440    }
441
442    public void setComPort(CommPort comPort) {
443        this.comPort = comPort;
444    }
445
446    public void setScanner(Scanner scanner) {
447        this.scanner = scanner;
448    }
449
450    public void setStop(boolean stop) {
451        this.stop = stop;
452    }
453
454    public void setX(float x) {
455        this.x = x;
456    }
457
458    public void setY(float y) {
459        this.y = y;
460    }
461
462    public void setZ(float z) {
463        this.z = z;
464    }
465
466    public void setSensitivty(double sensitivty) {
467        this.sensitivty = sensitivty;
468    }
469
470    public void setRobot(Robot robot) {
471        this.robot = robot;
472    }
473
474    public void setMouseInfo(PointerInfo mouseInfo) {
475        this.mouseInfo = mouseInfo;
476    }
477
478    public void setComPort(CommPort comPort) {
479        this.comPort = comPort;
480    }
481
482    public void setScanner(Scanner scanner) {
483        this.scanner = scanner;
484    }
485
486    public void setStop(boolean stop) {
487        this.stop = stop;
488    }
489
490    public void setX(float x) {
491        this.x = x;
492    }
493
494    public void setY(float y) {
495        this.y = y;
496    }
497
498    public void setZ(float z) {
499        this.z = z;
500    }
501
502    public void setSensitivty(double sensitivty) {
503        this.sensitivty = sensitivty;
504    }
505
506    public void setRobot(Robot robot) {
507        this.robot = robot;
508    }
509
510    public void setMouseInfo(PointerInfo mouseInfo) {
511        this.mouseInfo = mouseInfo;
512    }
513
514    public void setComPort(CommPort comPort) {
515        this.comPort = comPort;
516    }
517
518    public void setScanner(Scanner scanner) {
519        this.scanner = scanner;
520    }
521
522    public void setStop(boolean stop) {
523        this.stop = stop;
524    }
525
526    public void setX(float x) {
527        this.x = x;
528    }
529
530    public void setY(float y) {
531        this.y = y;
532    }
533
534    public void setZ(float z) {
535        this.z = z;
536    }
537
538    public void setSensitivty(double sensitivty) {
539        this.sensitivty = sensitivty;
540    }
541
542    public void setRobot(Robot robot) {
543        this.robot = robot;
544    }
545
546    public void setMouseInfo(PointerInfo mouseInfo) {
547        this.mouseInfo = mouseInfo;
548    }
549
550    public void setComPort(CommPort comPort) {
551        this.comPort = comPort;
552    }
553
554    public void setScanner(Scanner scanner) {
555        this.scanner = scanner;
556    }
557
558    public void setStop(boolean stop) {
559        this.stop = stop;
560    }
561
562    public void setX(float x) {
563        this.x = x;
564    }
565
566    public void setY(float y) {
567        this.y = y;
568    }
569
570    public void setZ(float z) {
571        this.z = z;
572    }
573
574    public void setSensitivty(double sensitivty) {
575        this.sensitivty = sensitivty;
576    }
577
578    public void setRobot(Robot robot) {
579        this.robot = robot;
580    }
581
582    public void setMouseInfo(PointerInfo mouseInfo) {
583        this.mouseInfo = mouseInfo;
584    }
585
586    public void setComPort(CommPort comPort) {
587        this.comPort = comPort;
588    }
589
590    public void setScanner(Scanner scanner) {
591        this.scanner = scanner;
592    }
593
594    public void setStop(boolean stop) {
595        this.stop = stop;
596    }
597
598    public void setX(float x) {
599        this.x = x;
600    }
601
602    public void setY(float y) {
603        this.y = y;
604    }
605
606    public void setZ(float z) {
607        this.z = z;
608    }
609
610    public void setSensitivty(double sensitivty) {
611        this.sensitivty = sensitivty;
612    }
613
614    public void setRobot(Robot robot) {
615        this.robot = robot;
616    }
617
618    public void setMouseInfo(PointerInfo mouseInfo) {
619        this.mouseInfo = mouseInfo;
620    }
621
622    public void setComPort(CommPort comPort) {
623        this.comPort = comPort;
624    }
625
626    public void setScanner(Scanner scanner) {
627        this.scanner = scanner;
628    }
629
630    public void setStop(boolean stop) {
631        this.stop = stop;
632    }
633
634    public void setX(float x) {
635        this.x = x;
636    }
637
638    public void setY(float y) {
639        this.y = y;
640    }
641
642    public void setZ(float z) {
643        this.z = z;
644    }
645
646    public void setSensitivty(double sensitivty) {
647        this.sensitivty = sensitivty;
648    }
649
650    public void setRobot(Robot robot) {
651        this.robot = robot;
652    }
653
654    public void setMouseInfo(PointerInfo mouseInfo) {
655        this.mouseInfo = mouseInfo;
656    }
657
658    public void setComPort(CommPort comPort) {
659        this.comPort = comPort;
660    }
661
662    public void setScanner(Scanner scanner) {
663        this.scanner = scanner;
664    }
665
666    public void setStop(boolean stop) {
667        this.stop = stop;
668    }
669
670    public void setX(float x) {
671        this.x = x;
672    }
673
674    public void setY(float y) {
675        this.y = y;
676    }
677
678    public void setZ(float z) {
679        this.z = z;
680    }
681
682    public void setSensitivty(double sensitivty) {
683        this.sensitivty = sensitivty;
684    }
685
686    public void setRobot(Robot robot) {
687        this.robot = robot;
688    }
689
690    public void setMouseInfo(PointerInfo mouseInfo) {
691        this.mouseInfo = mouseInfo;
692    }
693
694    public void setComPort(CommPort comPort) {
695        this.comPort = comPort;
696    }
697
698    public void setScanner(Scanner scanner) {
699        this.scanner = scanner;
700    }
701
702    public void setStop(boolean stop) {
703        this.stop = stop;
704    }
705
706    public void setX(float x) {
707        this.x = x;
708    }
709
710    public void setY(float y) {
711        this.y = y;
712    }
713
714    public void setZ(float z) {
715        this.z = z;
716    }
717
718    public void setSensitivty(double sensitivty) {
719        this.sensitivty = sensitivty;
720    }
721
722    public void setRobot(Robot robot) {
723        this.robot = robot;
724    }
725
726    public void setMouseInfo(PointerInfo mouseInfo) {
727        this.mouseInfo = mouseInfo;
728    }
729
730    public void setComPort(CommPort comPort) {
731        this.comPort = comPort;
732    }
733
734    public void setScanner(Scanner scanner) {
735        this.scanner = scanner;
736    }
737
738    public void setStop(boolean stop) {
739        this.stop = stop;
740    }
741
742    public void setX(float x) {
743        this.x = x;
744    }
745
746    public void setY(float y) {
747        this.y = y;
748    }
749
750    public void setZ(float z) {
751        this.z = z;
752    }
753
754    public void setSensitivty(double sensitivty) {
755        this.sensitivty = sensitivty;
756    }
757
758    public void setRobot(Robot robot) {
759        this.robot = robot;
760    }
761
762    public void setMouseInfo(PointerInfo mouseInfo) {
763        this.mouseInfo = mouseInfo;
764    }
765
766    public void setComPort(CommPort comPort) {
767        this.comPort = comPort;
768    }
769
770    public void setScanner(Scanner scanner) {
771        this.scanner = scanner;
772    }
773
774    public void setStop(boolean stop) {
775        this.stop = stop;
776    }
777
778    public void setX(float x) {
779        this.x = x;
780    }
781
782    public void setY(float y) {
783        this.y = y;
784    }
785
786    public void setZ(float z) {
787        this.z = z;
788    }
789
790    public void setSensitivty(double sensitivty) {
791        this.sensitivty = sensitivty;
792    }
793
794    public void setRobot(Robot robot) {
795        this.robot = robot;
796    }
797
798    public void setMouseInfo(PointerInfo mouseInfo) {
799        this.mouseInfo = mouseInfo;
800    }
801
802    public void setComPort(CommPort comPort) {
803        this.comPort = comPort;
804    }
805
806    public void setScanner(Scanner scanner) {
807        this.scanner = scanner;
808    }
809
810    public void setStop(boolean stop) {
811        this.stop = stop;
812    }
813
814    public void setX(float x) {
815        this.x = x;
816    }
817
818    public void setY(float y) {
819        this.y = y;
820    }
821
822    public void setZ(float z) {
823        this.z = z;
824    }
825
826    public void setSensitivty(double sensitivty) {
827        this.sensitivty = sensitivty;
828    }
829
830    public void setRobot(Robot robot) {
831        this.robot = robot;
832    }
833
834    public void setMouseInfo(PointerInfo mouseInfo) {
835        this.mouseInfo = mouseInfo;
836    }
837
838    public void setComPort(CommPort comPort) {
839        this.comPort = comPort;
840    }
841
842    public void setScanner(Scanner scanner) {
843        this.scanner = scanner;
844    }
845
846    public void setStop(boolean stop) {
847        this.stop = stop;
848    }
849
850    public void setX(float x) {
851        this.x = x;
852    }
853
854    public void setY(float y) {
855        this.y = y;
856    }
857
858    public void setZ(float z) {
859        this.z = z;
860    }
861
862    public void setSensitivty(double sensitivty) {
863        this.sensitivty = sensitivty;
864    }
865
866    public void setRobot(Robot robot) {
867        this.robot = robot;
868    }
869
870    public void setMouseInfo(PointerInfo mouseInfo) {
871        this.mouseInfo = mouseInfo;
872    }
873
874    public void setComPort(CommPort comPort) {
875        this.comPort = comPort;
876    }
877
878    public void setScanner(Scanner scanner) {
879        this.scanner = scanner;
880    }
881
882    public void setStop(boolean stop) {
883        this.stop = stop;
884    }
885
886    public void setX(float x) {
887        this.x = x;
888    }
889
890    public void setY(float y) {
891        this.y = y;
892    }
893
894    public void setZ(float z) {
895        this.z = z;
896    }
897
898    public void setSensitivty(double sensitivty) {
899        this.sensitivty = sensitivty;
900    }
901
902    public void setRobot(Robot robot) {
903        this.robot = robot;
904    }
905
906    public void setMouseInfo(PointerInfo mouseInfo) {
907        this.mouseInfo = mouseInfo;
908    }
909
910    public void setComPort(CommPort comPort) {
911        this.comPort = comPort;
912    }
913
914    public void setScanner(Scanner scanner) {
915        this.scanner = scanner;
916    }
917
918    public void setStop(boolean stop) {
919        this.stop = stop;
920    }
921
922    public void setX(float x) {
923        this.x = x;
924    }
925
926    public void setY(float y) {
927        this.y = y;
928    }
929
930    public void setZ(float z) {
931        this.z = z;
932    }
933
934    public void setSensitivty(double sensitivty) {
935        this.sensitivty = sensitivty;
936    }
937
938    public void setRobot(Robot robot) {
939        this.robot = robot;
940    }
941
942    public void setMouseInfo(PointerInfo mouseInfo) {
943        this.mouseInfo = mouseInfo;
944    }
945
946    public void setComPort(CommPort comPort) {
947        this.comPort = comPort;
948    }
949
950    public void setScanner(Scanner scanner) {
951        this.scanner = scanner;
952    }
953
954    public void setStop(boolean stop) {
955        this.stop = stop;
956    }
957
958    public void setX(float x) {
959        this.x = x;
960    }
961
962    public void setY(float y) {
963        this.y = y;
964    }
965
966    public void setZ(float z) {
967        this.z = z;
968    }
969
970    public void setSensitivty(double sensitivty) {
971        this.sensitivty = sensitivty;
972    }
973
974    public void setRobot(Robot robot) {
975        this.robot = robot;
976    }
977
978    public void setMouseInfo(PointerInfo mouseInfo) {
979        this.mouseInfo = mouseInfo;
980    }
981
982    public void setComPort(CommPort comPort) {
983        this.comPort = comPort;
984    }
985
986    public void setScanner(Scanner scanner) {
987        this.scanner = scanner;
988    }
989
990    public void setStop(boolean stop) {
991        this.stop = stop;
992    }
993
994    public void setX(float x) {
995        this.x = x;
996    }
997
998    public void setY(float y) {
999        this.y = y;
1000    }
1001
1002    public void setZ(float z) {
1003        this.z = z;
1004    }
1005
1006    public void setSensitivty(double sensitivty) {
1007        this.sensitivty = sensitivty;
1008    }
1009
1010    public void setRobot(Robot robot) {
1011        this.robot = robot;
1012    }
1013
1014    public void setMouseInfo(PointerInfo mouseInfo) {
1015        this.mouseInfo = mouseInfo;
1016    }
1017
1018    public void setComPort(CommPort comPort) {
1019        this.comPort = comPort;
1020    }
1021
1022    public void setScanner(Scanner scanner) {
1023        this.scanner = scanner;
1024    }
1025
1026    public void setStop(boolean stop) {
1027        this.stop = stop;
1028    }
1029
1030    public void setX(float x) {
1031        this.x = x;
1032    }
1033
1034    public void setY(float y) {
1035        this.y = y;
1036    }
1037
1038    public void setZ(float z) {
1039        this.z = z;
1040    }
1041
1042    public void setSensitivty(double sensitivty) {
1043        this.sensitivty = sensitivty;
1044    }
1045
1046    public void setRobot(Robot robot) {
1047        this.robot = robot;
1048    }
1049
1050    public void setMouseInfo(PointerInfo mouseInfo) {
1051        this.mouseInfo = mouseInfo;
1052    }
1053
1054    public void setComPort(CommPort comPort) {
1055        this.comPort = comPort;
1056    }
1057
1058    public void setScanner(Scanner scanner) {
1059        this.scanner = scanner;
1060    }
1061
1062    public void setStop(boolean stop) {
1063        this.stop = stop;
1064    }
1065
1066    public void setX(float x) {
1067        this.x = x;
1068    }
1069
1070    public void setY(float y) {
1071        this.y = y;
1072    }
1073
1074    public void setZ(float z) {
1075        this.z = z;
1076    }
1077
1078    public void setSensitivty(double sensitivty) {
1079        this.sensitivty = sensitivty;
1080    }
1081
1082    public void setRobot(Robot robot) {
1083        this.robot = robot;
1084    }
1085
1086    public void setMouseInfo(PointerInfo mouseInfo) {
1087        this.mouseInfo = mouseInfo;
1088    }
1089
1090    public void setComPort(CommPort comPort) {
1091        this.comPort = comPort;
1092    }
1093
1094    public void setScanner(Scanner scanner) {
1095        this.scanner = scanner;
1096    }
1097
1098    public void setStop(boolean stop) {
1099        this.stop = stop;
1100    }
1101
1102    public void setX(float x) {
1103        this.x = x;
1104    }
1105
1106    public void setY(float y) {
1107        this.y = y;
1108    }
1109
1110    public void setZ(float z) {
1111        this.z = z;
1112    }
1113
1114    public void setSensitivty(double sensitivty) {
1115        this.sensitivty = sensitivty;
1116    }
1117
1118    public void setRobot(Robot robot) {
1119        this.robot = robot;
1120    }
1121
1122    public void setMouseInfo(PointerInfo mouseInfo) {
1123        this.mouseInfo = mouseInfo;
1124    }
1125
1126    public void setComPort(CommPort comPort) {
1127        this.comPort = comPort;
1128    }
1129
1130    public void setScanner(Scanner scanner) {
1131        this.scanner = scanner;
1132    }
1133
1134    public void setStop(boolean stop) {
1135        this.stop = stop;
1136    }
1137
1138    public void setX(float x) {
1139        this.x = x;
1140    }
1141
1142    public void setY(float y) {
1143        this.y = y;
1144    }
1145
1146    public void setZ(float z) {
1147        this.z = z;
1148    }
1149
1150    public void setSensitivty(double sensitivty) {
1151        this.sensitivty = sensitivty;
1152    }
1153
1154    public void setRobot(Robot robot) {
1155        this.robot = robot;
1156    }
1157
1158    public void setMouseInfo(PointerInfo mouseInfo) {
1159        this.mouseInfo = mouseInfo;
1160    }
1161
1162    public void setComPort(CommPort comPort) {
1163        this.comPort = comPort;
1164    }
1165
1166    public void setScanner(Scanner scanner) {
1167        this.scanner = scanner;
1168    }
1169
1170    public void setStop(boolean stop) {
1171        this.stop = stop;
1172    }
1173
1174    public void setX(float x) {
1175        this.x = x;
1176    }
1177
1178    public void setY(float y) {
1179        this.y = y;
1180    }
1181
1182    public void setZ(float z) {
1183        this.z = z;
1184    }
1185
1186    public void setSensitivty(double sensitivty) {
1187        this.sensitivty = sensitivty;
1188    }
1189
1190    public void setRobot(Robot robot) {
1191        this.robot = robot;
1192    }
1193
1194    public void setMouseInfo(PointerInfo mouseInfo) {
1195        this.mouseInfo = mouseInfo;
1196    }
1197
1198    public void setComPort(CommPort comPort) {
1199        this.comPort = comPort;
1200    }
1201
1202    public void setScanner(Scanner scanner) {
1203        this.scanner = scanner;
1204    }
1205
1206    public void setStop(boolean stop) {
1207        this.stop = stop;
1208    }
1209
1210    public void setX(float x) {
1211        this.x = x;
1212    }
1213
1214    public void setY(float y) {
1215        this.y = y;
1216    }
1217
1218    public void setZ(float z) {
1219        this.z = z;
1220    }
1221
1222    public void setSensitivty(double sensitivty) {
1223        this.sensitivty = sensitivty;
1224    }
1225
1226    public void setRobot(Robot robot) {
1227        this.robot = robot;
1228    }
1229
1230    public void setMouseInfo(PointerInfo mouseInfo) {
1231        this.mouseInfo = mouseInfo;
1232    }
1233
1234    public void setComPort(CommPort comPort) {
1235        this.comPort = comPort;
1236    }
1237
1238    public void setScanner(Scanner scanner) {
1239        this.scanner = scanner;
1240    }
1241
1242    public void setStop(boolean stop) {
1243        this.stop = stop;
1244    }
1245
1246    public void setX(float x) {
1247        this.x = x;
1248    }
1249
1250    public void setY(float y) {
1251        this.y = y;
1252    }
1253
1254    public void setZ(float z) {
1255        this.z = z;
1256    }
1257
1258    public void setSensitivty(double sensitivty) {
1259        this.sensitivty = sensitivty;
1260    }
1261
1262    public void setRobot(Robot robot) {
1263        this.robot = robot;
1264    }
1265
1266    public void setMouseInfo(PointerInfo mouseInfo) {
1267        this.mouseInfo = mouseInfo;
1268    }
1269
1270    public void setComPort(CommPort comPort) {
1271        this.comPort = comPort;
1272    }
1273
1274    public void setScanner(Scanner scanner) {
1275        this.scanner = scanner;
1276    }
1277
1278    public void setStop(boolean stop) {
1279        this.stop = stop;
1280    }
1281
1282    public void setX(float x) {
1283        this.x = x;
1284    }
1285
1286    public void setY(float y) {
1287        this.y = y;
1288    }
1289
1290    public void setZ(float z) {
1291        this.z = z;
1292    }
1293
1294    public void setSensitivty(double sensitivty) {
1295        this.sensitivty = sensitivty;
1296    }
1297
1298    public void setRobot(Robot robot) {
1299        this.robot = robot;
1300    }
1301
1302    public void setMouseInfo(PointerInfo mouseInfo) {
1303        this.mouseInfo = mouseInfo;
1304    }
1305
1306    public void setComPort(CommPort comPort) {
1307        this.comPort = comPort;
1308    }
1309
1310    public void setScanner(Scanner scanner) {
1311        this.scanner = scanner;
1312    }
1313
1314    public void setStop(boolean stop) {
1315        this.stop = stop;
1316    }
1317
1318    public void setX(float x) {
1319        this.x = x;
1320    }
1321
1322    public void setY(float y) {
1323        this.y = y;
1324    }
1325
1326    public void setZ(float z) {
1327        this.z = z;
1328    }
1329
1330    public void setSensitivty(double sensitivty) {
1331        this.sensitivty = sensitivty;
1332    }
1333
1334    public void setRobot(Robot robot) {
1335        this.robot = robot;
1336    }
1337
1338    public void setMouseInfo(PointerInfo mouseInfo) {
1339        this.mouseInfo = mouseInfo;
1340    }
1341
1342    public void setComPort(CommPort comPort) {
1343        this.comPort = comPort;
1344    }
1345
1346    public void setScanner(Scanner scanner) {
1347        this.scanner = scanner;
1348    }
1349
135
```

```
    }
72   public void setSensitivity( double sensitivity ) {
73     this.sensitivity = sensitivity ;
74   }
75   public void stop() {
76     stop = true ;
77   }
78 }
```

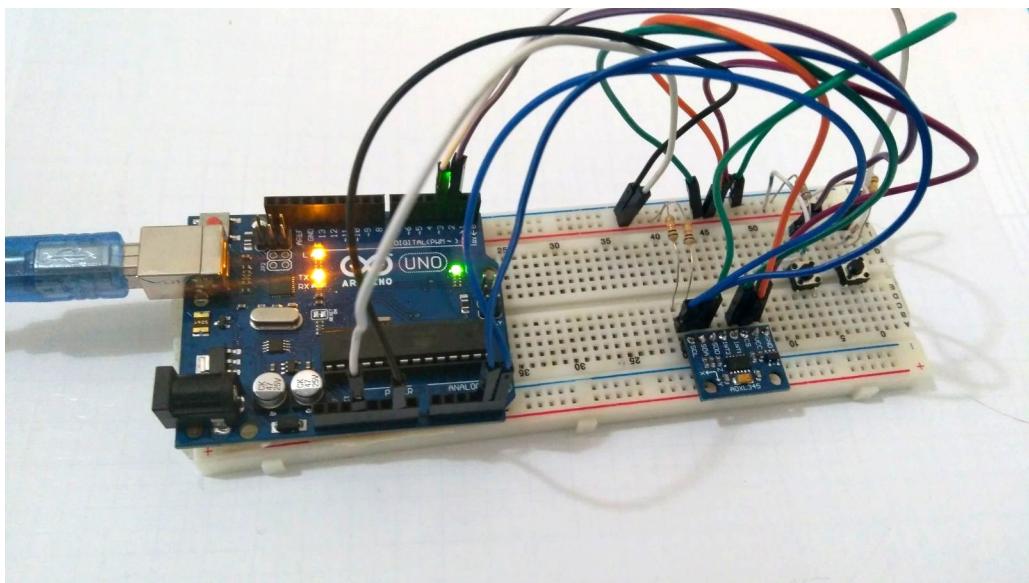
Bu sınıf Arduino'dan okuduğu veriyi kullanarak mouse konumunu ayarlamaktadır. Program çalışırken arayüz yardımıyla hassaslık değiştirilebilir. Sağ ve sol tık, eğer Arduino'dan gelen veride belirtilmişse uygulanır. Program için Java'nın Swing kütüphanesi (arayüz), jSerialComm (Arduino iletişim) kütüphanesi ve Java'nın Robot sınıfı (mouse aksiyonları) kullanılmıştır. Kaynak kodunun arayüzden sorumlu kısmı burada gösterilmemiştir. Bütün projeye [https://github.com/cakinalperen/tilt\\_mouse](https://github.com/cakinalperen/tilt_mouse) adresinden ulaşılabilir.



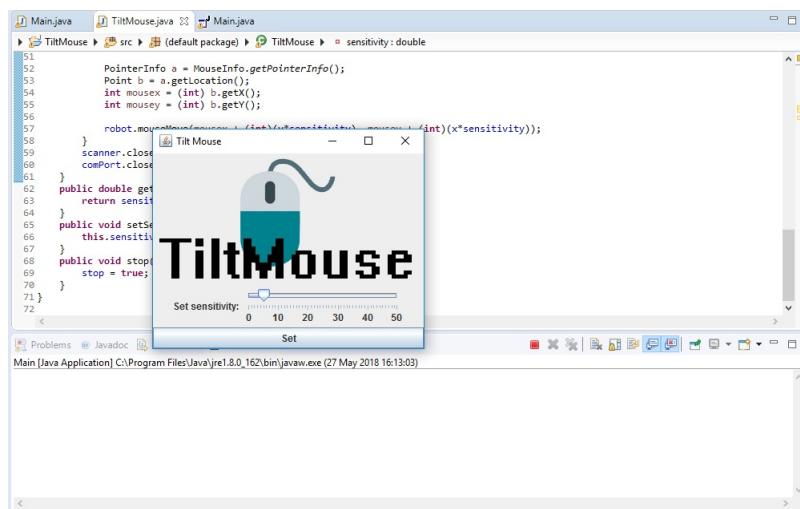
Sekil 7.34: Devre Semasi

Yerleştirilen iki butonun amacı sağ ve sol tık fonksiyonlarını sağlamaktır. ADXL345 ile i2c yoluyla iletişim kurmak için gerekli yerleşim yapılmıştır. Bu yerleşim hem gösterdiğim kaynakta, hem de <https://learn.sparkfun.com/tutorials/adxl345-hookup-guide> adresinde bulunabilir. Sensörün i2c moduna geçmek için CS pinine 1 verilir. Sensörün GND'si ve VCC'si uygun

pinlere bağlanır. Kalan pinler ise yukarıdaki adresin referansı olan, sensörün datasheet'ine göre bağlanmıştır. Bu belge [https://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf?\\_ga=2.29990975.1001805871.1527861113-2031533479.1520333148](https://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf?_ga=2.29990975.1001805871.1527861113-2031533479.1520333148) adresinde bulunabilir.



**Sekil 7.35:** Tilt Mouse



Sekil 7.36: Tilt Mouse



Şekil 7.37: Tilt Mouse

## 7.19 HC-05 Bluetooth Serial Module üzerinden Seri Haberleşme

### Proje Videosu:

<https://drive.google.com/drive/folders/1eAshRBBsoKc5xgdKKQh-5-MlhieSo1M?usp=sharing>

### Malzeme Listesi

- Arduino UNO
- HC-05 Bluetooth Module
- 4 adet 220Ω direnç
- Breadboard
- 4 adet LED
- Dişi - dişi jumper kablo
- Erkek - erkek jumper kablo

İlk olarak Arduino'ya bağladığım HC-05 modülü üzerinde isim, boud rate ve şifre ayarı yapıyorum. HC-05 modülü üzerinde bulunan LED,

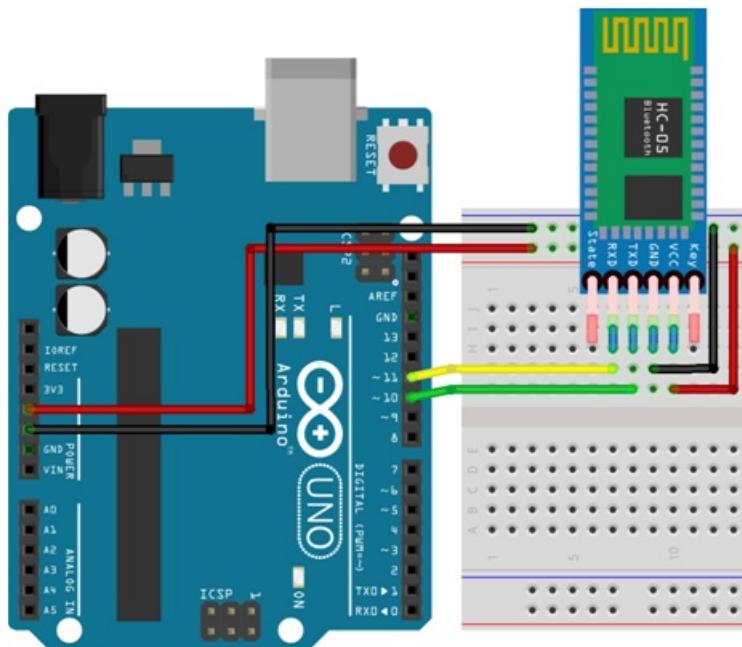
- Aralıksız yanıp sönyorsa, modül iletişim modundadır.
- 3'er saniyelik aralıklarla yanıp sönyorsa, modül konfigurasyon modundadır.

Modül, iletişim modunda iken diğer cihazlardan tarama yaptığımızda “Arduino UNO” ismi ile listede yer aldığı görebiliriz. Bir cihaz modüle bağlanmak istediği HC-05 üzerindeki LED 3 saniyede bir kısa kısa yanıp sönmektedir.

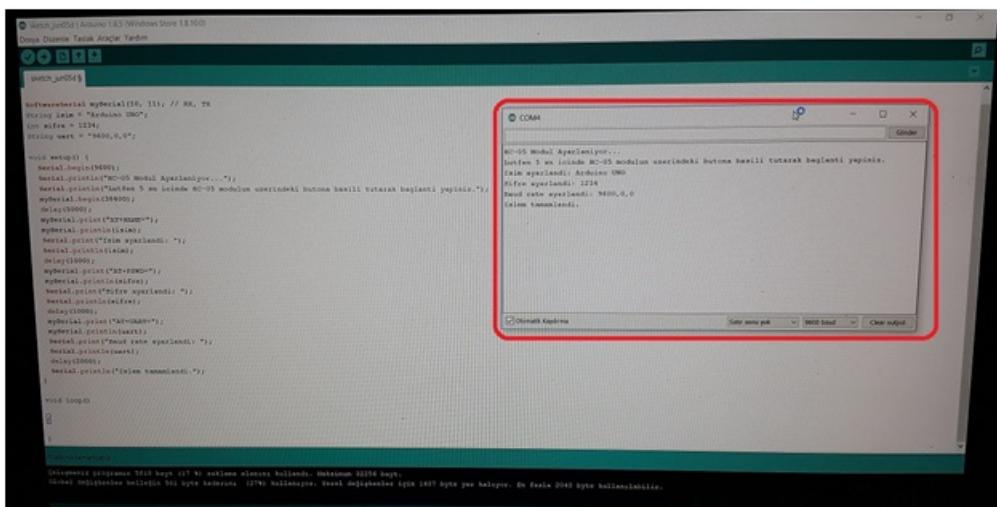
```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX
String device_name = "Arduino UNO";
int device_password = 1234;
String baud = "9600,0,0";
```

Kodda yer alan device\_name, device\_password ve baud değişkenleri üzerinden modülü istediğim şekilde ayarlayabiliyorum.



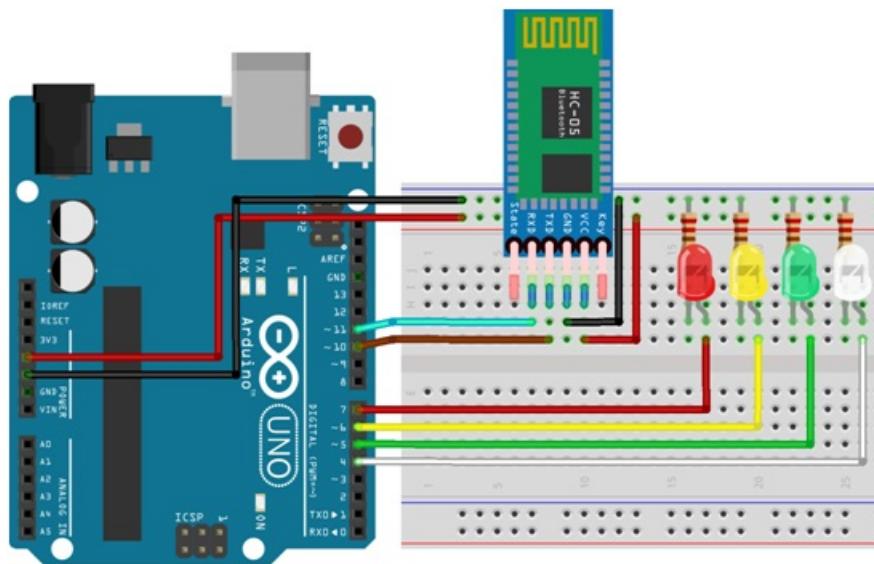
Şekil 7.38: Modül Konfigurasyonu için Devre Şeması



Şekil 7.39: Seri port ekranı

Seri port Ekranını açıp HC-05 modülünün üzerindeki butona basılı tutarak yaptığımız ayarlamaları ekranda görebiliriz.

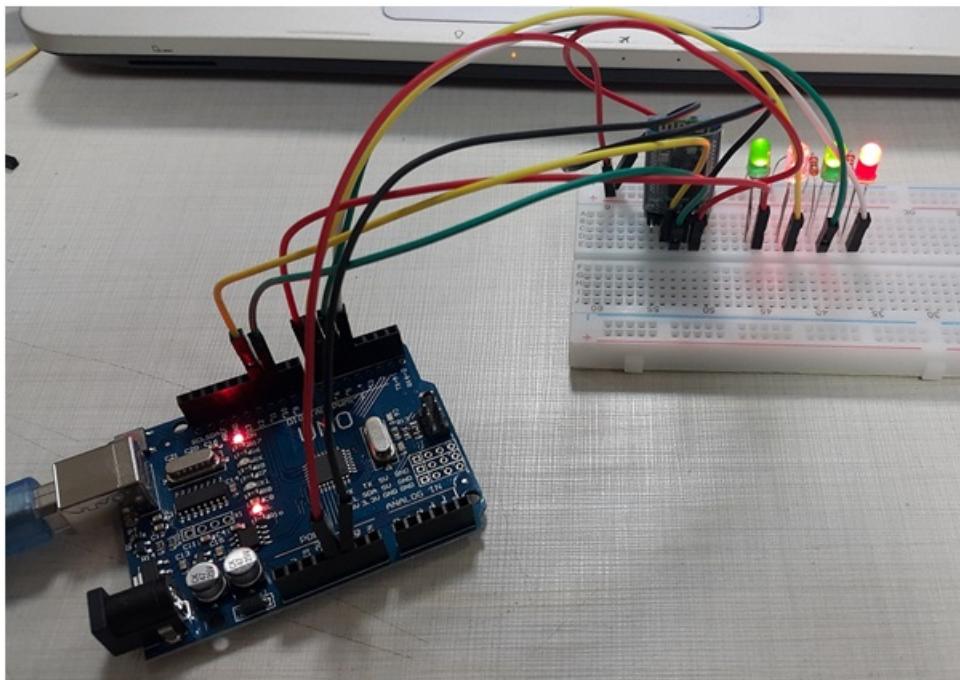
Ayarlamaları yaptıktan sonra, bluetooth modül üzerinden seri haberleşme yapacağım asıl devremi kuruyorum.



Şekil 7.40: Proje Devre Şeması

Devreyi kurup kodumu yükleme işlemini tamamladım. Şimdi bluetooth modülüne Windows bilgisayara tanıtmam gerekiyor. Modülümün iletişim modunun açık olduğuna emin olduktan sonra, sistemde “Bluetooth Aygıtları”na gelerek bluetoothı açıyorum. Aygit Tarama işlemi gerçekleştirip modülümü listede bulduktan sonra eşleştirme işlemi için önceden belirlediğim şifreyi sisteme giriyorum. Eşleştirme işlemini tamamlamış bulunmaktayım. Şimdi “Denetim Masası”ndan HC-05’in “Özellikler”ine girerek Donanım bilgilerinden COM port numarasının kaç olduğunu öğreniyorum. (Benimki COM5) Bunu yapmamın amacı, Andiuno’nun portu ile bluetooth modülümü portu aynı şekilde ayarlanmış olmalıdır. Andiuno Ayarlar sekmesinden portumu COM5 olarak ayarlıyorum. Artık programım yürütülmeye hazır durumda. Andiuno Seri Port Ekranı’nı açarak sağ alt köşeden baud rate’i 9600 olarak belirliyorum. Artık komutlarımı işletebilirim.

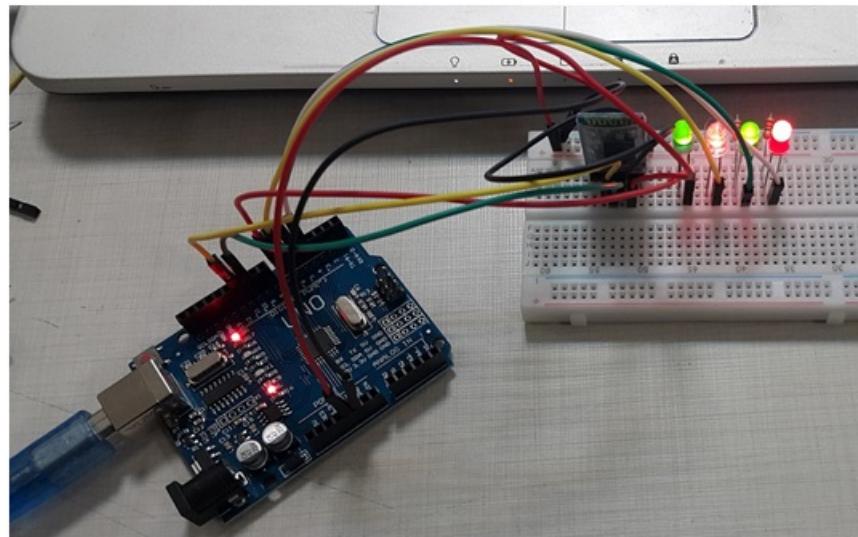
#### Code Komutları:



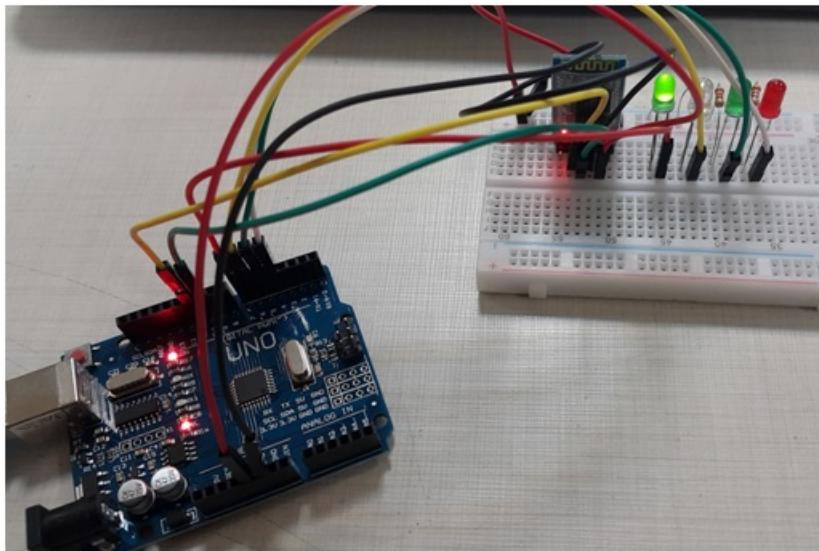
1. q -> 7 numaralı pine bağlı LED'i yakar & söndürür.
2. w -> 6 numaralı pine bağlı LED'i yakar & söndürür.
3. e -> 5 numaralı pine bağlı LED'i yakar & söndürür.
4. r -> 4 numaralı pine bağlı LED'i yakar & söndürür.
5. x -> Tüm LED'leri yakar.
6. z -> Tüm LED'leri söndürür.

```
char ch = mySerial.read();
if (ch == 'q')
{
    digitalWrite(led1, !digitalRead(led1));
    if (digitalRead(led1) == HIGH)
    {
        mySerial.println("LED 1 yandi");
    }
    else
    {
        mySerial.println("LED 1 sondu");
    }
}
if (ch == 'w')
{
    digitalWrite(led2, !digitalRead(led2));
```

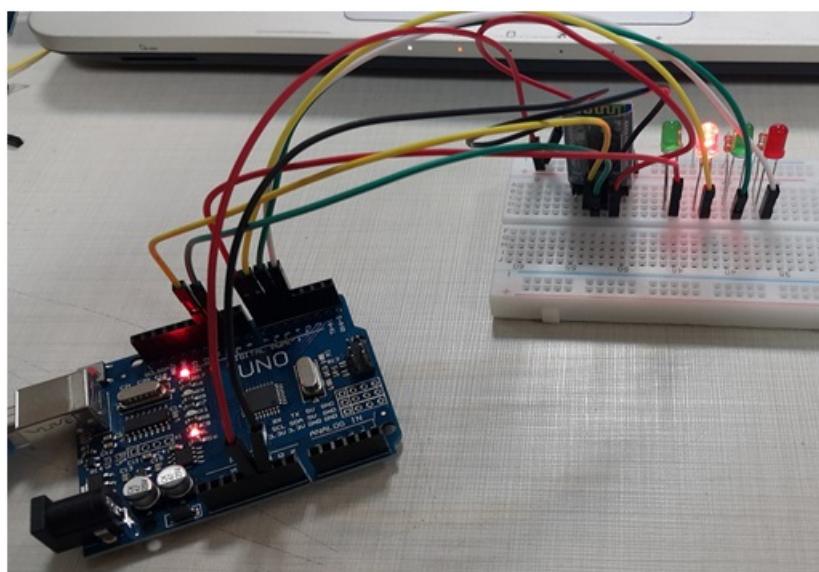
Her bir komutun kendi if bloğu içinde seri haberleşme kodu



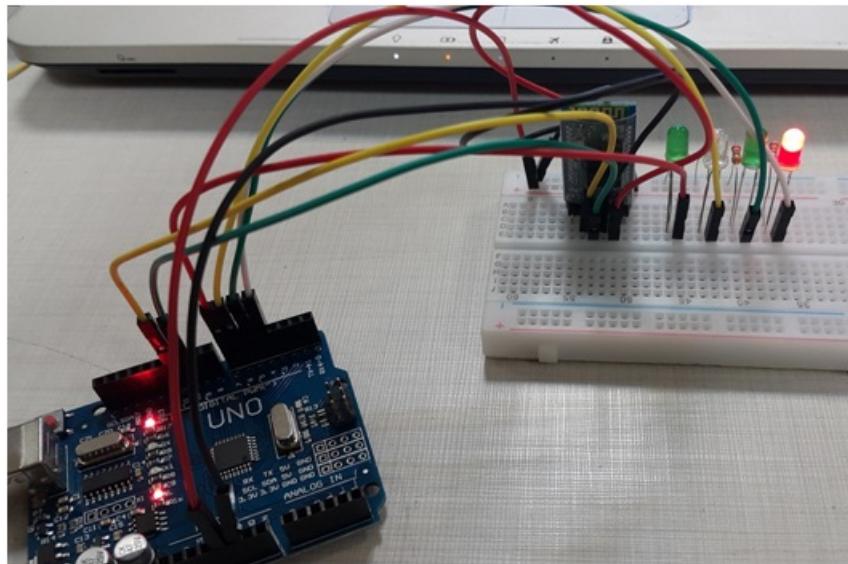
Şekil 7.41: X komutu yürütülüyor



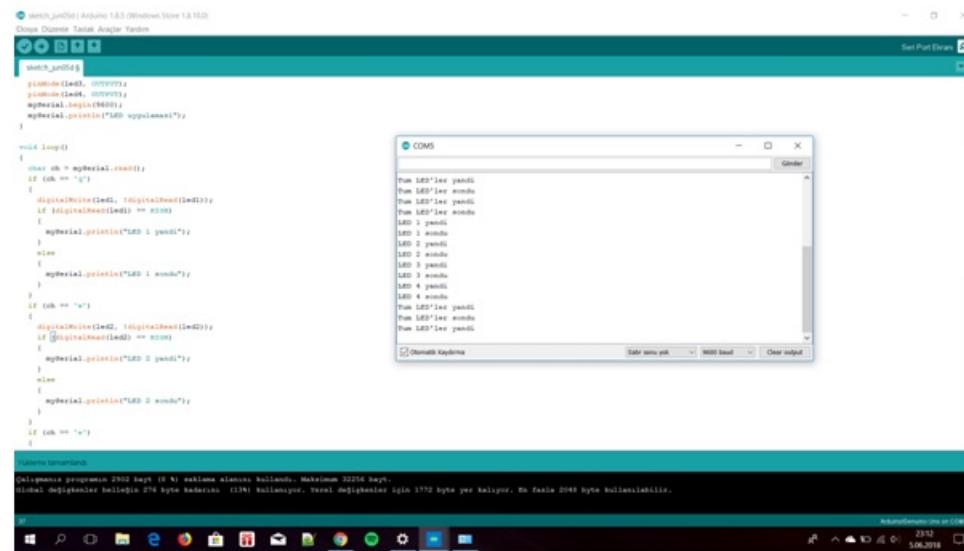
Şekil 7.42: Q komutu yürütülüyor



Şekil 7.43: W komutu yürütülüyor



Şekil 7.44: R komutu yürütülüyor



Şekil 7.45: Seri Port Ekran Çıktıları

## 7.20 RFID-RC522 Modülü ile Kapı Kilidi Uygulaması

### Malzeme Listesi

- Arduino UNO
- RFID-RC522 modülü
- Breadboard
- 1 adet LED
- Dişi - dişi jumper kablo
- Erkek - erkek jumper kablo
- $220\Omega$  direnç
- 2 adet RFID'e tanıtılacak kart
- 1 adet RFID anahtarlık
- 1 adet RFID tarafından tanımlanamayan kart

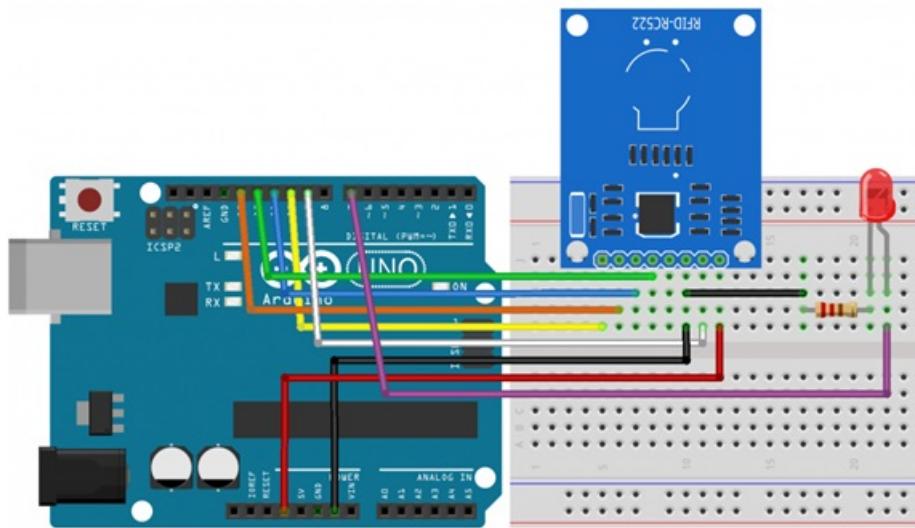
RFID, nesnelerin yaydığı radyo dalgaları üzerinden tanınmasına dayanan bir teknolojidir. Günlük hayatı kartlı geçiş sistemine sahip turnikelerde sıkça kullanılmaktadır. Burada okutulan her kartın birbirinden farklı, kendine ait UID numaraları vardır. Okuyucuya kartı veya anahtarlığı yaklaştırdığımızda UID numarası okunarak işlem gerçekleştirilir. İlk olarak projede gerçekleştirmem gereken aşama, RFID'e tanıtılacak iki kartın UID'lerini Arduino'nun EEPROM'una kaydetmektir. Daha sonra projede test edeceğim kartların UID'leriyle, memory'e kaydettiğim UID değerlerini karşılaştıracak düzeneği kuracağım.

Devre Şemasına kodu yüklemeden önce MFRC522 kütüphanesini indirerek libraries'e ekliyoruz.

```
#include <EEPROM.h>
```

Şimdi ilk olarak Arduino'nun EEPROM'una kartların UID'lerini kaydedecek kodu yükliyoruz ve Seri Port Ekranını açarak kartları tanıma işlemini gerçekleştiriyoruz.

1. Şekil 7.47 ve Şekil 7.48'deki kodu Ardiuno'ya yükliyoruz.



Şekil 7.46: Devre Şeması

2. Seri Port Ekranını açıyoruz.
  3. RFID modülüne,

ERROM'a kayıtlı olan 1 numaralı kartı okuttuğumuzda memory'deki UID değerleriyle karşılaştırıp, aynı değeri bulduğu için 7 numaralı pine bağlı olan LED'i yanıyor durumda ise söndürür, sönüklük durumda ise yakar.

EEROM'a kayıtlı olan 2 numaralı kartı okuttuğumuzda memory'deki UID değerleriyle karşılaştırıp, aynı değeri bulduğu için 7 numaralı pine bağlı olan LED'i yanıyor durumda ise söndürür, sönüklük durumda ise yakar.

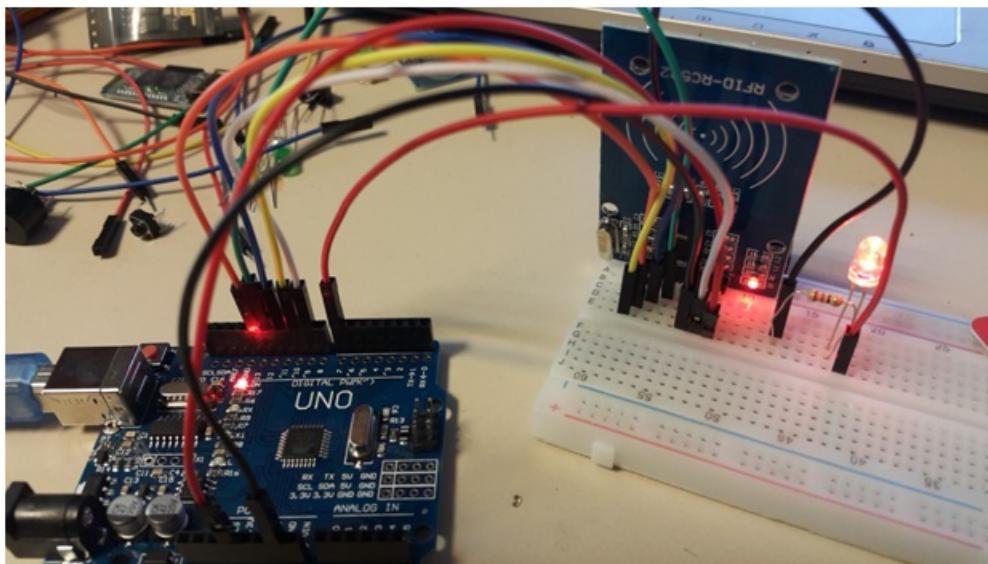
Eğer EEPROM'da UID'si kayıtlı olmayan bir kart okutursak LED'in durumunda herhangi bir değişiklik olmuyor. Ancak Seri Port ekranında okunan kartın UID değerini bize verir. Devrede ise herhangi bir geri dönüt gerçeklestirmez.

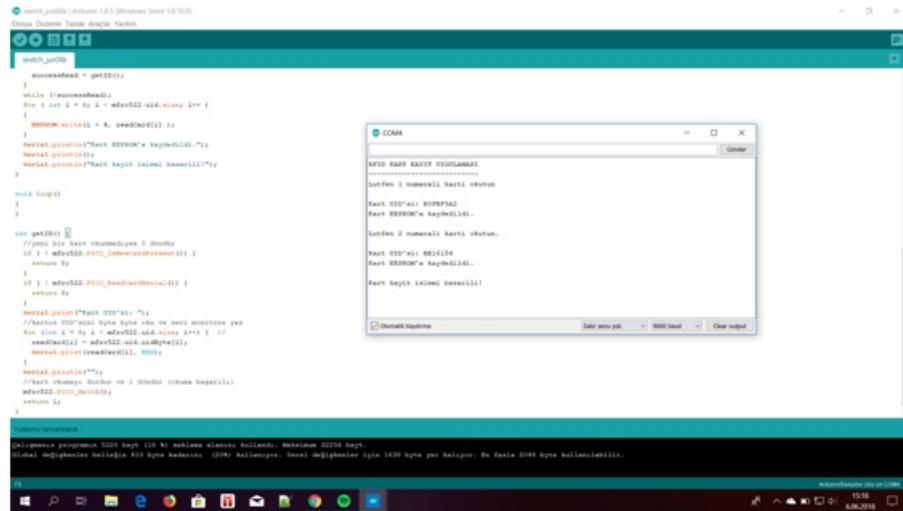
Eğer RFID tarafından tanımlanamayan bir kart okutulmaya çalışılırsa Seri Port Ekranında veya Devrede herhangi bir değişiklik gerçekleştirilmez.

```
MFRC522 mfrc522(SS_PIN, RST_PIN);

MFRC522::MIFARE_Key key;

void setup()
{
    Serial.begin(9600);
    SPI.begin();
    mfrc522.PCD_Init();
    Serial.println("RFID KART KAYIT UYGULAMASI");
    Serial.println("-----");
    Serial.println("Lutfen 1 numarali karti okutun");
    Serial.println();
    do {
        //okuma basarili olana kadar getID fonksiyonunu cagir
        successRead = getID();
    }
    while (!successRead);
    for ( int i = 0; i < mfrc522.uid.size; i++ )
    {
        //kartin UID'sini EEPROM'a kaydet
        EEPROM.write(i, readCard[i]);
    }
    Serial.println("Kart EEPROM'a kaydedildi.");
    Serial.println();
    Serial.println("Lutfen 2 numarali karti okutun.");
    Serial.println();
    do {
        successRead = getID();
    }
    while (!successRead);
    for ( int i = 0; i < mfrc522.uid.size; i++ )
    {
        EEPROM.write(i + 4, readCard[i]);
    }
```





Şekil 7.47: Kaynak kodu - 1

```

void loop()
{
    //yeni kart okunmadıkça devam etme
    if (!mfrc522.PICC_IsNewCardPresent())
    {
        return;
    }
    if (!mfrc522.PICC_ReadCardSerial())
    {
        return;
    }
    //kartın UID'sini oku, rfid isimli string'e kaydet
    String rfid = "";
    for (byte i = 0; i < mfrc522.uid.size(); i++)
    {
        rfid += mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ";
        rfid += String(mfrc522.uid.uidByte[i], HEX);
    }
    //string'in boyutunu ayarla ve tamamını büyük harfe çevir
    rfid.trim();
    rfid.toUpperCase();

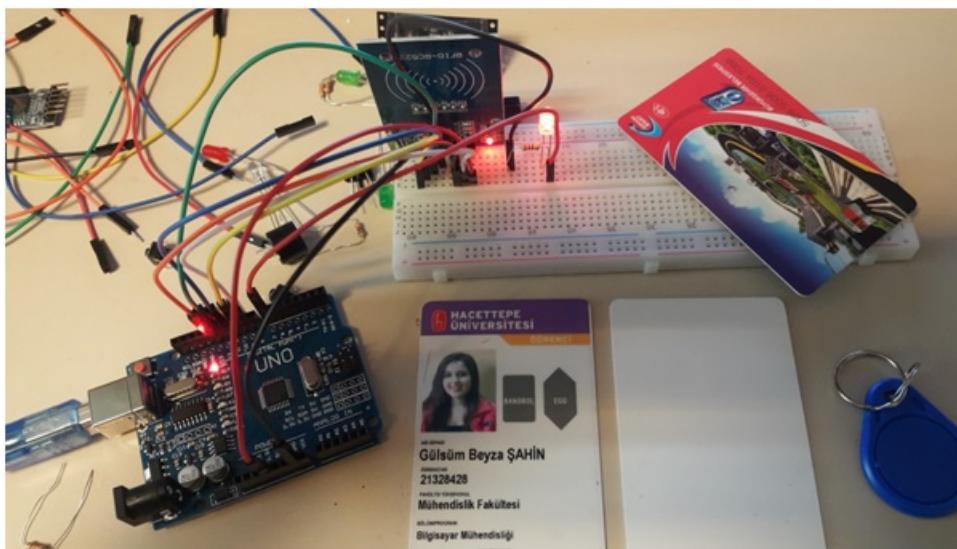
    if (rfid == lastRfid)
        return;
    lastRfid = rfid;

    Serial.print("Kart 1: ");
    Serial.println(rfid);
    Serial.print("Kart 2: ");
    Serial.println(rfid);
    Serial.print("Okunan: ");
    Serial.println(rfid);
    //1 nolu kart okunduysa LED'i yak, 2 nolu kart okunduysa LED'i sönür
    if (rfid == kart1)
    {
        digitalWrite(ledPin, HIGH);
        Serial.println("LED yandı.");
    }
    if (rfid == kart2)
    {
        digitalWrite(ledPin, LOW);
        Serial.println("LED söndür.");
    }
    Serial.println();
    delay(2000);
}

void readEEPROM()
{
    //EEPROM'dan ilk kartın UID'sini oku (ilk 4 byte)
    for (int i = 0 ; i < 4 ; i++)
    {
        kart1 += EEPROM.read(i) < 0x10 ? " 0" : " ";
        kart1 += String(EEPROM.read(i), HEX);
    }
    //EEPROM'dan ikinci kartın UID'sini oku
    for (int i = 4 ; i < 8 ; i++)
    {
        kart2 += EEPROM.read(i) < 0x10 ? " 0" : " ";
        kart2 += String(EEPROM.read(i), HEX);
    }
    //Okunan stringleri düzelle
    kart1.trim();
    kart1.toUpperCase();
    kart2.trim();
    kart2.toUpperCase();
}

```

Şekil 7.48: Kaynak kodu - 2



```

// sketch_jun080 | Arduino 1.8.5 (Windows Store 1.8.10.0)
// Diese Datei wurde unter Angabe von
// https://www.arduino.cc/en/Tutorial/SerialPrint
// erstellt.

void setup() {
    // initialize serial communication at 9600 bits per second:
    Serial.begin(9600);
}

void loop() {
    if (rfid == kart1) {
        digitalWrite(LEDPin, HIGH);
        Serial.println("LED prendi.");
    }
    if (rfid == kart2) {
        digitalWrite(LEDPin, LOW);
        Serial.println("LED sonda.");
    }
    Serial.println("delay(2000);");
}

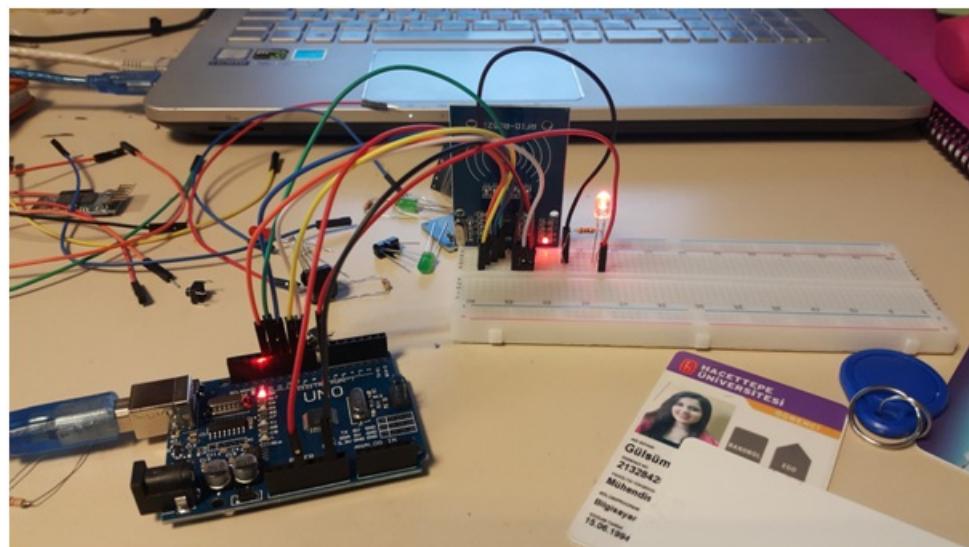
void rfid(RFIDModule)
{
    //RFIDON'dan LIX Kartın UID'sini oku (114 byte)
    for (int i = 0; i < 4 * 34) {
        kart1 = RFIDON.read(i) + (0x10 * i) + " ";
        kart2 = String(RFIDON.read(i), HEX);
    }
    //RFIDON'dan ikinci Kartın UID'sini oku
    for (int i = 0; i < 4 * 34) {
        kart2 = RFIDON.read(i) + (0x10 * i) + " ";
        kart2 = String(RFIDON.read(i), HEX);
    }

    //Okunan stringleri dosyeye
    kart1.tostring();
    kart1.tostring(kart1);
    kart2.tostring();
    kart2.tostring(kart2);

    //LED yanıp sönsele
    digitalWrite(LEDPin, HIGH);
    Serial.println("LED yanıp sönsele.");
    delay(2000);
    digitalWrite(LEDPin, LOW);
    Serial.println("LED sonda.");
}

```

The code is an Arduino sketch named 'sketch\_jun080'. It uses an RFID module (RFIDON) to read two specific cards (kart1 and kart2). If the read card matches kart1, it turns on an LED (HIGH). If it matches kart2, it turns off the LED (LOW). The code also includes a delay of 2 seconds between reads. The comments indicate that the code was generated by the Arduino IDE from a template.



## 7.21 Arduino Web Server LED Kontrol

Projede amaç serverdaki bir sayfa üzerinden arduinoya bağlanan ledleri kontrol etmek. Bunun için arduino'nun internete bağlanması gerekmektedir.

Bu bağlantı için esp8266 kullanılmıştır. Burada da espyi internete bağlamak için AT komutları kullanılmıştır. Daha sonra AT+CIPSERVER komutuyla bağlantı sağlanmış ve html sayfada oluşturulacak on ve off butonları arduino kodundan gönderilmiştir. Bir kısım AT komutları :

**Kullanılan kod kesimi de şu şekildedir:**

```

1 void setup() { // put your setup code here, to run once:
2   Serial1.begin(115200);   Serial.begin(115200);      ATkomut("AT+RST\r\n",1000,true);
3   ATkomut("AT+CWMODE=1\r\n",4000,true);
4   ATkomut("AT+CWJAP="\\"\\","\\r\\n",8000,true);
5   ATkomut("AT+CIPMUX=1\r\n",4000,true);
6   ATkomut("AT+CIFSR\r\n",4000,true);
7   ATkomut("AT+CIPSERVER=1,80\r\n",4000,true); //ATkomut("AT+
8   CIPSTART=4,\\"TCP\\",\\192.168.0.16\\",80\r\n",12000,true);
9   pinMode(LED_BUILTIN, OUTPUT);
10 }
11
12 void loop() { // put your main code here, to run repeatedly:
13   if (Serial1.available()) {
14     if(Serial1.find("+IPD,")) // Recieved data
15       delay(500);
16     int baglanti = Serial1.read()-48; //ASCII to int convert
17     String yolla = "<head> Sedat Burak </head>";
18     yolla+="

```

```
32 }  
33 String ATkomut( String komut, const int sure, boolean durum) {  
34     String gelen="";  
35     Serial1.print(komut);  
36     long int zaman = millis();  
37     while((zaman+sure)> millis()) {  
38         while(Serial1.available()) {  
39             char c = Serial1.read();  
40             gelen+=c;  
41         }  
42     }  
43     if(durum) {  
44         Serial.print(gelen);  
45         if(gelen.indexOf(":GET \?pin=ON")>1)  
46             digitalWrite(LED_BUILTIN, HIGH);  
47         if(gelen.indexOf(":GET \?pin=OFF")>1)  
48             digitalWrite(LED_BUILTIN, LOW);  
49     }  
50 }  
51  
52 return gelen;  
}
```

## AT COMMAND LISTING

	COMMAND	FUNCTION
1	AT	Test UART Connection
2	AT+RESET	Reset Device
3	AT+VERSION	Querry firmware version
4	AT+ORGL	Restore settings to Factory Defaults
5	AT+ADDR	Query Device Bluetooth Address
6	AT+NAME	Query/Set Device Name
7	AT+RNAME	Query Remote Bluetooth Device's Name
8	AT+ROLE	Query/Set Device Role
9	AT+CLASS	Query/Set Class of Device CoD
10	AT+IAC	Query/Set Inquire Access Code
11	AT+INQM	Query/Set Inquire Access Mode
12	AT+PSWD	Query/Set Pairing Passkey
13	AT+UART	Query/Set UART parameter
14	AT+CMODE	Query/Set Connection Mode
15	AT+BIND	Query/Set Binding Bluetooth Address
16	AT+POLAR	Query/Set LED Output Polarity
17	AT+PIO	Set/Reset a User I/O pin
18	AT+MPIO	Set/Reset multiple User I/O pin
19	AT+MPIO?	Query User I/O pin
20	AT+IPSCAN	Query/Set Scanning Parameters
21	AT+SNIFF	Query/Set SNIFF Energy Savings Parameters
22	AT+SENM	Query/Set Security & Encryption Modes
23	AT+RMSAD	Delete Authenticated Device from List
24	AT+FSAD	Find Device from Authenticated Device List
25	AT+ADCN	Query Total Number of Device from Authenticated Device List
26	AT+MRAD	Query Most Recently Used Authenticated Device
27	AT+STATE	Query Current Status of the Device
28	AT+INIT	Initialize SPP Profile
29	AT+INQ	Query Nearby Discoverable Devices
30	AT+INQC	Cancel Search for Discoverable Devices
31	AT+PAIR	Device Pairing
32	AT+LINK	Connect to a Remote Device
33	AT+DISC	Disconnect from a Remote Device
34	AT+ENSNIFF	Enter Energy Saving mode
35	AT+EXSNIFF	Exit Energy Saving mode

## ERROR CODES

ERROR CODE	VERBOSE
0	Command Error/Invalid Command
1	Results in default value
2	PSKEY write error
3	Device name is too long (>32 characters)
4	No device name specified (0 lenght)
5	Bluetooth address NAP is too long
6	Bluetooth address UAP is too long
7	Bluetooth address LAP is too long
8	PIO map not specified (0 lenght)
9	Invalid PIO port Number entered
A	Device Class not specified (0 lenght)
B	Device Class too long
C	Inquire Access Code not Specified (0 lenght)
D	Inquire Access Code too long
E	Invalid Inquire Access Code entered
F	Pairing Password not specified (0 lenght)
10	Pairing Password too long (> 16 characters)
11	Invalid Role entered
12	Invalid Baud Rate entered
13	Invalid Stop Bit entered
14	Invalid Parity Bit entered
15	No device in the Pairing List
16	SPP not initialized
17	SPP already initialized
18	Invalid Inquiry Mode
19	Inquiry Timeout occurred
1A	Invalid/zero lenght address entered
1B	Invalid Security Mode entered
1C	Invalid Encryption Mode entered

Şekil 7.49: AT komutları

## 7.22 Arduino Uno ve Ethernet Shield ile İnternet Üzerinden Fan Kontrol Sistemi

**Proje videosu:**

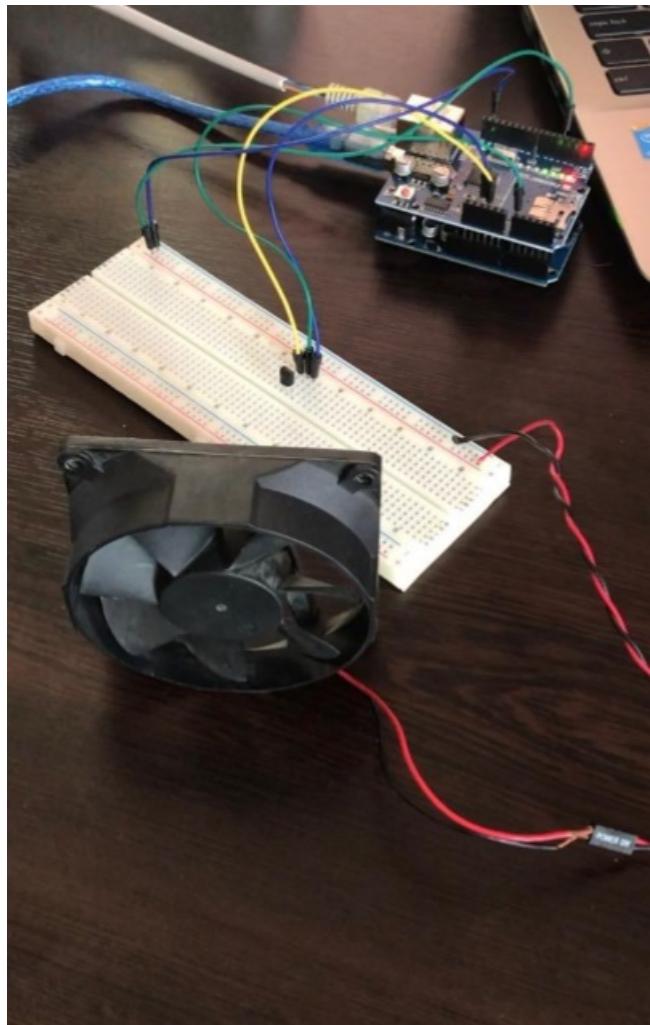
<https://drive.google.com/open?id=1MytQaQ4N4au5dsBcgJjGGAlRsR7LfG1V>

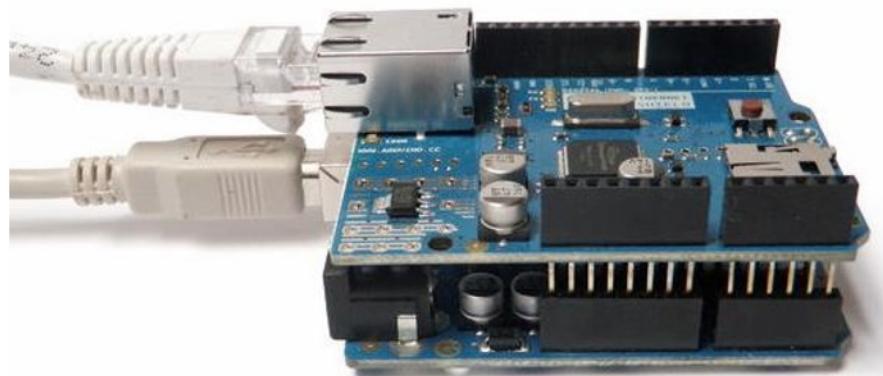
**Malzeme listesi**

- Arduino Uno
- Ethernet Shield modülü
- Jumper kablolar
- Isı sensörü
- Bilgisayar fanı

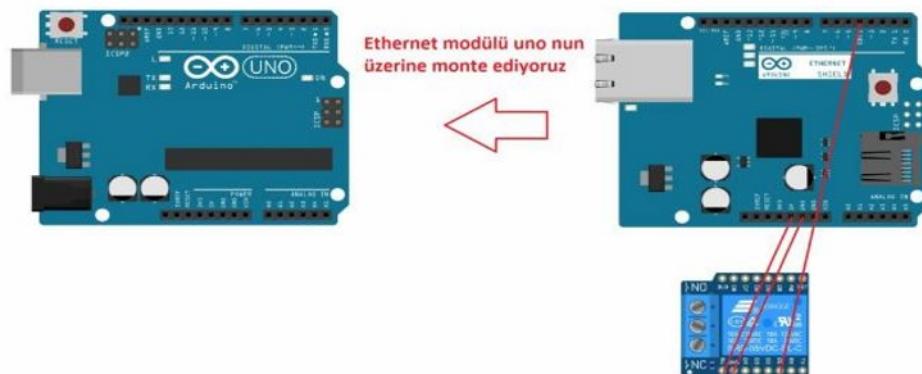
**Projenin amacı ve çalışma mantığı** Ardunio Ethernet Shiled ile Arduino kartımızı internete bağlayabilir, internet üzerinden kartımıza komut gönderebiliriz. Şekil 7.50'de arduino uno ile ethernet shield bağlantısının nasıl yapıldığını ve bağlantı şemasını görmekteyiz.

Ethernet kablomuzu ethernet shield'e bağladıkten sonra Arduino'muz internete bağlanmaya hazır gelecektir. Burda dikkat edilmesi gereken ethernet modülünün 3.3V pinine bağlanacak olmasıdır. İnternet bağlantımızı yaptıktan sonra web server kuracağız. Öncelike bağlı olduğumuz modemim bize verdiği IP yi öğrenmemiz gereklidir. Bunu öğrenmenin birkaç yolu var. En basit olanı modeme sormak. Bu soru işlemini Arduino'ya eklediğimiz kütüphane nin DHCP isimli sketch' ini kullanarak yapabiliriz. Bu yazılımı Arduino' ya attığımız zaman mac adresimiz yardımcı ile modemim bize verdiği ip adresini Serial monitör' den görmüş olacağız. Daha sonra biz web server' imizi bu ip üzerine kuracağız. Aşağıda DHCP chat Server 'imizda kullandığım kodu göreceğiz.





Şekil 7.50: Bağlantı şeması - 1



Şekil 7.51: Bağlantı şeması - 2

```
/*
2 #include <SPI.h> #include <Ethernet.h>
4 // Enter a MAC address and IP address for your controller
// below. // The IP address will be dependent on your local
// network. // gateway and subnet are optional:
6 byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02 };
IPAddress ip(192, 168, 1, 33);
IPAddress myDns(192,168,1, 1);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 0, 0);

12 // telnet defaults to port 23 EthernetServer server(23);
13 boolean gotAMessage = false; // whether or not you got a
14 message from the client yet

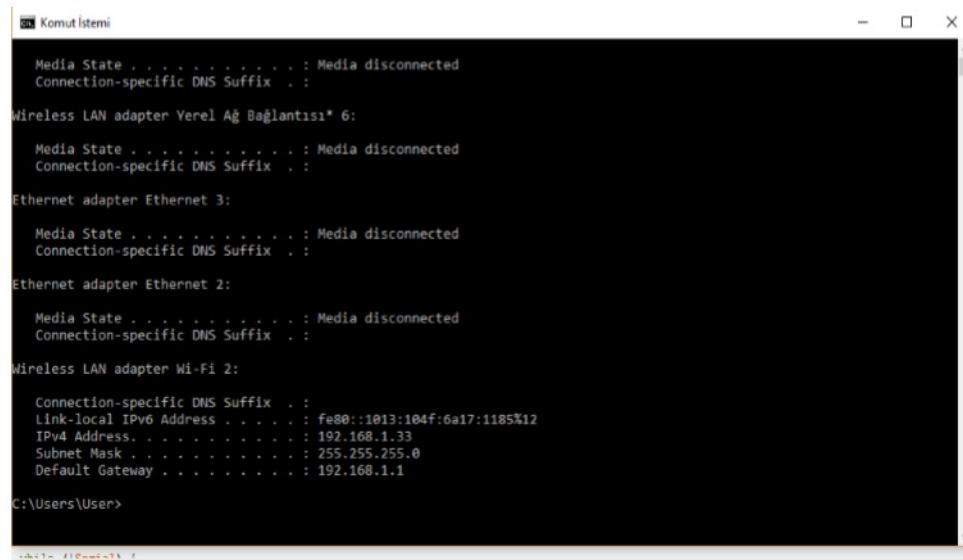
void setup() { // Open serial communications and wait for
15   // port to open: Serial.begin(9600); // this check is only
16   // needed on the Leonardo:
17   while (!Serial) {
18     ; // wait for serial port to connect. Needed for native USB
19     // port only
20   }

22 // start the Ethernet connection:
23 Serial.println("Trying to get an IP address using DHCP");
24 if (Ethernet.begin(mac) == 0) {
25   Serial.println("Failed to configure Ethernet using DHCP");
26   // initialize the Ethernet device not using DHCP:
27   Ethernet.begin(mac, ip, myDns, gateway, subnet);
28 }
29 // print your local IP address:
30 Serial.print("My IP address: ");
31 ip = Ethernet.localIP();
32 for (byte thisByte = 0; thisByte < 4; thisByte++) {
33   // print the value of each byte of the IP address:
34   Serial.print(ip[thisByte], DEC);
35   Serial.print(".");
36 }
37 Serial.println();
38 // start listening for clients
39 server.begin();
40 }

void loop() { // wait for a new client: EthernetClient
41   client = server.available();
```

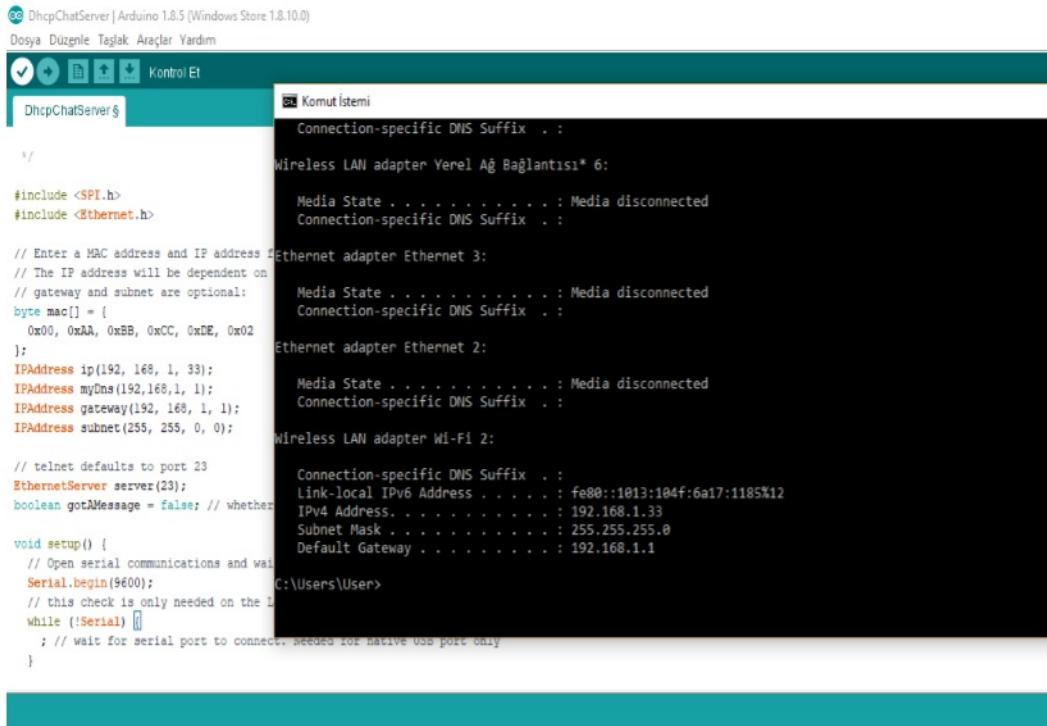
```

40 // when the client sends the first byte, say hello:
41 if (client) {
42   if (!gotAMessage) {
43     Serial.println("We have a new client");
44     client.println("Hello, client!");
45     gotAMessage = true;
46   }
47
48   // read the bytes incoming from the client:
49   char thisChar = client.read();
50   // echo the bytes back to the client:
51   server.write(thisChar);
52   // echo the bytes to the server as well:
53   Serial.print(thisChar);
54   Ethernet.maintain();
55 }
56 }
```



Sekil 7.52: Cmd'den ip adresi okuma

Sekil 7.52'de cmd'den okuduğumuz ip adresimizi bu kodda kendimiz düzenliyoruz. Yani static olarak alıyoruz. Sekil 7.53'te bu düzenlemenin ekran alıntısını göreceğiz.



Şekil 7.53: Cmd'den okunan ip ile kodun düzenlenmesi

Burda gördüğümüz üzere bağlı olduğumuz modem ip adresini cmd ile öğrenip dhcp chat server sayfamıza statik olarak yazdık. Bundan sonraki adım bilgisayardan yada mobil cihazdan bir arayüz sayesinde bu sistemimizi kontrol etmek. Bu işlemleri yapmamızı sağlayan kodumu aşağıda paylaşıyorum.

```
1 include <SPI.h>
2 #include <Ethernet.h>
3 #include <Servo.h>
4
5 float oda_sicakligi;           //oda sicakligi degiskeni
6     tanimlandi
7 int sensor = 0;           // LM35 sensoronun ortasindaki sinyal ucu
8     Analog 0 baglandi.
9
10 int led = 4;
11 Servo microservo;
12 int pos = 0;
13 byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };    // physical mac address
```

```

byte ip[] = { 192, 168, 1, 37 }; // ip in
    lan (that's what you need to use in your browser. ("192.168.1.178"))
13 byte gateway[] = { 192, 168, 1, 1 }; //internet access via router
14 byte subnet[] = { 255, 255, 255, 0 }; //subnet mask
15 EthernetServer server(45); //server port
16 String readString;
17
18 void setup() {
19 // Open serial communications and wait for port to open:
Serial.begin(9600);
20 while (!Serial) {
21 ; // wait for serial port to connect. Needed for Leonardo only
22 }
23 pinMode(led, OUTPUT);
24 microservo.attach(7);
25 // start the Ethernet connection and the server:
26 Ethernet.begin(mac, ip, gateway, subnet);
27 server.begin();
28 Serial.print("server is at ");
29 Serial.println(Ethernet.localIP());
30 }
31
32 float getTemp() {
33
34 oda_sicakligi = analogRead(sensor); //Analog cikis
35     aldig.
36
37 oda_sicakligi = oda_sicakligi * 0.48828125; // Voltajimizi derece cinsine cevirmek icin carptik. 0.48828125
38     ifadesi [(5V*1000)/1024]10 hesaplamasindan geliyor.
39     return oda_sicakligi;
40 }
41
42 void loop() {
43
44 // Create a client connection
EthernetClient client = server.available();
45 if (client) {
46     float tem = getTemp();
47
48     while (client.connected()) {
49         if (client.available()) {
50             char c = client.read();
51
52             //read char by char HTTP request

```

```

53   if (readString.length() < 100) {
54     // store characters to string
55     readString += c;
56     Serial.print(c);
57   }
58
59   // if HTTP request has ended
60   if (c == '\n') {
61     Serial.println(readString); // print to serial monitor for
62       debuging
63
64     client.println("HTTP/1.1 200 OK"); //send new page
65     client.println("Content-Type: text/html");
66     client.println();
67     client.println("<HTML>");
68     client.println("<HEAD>");
69     client.println("<meta http-equiv='refresh' content='4' />");
70
71     client.println("<meta name='apple-mobile-web-app-capable' content='yes' />");
72     client.println("<meta name='apple-mobile-web-app-status-bar-style' content='black-translucent' />");
73     client.println("<link rel='stylesheet' type='text/css' href='http://randomnerdtutorials.com/ethernetcss.css' />");
74     client.println("<TITLE>FAN CONTROL SYSTEM</TITLE>");
75     client.println("</HEAD>");
76     client.println("<BODY>");
77     client.println("<H1>FAN CONTROL SYSTEM</H1>");
78     client.println("<hr />");
79     client.println("<br />");
80     client.println("<H2>Arduino with Ethernet Shield</H2>");
81     client.println("<br />");
82     client.println("<a href=\"/?button1on\">Turn On FAN</a> ");
83     client.println("<a href=\"/?button1off\">Turn Off FAN</a><br /> ");
84     client.println("<br /> ");
85     client.println("<br /> ");
86     // client.println("<a href=\"/?button2on\">Rotate Left </a> ");
87     // client.println("<a href=\"/?button2off\">Rotate Right</a><br /> ");
88     client.print("ODA SICAKLIGINIZ : ");
89     client.print(temp);
90     client.println(" DERECEDIR ");
91     client.println("<br /> ");
92     client.println("</BODY> ");
93

```

```

95  client.println("</HTML>");  

96  //controls the Arduino if you press the buttons  

97  //Voltajimizi derece cinsine cevirmek icin carpik. 0.48828125  

98  // ifadesi [(5V*1000) /1024]10 hesaplamasindan geliyor.  

99  

100 while (25 < tem){  

101   Serial.print("Sicakligimiz :");  

102   Serial.print(oda_sicakligi);  

103   Serial.print("enerji verildi");  

104   digitalWrite(led, HIGH);  

105   tem = getTemp();  

106   break;  

107   if (tem<25){  

108     digitalWrite(led, LOW);  

109   }  

110  

111   delay(1);  

112   //stopping client  

113   client.stop();  

114   if (readString.indexOf("?button1on") >0){  

115     Serial.print("enerji verildi");  

116     digitalWrite(led, HIGH);  

117   }  

118   if (readString.indexOf("?button1off") >0){  

119     digitalWrite(led, LOW);  

120   }  

121   if (readString.indexOf("?button2on") >0){  

122     for(pos = 0; pos < 180; pos += 3) // goes from 0 degrees to  

123     // 180 degrees  

124     {  

125       microservo.write(pos); // tell servo to go to  

126       // position in variable 'pos'  

127       delay(15); // waits 15ms for the servo  

128       // to reach the position  

129     }  

130   }  

131   if (readString.indexOf("?button2off") >0){  

132     for(pos = 180; pos>=1; pos-=3) // goes from 180 degrees to  

133     // 0 degrees  

134     {  

135       microservo.write(pos); // tell servo to go to  

136       // position in variable 'pos'  

137       delay(15); // waits 15ms for the servo  

138       // to reach the position  

139     }  

140   }  

141   //clearing string for next read  

142   readString="";  


```

```
137 }  
139 }  
141 }  
143 }
```

Bu kod parçasını yazıp programımızı yüklediğimizde aynı modem üzerinden yani aynı ip adresi üzerinden sistemimizi kontrol edebileceğimiz arayüze ulaşmış oluyoruz. Bu arayüz sayesinde fanımızı manuel olarak açıp kapatabiliyoruz. Ayrıca sistemimize eklediğimiz sıcaklık sensörü sayesinde de anlık olarak ortam sıcaklığını ölçebiliyoruz. Yukarıdaki kodda sıcaklık sensörünün çalışması için gerekli metodlar yazılmıştır ve sorunsuz olarak çalışmaktadır. Sıcaklık sensörünü şemasını da bu yazımada anlatacağım. Aynı ip adresi üzerinden(192.168.1.37) bağlantı sağladığımız ekran görüntüsünü Şekil 7.54'te görebiliriz.

Fakat mobil veri üzerinden yada başka bir ağdan bağlanmak istersek o zaman modemimizin arayüzüne girerek dış port açmamız gereklidir. Bunun da ekran görüntüsünü Şekil 7.55'te paylaştım.

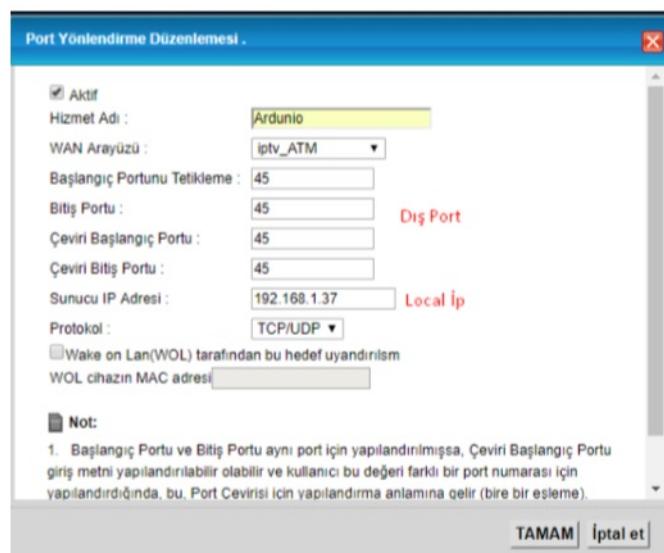
Ve kodumuzun içerisinde bulunan:

```
EthernetServer server(45); //server port
```

kısmina da port yönlendirme düzenlemesinde verdigimiz sayı değerimizi yazdım. Bu işlemlerden sonra programımızı derleyip çalıştığımızda sistem mobil veriden erişilebilir hale gelecektir. Dış ip'den erişebilmek için son olarak yapmamız gereken <https://whatismyipaddress.com/tr/ip-im> ile ip adresimizi öğrenmek ve dış portumuzu yazmak. Örneğin ip adresimiz 78.170.129.4 ise sonuna :45 ekleyerek(78.170.129.4:45) mobil veri yada herhangi başka bir ağdan fan kontrol sistemimize erişim sağlayabiliriz. Mobil veriden erişimimin ekran görüntüsünü de Şekil 7.56'da paylaşıyorum.



Şekil 7.54: Arayüz ekran görüntüsü

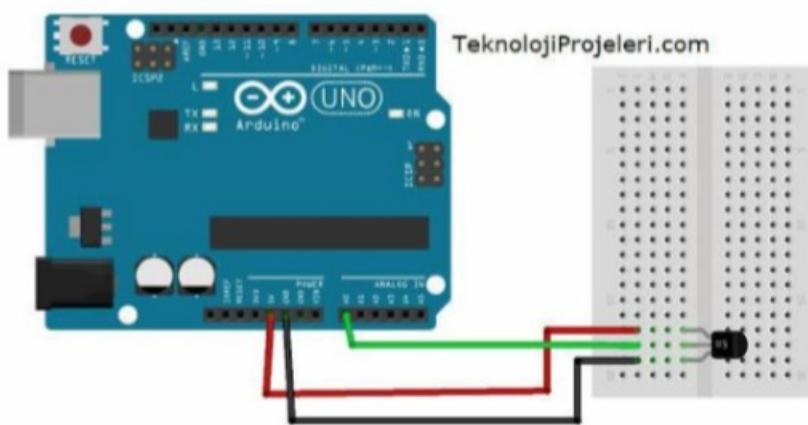


Sekil 7.55: Modem arayüzünden dış port açma



Şekil 7.56: Mobil veriden uygulamaya erişim

Sistemimizdeki sıcaklık sensörünün anlık olarak sıcaklık değerimizi ölçtüğünü ve arayüz ekranında bu sıcaklık değerini gösterebildiğimizi belirtmiştik. Bu sıcaklık sensörsümüzün ölçüdüğü sıcaklık değeri eğer 25 santigrat dereceyi geçerse fanımız sıcaklık değeri 25 derecenin altına düşünceye kadar çalışmaya devam edecektir. 25 derecenin altına ulaşıldığında ise fanımız otomatik olarak duracaktır. Bu durumu videomda da ayrıntılı olarak anlattım. Sıcaklık sensörsümüzün devre şeması Şekil 7.57'de görüldüğü gibidir.



Şekil 7.57: Sıcaklık sensörü devre şeması

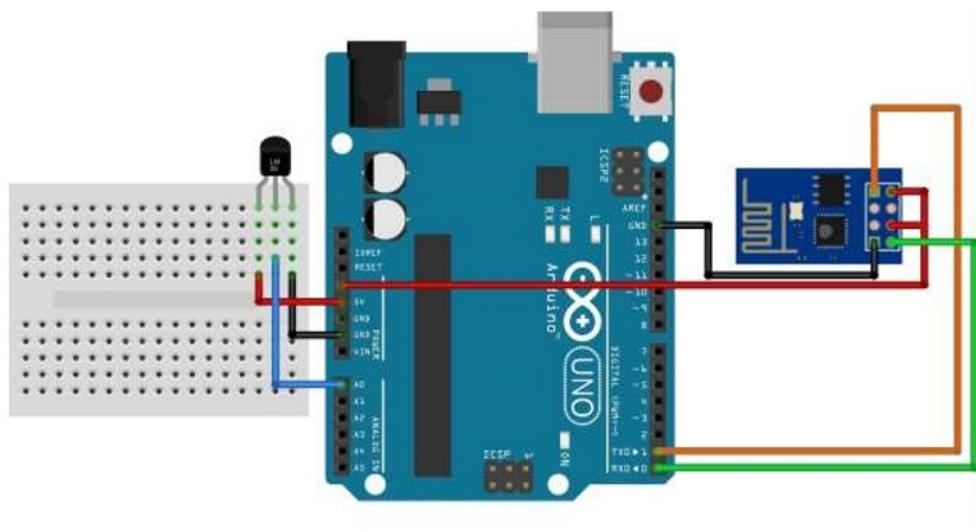
Ethernet shield üzerinde de yine aynı şekilde Vcc, GND ve analog 0 pinlerine bağlarız.

Sonuç olarak ortam sıcaklığına göre otomatik olarak açılıp kapanabilen ve aynı zamanda manuel olarak internet üzerinden kontrol edilebilen bu arduino projemiz kullanılır hale gelmiş bulunmaktadır.

## 7.23 LM35 ve ESP8266 ile İnternet Termometresi

### Devre Kurulumu

Bu proje Arduino'nun LM35'den aldığı sıcaklık değerini ESP8266'ya gönderir. ESP8266 da ThingSpeak'a göndererek bir veri günlüğü olmasını sağlar. Öncelikle bu modülü Arduino'ya takacağımız için 3.3V-5V farkını gidermemiz gereklidir. Bunun için hazır olarak satılan regülatörler yada Arduino'nun 3.3V çıkış birimi kullanılabilir. Bundan başka gerilimi bölmek için gerilim bölücü direnç kullanabiliriz. Önemli olan 3.3V ile çalışan ESP8266 modülüne hiçbir şekilde aşırı gerilim uygulanmasını engellemektir. Devreyi kurmak için bir adet Arduino Uno, bir adet ESP8266 modülüne bir adet de LM35 entegresine ihtiyacımız vardır.



Şekil 7.58: Devre şeması

Seri iletişim bağlantısı için USB-TTL çeviri modülü değil de Arduino'yu kullandım. Arduino'nun SoftwareSerial kütüphanesi bu seri iletişimini herhangi bir bacaktan yazılım tabanlı kullanmamızı sağlar. 0 ve 1 ayakları normalde RX ve TX yani deri iletişim ayakları olarak da kullanılabilir. Ama program atarken ve Arduino'nun seri iletişimini kullanırken bir sıkıntı çekmaması için böyle bir yol izledim. Bu devreyi kurmadan önce ESP8266'yı ağa bağlayıp kullanmamız gerekmektedir. Bunu ESP'yi güncelleyerek yapabiliriz. Ağa bağlantısını yaptıktan sonra ESP8266 bunu içinde kaydedecektir ve sürekli aynı

işlemden kaçınılacaktır.

### ThingSpeak Kanal Oluşturma

Time Zone kısmını doğru girmenizi tavsiye ederim çünkü daha sonra yolladığımız veriler ile grafik oluşturduğumuzda zaman bilgisini doğru alması açısından önemli. Kayıt olup giriş yaptıktan sonra üstteki Channels menüsünden My Channels seçeneği seçildiğinde karşımıza aşağıdaki gibi bir menü geliyor.



Şekil 7.59: ThingSpeak kanal oluşturma - 1

Bu menüye geldiğinizde New Channel sekmesine tıklayın ve karşımıza gelen menü ile yeni bir kanal açma işlemini başlatabiliriz.

### New Channel

A screenshot of the 'New Channel' configuration form. The form has several input fields: a 'Name' field with a placeholder 'My Temperature Sensor', a 'Description' field with a placeholder 'My temperature sensor', a 'Field 1' section containing a 'Field Label 1' input field and a checked checkbox, and a 'Field 2' section which is currently empty. Below the form, there is a large red 'Create Channel' button.

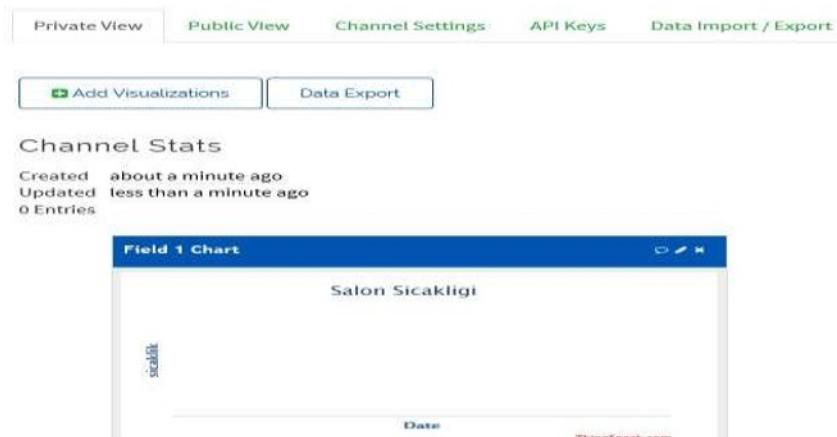
Şekil 7.60: ThingSpeak kanal oluşturma - 2

Gelen sayfada Name yazan yere kanalımızın ismini yazıyoruz. Ben buraya şimdilik salon sıcaklığı yazıyorum. Siz istediğiniz şekilde yazabilirsiniz. Zaten daha sonradan bu ayarları değiştirebilmek mümkün. Önemli olan Türkçe karakter kullanmamamız. Description kısmına ise kanalımız ile ilgili bir açık-

lama yazabiliriz. LM35 ile sıcaklığının olculmesi yazıyorum buraya da. Field1 yazan kısım ise yolladığımız veriler ile oluşturacağımız grafiğin ne grafiği olduğunu doğudur. Bu örneğimizde Sıcaklık-Zaman grafiği çizdireceğimiz için bu kısma sıcaklık yazıyorum.

Birden çok LM35 sensörü ile evin tamamının sıcaklığını ölçme ya da sıcaklık haricinde başka veriler de yollamak isterseniz Field 2, Field 3 gibi başka grafikler de oluşturabilirsiniz. Fakat şuan için sadece doldurdugumuz kısımlar bize yeterli olacak.

Bilgileri girip Save Channel ‘a tıkladığımızda kanalımız oluşturuluyor ve Şekil 7.61’deki gibi bir sayfaya yönlendiriliyoruz.



Şekil 7.61: ThingSpeak kanal oluşturma - 3

Bu sayfada yolladığımız verileri Sıcaklık-Zaman grafiği şeklinde görebileceğiz. Üst menüden biraz bahsetmek gerekirse Private View sekmesi grafiği sadece bizim gördiğimiz sekmedir. Eğer verileri herkes ile paylaşmak istersenek hemen sağındaki Public View sekmesine tıklayıp sayfa linkini paylaşmamız yeterli olacaktır. Channel Settings sekmesinden az önce ayarladığımız tüm bilgileri değiştirebilir veya yeni grafikler ekleyebiliriz. API Keys sekmesi ise bizim için önemli bir kısım, çünkü Arduino ve ESP modülümüz ile kanalımızı haberleştirirken bu sekme içinde bulunan, bize özel API keyimizi kullanacağız. Bu key tamamen bize özel olup, keyi kanalımız ile ESP modülümüz arasında bir şifre gibi düşünebilirsiniz. En sağdaki Data Import/Export sekmesinde ise daha önce elimizde kanalımıza ait bir veri var ise

bu veriyi grafiğimize eklemeye ya da grafiğimizdeki verileri bilgisayarımıza indirmeye yarar.

**NOT:** Devreyi kurup Arduino'ya kod yüklemeye çalışığınızda Arduino IDE hata verecektir. Bunun nedeni, Arduino'nun RX ve TX pinlerinin o an bağlı ESP modülüümüze bağlı olmasından dolayıdır. Yükleme işleminden önce RX ve TX bağlantılarını söküp, kod yüklendikten sonra takarsanız bu sorunu yaşamazsınız.

### Kod kısmı

```
1 #define ag_ismi "WiFi_Ismi"
2 #define ag_sifresi "WiFi Sifresi"
3 #define IP "184.106.153.149" //thingspeak.com IP adresi

5 float sicaklik;
6 void setup()
7 {
8     //Seriport'u aciyoruz. Guncelledigimiz
9     //ESP modulunun baudRate degeri 115200 oldugu
10    //icin biz de Seriport'u 115200 seklinde seciyoruz
11
12    Serial.begin(115200);
13    Serial.println("AT");

15    //ESP modulumuz ile baglanti kurulup
16    //kurulmadigini kontrol ediyoruz.
17    //ESP ile iletisim icin 3 saniye bekliyoruz.

19    delay(3000);

21    analogReference(INTERNAL);

23    if(Serial.find("OK")) {

25        //esp modulu ile baglantiyi kurabilmesek modul
26        //"AT" komutuna "OK" komutu ile geri donus yapiyor.

27        Serial.println("AT+CWMODE=1");
28        //esp modulumuzun WiFi modunu STA sekline getiriyoruz.
29        //Bu mod ile modulumuz baska aglara baglanabilecek.

31        delay(2000);
32        String baglantiKomutu=String("AT+CWJAP=\\"") + ag_ismi + "\",\""
33        + ag_sifresi + "\";
34        Serial.println(baglantiKomutu);
35        delay(5000);
36    }
37}
```

```

37 }

39 void loop(){
40     float sicaklik = analogRead(A0) / 9.31;
41     Serial.println(sicaklik);
42     sicaklik_yolla(sicaklik);
43     // dakikada 1 güncellenmesi için 1 dakika bekle
44     delay(60000);
45 }

47 void sicaklik_yolla(float sicaklik){

49     Serial.println(String("AT+CIPSTART=\\"TCP\\",\"" + IP + "\",80
50                     ));

51     //thingspeak sunucusuna bağlanmak için bu kodu kullanıyoruz.
52     //AT+CIPSTART komutu ile sunucuya bağlanmak için sunucudan
53     //izin istiyoruz.
54     //TCP burada yapacağımız bağlantı cihazını gösteriyor.
55     //80 ise bağlanacağımız portu gösteriyor

56     delay(1000);
57     if(Serial.find("Error")){
58         //sunucuya bağlanamazsa ESP modülü bize
59         //"Error" komutu ile donuyor.
60         Serial.println("AT+CIPSTART Error");
61         return;
62     }

63     String yollanacakkomut = "GET /update?key=64TOOS3R10EAYUML &
64         field1=\"";
65     // Burada 64TOOS3R10EAYUML yazan kısım bizim API Key den
66     // aldigımız Key. Siz buraya kendi keyinizi yazacaksınız.
67     yollanacakkomut += (int(sicaklik));
68     // Burada ise sıcaklığımız float değişkenine atayarak
69     // yollanacakkomut değişkenine ekliyoruz.
70     yollanacakkomut += "\r\n\r\n";
71     // ESP modulumuz ile seri iletişim kurarken yazdığımız
72     // komutların module iletilebilmesi için Enter komutu yani
73     delay(3000);
74     // /r/n komutu kullanmamız gerekiyor.

75     Serial.print("AT+CIPSEND=");
76     //veri yollayacağımız zaman bu komutu kullanıyoruz. Bu komut
77     // ile önce kaç tane karakter yollayacağımızı söylememiz
78     // gerekiyor.
79     Serial.println(yollanacakkomut.length()+2);
80     //yollanacakkomut değişkeninin kaç karakterden olustugunu .
81     //length() ile bulup yazdırıyoruz.

```

```

77   delay(1000);

79   if(Serial.find(">")){
80     //eger sunucu ile iletisim saglayip komut uzunlugunu
81     //gonderebilmissek ESP modulu bize ">" isareti ile geri
82     //donuyor.

83   // arduino da ">" isaretini gordugu anda sicaklik verisini
84   // esp modulu ile thingspeak sunucusuna yolluyor.
85   Serial.print(yollanacakkomut);
86   Serial.print("\r\n\r\n");
87 }
88 else{
89   Serial.println("AT+CIPCLOSE");
90 }
91 }
```

Bu kodu kendi WiFi adres ve şifremiz ile düzenledikten sonra kodu Arduino'ya yüklediğimizde verileri ThingSpeak'de oluşturduğumuz grafikte görmeye başlıyoruz.

#### Channel Stats

Created: [about a month ago](#)  
 Updated: [about 6 hours ago](#)  
 Last entry: [about 14 hours ago](#)  
 Entries: 136



Şekil 7.62: ThingSpeak grafiği

## 7.24 Seri Haberleşme (UART)

Arduino ve 74HC595 entegresini kullanarak seri iletişim ile bilgisayar kontrolü LED uygulaması gerçekleştirmi... .

### Kullanılan malzemeler

- Arduino UNO
- Breadboard
- 74HC595 shift register entegresi
- 8 adet LED
- 8 adet  $220\Omega$
- İki ucu erkek jumper kablo

Projede kullanacağımız LED'lerimizi bilgisayarımızdan vereceğimiz komutlar ile yakıp söndüreceğiz.

### Seri Haberleşme

COM portu, seri haberleşme (UART) dediğimiz bir bağlantı türü için kullanılmaktadır. Bilgisayarımızla Arduino'muzu haberleştirmek için Arduino yazılımında bulunan "Seri Port Ekrani"nı kullanacağız.

### Bağlantı Şeması

### Kod Kısmı

```

1 int latchPin = 5;
2 int clockPin = 6;
3 int dataPin = 4;
4
5 byte leds = 0;
6
7 void setup()
8 {
9     pinMode(latchPin, OUTPUT);
10    pinMode(dataPin, OUTPUT);
11    pinMode(clockPin, OUTPUT);
12    regiszeraYaz();
13    Serial.begin(9600);
14    while (! Serial);

```

```
16     Serial.print("1 ile 8 arasında bir LED numarası girin veya ");
17     ;
18   Serial.println("x ile hepsini sondurun");
19 }
20
21 void loop()
22 {
23   if (Serial.available())
24   {
25     char ch = Serial.read();
26     if (ch >= '1' && ch <= '8')
27     {
28       int led = ch - '1';
29       bitSet(leds, led);
30       registeraYaz();
31       Serial.print(led +1);
32       Serial.println(" numaralı LED yandi");
33     }
34     if (ch == 'x')
35     {
36       leds = 0;
37       registeraYaz();
38       Serial.println("Tüm LED'ler sondu");
39     }
40   }
41
42 void registeraYaz()
43 {
44   digitalWrite(latchPin, LOW);
45   shiftOut(dataPin, clockPin, LSBFIRST, leds);
46   digitalWrite(latchPin, HIGH);
}
```

Kodumuzun setup kısmında yer alan `Serial.begin(9600)` komutu, Arduino'muzun seri portunu 9600 baud'da (seri iletişim hızı) çalışacak şekilde ayarlıyor. `Serial.print` komutu ise Arduino'nun seri porttan bilgisayarla iletişime geçerek tırnak içinde yazının seri port ekranında görünmesini sağlıyor.