(ÜŞENGEÇLER İÇİN)

ARDUINO PROGRAMLAMA

YILMAZ ALACA | EMİNE YALÇIN

Yılmaz ALACA

2001 Adana doğumlu yazar. İlk ve orta öğrenimini Gazi İlk ve Ortaöğretim Okulu'nda tamamlamıştır. Orta okul döneminde C++ yazılım dili ile tanışmış ve oyun geliştirme üzerine çalışmalar gerçekleştirmiştir. Lise öğrenimini Şehit Temel Cingöz Anadolu Lisesi'nde tamamlayan yazar, Lise döneminde TÜBİTAK dahil olmak üzere çeşitli Robotik ve Kodlama şenliklerine katılmış, İşitme engellilerin hayatını kolaylaştırmak, fast food zincirleri için garson robot ve yüksek ateş hastalığından çocukları korumak amacıyla projeler geliştirmiştir. Üniversite öğrenimine Adana Alparslan Türkeş Bilim ve Teknoloji Üniversitesi'nde Elektrik ve Elektronik Mühendisliği bölümüne başlamış, kurduğu İNOVASYON Topluluğu'nda C++, Python ve gömülü sistemler üzerine eğitimler vermiştir. Şu an üniversite 3. sınıf öğrencisi olan yazar, okuduğu dönem boyunca Udemy'de eğitmenliğe başlamış ve 2,5 yılda 80 farklı ülkeden 20 Bin'den fazla öğrenciye ulaşmıştır. Udemy'de C++, Python, Gömülü Sistemler, Makine Öğrenmesi, Derin Öğrenme, Oyun Geliştirme ve Görüntü İşleme konularında eğitimleri bulunmaktadır. 2022 yılından itibaren Dene Yap Türkiye'de Robotik ve Kodlama eğitmenliği yapmaktadır.

Ayrıca, SkillShare ve Tutorialspoint online eğitim sitelerinde Raspberry Pi ve Bilgisayarlı Görü (Computer Vision) üzerine eğitimleri bulunmaktadır.

Teşekkür

Bugünlere gelmeme vesile olan başta annem Fatma ve babam Devrim olmak üzere, kardeşim Uygar'a ve lise dönemimde elektroniğe karşı heveslenmemi sağlayan, değerli Can Günöz öğretmenime bir teşekkürü borç bilirim.

Emine YALÇIN

2001 Gaziantep doğumlu yazar, ilkokul ve ortaokulu; Nuriye Zekeriya Kına İlköğretim Okulu, Liseyi; İstanbul Gaziantepliler Anadolu Lisesi'nde dereceyle tamamlamıştır. Lisans eğitimine; Adana Alparslan Türkeş Bilim ve Teknoloji Üniversitesi, Havacılık ve Uzay Mühendisliği Bölümü'nde 3. Sınıf öğrencisi olarak devam etmektedir.

Python, C++ ve Matlab programlama dilleri üzerine çalışmakta ve projeler gerçekleştirmektedir. Python ve Matlab programlama dilleri üzerine yazmış olduğu el kitapları bulunmaktadır. AutoCad ve SolidWorks ile üç boyutlu çizimler yapmaktadır. Roketler ve hava araçları üzerine projelerde mentorluk yapmaktadır. Bireysel olarak lise öğrencilerine Matematik ve Yazılım dersleri vermektedir.

Teşekkür

Her zaman desteklerini yanımda hissettiğim; bana iyi bir insan olmayı öğreten sevgili annem ve babama, sevgiyi ve paylaşmayı öğreten kardeşlerime, güzeller güzeli Mihrimah'ıma, beni yazılıma teşvik eden değerli hocam Yılmaz Alaca'ya, Hayatım boyunca bana öğreten herkese sonsuz teşekkürlerimi sunarım.

İçindekiler

1

Giriş

Bu Bölümde

Bu bölümde, kitabımızın amacını öğrenip, programlama dilleri ve bilgisayarlar hakkında bilgi sahibi olacağız.

Ayrıca, C++ yazılım dilinin tarihçesini ve kullanım alanları hakkında bilgi sahibi olup, neden *C++ ve Arduino öğrenmeliyiz?* sorusunun cevabını öğreneceğiz. Ek olarak, gömülü sistemin ne olduğunu, yazılım ve matematik ilişkisini ve Algoritma hakkında da kısaca bilgi sahibi olacağız.

KİTABIN AMACI

Üşengeçler için Arduino Programlama kitabı ile birlikte C++ yazılım dilinin dünyasına giriş yaparak, elde ettiğimiz bilgilerle Arduino kartımızı nasıl kodlayabileceğimizi öğreneceğiz.

Bu kitap siz değerli okurlarıma iyi bir kaynak, özgün proje fikirleri ortaya çıkarmak, C++ yazılım dili ile Arduino kartını bir arada kullanabilmek amacıyla oluşturulmuştur. Ayrıca, günümüzde Türkçe ya da farklı dillerdeki Arduino kitaplarından ayrı olarak siz değerli okurlarıma yapabileceklerinizin yalnızca hayal gücüyle sınırlı olmadığını göstermek olacaktır.

Özellikle öğrenciler, yazılım ve programlamaya adım atarken aslında öğrendiği yazılım dillerini özgün projeye dökememekte ya da bu konuda ciddi anlamda zorlanmakta. Bu sebeple, sizlerle birlikte öncelikli olarak C++ yazılım dilinin temellerine giriş yapacağız. Bununla da sınırlı kalmayıp Arduino ile birlikte çeşitli modüllerin, ekranların ve sensörlerin kullanımı hakkında da bilgi sahibi olacağız.

PROGRAMLAMA DİLLERİ VE BİLGİSAYARLAR

Özellikle günümüzde hemen herkes tarafından gözde hale gelmiş programlama dilleri aslında bilgisayarı kontrol edebilmemiz için gerekli olan ve haberleşmesini sağlayan araçlardır. Bir bilgisayar genel olarak 2 kısımdan oluşur. Yazılım ve Donanım. **Yazılım,** donanımlarımıza anlam katan, gerekli komutları yaptırabilen yapıdır. **Donanım ise,** bugün gözle gördüğümüz her bilgisayar parçasıdır ve bu donanımlar gerekli görevleri yapabilmesi için yazılımlara ihtiyaç duyar. Elbette yazılımlar da programlama dilleri sayesinde oluşturulur.

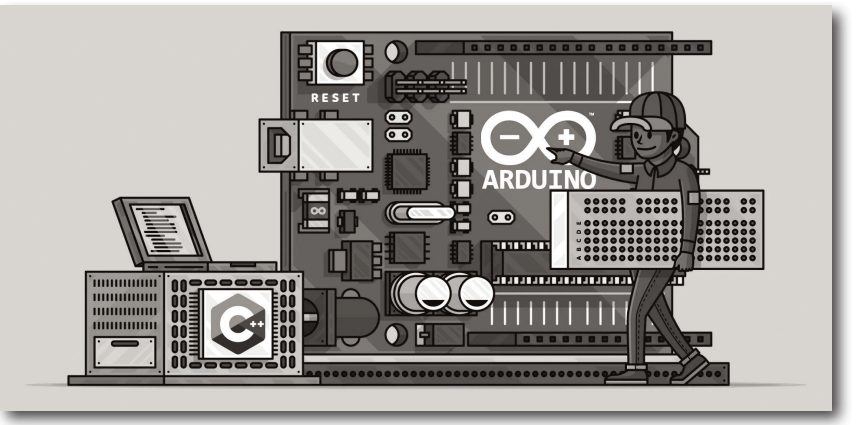
C++ Yazılım Dili

C++ yazılım dili, 1979 yılında geliştirilmeye başlanmış orta seviyeli bir yazılım dilidir. Günümüzde gömülü sistemler, işletim sistemleri, internet tarayıcıları ve oyun yapımında kullanılan bir dildir. Uzantısı *.cpp* olup, halen birçok şirket tarafından desteklenmektedir. Son sürümü 2011 yılında çıkmıştır. Bugün Windows, MacOS, Linux vb. işletim sistemlerinde çalışabilmektedir.

Neden C++ ve Arduino Öğrenmeliyiz?

Elbette bu kitabı okurken kendinize bu soruyu soruyorsunuzdur. Haydi açıklamasını yapalım.

C++ yazılım dili, bugün gömülü sistem adını verdiğimiz, yani işlemcisi ve kendine özgü elektronik parçaları olan donanımları kodlamamıza olanak sağlayan bir yazılım dilidir. Arduino kartımızı kodlayabilmemiz için C++ yazılım diline ihtiyaç duymaktayız. Böylelikle, C++ yazılım dili ile birlikte Arduino kartımızı kodlayacağız. Arduino tarafında ise hem endüstriyel hem de hobi amaçlı projeleri, Arduino kartlarımızla birlikte yapabilmekteyiz. Üzerinde bulunan güç pinleri sayesinde çeşitli modülleri kart üzerinde çalıştırabilir ve kontrol edebiliriz. Üzerinde bulunan dijital pinler sayesinde ise buton veya dijital veri okuyabilen sensörlerdeki verilere göre işlemler yapabiliriz. Ayrıca, modülleri kontrol edebilmemiz için gerekli sinyalleri yollayabiliriz. Üzerinde bulunan analog pinler sayesinde çeşitli sensörleri kullanabilir ve bu sensörlerdeki verilere göre projeler geliştirebiliriz.



2

Temel C++ Bilgisi

Bu Bölümde

Kitabımızın bu bölümünde C++ yazılım dilinin dünyasına giriş yapacağız.

Bu bölümden elde ettiğimiz bilgiler ve kazanımlar sayesinde Arduino kartımızın kodlayabilmemiz için gerekli olan C++ bilgisini öğreneceğiz. Ayrıca, bu bölüm ile birlikte C++ yazılım dilinde kodlarımızı yazabilmemiz için gerekli olan ortamımızın kurulumuna, yazım biçimine (syntax), değişken kavramına, koşullu yapılara, döngülere, fonksiyonlara, dizilere ve kütüphanelere giriş yapacağız.

IDE NEDİR?

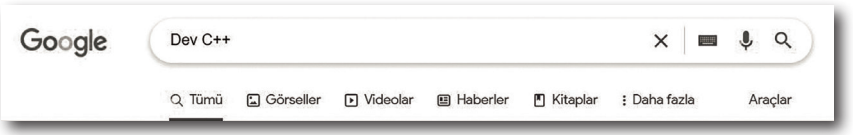
İngilizce yazılışını incelediğimizde Türkçe'si, **Tümleşik Geliştirme Ortamı** olarak isimlendirilmektedir. IDE, programcıların kodlarını kolay bir şekilde yazabilmesi için oluşturulmuş yazılımlardır. Günümüzde bir çok yazılım dilinin birbirinden farklı IDE'si bulunmaktadır. Ek olarak, bugün bu yazılımlar sayesinde kodlarımızda bulunan hataları kolaylıkla bulabilir, kodlarımızı derleyebilir ve çalıştırabiliriz. Biz de C++ kodlarımızı yazabilmemiz için Dev C++ IDE'sinden yararlanacağız.

DEV C++ KURULUMU

Günümüzde C++ kodlarımızı yazabilmemiz için kullanım kolaylığı sebebiyle Dev C++ IDE'si halen kullanılmaktadır. Biz de bu IDE üzerinden kodlarımızı yazacağız. Haydi şimdi adım adım kurulumu gerçekleştirelim.

1. Adım

Google'a Dev C++ yazalım ve aratalım.



2. Adım

sourceforge.net üzerinden kurulumumuzu gerçekleştireceğimiz için ***sourceforge.net*** sayfasına girelim.

3. Adım

Girdiğimiz web sayfasında bulunan yeşil renkli **Download** butonuna tıklayalım.

4. Adım

Kurulumumuz gerçekleştikten sonra indirilen dosyaya tıklayalım.

5. Adım

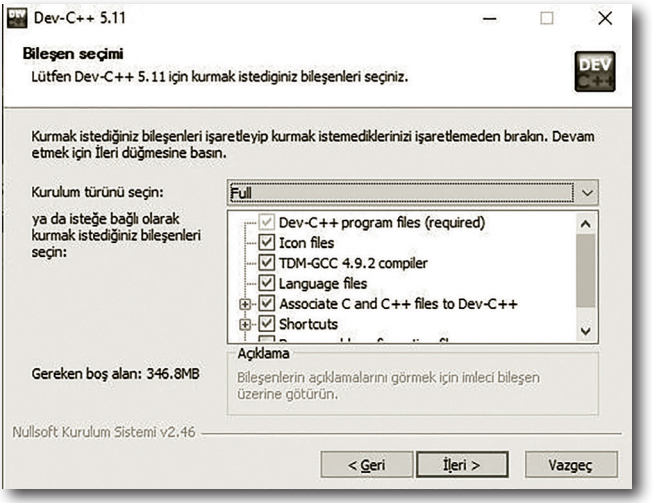
İlk olarak dil seçmemiz gerekli. İstediğiniz dili seçtikten sonra **OK** tuşuna basalım.

6. Adım

Karşımıza çıkan lisans sözleşmesini onaylayalım.

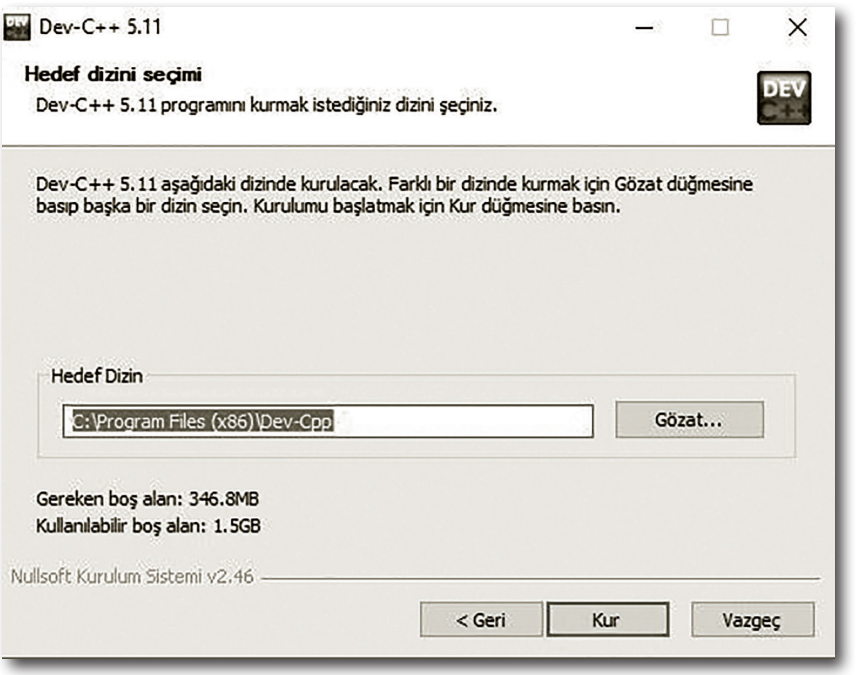
7. Adım

Bileşen seçimi kısmından seçebileceğimiz seçeneklerin hepsini seçelim ve **İleri** tuşuna basalım.



8. Adım

Kurulum ekranında son yapacağımız işlem ise kurmak istediğimiz klasörü ayarlamak (Varsayılan olarak bırakmanızı tavsiye ediyorum). Hedef klasörünü seçtikten sonra **Kur** butonuna tıklayarak devam edelim.



9. Adım

Kurulumu başarıyla tamamladık. Şimdi Dev C++ IDE'sini açalım. Karşımıza tekrardan bir dil seçme ekranı geldi. Buradan kullanacağımız IDE üzerindeki her şeyin hangi dilde olduğunu belirtmemiz gerekli. Dilimizi ayarladıktan sonra **Next** tuşuna basalım.

10. Adım

Bu adımda yazacağımız kodların fontunu, kodları yazdığımız kaynak kodu dosyalarının temasını ve IDE içerisinde bulunan ikonları ayarlamamız gerekli. Dilediğinizce ayarlamanızı yaptıktan sonra tekrardan "Next" tuşuna basalım.

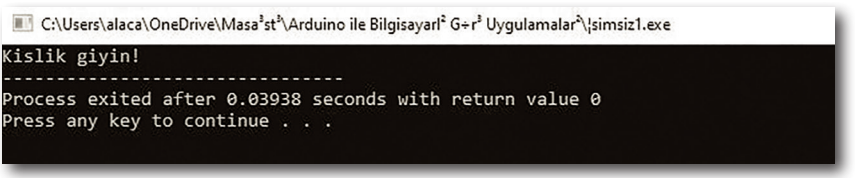
11. Adım

Başarıyla kurulumu tamamladık. Tebrikler :).

rumda, yani yanlış olan her durumda kullanılmasıdır. Haydi aşağıdaki örneği inceleyelim.

Örnek Kod

```
#include<iostream>
using namespace std;
int main()
{
  bool havaSicak=false;
  if(havaSicak==true)
  {
          cout<<"Yazlik giyin!";
  }
  else
  {
          cout<<"Kislik giyin!";
  }
}
```



Yukarıdaki örnekte `bool` değişken tipinde bir değişken oluşturduk. Bildiğiniz gibi mantıksal değişkenlerimiz true, yani **doğru**. false, yani **yanlış** değerler alırdı. Bizde `havaSicak=false` şeklinde bir değişken tanımlayarak havanın sıcak olmadığını belirttik. Ancak, `if` koşulumuzu oluştururken `havaSicak` değişkeninin `true` değerine eşit olup olmadığını kontrol ettik. Eşitliği `true` değerine göre yaptığımız için ve `havaSicak=false` olduğu için program `else` koşulu içerisine girdi ve bize **`Kislik giyin!`** şeklinde çıktı verdi.

Peki, birden fazla koşullu durumu kontrol edebilmemiz için ne yapmamız gerekli?

Bunun için ise `else if` koşulundan yararlanmamız gerekli. Haydi else if koşulu hakkında bilgi sahibi olalım.

oluşturulmalıdır. Bu formatta bahsedilen değişken tipi; `int`, `string`, `float` şeklinde olabilmektedir.

Değişken ismi ise diziye verdiğimiz isimdir. **Köşeli parantez** içerisindeki eleman sayısı kısmına, dizinin içerisinde kaç değer varsa o sayı girilebilir ya da daha yüksek bir değer girilebilir ancak daha düşük bir değer girilemez. Örneğin: eleman sayısı kısmında 10 değeri varsa dizi içerisine maksimum 10 tane değer girilebilir. Daha sonra dizi oluşturmak için süslü parantezlerimizi kullanarak, içerisine gireceğimiz her bir değerin arasına **virgül** koyarız. Bir örnek üzerinden görelim.

```
int tek_rakamlar={1,3,5,7,9};
```

Dizi içerisindeki değerlerin her biri için bir `indis` değeri bulunmaktadır. Baştan sona doğru ilk değer 0. indisi, ikinci 1. indisi, üçüncü 2. indisi temsil eder ve bu şekilde numaralandırmaya devam edilir. Bizler bu indeks numaralarına göre dizi içerisindeki değerlere ulaşabiliriz. Peki bu değerlere nasıl ulaşırız ve ulaşarak neler yapabiliriz bunu öğrenelim.

Dizi içerisinde herhangi bir değeri çağırmak istersek; `cout<<degisken_ismi[indis no];` formatında istenen değerin indeks numarasını girerek, o değere ulaşabiliriz. Eğer dizi içerisinden herhangi bir elemanın değerini değiştirmek istersek, `degisken_ismi[indis no]=yeni değer;` şeklinde ifade etmemiz gerekir. Herhangi bir elemanın yerine kullanıcıdan çıktı alarak başka bir değer girme işlemini yaptırmak istersek, `cin>>degisken_ismi[indis no];` formatını kullanırız. Örnek üzerinden inceleyelim:

```
int rakamlar[10]={0,1,2,3,4,5,6,7,8,9}
```

Cout<<rakamlar[0];

Çıktı olarak 0. İndiste bulunan 0 değerini alırız.

rakamlar[5]=7;

Artık dizimizdeki 5. İndis yerine 7 değeri atanacaktır.

Cin>>rakamlar[3];

3. indisteki değer yerine kullanıcıdan bir değer girilmesini ister ve girilen değeri 3. İndis yerine atar.

Örnek 7 (Dizi Elemanlarını Oluşturma ve Toplama İşlemi Yaptırma)

Bizler bu örneğimizde, 6 tane değer atama işlemi yapıp her bir değeri kullanıcıdan alacağız. Sonrasında ise her bir değeri birbiriyle toplayıp çıktı alacağız. Şimdi bu işlemleri nasıl yapacağımızı kodlarımız üzerinden inceleyip yorumlayalım.

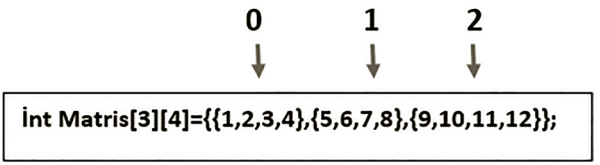
Örnek Kod

```
#include<iostream>
using namespace std;
int main()
{
  int degerler[6];
  for (int i=0;i<6;i++)
  {
    Cout<<"lutfen dizinin "<<i<<".indesksini giriniz: ";
    cin>>degerler[i];
  }
  int toplam=0;
  for(int x=0;x<6;x++)
  {
          toplam+=degerler[x];
  }
  cout<<"Kullanicinin girdigi degerler toplami:"<<toplam;
}
```

Kodlarımızı incelediğimizde öncelikle **iostream** kütüphanemizi çağırdığımızı görüyoruz. Standart isim uzayını çağırıp, ana fonksiyonumuzu oluşturduk. Sonrasında 6 elemanlı `degerler` isminde bir dizi oluşturduk. Her bir indise değer atamak için, 0. indisten başlanarak 6. indise gelene kadar birer birer artacak şekilde döngü oluşturduk. `cin` kullanarak kullanıcıdan dizi için değerler istedik. Daha sonra `toplam` isminde bir değişken oluşturup 0 değerini atadık. Değerlerin üstüne her seferinde x kadar indeksin eklenmesini sağlayarak, toplama işlemini yapacak sayaçlı döngü oluşturduk. Böylelikle kullanıcıdan aldığımız her değer `"Kullanicinin girdigi degerler toplamı:"` şeklinde çıktı vererek hesaplanacaktır.

Peki, çok boyutlu bir dizi, yani satır ve sütunları bulunan bir dizi (matris) oluşturma işlemini nasıl gerçekleştiririz bunun hakkında bilgi sahibi olalım. Bizler bu işlem için çoklu köşeli parantezlerden faydalanırız. `"degisken tipi degisken_ismi[satır sayısı][sütun sayısı]={};"` şeklinde oluşturmamız

gerekmektedir. Burada bilinmesi gereken bir diğer husus da **süslü parantezin** içerisine değerleri nasıl gireceğimizdir. Satırları ayırmak için süslü parantezin içerisine, kaç tane satır varsa o kadar süslü parantez ekleriz. Bu süslü parantezlerin her biri bir satırı temsil etmektedir ve aralarına **virgül** konulmalıdır. Her bir parantezin ise bir indeks numarası bulunmaktadır. Bunlar soldan sağa 0,1,2... şeklinde numaralandırılmaktadır. Satırların içerisine değer girerken, her bir satıra sütun sayısı kadar ve aralarına virgül koyularak değer girilmektedir. Satır içerisindeki değerlerin de yine indeks numaraları bulunmaktadır. İndeks numarası verilirken her bir satır birbirinden bağımsız olarak 0'dan başlanarak değer verilmektedir. Şimdi ise bu anlatılanları örnek üzerinden daha iyi anlayalım.



Örneğimizde Matris adında bir dizi oluşturduk. İlk köşeli parantezimizin içinde belirttiğimiz 3 değeri satır sayımızı temsil ederken, 4 değeri sütun sayımızı temsil etmektedir. Satır sayısı 3 olduğundan 3 tane süslü parantez oluşturduk ve her birinin içerisine 4 (sütun sayısı kadar) adet sayı değeri girdik. Satırların yukarısında belirtilen değerler ise her bir satırın indeks numarasını temsil etmektedir.

Matris içerisindeki değerlere nasıl ulaşacağımız hakkında bilgi sahibi olalım. Bunun için `cout<<degisken_ismi[satırın indeks numarası][satır` içerisindeki sayının indeks `numarası];` formatında kodumuzu oluşturmamız gerekmektedir. Şimdi bu durumu Matris adında oluşturduğumuz örnek dizi üzerinden inceleyelim. Örneğin bizler ikinci satırdaki 6 değerine ulaşmak istersek, `cout<<Matris[1][1];` şeklinde ifade etmemiz gerekir. Buradaki ilk parantezdeki 1 değeri 2. satırın indeks numarasını, ikinci parantezdeki 1 değeri ise o satırın içerisindeki 6 değerinin indeks numarasını temsil etmektedir. Eğer bizler matrisin içerisindeki tüm değerlere ulaşmak istersek, matrisi tanımlayıp, iç içe `for` döngüsü oluşturmamız gerekmektedir. Kodumuz üzerinden inceleyebilirsiniz.

3

Arduino Temelleri, Uygulamalar ve Projeler

Bu Bölümde

Bu bölümde, sizlerle birlikte Arduino dünyasına giriş yapacağız.

Öncelikle, Arduino UNO kartı hakkında bilgi sahibi olup, kartımızı kodlayabilmemiz için gerekli olan temel fonksiyonlarımızı öğreneceğiz. Elde ettiğimiz temel bilgiler ile birlikte çeşitli modülleri, sensörleri, motorları ve ekranları kullanmayı öğrenip, çeşitli projeler geliştireceğiz.

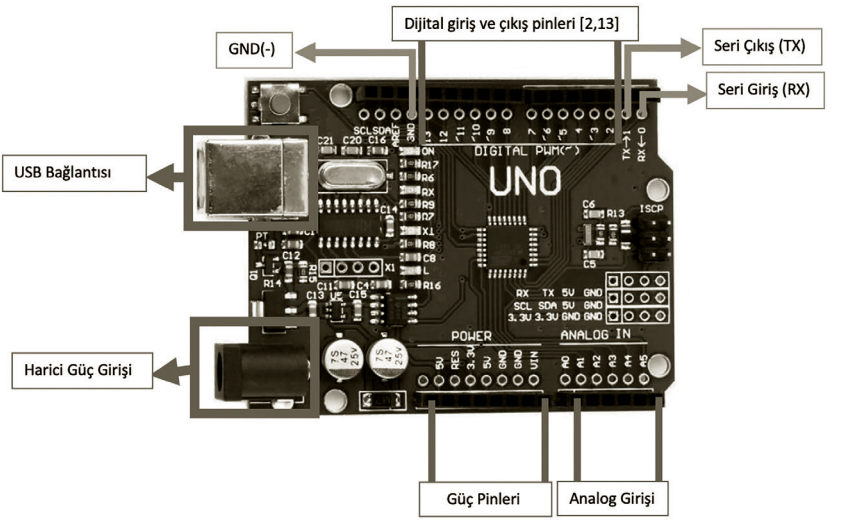
Öğrendiğimiz temel C++ bilgisi, Arduino kartlarımızı kodlarken işimizi oldukça kolaylaştıracak.

ARDUINO NEDİR?

Arduino, çeşitli elektronik ve gömülü sistem projelerini yapabildiğimiz elektronik karttır. Üzerinde çeşitli giriş ve çıkış pinleri bulunmakta olup, yalnız veya farklı cihazlarla eş zamanlı çalışabilmektedir. Bugün dünya da oldukça popülerdir ve hemen her hobi projelerinde, okul projelerinde, prototip cihazlarda ve endüstriyel cihazlarda Arduino kullanılmaktadır.

Yani, kullanım alanı oldukça geniş bir karttır. Örneğin; bir uzaktan kumandalı araba yapmak istediğimizde, uzaktan kumanda kontrolünü ve aracın kontrolünü Arduino ile birlikte gerçekleştirebiliriz. Arduino kartının çeşitli modelleri bulunmaktadır (Arduino UNO, Arduino Pro Mini, Arduino Mega vs. gibi). Biz de projelerimizi ve örneklerimizi geliştirebilmek için Arduino UNO kartından yararlanacağız. Haydi kartımıza yakından bakalım.

ARDUINO UNO



Yukarıda kartımızı incelediğimizde çok fazla pinlerin olduğunu görmekteyiz. Bu sizlere korkutucu gelmesin. Aslında pinleri bir ya da iki kez kullandığımızda eliniz alışacaktır. Dilerseniz pinlerimizi inceleyelim;

Dijital (PWM) Pinleri: Dijital pinler çok amaçlı kullanılabilir. Bunlar giriş, çıkış ve PWM olarak adlandırılabilirler. Bu pinler gerektiğinde güç çıkışı, gerektiğinde gücü kesme, gerektiğinde sinyal kontrolü ve gerektiğinde de güç kontrolü yap-

mamıza olanak sağlayan pinlerdir. Örneğin; biz butona bastığımızda basılıp basılmadığını bu pinler ile kontrol ederiz. Biz motorun hızını kontrol etmek istediğimizde PWM hatlarını kullanarak motor hız kontrolü sağlayabiliriz. Örneğin; bir LED'i yakmak istediğimizde herhangi bir dijital pini kullanarak ledi yakabiliriz. Tabii ki yaktıktan sonra verdiğimiz güçleri kökten de kesebiliriz. Burada önemli olan biz ne istersek pinlerimiz üzerinden de isteklerimizi yerine getirebilmemiz. Bu sebeple de Arduino kartı fazlasıyla işimizi görecektir.

USB Bağlantısı: Tabii ki Arduino kartımızı kontrol etmek istediğimizde, Arduino kartımızdaki verileri monitör aracılığıyla okumak istediğimizde veya kartımızın çalışıp çalışmadığını kontrol etmek istediğimizde Arduino'daki USB bağlantı portundan yararlanırız. Bu port üzerinden kartımızı bilgisayara bağlayabilir ve yazdığımız kodları kolaylıkla kartımıza yükleyebiliriz.

Analog Girişi: Arduino kartındaki analog pinler ile birlikte sensörlerin okuduğu verileri öğrenebilir ve bu veriler üzerinden işlemler gerçekleştirebiliriz. Örneğin: ortamın ışık şiddetini ölçmek istediğimizde bu pinlerden yararlanmamız gerekli.

Güç pinleri: Şüphesiz bir cihazı kontrol etmek istediğimizde bizlere + ve – hatları gerekmektedir. Örneğin: bir LCD ekranı kontrol etmek istediğimizde ekranımıza güç verebilmek için Arduino'daki güç hatlarından yararlanabiliriz. Kartımız bize bu olanağı sağlamakla birlikte, 5 veya 3.3V olarak iki seçenek sunuyor. Özellikle elektronik devrelerde bu 2 farklı volt değeri çok sık kullanılmaktadır. Örneğin kablosuz haberleşme modüllerinde 3.3V kullanmamız gerekirken, ekran kullanımında 5V hattından yararlanmamız gereklidir.

Harici Güç Girişi: Arduino kartımızı pil yardımıyla ya da şarj cihazlarıyla kullanabilmek amacıyla harici güç yuvası bulunmaktadır. Tabii ki USB üzerinden de kartımıza güç gelmektedir ancak projemizi geliştirdikten sonra bilgisayar olmadan kullanmak istediğimiz de harici güç yuvasından yararlanabiliriz.

GND: Kartımızda – (eksi) hatlardır. Elektronik devrelerin – (eksi) hattı, Arduino'daki GND hatlarına bağlanmaktadır.

Seri Çıkış(TX) ve Seri Giriş(RX): Arduino kartımızda seri haberleşmenin yapıldığını söylemiştik. Eğer kartımızdan veri gönderme işlemi yapacaksak seri çıkış hattını, veri alma işlemi yapacaksak seri giriş hattını kullanabiliriz.

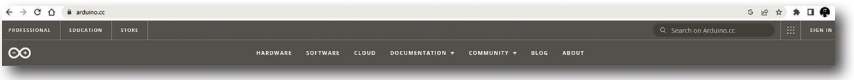
Kartımızın genel yapısını öğrendiğimize göre, kartımızı kodlayabilmemiz için gerekli olan Arduino IDE'sinin kurulumunu gerçekleştirelim.

ARDUINO IDE KURULUMU

Bu başlık altında Arduino IDE'sinin kurulumunu öğreneceğiz. Arduino IDE'sinin kurulumu oldukça basit ve ücretsizdir. Bizler kolay bir şekilde kurulum yapabilir ve kodlarımızı yazabiliriz. Haydi adım adım kurulumu gerçekleştirelim.

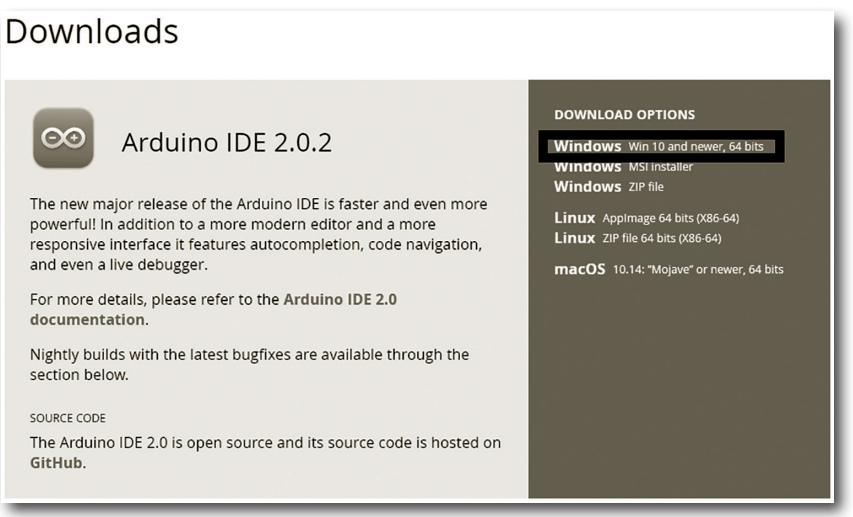
1. Adım

Arduino IDE'sinin kurulumu için Arduino'nun resmi web sayfasını açmamız gereklidir (***arduino.cc***). Sitemizi açtıktan sonra **SOFTWARE** seçeneğine tıklayalım.



2. Adım

SOFTWARE bölümüne tıkladıktan sonra bizleri IDE sürümü ve kurulum dosyaları karşılıyor. Şu anda bizim indirdiğimiz sürüm 2.0.2'dir. Tabii ki ilerleyen zamanda bu sürüm artabilir. Sizler de herhangi bir sıkıntı yaşamadan yeni sürümler üzerinden de çalışabilirsiniz. Sağ kısımda görüldüğü üzere indirme seçenekleri bulunmaktadır. Bizlere Windows, Linux ve MAC işletim sistemleri için indirme seçenekleri sunuyor. Ben Windows bilgisayar kullandığım için Windows için kurulum yapacağım. Sizler farklı işletim sistemi kullanıyorsanız, kendi işletim sistemine göre olanını seçin ve indirin.



Eğer Windows bilgisayar üzerinden kurulum yapacaksanız, siyah kutucuk içerisine alınmış seçeneği seçin ve indirin.

ARDUINO KARTININ KODLANMASI

Buraya kadar olan yazılarımızda C++ yazılım dilini, Arduino kartını ve Arduino IDE'sinin kullanımı hakkında bilgi sahibi olduk. Artık yapmamız gereken kartımızı kodlamak. Kodlamalara başlamadan önce Arduino kartını kodlarken kullanabileceğimiz en temel fonksiyonların kullanımı hakkında bilgi sahibi olalım.

PİNMODE FONKSİYONU

Arduino kartımızda bulunan pinleri giriş veya çıkış olarak ayarlamak için bizler `pinMode` fonksiyonundan yararlanırız. Akım ihtiyacı bulunan elemanlar çıkış birimi olarak, veri okuduğumuz elemanlarsa giriş birimi olarak tanımlanmaktadır. Buzzer, Motor, LED gibi elemanlar çıkış biriminde tanımlanırken, **sensör** ve **buton** gibi elemanlar giriş birimi olarak tanımlanmaktadır.

`pinMode` fonksiyonunu bizler `void setup( )` fonksiyonu içerisinde kullanmalıyız. `pinMode` fonksiyonunu kullanırken 2 parametre tanımlamalıyız. İlk parametreye işlem yapmak istediğimiz pin numarası ya da ismi, mod kısmına ise `INPUT` veya `OUTPUT` yazılmalıdır. İkinci parametreye `INPUT` yazdığımızda, pin kısmındaki değeri giriş pini olarak ayarlarız. İkinci parametreye `OUTPUT` yazdığımız zaman, pin kısmındaki değeri çıkış pini olarak ayarlarız. pinMode fonksiyonu `pinMode(pin_ismi,mod)` olarak kullanılmalıdır.

Şimdi ise fonksiyonu örnekler üzerinden gösterelim.

Örnek Kod

```
int ledPin=7;
int sensorPin=A2;
int butonPin=6;
void setup() {
  pinMode(ledPin,OUTPUT);
  pinMode(sensorPin,INPUT);
  pinMode(butonPin,OUTPUT);
}
```

Örneği incelediğimizde ilk olarak pinlerimizi değişken içerisinde tanımladık. Bu örneği gerçekleştirirken aynı zamanda pinleri tam sayı değişkenleri içerisinde tanımlayabileceğimizi de gözlemledik. Örneğimizdeki `ledPin` ve `butonPin` değişkenleri dijital pinlerimizi tutmaktadır. `ledPin` değişkenine LED bağladığımız için (temsili olarak) ve akım vermemiz gerektiğinden `OUTPUT` olarak ayarladık.

butonPin değişkeni ise daha farklı. Butona basılıp basılmadığını kontrol edebilmek için INPUT olarak ayarladık. Tabii ki birde sensorPin değişkenimiz var. Sensörlerin hepsini INPUT olarak ayarlarız. Çünkü sensörler üzerinden veri okuruz ve bu verileri okuyabilmek için sensör pinlerini INPUT olarak ayarlamamız gerekmektedir. Örneğin yağmur sensörünü kullandığımızda INPUT olarak tanımlama yaparız. Böylelikle, dışarıda yağmurun yağıp yağmadığı, kart üzerinden elde ettiğimiz verilere göre anlaşılabilir.

delay Fonksiyonu

delay fonksiyonu, projelerimize tepkime süresi eklememize olanak sağlar. Fonksiyon 1 parametre alır ve milisaniye cinsindendir. Kullanımı delay(milisaniye) şeklinde olmalıdır. delay(1000) şeklinde bir fonksiyon tanımladığımızda Arduino kartımız 1 saniye beklemektedir.

digitalWrite Fonksiyonu

Bu fonksiyon ile birlikte çıkış olarak ayarlanan dijital pini kontrol edebiliriz. Bu fonksiyonu 2 temel özelliği bulunmaktadır. Bunlar, çıkış olarak ayarlanan pine +5V göndermek ve çıkış olarak ayarlanan pine gönderilen voltajı kesmek. Fonksiyonu kullanmak istediğimizde 2 farklı parametre tanımlamamız gerekmektedir. İlk parametre, çıkış olarak tanımlanmış pin numarası ya da pinin atandığı değişken ismi olmalıdır. İkinci parametre, özel olarak belirlenmiş HIGH veya LOW komutlarıdır. HIGH komutu **5V** geçişine izin verirken, LOW komutu **volt** geçişine izin vermemektedir.

Bir LED'i düşünelim. Eğer LED'i yakmak istiyorsak HIGH komutu vermeli, eğer söndürmek istiyorsakta LOW komutu vermeliyiz. digitalWrite fonksiyonu, digitalWrite(pin_no,ozel_parametre) şeklinde kullanılmalıdır. Daha iyi anlayabilmek amacıyla aşağıdaki örneği inceleyelim.

Örnek 1 (Göz Kırpma)

Bu örneğimizde göz kırpma örneği gerçekleştireceğiz. Bunun için bir adet LED diyota, bir adet Arduino kartına ve 2 adet erkek-dişi jumper kablosuna ihtiyacımız bulunmaktadır. Bağlantımız ise oldukça basit. LED diyotun 2 bacağı bulunmaktadır. Bu bacaklardan uzun olanı + hattıdır. Yani, Arduino kartımızdaki dijital pinlerden bir tanesine bağlamamız gerekli (Bu örneğimizde 6. dijital pini kullanacağız). Kısa olan bacak ise - hattıdır. Yani, Arduino kartındaki GND hattına bağlanmalıdır.

analogWrite Fonksiyonu

Arduino kartı ile birlikte PWM işlemleri yapılabilmektedir. Peki, PWM nedir? Türkçe anlamı, Darbe Genişliği Modülasyonudur.

Elektrik sinyalleri tarafından sağlanan ortalama gücü, hızlı bir şekilde parçalara bölerek azaltma işlemine darbe genişliği modülasyonu diyebiliriz. Bilindiği üzere Arduino UNO kartımızdaki gücümüz 5V değerindedir. Eğer 3.75V değerinde bir güç elde etmek istiyorsak, gönderdiğimiz sinyali milisaniyeler eşliğinde açıp kapamamız gerekir. Tabii ki biz bu işlemi manuel bir şekilde yapmıyoruz. Bunun için `analogWrite` fonksiyonundan yararlanırız. Bu fonksiyon, Arduino'daki özel dijital hatlar üzerinden PWM işlemleri yapmamıza olanak sağlar.

Örneğin; motor hızını ayarlamak istediğimizde veya LED'imizin ışık şiddetini ayarlamak istediğimizde darbe genişliği modülasyonundan yararlanabiliriz. Arduino UNO kartında PWM işlemleri gerçekleştirmek istiyorsak, dijital pinlerimizden 3, 5, 6, 9, 10 ve 11. pinlerimizi kullanmamız gereklidir. Diğer pinlerimiz PWM işlemleri için uygun değildir. Tabii ki kullanabilmemiz için de `analogWrite` fonksiyonunu öğrenmeliyiz.

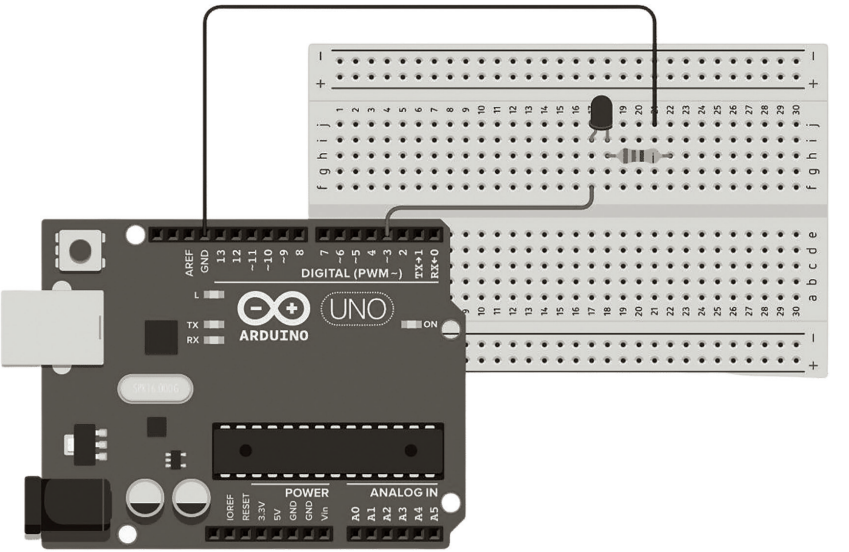
Kullanımı; `analogWrite(pin_numarasi, deger)` şeklindedir. İkinci parametre görev döngüsünü temsil eder. Bu parametreye `[0,255]` aralığında değerler girilmelidir. Kısaca, ikinci parametreye düşük değer girdiğimizde aktarılan güç düşük olurken, yüksek değer girdiğimizde aktarılan güç fazladır. 0 değerini girdiğimizde 0V aktarılır. Yani, herhangi bir volt aktarılmaz. 255 değerinde ise 5V gerilim sağlanır. Tabii ki buradaki örnekleri özellikle verdim. Doğru orantı yapılarak istenilen voltta gerilim sağlanabilir(0-5V arası). Haydi LED'imizin ışık şiddetini ayarlayan bir örnek yapalım ve fonksiyon kullanımını da daha iyi bir şekilde öğrenelim.

Örnek 5 (LED Işık Şiddeti Ayarlama)

Bu örneğimiz ile birlikte Arduino kartımızda PWM işleminden yararlanarak ledimizin ışık şiddetini ayarlayacağız. Bunun için `analogWrite` fonksiyonundan ve özel dijital hatlarımızdan yararlanacağız.

Kullanacağımız malzemeler 1 adet Arduino UNO, 1 adet breadboard tahtası, 1 adet 220 ohm direnç, 1 adet LED diyot ve 2 adet erkek-erkek jumper kablo. LED'imizin uzun bacağını özel dijital hatlarımızdan olan 3. pine bağlayalım. Böylelikle istediğimiz gibi ışık şiddetini ayarlayabiliriz. Önceki örneklerimizden bildiğiniz gibi direnç bağlantısını ve kısa bacağın nereye bağlanması gerektiğinden bahsetmeyeceğim :). Çoktan öğrendiniz.

Aşağıdaki görselimizden de bağlantı şemamızı inceleyebilirsiniz.



Haydi şimdi projemizin kodlarını inceleyelim.

Örnek Kod

```
int ledPin=3; // Led Pini
void setup() {
  pinMode(ledPin,OUTPUT); //  Led pinini çıkış olarak ayarladık.
}
void loop(){
analogWrite(ledPin,0); // Ledimize giden gerilim yok.
delay(500);
analogWrite(ledPin,127); // Ledimize giden gerilim %50
delay(500);
analogWrite(ledPin,255); // Ledimize giden gerilim %100 (Maksimum ışık)
delay(500);
}
```

Kodlarımızı incelediğimizde, oluşturduğumuz örnekte 3 adet `analogWrite` fonksiyonunu kullandık. İlk fonksiyonumuzda LED'imize gönderdiğimiz gerilim 0'dır. Bu sebeple, ledimizde herhangi bir ışık yanmayacak. İkinci fonksiyonumuzda ise LED'imize giden gerilimi %50 olarak ayarladık. Böylelikle ışık şidde-

timiz, maksimum gerilimdeki ışık şiddetinin yarısı kadar olacak. Üçüncü ve sonuncu fonksiyonumuzda hattımıza giden gerilimi 255 olarak ayarladık ki, bu da LED'imize giden gerilimin maksimum değeridir. Böylelikle, LED'imiz maksimum ışık şiddetiyle yanacaktır. Örneğimizi yorumlamak istersek, yarım saniye ledimiz sönük, yarım saniye LED'imiz yarı ışıkta ve yarım saniye LED'imiz maksimum ışıkta yanmaktadır.

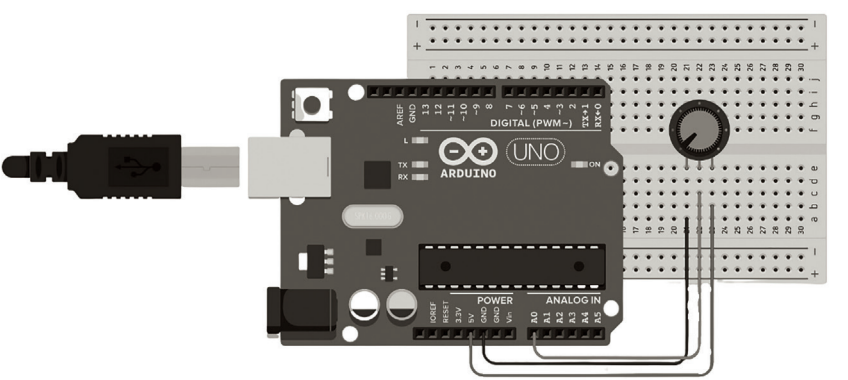
POTANSİYOMETRE

Özellikle PWM ve analog okuma işlemlerini öğrendiğimize göre potansiyometre kullanımı da öğrenmemiz faydamıza olacaktır. Potansiyometreye kısaca ayarlanabilir direnç denmektedir. Farklı bir açıklamayla, gerilimi bölmemize olanak sağlarlar. Kullanımı ise oldukça basittir. Potansiyometremizin üzerinde 3 farklı bacak bulunmaktadır. Sol ve sağdaki bacaklara Arduino kartında 5V ve GND hatları bağlanır.

Yani, soldaki bacağa GND bağlıyorsak, sağdaki bacağa 5V hattını bağlamamız gerekir. Ortadaki bacağa ise analog hatlarımızdan bir tanesi bağlanır. Böylelikle potansiyometremizi döndürdüğümüzde gerilimi ayarlayabilir ve buna bağlı olarak sinyalleri ölçebiliriz. Tabii ki şu konuda da bilgi sahibi olmamızda fayda var. Sol ve sağ bacaklara ters bağlantı da gerçekleştirilebilir. Yani, soldakine 5V ve sağdakine GND bağlanabilir. Haydi şimdi örneğimize geçelim ve potansiyometre kullanımını daha iyi bir şekilde öğrenelim.

Örnek 6 (Potansiyometreden Veri Okuma)

Bu örneğimizle birlikte Arduino kartını kullanarak potansiyometre kullanımı hakkında bilgi sahibi olacağız. Kullanacağımız malzemeler 1 adet Arduino UNO, 1 adet potansiyometre, 1 adet breadboard tahtası ve 3 adet erkek-erkek jumper kablo. Haydi bağlantı şemamızı inceleyelim.



mızı belirttik. DC motorumuzun hızını ayarlamak ve yönünü ayarlamak için, setup fonksiyonumuzda değişkenlerimizi çıkış olarak ayarladık ve motorumuzun ne yönde döneceğini belirledik. Örneğimizde in1 hattımız motorumuzu sağa, in2 hattımız ise motorumuzu sola döndürmektedir.

Burada ki sağa ve sola dönme yönlerini önceden test ettiğim için biliyorum. Siz de kodlarınızı oluşturduktan sonra bağlantınıza göre sağ ve sol yönlerinizi öğrenebilirsiniz. Biz de motorumuzun sağa dönmesini istediğimiz için in1 hattına HIGH, in2 hattına LOW komutunu verdik. Daha sonra hızı ayarlamak için hizdegeri isminde bir fonksiyon oluşturduk. Hız ayarlarken 0 değeri motorumuzu döndürmüyorken, 255 değeri maksimum hızda döndürmemizi sağlamaktadır. Biz ise örneğimizde hızımızı 100 olarak ayarladık.

Ayrıca, sayaçlı bir döngü oluşturarak 0'dan 100'e kadar hızlanırken, 100'den sonra aynı hızla sabit çalışmasını sağladık. Burada 100 değerine takılmayalım. Bu örneğimizde fonksiyona atadığımız değer 100 olduğu için 0'dan 100'e hızlanacaktır. Eğer siz farklı bir değer tanımlarsanız, o değere göre 0'dan başlar ve tanımladığınız değere kadar hızlanır. Ek olarak hız artarken tepkime süresini de 20 milisaniye olarak ayarladık. Bunun sebebi, normal arabalarda da bir anda hızlanma gerçekleşmez. Yavaş yavaş hızlanırlar. Biz de bu tepkime süresiyle bu izlenimi verdik.

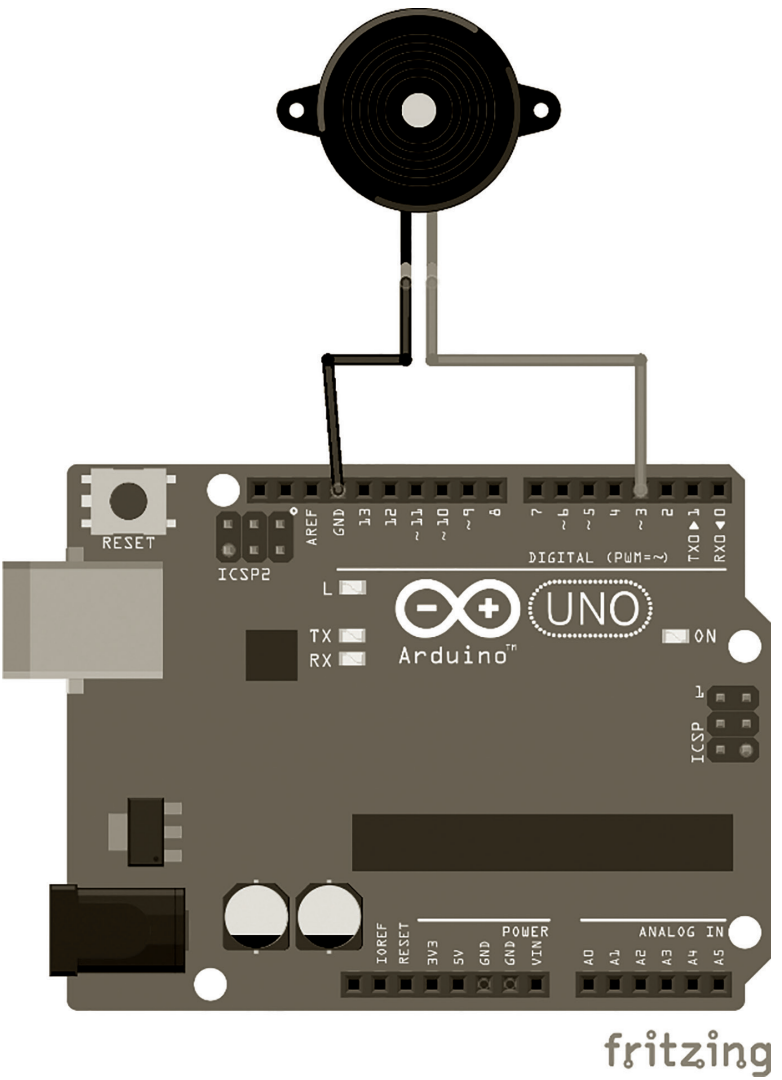
Buzzer

Bu başlığımızda buzzer hakkında bilgi sahibi olacağız. Buzzer, genel olarak sesli uyarı vermek amacıyla kullanılan mini hoparlör çeşididir. Kullanım alanı ise oldukça yaygın. Örneğin alarm sistemlerinde, bilgisayar anakartlarında ve makinelerde sesli uyarı vermek amacıyla kullanılabilmektedir. Bizler de Arduino ile birlikte kullanırken uyarı vermek amacıyla ya da şarkı notalarını çalmak amacıyla kullanmaktayız. Kullandığımız buzzerların 2 bacağı bulunmaktadır. Bu bacaklar gözle ayırt edilebilmekle birlikte, biri kısa diğeri uzundur. Uzun olan bacağımız sinyal pinimiz iken kısa olanı GND hattımızdır. Sinyal pinini Arduino'daki PWM hatlarından yararlanarak kullanabilir ve istediğimiz tizlikte sesle çıkartabiliriz. Haydi şimdi örneğimize geçelim ve iyi bir şekilde buzzer kullanımını öğrenelim.

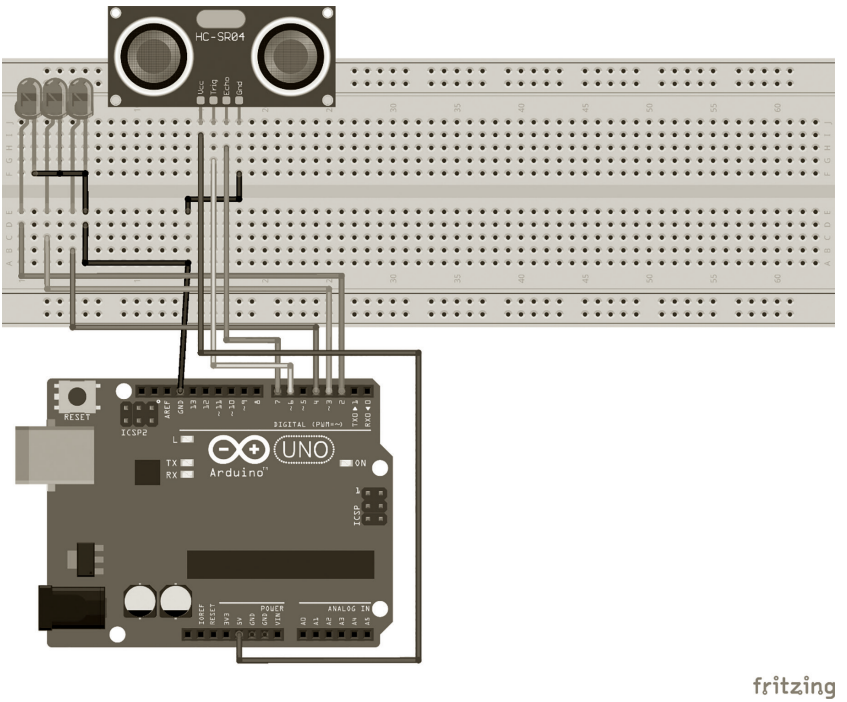
Örnek 16 (Arduino ile Buzzer Kullanımı)

Bu örneğimizle birlikte Arduino kartlarımızda buzzer kullanımı hakkında iyice bilgi sahibi olacağız. Buzzer hakkında bilgi sahibi olurken 2 bacağımız olduğunu belirtmiştik. Bu bacaklardan uzun olanını Arduino kartlarımızda ki PWM hatlarından birine, kısa olanını ise Arduino kartımızdaki GND hattına bağlamamız

gereklidir. Kullanacağımız malzemeler 1 adet Arduino UNO, 1 adet buzzer ve 2 adet erkek-dişi jumper kablo. Yapacağımız örnekte buzzer kullanarak 3 farklı tiz sesi yarım saniye arayla çaldıracağız. Bağlantı şemamızı görüntümüz üzerinden inceleyebilirsiniz. Gayet kullanımı basit olduğu için şemamızı yorumlamamıza gerek yoktur. Yapılması gereken, uzun bacağımızı PWM hatlarından birine(biz 3 numaralı PWM hattını kullanacağız), kısa bacağımızı ise GND hattımıza bağlayacağız.



bilgi vereceğiz. Bu projemizi ev koşullarında yaptığımız için ve ölçülen mesafeler kısa olduğu için, kontrollerimizi kısa mesafeye göre ayarlayacağız. Örneğin, ölçülen mesafe 20 cm ve 15 cm aralığındaysa 1 adet LED'i yakacağız. 15 cm ve 7 cm aralığındaysa 2 adet LED'i ve 7 cm ve aşağısındaysa da 3 adet LED'imizi yakacağız. Böylelikle, sürücü geri geri giderken artık durması gerektiğini yaktığımız ışıklara göre anlayacak. Projemiz malzemeleri ve mantığı hakkında bilgi sahibi olduğumuza göre haydi bağlantı şemamızı inceleyelim ve yorumlayalım.



Bağlantı şemamızı incelediğimizde LED diyotlarımızın GND hatlarını ve mesafe sensörümüzün GND hattını ortak bağladık. Bu bize herhangi bir sıkıntı yaşatmaz. Park sensörümüzde uyarı verirken sırasıyla soldan sağa doğru LED'lerimi yakacağız. Bu sebeple biz de projemizde bu LED'lerin dijital bağlantısını sırasıyla 2, 3 ve 4 numaralı dijital pin hatlarına yaptık.

Mesafe sensörümüz 5V gerilime ihtiyaç duyduğu için **vcc** hattını 5 hattın bağladık. **Trig** ve **Echo** pinlerinden de dijital sinyal gönderip, alacağımız için dijital pin hatlarımızdan olan 7 numaralı pini Echo pinine, 6 numaralı pini ise Trig pinine bağladık. Tabii ki bu hatlar bildiğiniz gibi zorunlu değil. Sadece bu projede bu

Yani bizler 16x2 LCD ekranımızla toplamda 32 farklı karakter görüntüleyebiliriz. Eğer kaydırma işlemi yaparsak 32'den fazla karakter de görüntüleme yapabilmekteyiz. 16x2 LCD ekranımızın üzerinde 16 adet pin bulunmaktadır. Bu pinler sayesinde Arduino kartımız ile LCD ekranımız arasındaki bağlantıyı sağlarız. Fakat I2C protokolünü kullanırsak 16 pin yerine yalnızca 4 pin hattını kullanırız.

Şimdi birlikte 16 tane pin hatlarının Arduino üzerindeki hangi pinlere bağlanacağını öğrenelim. LCD ekranımızın üzerinden soldan sağa doğru sırasıyla, en soldaki hattımız VSS (GND) yani toprak hattıdır. VSS hattını Arduino kartımızdaki GND hattına bağlarız. İkinci sıradaki hattımız ise VDD, yani besleme gerilimi hattıdır. 5V ile çalıştığından kartımızdaki 5V hattına bağlarız. Üçüncü sıradaki hattımız V0 (Vee) hattıdır. İçerisinde potansiyometre bulunur ve kontrast ayarı yapmamızı sağlar. Arduino kartımızdaki GND hattına bağlanır. Dördüncü sıradaki hattımız RS hattıdır. RS hattı Komut Register'i ile Veri Register'i arasında geçiş yapmamızı sağlar. Beşinci hattımız RW hattıdır. RW hattımız LCD ekranımızda okuma ve yazma işlemlerini gerçekleştirmek için kullanılmaktadır.

RW hattını, Arduino kartımızdaki GND hattına bağlarız. Altıncı sıradaki hattımız ise EN hattıdır. Ekrana yazı yazılmasını aktifleştiren pin hattıdır. Eğer bu pin kullanılmazsa ekrana da yazı yazdırılamaz. EN pin hattından sonra LCD ekranımızda sırasıyla D0'dan D7'ye kadar 8 tane Data pinlerimiz bulunmaktadır. Bu pinler 8 bitlik pinlerdir. Eğer 8 bitlik yazı yazmak istersek buradaki tüm hatları kullanmamız gerekir.

Son olarak LCD ekranımızda, soldan sağa doğru sıralandığında 15. ve 16. Pinlerimiz, yani A ve K pinlerimiz bulunmaktadır. A pin hattı (led+) LCD'nin arka plan aydınlatmasının açılmasını sağlar. K pin hattı (led-) LCD'nin arka plan aydınlatmasının kapanmasını sağlar. Eğer A hattını HIGH olarak ayarlarsak, aydınlatma yanar. Eğer K hattını HIGH olarak ayarlarsak, aydınlatma söner.

I2C Nedir?

Yukarıda bahsettiğimiz I2C modülünü tanıyalım ve LCD ile kullanımını öğrenelim. I2C, iki telli bir seri haberleşme protokolüdür. Veri için SDA teli ve saat için SCL teli kullanılmaktadır. I2C, hat sayısı fazla olan sistemlere bağlanarak bu hat sayılarını 4'e kadar düşürmektedir. Böylece kısa mesafeli veri aktarımını sağlamamıza yardımcı olmaktadır. I2C protokolü üzerinde bulunan 4 hattımız yukarıdan aşağıya doğru; GND, VCC, SDA ve SCL şeklindedir. Bizler bu hatları Arduino kartımıza bağladığımızda, LCD ekranımızla Arduino kartımız arasında da bağlantı sağlamış oluruz. Her cihazın önceden belirlenmiş bir cihaz adresi

bulunmaktadır. I2C ile haberleşme sağlanırken, dizilerdeki adresler sırasıyla veri aktarır. Şimdi I2C modülünü, Arduino kartımızdaki hangi pinlere bağlamamız gerektiğini tablo üzerinden inceleyelim. Tablomuzda Arduino türlerine göre, I2C hatlarını hangi pinlere bağlamamız gerektiği gösterilmiştir.

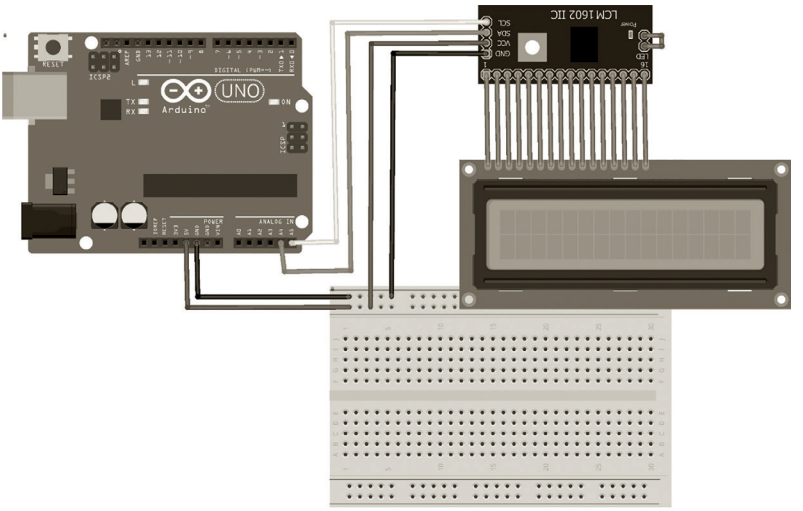
Arduino Türü	SDA Pini	SCL Pini
Arduino UNO	A4	A5
Arduino MEGA	20	21
Arduino Leonardo	2	3
Arduino DUE	20	21
Arduino Nano	A4	A5

Tablomuzuda incelediğimize göre şimdi örneklerimize geçelim ve Arduino ile LCD ekran kontrolü nasıl yapılır bunu öğrenelim.

Örnek 20 (LCD Ekranda Çıktı Almak)

Bu örneğimizde I2C destekli LCD ekranı kullanarak, lcd ekranda nasıl çıktı alabileceğimizi öğreneceğiz. Örneğin gelelim ve merhaba yazısını LCD ekranda çıktı alalım. Bunun için kullanacağımız malzemeler 1 adet I2C destekli LCD ekran, 1 adet Arduino UNO kartı, 1 adet breadboard tahtası ve 4 adet erkek-erkek jumper kablo.

I2C destekli LCD ekranda çıktı almak istiyorsak, yeni bir kütüphaneden yararlanmalıyız. Kütüphanemizin ismi, LiquidCrystal_I2C.h kütüphanesidir. Tabii ki lcd ekran üzerinden işlemler yapmak için, öncelikle özel belirlenmiş sınıfımızı çağırıp daha sonra ise nesnemizi tanımlamalıyız. Sınıfımızın ismi LiquidCrystal sınıfıdır. Bu sınıfla birlikte LCD ekranda karakter kontrolü, konumu, çıktıları ve özel karakter oluşturma gibi işlemler kolaylıkla yapılabilmektedir. Sınıfımız nesnesi örneğin ekranim olsun. `ekranim` nesnesine 3 farklı parametre tanımlamamız gereklidir. İlki LCD ekranımızın adres bilgisi. Genel olarak bu değer 0x27 ya da 0x3F adresiyle tanımlanmaktadır. Sizin de LCD ekranınızın adresi bu iki değerden biridir. İkinci parametre LCD ekranınızın sütun sayısı. Aslında buna yan yana bulunabilecek karakter sayısıda diyebiliriz. Benim kullandığım LCD ekran 16 sütundan oluşmaktadır. Son gireceğimiz parametre ise satır sayısıdır. Ekranınıza al alta yazdığınız her bir bölme satırı ifade etmektedir. Benim ekranımın satır sayısı da 2'dir. Haydi şimdi bağlantı şemamıza geçelim ve Arduino ile LCD ekranın(i2c destekli) bağlantısını gerçekleştirelim.



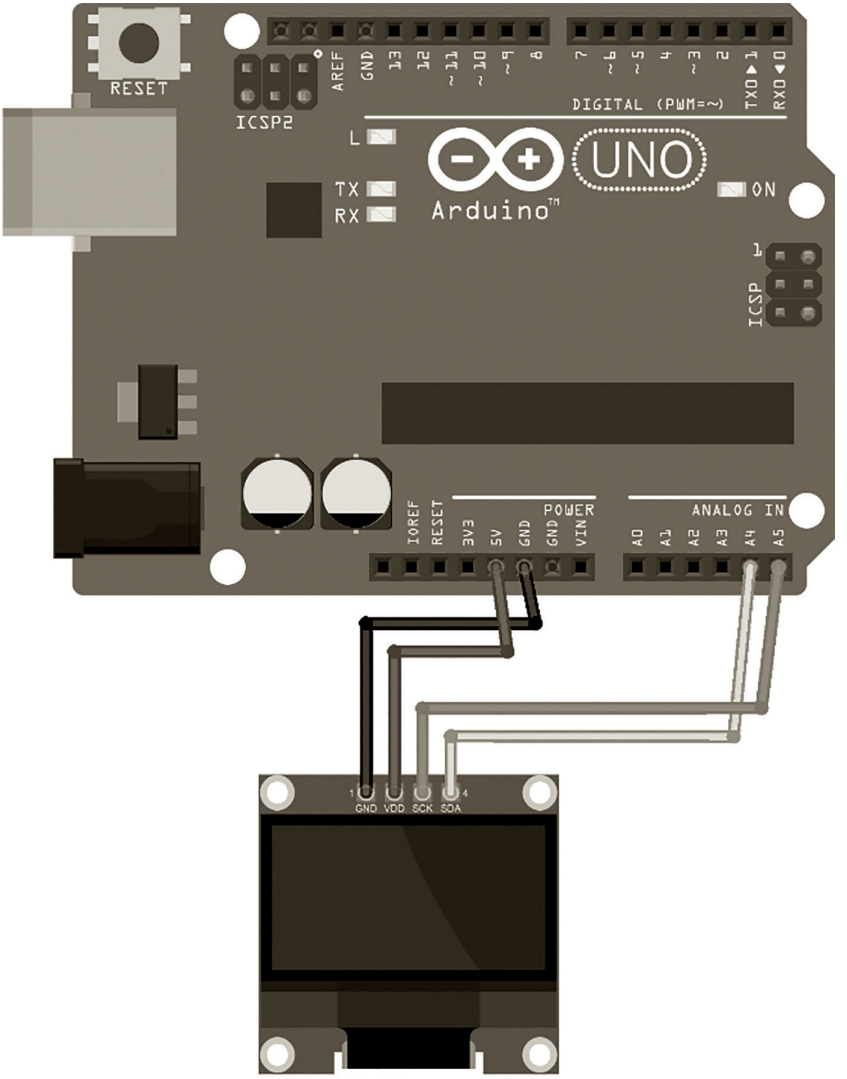
Bağlantımızı incelediğimizde aslında OLED ekranımızın bağlantısıyla tıpatıp aynı. Burada sadece breadboard tahtasından da yararlandık. LCD ekranımızın gerilimi 5V ile sağlanmaktadır. Bu sebeple, VCC hattını Arduino'daki 5V hattına bağlamalıyız. GND hattını Arduino'daki GND'ye, SDA hattını Arduino'daki A4 hattına ve SCL hattını ise A5 hattına bağlamamız gereklidir. Böylelikle Arduino ile LCD ekran kontrolü sağlayabiliriz. Haydi şimdi kodlarımızı inceleyelim ve yorumlayalım.

Örnek Kod

```
// Kütüphanemizi dahil ettik
#include <LiquidCrystal_I2C.h>
// LCD ekranımızı tanımladık
LiquidCrystal_I2C ekranim(0x27,16,2);
void setup() {
// Ekranımızı başlattık
  ekranim.begin();
// Ekranımızda çıktı aldık
  ekranim.print("MERHABA");
  delay(2000);
// Ekranımızı temizledik
  ekranim.clear();

}
void loop() {}
```

Bu kadar yazılı açıklamadan sonra artık devre şemanızı oluşturmanız ve ekranınız için gerekli kodları yazmanız hakkınızdır. Haydi şimdi ekranımızın bağlantı şeması hakkında bilgi sahibi olalım ve yorumlayalım.



fritzing

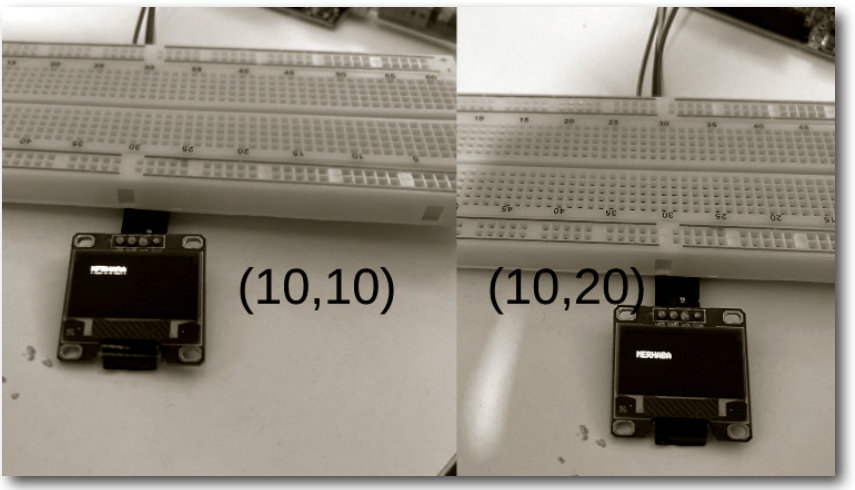
Bağlantı şemamız gördüğünüz gibi oldukça basit. OLED ekranımıza güç gelebilmesi için, ekrandaki VCC hattını 5V hattına, GND hattını ise GND hattına bağlamamız gereklidir. Ekranımız üzerinden işlemler yapabilmek için SCL hattını A5 hattına ve SDA hattını ise A4 hattına bağlamamız gereklidir. Bağlantı şemamızı incelediğimize göre kodlarımızı oluşturalım ve yorumlayalım.

Örnek Kod

```
// Kütüphanemizi tanıttık
#include <OLED_I2C.h>
// Arduino Uno
// ----------------------
// SDA pin   -> A4
// SCL pin   -> A5
// Ekranımızı tanıttık
OLED ekranim(A4,A5);
// Fontumuzu tanıttık
extern uint8_t SmallFont[];
void setup()
{
// Ekranımızın sürücüsünü girerek başlattık
ekranim.begin(SSD1306_128X64);
// Ekranımızı temizledik
ekranim.clrScr();
//Fontumuzu ekrana tanıttık
ekranim.setFont(SmallFont);
// x-> 10 y->20 MERHABA çıktısı aldık
ekranim.print("MERHABA",10,20);
// Ekranımızı güncelledik
ekranim.update();
// 4 saniye MERHABA yazılı kalır
delay(4000);
// Ekranımızı tekrar temizledik
ekranim.clrScr();
// Yazımız silindi
ekranim.update();
delay(500);
}
void loop()
{
}
```

Kodlarımızı incelediğimizde ilk olarak kütüphanemizi tanımladık ve nesnemizi oluşturup, SDA ve SCL hatlarımızı tanımladık. Tekrardan uyaralım. Her Arduino'nun kendine ait I2C hatları bulunmaktadır. Bu sebeple buna dikkat etmelisiniz.

Ekranımızı tanımladıktan sonra ekranımızda yazı yazdıracağımız için font değişkenimizi tanımladık. Bilinmelidir ki bu `font` değişken ismi `DefaultFonts.c` dosyasında bulunmaktadır ve ayarlanmıştır. Bu sebeple `font` ismi özel bir değişkendir. `setup` fonksiyonumuzu incelediğimizde ilk olarak ekranımızı başlatıp, kullandığımız ekranın sürücüsünü `begin` fonksiyonumuz içerisinde parametre olarak tanımladık. Ekranımızda önceki işlemlerden herhangi bir çıktı vb. işlemler yapıldıysa, bunu temizlemek amacıyla ekranımızı temizledik. Daha sonra ise ekranımızda kullanacağımız fontu, ekranımıza tanımladık. Font tanımlama işleminden sonra yazdırmak istediğimiz yazılar için `print` fonksiyonumuzu çağırdık ve başlangıç konumunu (10,20) olarak ayarladık. Böylelikle "`MERHABA`" çıktısı soldan sağa doğru 10 piksel öteden ve yukarıdan aşağıya doğru 20 piksel öteden bizlere görüntülenecektir. Tabii ki çıktı işleminden hemen sonra da ekranımızı güncellememiz gerekmektedir. "`MERHABA`" çıktısı 4 saniye ekranda kalacak ve sonra ekranımızı temizle komutu verdiğimiz için silinecektir. Yaptığımız örneğimizin görüntüsünü görsel üzerinden inceleyebilirsiniz. 2 farklı konum ayarlanmıştır. (10,10) ve (10,20) şeklinde.



Örnek 18 (OLED Ekranda Şekil Çizdirmek)

Bu başlığımız altında hazır fonksiyonlarımızdan yararlanarak OLED ekranımızda şekil çizdirmeyi öğreneceğiz. Örneğimizle birlikte OLED ekranımızda çizgi ve dikdörtgen çizdirebilmeyi öğreneceğiz.

Bildiğiniz gibi önceki örneğimizde oled ekranımızın arduino ile bağlantısını gerçekleştirdik. Bu örneğimizde de aynı devre şemasından yararlanacağımız için tekrardan şema incelememize gerek yoktur. OLED ekran kütüphanemiz olan **OLED_I2C.h** kütüphanesini kullanarak kolaylık çizgi ve dikdörtgen çizdirebiliriz.

Eğer dikdörtgen çizdirmek istiyorsak, `drawRect()` metotundan yararlanmamız gerekli. Metotumuz 4 farklı parametre almaktadır. Bunlar sırasıyla başlangıç x, başlangıç y, bitiş x ve bitiş y parametreleridir. Bu parametreler, tam sayı cinsindendir. Bilindiği gibi ekranımızda piksellerden oluştuğu için yarım piksel kavramı bulunmamaktadır. `drawLine()` metotu ise çizgi çizdirmemize olanak sağlayan fonksiyonumuzdur.

Aynı şekilde `drawLine` metotumuzda 4 farklı parametreye ihtiyaç duymaktadır. Parametrelerimiz sırasıyla başlangıç x, başlangıç y, bitiş x ve bitiş y parametreleridir. Aynı şekilde bu fonksiyonumuzun parametreleri de tam sayı cinsindendir. Haydi şimdi kodlarımızı oluşturalım ve yorumlayalım.

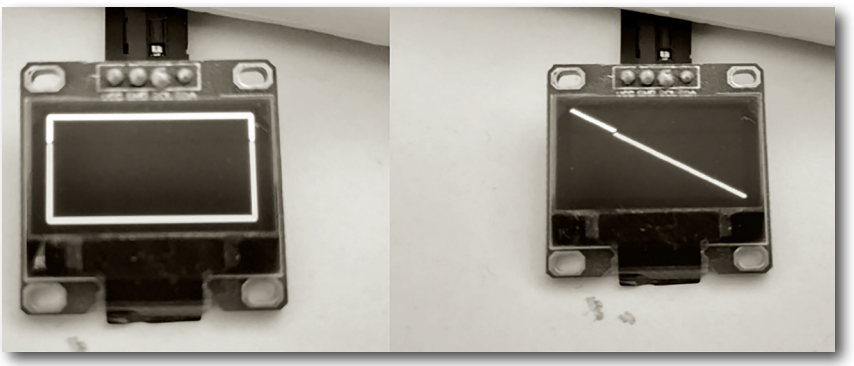
Örnek Kod

```
// Oled ekran kütüphanemizi tanımladık
#include <OLED_I2C.h>
// Arduino Uno
// ----------------------
// SDA pin   -> A4
// SCL pin   -> A5
// Ekranımızı tanıttık
OLED ekranim(A4,A5);
void setup() {
// Ekranımızı başlattık
ekranim.begin(SSD1306_128X64);
// Ekranımızı temizledik
ekranim.clrScr();
//Ekranımızı güncelledik
ekranim.update();
}
void loop() {
```

```
// Çizgi ve dikdörtgen karışmaması için temizledik
ekranim.clrScr();
//Dikdörtgen çizdirdik
ekranim.drawRect(0,0,127,63);
ekranim.update();
delay(2000);
//2 saniye sonra ekranımızı temizledik
ekranim.clrScr();
// Çizgi çizdirdik
ekranim.drawLine(0,0,127,63);
ekranim.update();
delay(2000);
}
```

Kodlarımızı incelediğimizde öncelikle kütüphanemizi dahil ettik ve ekranımızı tanımladık. Daha sonra ise ekranımızı başlatarak, ekran modelimizi belirttik. Tabii ki önceki işlemleri görüntülememek için de ekranımızı temizleyip, güncelledik. Döngü fonksiyonumuz içerisine girdiğimizde, öncelikle ekranımızı temizledik. Bunun sebebi aşağıda çizgi çizdirdiğimiz için, çizgi ve dikdörtgenin birbiriyle karışmasını istememizdendir.

Görüldüğü gibi `drawRect` fonksiyonumuzu çağırdık ve konumunu 0, 0, 127 ve 63 olarak belirttik. Bu şekilde bir işlem yaptığımızda ekranımızın köşelerinde bir dikdörtgen çizdirmiş olacağız ve bu dikdörtgen 2 saniye ekranda gösterilecek. 2 saniye gösterildikten sonra ekranımızı tekrardan temizliyoruz. Bunun sebebi ise dikdörtgen yerin artık çizgimizi göstereceğiz. Görüldüğü gibi drawLine fonksiyonunda belirttiğimiz parametreler sırasıyla 0, 0, 127 ve 63'tür. Bu şekilde tanımlama yaptığımızda sol üst köşeden sağ alt köşeye bir köşegen çizdirmiş oluruz. Örneğimizin çıktılarını görselimizden inceleyebiliriz.



Yılmaz ALACA

instagram.com/yilmazalacaa

twitter.com/alacayilmaz01

linkedin.com/in/yilmazalaca

udemy.com/user/yilmaz-alaca

youtube.com/@yilmazalaca

yilmazalaca01@gmail.com

Emine YALÇIN

https://bit.ly/3vqHVAx

https://bit.ly/3PQVhzd

emineyalcin015@gmail.com