Python ile Gömülü Sistemlerde

ARDUINO

için Arayüz Geliştirme

Yılmaz ALACA

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 22 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. İlk kitabı, Şubat 2023 yılında, Üşengeçler için Arduino ismiyle yayınlanmıştır.

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

Teşekkür

İkinci kitabımı oluşturmamda bana desteklerini esirgemeyen kıymetli büyüklerim YTY Metal Yönetim Kurulu Başkanı Yasin Çetin ve Adana Büyükşehir Belediye Kültür ve Sosyal Daire Başkanı Mahmut Göğebakan'a bir teşekkürü borç bilirim.

6 Şubat 2023 tarihinde saat 04:17'de kaybettiğimiz tüm canlara, yaralılara ve etkilenen herkese ithafen...

Yılmaz ALACA

instagram.com/yilmazalacaa

twitter.com/alacayilmaz01

linkedin.com/in/yilmazalaca

udemy.com/user/yilmaz-alaca

youtube.com/@yilmazalaca

yilmazalaca01@gmail.com

Değerli okurlarım kitabımıza başlamadan önce bir ricam olacak.

Kitapta hata gördüğünüzde hemen kızmayın, sizlerin de desteğiyle hatalarımızı ve eksikliklerimizi tarafıma bildirmeniz durumunda bir sonraki baskıda düzeltilecektir. :)

İçindekiler

1

Giriş

Bu Bölümde

Bu bölümde, sizler ile birlikte Python yazılım dilinin ne olduğunu, Python ile Arduino'yu kontrol edebilmemiz için gerekli modülün kurulumunu, arayüz tasarlaması yapabilmemiz için gerekli olan modülün kurulumunu, Python ve Arduino kodlarını yazabilmemiz için gerekli ortamların kurulumunu ve Arduino IDE'sine nasıl kütüphane ekleyebileceğimiz hakkında bilgi sahibi olacağız. Haydi başlayalım... :)

PYTHON NEDİR?

Python yazılım dili yüksek seviyeli programlama dilidir. Nesne yönelimli programlama dil yapısına sahip olmakla birlikte yazım biçimi kolay bir dildir. İlk sürümü 1990 yılında yayınlanmıştır. Günümüzde web uygulamaları, oyun geliştirme, arayüz oluşturma, makine öğrenmesi, görüntü işleme ve bilimsel çalışmalarda sıkça kullanılmaktadır. Bugün birçok yazılım geliştiricisi tarafından geliştirilmekle birlikte zengin kütüphanelere sahiptir. Biz de Arduino kartımızı kontrol etmek amacıyla arayüz oluştururken ve seri haberleşme yaparken Python yazılım dilini ve Python'daki kütüphanelerden yararlanacağız.

PYSERİAL MODÜLÜ

Python ile port kontrolü yapabilmek için oluşturulmuş bir modüldür. Bizler Python ile Arduino kartımızdan veri alırken veya veri iletirken bu modülden yararlanacağız. PySerial modülünü bilgisayarınıza kurabilmeniz için, kullandığınız IDE üzerinden terminal (komut satırı) açmalı ve aşağıdaki kodu çalıştırmalısınız;

```
pip install pyserial
```

NOT Herhangi bir IDE'yi kullanmanız durumunda proje klasörünüzün ya da oluşturduğunuz Python dosyasınının dizininde, Türkçe karakter bulunmamalıdır.

Doğru Kullanım

```
C:\trress\projects
```

Hatalı Kullanım

```
C:\projects\iş_amaçlı_klasör
```

NOT Python dosyasında Türkçe karakter bulunmamalıdır(ş, ç, ğ, ü, ı, ö).

Doğru Kullanım

```
proje1.py
```

Hatalı Kullanım

```
iş_proje.py
```

Pygame Modülü

Arduino'muzu kontrol etmek için veya Arduino'daki verileri güzel arayüzler eşliğinde görüntüleyebilmek amacıyla **Pygame** modülünü kullanacağız. **Pygame modülü,** Python yazılım dili için oluşturulmuş kütüphanedir. Genel olarak oyun amaçlı ve arayüz oluşturma amacıyla kullanılmaktadır. Biz de projelerimizin arayüz bölümünü Pygame modülüyle oluşturacağız. Pygame modülünü bilgisayarınıza kurabilmeniz için, kullandığınız IDE üzerinden terminal(komut satırı) açmalı ve aşağıdaki kodu çalıştırmalısınız;

```
pip install pygame
```

Kullanacağımız Kod Editörü: Visual Studio Code

Arayüzlerimizi Python ile Visual Studio Code editöründe oluşturacağız. VSCode kullanımı kolay, sürekli geliştirilen ve Microsoft tarafından yayınlanmış kod editörüdür. Sizler farklı kod editörü ya da IDE kullanabilirsiniz. Ancak, kitap boyunca benimle birlikte aynı kod editörünü kullanmanızı tavsiye ederim. Haydi şimdi kurulumunu gerçekleştirelim ve aynı zamanda kod editörüne Python kurulumunu gerçekleştirelim.

VSCode Kurulumu

Şimdi ise kullanacağımız editörün kurulumunu gerçekleştirelim.

1. Adım:

https://code.visualstudio.com/ sayfası üzerinden işletim sistemimize göre kurulum dosyasını indirelim.

2. Adım:

Kurulum dosyasını çalıştıralım ve yönergeleri takip ederek kurulumu tamamlayalım.

3. Adım:

Editörümüzü açalım ve File (Dosya) seçeneğine basalım.

4. Adım:

Open Folder (Klasör Aç) seçeneğine tıklayalım ve proje klasörümüzü oluşturalım. **New Folder** (Yeni Klasör) seçeneğinden klasör oluşturulabilir ve klasöre isim verilebiliriz. Bir sonraki görseli inceleyebilirsiniz.

2

Python ve Arduino Kartının Haberleşmesi

Bu Bölümde

Bu bölümde, sizler ile birlikte Python yazılım dilinin ne olduğunu, Python ile Arduino'yu kontrol edebilmemiz için gerekli modülün kurulumunu, arayüz tasarlaması yapabilmemiz için gerekli olan modülün kurulumunu, Python ve Arduino kodlarını yazabilmemiz için gerekli ortamların kurulumunu ve Arduino IDE'sine nasıl kütüphane ekleyebileceğimiz hakkında bilgi sahibi olacağız. Haydi başlayalım... :)

Bu başlığımız altında Python yazılım dilini kullanarak Arduino kartını nasıl kontrol edebileceğimizi ve nasıl veri okuyabileceğimizi öğreneceğiz. Python yazılım dilini kullanarak elektronik karta veri göndermek ya da almak istediğimizde kullanmamız gereken bir modül bulunmaktadır. Bu modülün ismi `pySerial` modülüdür. **pySerial** modülü, Windows, Mac ve Linux işletim sistemlerindeki USB portlarına erişmemizi ve kontrol etmemizi sağlar. Her ne kadar pySerial olarak isimlendirilse de, kodlarımızı oluştururken `serial` şeklinde modülü kullanmamız gerekmektedir. Aşağıdaki gibi:

```
import serial
```

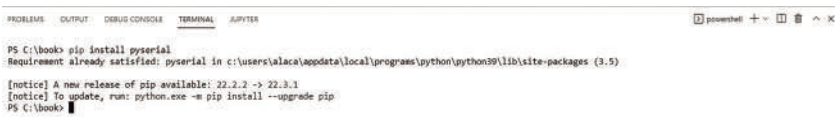
Tabii ki **as** yapısını kullanarak daha kısa bir şekilde kullanabilmemiz mümkündür. Aşağıdaki gibi:

```
import serial as s
```

`pySerial` modülü varsayılan bir şekilde Python modülü olarak hazır gelmemektedir. Bunun için bilgisayarımıza bu modülü kurmamız gereklidir. Aşağıdaki komutu kullandığımız IDE'nin Terminal bölümünde kullanmamız yeterli olacaktır.

Komut:

```
pip install pyserial
```



Modül kurulumunu gerçekleştirdiğimize göre yakından tanıma vakti.

Modül İçerisindeki Sınıf ve Metodların Kullanımı

Python tarafında serial modülünü dahil ettiğimizde, kullanmamız gereken bir sınıf bulunmaktadır. Bu sınıfın ismi ise `Serial()` sınıfıdır. 2 farklı parametre alır. İlk parametre, Arduino kartını bağladığımız USB portunun ismi. Örneğin, Arduino kartımız **com5** USB portuna bağlı olsun. Sınıfta tanımlamamız gereken port ismi parametresi **com5** şeklinde string tipinde olmalıdır. İkinci parametre ise seri haberleşme hızı parametresi. Seri haberleşme hızını genel olarak 9600 ya da 115200 ayarlarız. Burada dikkat etmemiz gereken en önemli nokta, Python'da tanımladığımız seri haberleşme hızı ile Arduino'daki seri haberleşme hızı

aynı olmalıdır. Aksi halde kart ve yazılım dili haberleşemez ve sorunla karşılaşabiliriz. `Serial()` sınıfının örnek kullanımı aşağıdaki gibidir (Port bilgisi: com5, Seri haberleşme hızı: 9600 olarak belirlenmiştir).

```
import serial
port=serial.Serial("com5",9600)
```

Kodumuzu incelediğimizde ilk olarak serial modülümüzü dahil ettik ve Serial sınıfını kullanabilmek için `port` isminde nesne oluşturup, `Serial` sınıfını çağırdık. Daha sonra ise port bilgisini ve seri haberleşme hızını tanımladık. Burada kafanızı karıştırabilecek durum, okunuşları aynı olan 2 kelimeyi yan yana kullanmamız olabilir. Buna hiç gerek yok :).

İlk yazdığımız `serial` bizim modülümüzdür. İkinci yazdığımız `Serial` ise bizim sınıfımızdır. Sınıfımız hakkında bilgi sahibi olduğumuza göre şimdi sınıfımız içerisinde bulunan `write()` metodunu öğrenelim. Bu metot, Python'dan Arduino kartına sinyal göndermemize olanak sağlayan metodumuzdur. İleride projelerimizi de gerçekleştirirken çok sık kullanacağız. Metodumuzu Arduino ile haberleştirebilmek için bir adet parametre tanımlamalıyız. Kullanımı, `nesne_adi.write(b'D')` şeklindedir. Buradaki nesne adı, sınıfımızı atadığımız değişkendir. `write()` metodu içerisinde tanımladığımız `b` harfi, Arduino kartına verileri byte tipinde gönderebilmemizi sağlar.

Ayrıca, Arduino'ya veri gönderirken her zaman byte tipinde göndermemiz gereklidir (ileri de modül güncelleştirmelerine bağlı olarak değişiklik olabilir). Bunun sebebi, gönderdiğimiz verileri byte(bayt) olarak gönderdiğimizde Arduino kartı gönderilen veriyi anlar. Python kodumuzda **'D'** verisini bayt olarak Arduino kartımıza gönderirsek, Arduino kartı da aynı şekilde bu veriyi **'D'** olarak anlar. `Serial` sınıfımızı kullanırken iki farklı parametre tanımladık. Bu iki farklı parametrenin değişken adlarını çağırarakta parametrelerini tanımlayabiliriz. Daha iyi anlamak amacıyla aşağıdaki kodumuzu inceleyelim ve açıklayalım.

```
import serial
arduino=serial.Serial()
arduino.port="com5 " # Port tanımladık
arduino.baudrate=9600 # Seri haberleşme hızını tanımladık
```

Kodumuzu incelediğimizde `arduino` adında bir nesne oluşturduk ve `Serial` sınıfımızı tanımladık. `Serial` sınıfımızdaki ilk parametrenin ismi `port` olarak

tanımlanmıştır. Bu sebeple, bizde nesnemizi çağırıp, sınıfımızdaki `port` değişkenimize ulaştık ve değerini "`com5`" olarak ayarladık. `Serial` sınıfımızdaki ikinci parametrenin ismi ise `baudrate` olarak tanımlanmıştır (seri haberleşme hızı). Bu sebeple, hızımızı ayarlamak amacıyla, nesnemizi çağırdık ve sınıfımızdaki `baudrate` parametresinin değerini `9600` olarak ayarladık. Kanıtlamamızı ister misiniz? Haydi kanıtlayalım.

Bunun için yapmamız gereken çok basit. `print` fonksiyonumuzla nesnemizi çıktı alalım.

```
print(arduino)
```

Çıktı aldığımızda bizler aşağıdaki gibi bir çıktıyla karşılaşacağız.

```
Serial<id=0x2038b432d00, open=False>(port='com5 ', baudrate=9600, bytesize=8,
parity='N', stopbits=1, timeout=None, xonxoff=False, rtscts=False, dsrdtr=False)
```

Tabii ki burada dikkat etmemiz gereken nokta, şu an için kodumuzun sağlıklı bir şekilde çalışmasıdır. İleride projelerimizi oluştururken hem modülümüzü daha iyi bir şekilde öğreneceğiz hem de nesne içerisindeki bilgileri kavrayacağız. Metodumuz ve sınıfımız hakkında bilgi sahibi olduğumuza göre ilk projemizi geliştirelim ve projeler üzerinden sınıfımızı daha iyi bir şekilde öğrenelim.

Proje 1: Python ile Arduino'ya Bağlı LED Kontrolü

Bu projemizde Python yazılım dilini kullanarak Arduino'ya bağlı olan LED diyotun kontrolünü sağlayacağız. Bu, bizim yapacağımız ilk ve basit projemiz olacak. Tabii ki, ilerleyen projelerimizde üst seviye projeler geliştirebilmek için temeli ve mantığı iyi bir şekilde kavramamız gereklidir. Bu projede kullanacağımız malzemeler;

Malzeme Listesi

» 1 x Adet Arduino UNO

» 1 x Adet 220 ohm direnç

» 1 x Adet LED diyot

» 1 x Adet Breadboard

» 2 x Adet erkek-erkek Jumper kablo

Malzemelerimizi öğrendiğimize göre ilk olarak Python kodlarımızı oluşturalım ve açıklayalım.

Python Kodları ve Açıklaması

```
import serial
#Seri haberleşme için gerekli tanımlamalar
arduino=serial.Serial("com5",9600)
# Program döngümüz
while True:
    #Kullanıcıdan led açıp kapatabilmek için veri istedik
    veri=input("""
    Lutfen LED açmak icin 'a' harfine,
    sondurmek icin 'b' harfine basiniz(cikmak icin 'z' harfi):""")
   # Koşullarımıza göre led kontrolü ve program sonlandırma işlemi gerçekleştirdik
    if(veri=="a"):
        arduino.write(b'a')
    elif(veri=="b"):
        arduino.write(b'b')
    elif(veri=="z"):
        print("Gule Gule!")
        break
```

Kodlarımızı incelediğimizde ilk olarak seri haberleşme modülümüzü Python kodlarımıza dahil ettik. Daha sonra ise, `Serial()` sınıfımızı çağırdık ve `arduino` ismindeki oluşturduğumuz nesneye bu sınıfı atadık. Görüldüğü gibi **port bilgimizi** ve **seri haberleşme** hızlarımızı da tanımladık.

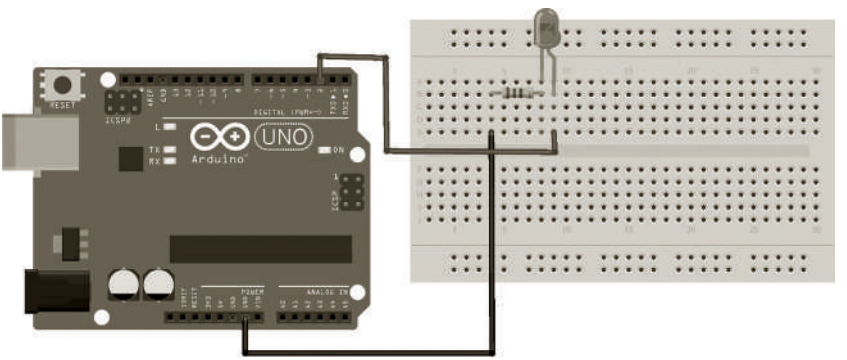
Sırasıyla "`com5`" ve `9600` değerleridir. Tabii ki burada tanımladığımız port bilgisini kafamıza göre belirlemiyoruz. Arduino kartımızı bilgisayarımıza bağladığımızda bu port bilgisini Arduino IDE'si üzerinden görüntülediğimiz için "`com5`" şeklinde tanımladık (Arduino kodlarımızı yazarken daha iyi öğreneceğiz). Gerekli tanımlamalarımızı yaptıktan sonra sonsuz bir döngü oluşturduk ve bu döngü içerisine girip, kullanıcıdan led açıp, kapatabilmesi için girdi girmesini istedik. Girilen değeri de veri değişkenine atadık. Böylelikle, kullanıcının girdiği her bir değer, veri değişkeninde atanmış olacak.

Devamında ise koşullu yapılarımızı oluşturduğumuzu gözlemliyoruz. Eğer veri değişkeni "a" değerine eşitse, Arduino kartına "a" sinyalini bayt olarak gönder şeklinde koşulumuzu oluşturduk.

Hocam burada veri kontrolünü (veri=="a") neden string tipinde gerçekleştirdik? şeklinde bir soru sorduğunuzu duyar gibiyim.

Cevabı, `input` fonksiyonuna bir veri girildiğinde, bu sayısal ya da karaktersel hiç farketmez `string` olarak tanımlanır. Bu sebepten dolayı, kontrol işlemlerimizde değişken denkliğine bakarken, string olarak denkliğin doğruluğunu kontrol ederiz. Kodumuzu incelemeye devam ettiğimizde, eğer veri değişkeni "b" değerine eşitse, Arduino kartına "b" değerini bayt olarak yollaması gerektiğini belirttik. Son koşullu yapımızın Arduino ile bir alakası bulunmamaktadır. Sonsuz bir döngü oluşturduğumuz için klavyeden "z" verisini programa gönderdiğimizde sonsuz döngümüzün sonlanması için koşulumuzu oluşturduk. Python kodlarımızı öğrendiğimize göre projemizin Arduino bağlantı şemasını oluşturalım ve inceleyelim.

Bağlanti Şemasi ve Açiklamasi



Bağlantı şemamız görüldüğü üzere oldukça basit. LED diyotumuzun 2 farklı bacağı bulunur. Bunlar artı (uzun bacak) ve eksi (kısa bacak) hatlarıdır. Arduino kartımızla LED kontrolü yapabilmemiz için dijital pin hatlarımızdan yararlanmamız gereklidir. Biz LED kontrolü için 2 numaralı dijital pin hattını kullanacağız. Tabii ki, dijital pin hattımıza LED diyotumuzun artı hattını bağlamamız gerektiğini de bilmemiz önemlidir. Aksi halde, LED diyotumuz ve Arduino kartımız zarar görebilir. LED diyotumuzun kısa bacağına 220 Ohm direnç bağladık. Bunun sebebi, Arduino UNO kartında bulunan dijital çıkış gerilimi 5V'dir.

Eğer 220 Ohm direnci LED diyotumuza bağlamazsak, LED diyotumuz yanma riskiyle karşı karşıya kalabilir. LED diyotumuzun eksi hattına direncimizi bağladığımıza göre direncin boşta kalan bacağına da Arduino'da bulunan GND hattını bağladık. Böylelikle, LED için gerekli olan + ve – hatların bağlantısını tamamlamış olduk. Bağlantı şemamızı öğrendiğimize göre şimdi Arduino kodlarımızı oluşturalım ve yorumlayalım.

3

Arayüz için Temel PyGame

Bu Bölümde

Bu bölümde, Python yazılım dilinde bulunan Pygame modülünü kullanarak, Arduino için nasıl arayüz geliştirebileceğimiz hakkında bilgi sahibi olacağız. En basit haliyle pencere oluşturulmasından, çeşitli şekillerin eklenebilmesi, arka plan görüntülerinin eklenebilmesi, pencerede yazı yazdırılabilmesi, şarkı eklenebilmesi ve Arduino'daki verinin arayüzde gösterilebilmesine kadar detaylıca bilgi sahibi olacağız.

Bu bölüm ile birlikte Arduino kartımızı arayüz üzerinden nasıl kontrol edebileceğimizi ve arayüzleri nasıl oluşturacağımız hakkında bilgi sahibi olacağız.

Peki, PyGame nedir?

PyGame, Python yazılım dilinde oyun ve kullanıcı arayüzleri oluşturmak için kullandığımız Python modülüdür. Kullanımı kolay olmakla birlikte, biz de çeşitli sensör verilerini okumak ya da Arduino kartını kontrol etmek amacıyla bu modül hakkında bilgi sahibi olup, çeşitli projeler gerçekleştireceğiz. Bildiğiniz gibi önceki bölümde Arduino kartı ve Python yazılım dilinin nasıl haberleştirilebildiğini öğrendik. Şimdi öğrendiğimiz bilgileri daha güzel ve şık bir projeye dönüştürebilmek için Pygame hakkında bilgi sahibi olalım.

PyGame Modülünün Kurulumu

PyGame modülünün kurulumu oldukça basittir. Bunun için aşağıdaki kodumuzu, VSCode üzerinden açtığımız komut satırında (terminal) çalıştıralım.

```
pip install pygame
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\book> pip install pygame
Requirement already satisfied: pygame in c:\users\alaca\appdata\local\programs\python\python39\lib\site-packages (2.1.2)

[notice] A new release of pip available: 22.2.2 -> 22.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\book>
```

Modül kurulumunu gerçekleştirdiğimize göre temel Pygame modülünü öğrenelim.

Pygame Modülünde Pencere Oluşturulması

Bu başlığımız altında Pygame modülünü kullanarak ilk penceremizi oluşturacağız. Tabii ki Python dosyamıza ilk olarak pygame modülünü dahil etmemiz gerekir. Daha sonra Pygame modülündeki tüm fonksiyon, sınıf ya da özel değişkenleri kullanmak istiyorsak Pygame modülünü hazırlamamız gereklidir. Bunun için `init()` fonksiyonundan yararlanırız. Bu fonksiyon herhangi bir parametre almaz ve görevi, bütün modül içerisindeki kullanacağımız işlevleri başlatmaktır. Şu ana kadar modülümüzü dahil ettik ve tüm işlevlerini kullanabilmek için `init()` fonksiyonumuzu çağırdık. Aşağıdaki gibi.

```
import pygame
# Modülü Başlattık
pygame.init()
```

Modülümüzü dahil edip, başlattıktan sonra yapmamız gereken, pencere oluşturabilmek için genişlik ve yükseklik değerlerini tanımlamak. Bunun için demet oluşturmalı ve demet içerisine sırasıyla **genişlik** ve **yükseklik** değerlerini tanımlamalıyız.

```
# 450 değeri genişlik, 360 değeri yükseklik
pencere_boyut= (450,360)
```

Görüldüğü gibi `pencere_boyut` isminde bir demet oluşturduk ve genişlik değerini `450`, yükseklik değerini ise `360` olarak ayarladık. Tabii ki pencere boyutumuzu tanımladıktan sonra artık yapmamız gereken penceremizi oluşturmak için gerekli olan fonksiyonumuzu çağırmak. Fonksiyonumuzun ismi `set_mode()` ve fonksiyonumuz pygame modülü içerisindeki **display** dosyasında bulunmaktadır. Tek bir parametre tanımlamamız gereklidir ve bu parametre ise pencere boyutunu ayarladığımız demettir. İnceleyelim.

```
# Penceremizi oluşturduk
pencere=pygame.display.set_mode(pencere_boyut)
```

Görüldüğü gibi `set_mode` fonksiyonumuzu çağırdık ve `pencere_boyut` demetini fonksiyon içerisinde tanımladık. Tabii ki fonksiyonumuzu da bir değişken içerisinde tanımlamamız gerekir. Sebebi, oluşturduğumuz oyun penceresi üzerinde değişiklik yapmak istediğimizde bu değişkene ihtiyacımız olacak. Değişkenimizin ismi pencere. Pencere oluşturmak için gerekli fonksiyonumuzu çağırdığımıza göre artık yapılması gereken döngü oluşturmak.

Peki, neden? Bugün bir oyun yapmak istediğimizde, penceremiz üzerinde değişiklik yapmak istediğimizde veya pencereyi ekranda tutmak istediğimizde döngüye ihtiyaç duyarız. Aksi durumda penceremiz direkt oluşturulur ve kapanır. Penceremizi ekranda tutmak istiyorsak ve istediğimiz şekilde hareket ettirmek istiyorsak, döngü içerisinde `for` döngüsü oluşturmalı ve event dosyası içerisindeki `get()` fonksiyonunu çağırmamız gereklidir. Haydi kodlarımız üzerinden inceleyelim ve yorumlayalım.

```
# Kontrol değişkeni
durum=True
while durum:
# Pencere kapatmak için döngü oluşturduk
    for etkinlik in pygame.event.get():
        print(etkinlik)
        # Etkinliğin tipini koşula girdirdik
        if etkinlik.type==pygame.QUIT:
            durum=False
```

Kodlarımızı incelediğimizde ilk olarak durum isminde bir değişken oluşturduk ve değerini `True` olarak ayarladık. Böylelikle döngümüz sürekli bir şekilde çalışacak ve penceremizi kapatmayana kadar da döngü sonlanmayacak. Bu sebeple, döngü kontrolünü durum değişkenine göre sağladık. Pygame modülünde yaptığımız tüm işlemler event dosyası içindeki `get()` fonksiyonunda tutulmaktadır. Örnek olarak mouse hareketleri, pencere büyütme ya da kapatma veya klavye hareketleri verilebilir. Yaptığımız tüm etkinlikleri görüntülemek istiyorsak `for` döngüsü oluşturmalı ve `get` fonksiyonundaki verilere ulaşmalıyız. Bunun için, `etkinlik` isminde eleman değişkeni oluşturduk ve get içindeki verilerin yerini almasını sağladık. Yaptığımız etkinlikleri görüntülemek için print fonksiyonuyla etkinlik değişkenini çıktı aldık. Kodumuzu çalıştırdığımızda konsol ekranında tüm etkinlikleri görüntüleyebileceğiz.

Program Çıktısı

```
<Event(1024-MouseMotion {'pos': (269, 51), 'rel': (7, -7), 'buttons': (0, 0,
0), 'touch': False, 'window': None})>
<Event(1024-MouseMotion {'pos': (275, 43), 'rel': (6, -8), 'buttons': (0, 0,
0), 'touch': False, 'window': None})>
<Event(32784-WindowLeave {'window': None})>
<Event(32787-WindowClose {'window': None})>
<Event(256-Quit {})>
```

Gördüğünüz gibi mouse ile gezindiğimizde, koordinat bilgisine kadar çıktı alabiliyoruz. Tabii ki bir de oluşturduğumuz pencereleri kapatmamız gereklidir. Bunun için, etkinlik değişkeni içerisine tanımlanmış type verisinden yararlanmamız gereklidir. Böylelikle, koşulumuzu oluşturduk ve `etkinlik.type` durumu pencere kapatma değişkenine eşitse, durum değişkenini False olarak ayarladık. `pygame.QUIT` değişkeni, oluşturulan penceredeki çarpı düğmesidir.

Pygame Modülünde Pencereye Yazı Yazdırma

Bu başlığımızda, oluşturduğumuz pencerede nasıl yazı yazabileceğimizi öğreneceğiz. Bunun için öğrenmemiz gereken bazı fonksiyonlar bulunmaktadır. Bu fonksiyonlar, hem varsayılan fontları kullanmamıza hem de harici fontları kullanmamıza olanak sağlarlar. Haydi ilk olarak varsayılan fontlara nasıl ulaşabileceğimizi öğrenelim.

Varsayılan Fontlara Ulaşmak

Bu işlem için yapmamız gereken pygame modülü içerisindeki font dosyasına ulaşıp, `get_fonts()` fonksiyonunu çağırmamız gereklidir. Böylelikle, Var olan tüm fontları liste içinde tanımlayabilir ve hepsini çıktı alabiliriz. Kullanımı, `pygame.font.get_fonts()` şeklindedir. Haydi şimdi varsayılan fontlarımızı kod üzerinden görüntüleyelim.

```
import pygame
# Modülü başlattık
pygame.init()
# Font listesi oluşturduk
font_listesi=pygame.font.get_fonts()
# Fontları çıktı aldık
for font in font_listesi:
    print(font)
```

Kodlarımızı incelediğimizde direkt olarak fontlara ulaşabilmek için `font_listesi` adında bir değişken oluşturduğumuzu ve bu değişkene `get_fonts()` fonksiyonunu atadığımızı gözlemliyoruz.

Peki, bu değişkenimizin tipi ne olur?

Aslında kullandığımız `get_fonts()` fonksiyonu, var olan tüm varsayılan fontları liste içerisinde tuttuğu için, `font_listesi` değişkeninin tipi de listedir. Tabii ki liste elemanlarına ulaşabilmenin en kolay yolu da `for` döngüsünden yararlanmaktır. Bu sebeple, biz de `for` döngüsü oluşturduk ve `font` adlı eleman değişkenimizin liste elemanlarının yerini alması gerektiğini belirttik. Böylelikle, varsayılan tüm fontları kolaylıkla çıktı alacağız. Program çıktısını inceleyerek daha iyi bir şekilde var olan fontları görebiliriz. Tabii ki, bazı fontlar sizin bilgisayarınızda olmayabilir ya da olabilir.

Program Çıktısı

```
arial
arialblack
bahnschrift
calibri
cambria
cambriamath
candara
comicsansms
consolas
constantia
corbel
couriernew
ebrima
franklingothicmedium
```

VARSAYILAN FONTLARI KULLANARAK PENCEREDE ÇIKTI ALMAK

Varsayılan fontlarımızı öğrendiğimize göre şimdi bir tanesini kullanalım ve ekrana **Merhaba** kelimesini yazdıralım.

Örneğin **Arial** fontunu kullanarak **Merhaba** çıktısı alalım. Tabii ki fontlar üzerinden işlem yapacağımız için **pygame** içerisindeki **font** dosyasında bulunan fonksiyonları kullanmalıyız. Pencerede yazı yazdırabilmek için ilk olarak `font` değişkeni oluşturmamız gereklidir. Örneğin `arayuz_font` şeklinde bir değişken oluşturabiliriz. Daha sonra bu değişkene, font içerisinde bulunan `SysFont()` fonksiyonunu atamalıyız. Bu fonksiyon 2 farklı parametre almaktadır. Birinci parametre string tipinde olmalı ve fontun adını yazmamız gereklidir. Biz **Arial** fontunu kullanacağımız için fonksiyon birinci parametresine tanımı "`arial`" şeklinde olmalıdır. İkinci parametre ise yazı fontunun büyüklüğüdür ve tam sayı şeklinde tanımlanmalıdır. Tabii ki büyüklük cinsininde piksel olduğunu bilelim.

Örneğin ikinci parametreyi 64 olarak tanımladığımızda yazı fontunun büyüklüğü 64 piksel büyüklüğünde olacaktır. Kullanımı, `pygame.font.SysFont(font_ismi,boyut)` şeklindedir. Font tanımlaması yapıldıktan sonra bu font için yazı yazmamız gerekir. Bunun için yeni bir değişken oluşturmalı ve bu değişkene, oluşturduğumuz yazı fontu değişkeninde tanımlı `render()` metodunu atamamız gerekli.

Bu metot, eklediğimiz fontu kullanarak istediğimiz yazıları yazdırır. 3 farklı parametre tanımlanmalıdır. Kullanımı, `font_degiskeni.render("Yazacagimiz`

4

Genel Proje 1: Röle Kontrol Arayüzü

Bu Bölümde

Bu bölümde, sizler ile birlikte ilk ileri seviye projemizi geliştireceğiz. Gerçekleştireceğimiz proje ile birlikte buton kontrolü yapabildiğimiz arayüzü geliştirip, oluşturduğumuz arayüzle Arduino'ya bağlı rölenin kontrolünü gerçekleştireceğiz. Haydi başlayalım. :)

PROJEYE GİRİŞ

Bu başlığımızla birlikte genel projelerimizi geliştirmeye başlayacağız. İlk geliştireceğimiz proje ile birlikte oluşturduğumuz arayüzle birlikte Arduino'ya bağlı röle kontrolü gerçekleştireceğiz. Arayüzümüzde kullanacağımız tüm kaynaklara Drive üzerinden ulaşabilir ve gönül rahatlığıyla kullanabilirsiniz. Yapacağımız arayüzde 3 farklı görüntü kullanacağız. Bunların ikisi, açma ve kapatma işlemi yapacağımız buton görüntülerimizdir. Diğer görüntümüz ise penceremizin arka plan görüntüsü olacak. Projemizde kullanacağımız malzemeler;

» 1 x Adet Arduino UNO

» 1 x Adet 5V Röle Modülü

» 1 x Adet Breadboard

» 1 x Adet LED Diyot

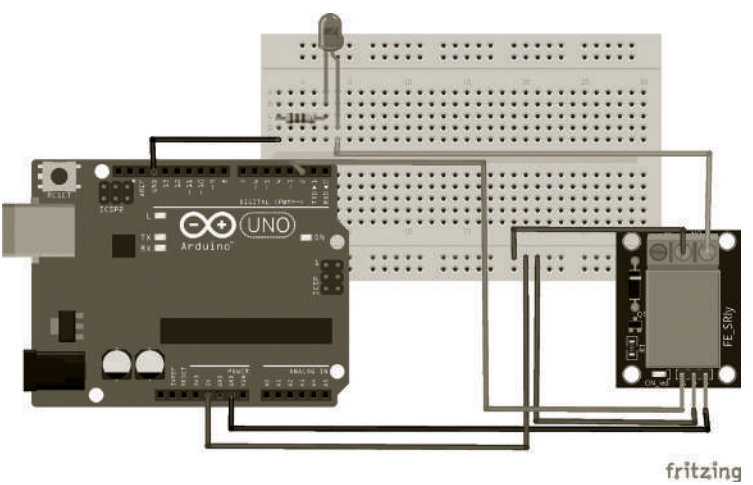
» 1 x Adet 220 Ohm Direnç

» 3 x Adet erkek-dişi Jumper Kablo

» 4 x Adet erkek-erkek Jumper Kablo

Projede kullanacağımız malzemelerimizi öğrendiğimize göre röle modülümüz hakkında kısaca bilgi sahibi olalım. Röle, elektrikle çalıştırılan ve anahtar görevi gören elektromanyetik bir anahtardır. Biz de oluşturacağımız arayüz ile birlikte rölemizi kontrol ederek LED diyotumuzu yakıp, söndüreceğiz. Haydi ilk olarak bağlantı şemamızı inceleyelim ve yorumlayalım.

BAĞLANTI ŞEMASI VE AÇIKLAMASI



Bağlantı şemamızı incelediğimizde öncelikle rölenin üst bölgesinden başlayalım. LED'imizin uzun bacağını röledeki NO (Normalde açık) hattına bağladık. Böylelikle, röle modülüne Arduino ile HIGH komutu gönderdiğimiz zaman LED diyotumuz yanacak. LOW komutu gönderdiğimizde ise sönecek. Tabii ki **NO** bağlantısının yanındaki **com** portuna(ortadaki hat) ise 5V gerilim hattımızı bağladık.

Peki, neden?

Röledeki com portu, enerji hattıdır. Bu porta kontrol etmek istediğimiz cihazın gerilimi bağlanır. Biz de Arduino'daki 5V gerilim hattını rölenin com portuna bağladık. Rölenin alt bölgesini incelediğimiz de 3 farklı hattımız bulunduğunu görüntülüyoruz. Solda kalan hat, Arduino'daki dijital pin hatlarına bağlanır. Biz bu projede 2 numaralı dijital pin hattından yararlanacağız. Görevi, röleyi kapatmak ya da açmaktır. Ortadaki hat rölenin çalışabilmesi için gerekli olan gerilimdir. Rölemiz 5V ile çalıştığı için, ortadaki hattı Arduino'nun 5V gerilim hattına bağladık. Sağda kalan hat ise rölenin eksi hattıdır. Bu hattı da Arduino'daki GND hattına bağladık. Gördüğünüz gibi LED'imizin diğer bacağını incelediğimizde direnç bağladığımızı gözlemliyoruz. Kullandığımız LED diyota direkt olarak 5V gerilim gönderdiğimizde yanabilmektedir. Biz de yanmanın önüne geçmek için dirençten faydalandık. Direncin de diğer bacağına Arduino'daki GND hattını bağladık. Bağlantı şemamız hakkında bilgi sahibi olduğumuza göre şimdi Python kodlarımızı oluşturalım ve yorumlayalım.

Python Kodları ve Açıklaması

```
import pygame
import serial
# Modülü başlattık
pygame.init()
# Seri haberleşme için port tanımlama
port=serial.Serial("com5",9600)
# Pencere oluşturduk
GENISLIK=1280
YUKSEKLIK=720
pencere=pygame.display.set_mode((GENISLIK,YUKSEKLIK))
# Butonu oluşturduk
buton_on=pygame.image.load("ac.png")
buton_on_koordinat=buton_on.get_rect()
buton_on_koordinat.topleft=(GENISLIK/2-100,YUKSEKLIK/2)
buton_kapa=pygame.image.load("kapa.png")
buton_kapa_koordinat=buton_kapa.get_rect()
```

5

Genel Proje 2: Sensör Arayüz Projesi (DHT-11 ve Yağmur Sensörü)

Bu Bölümde

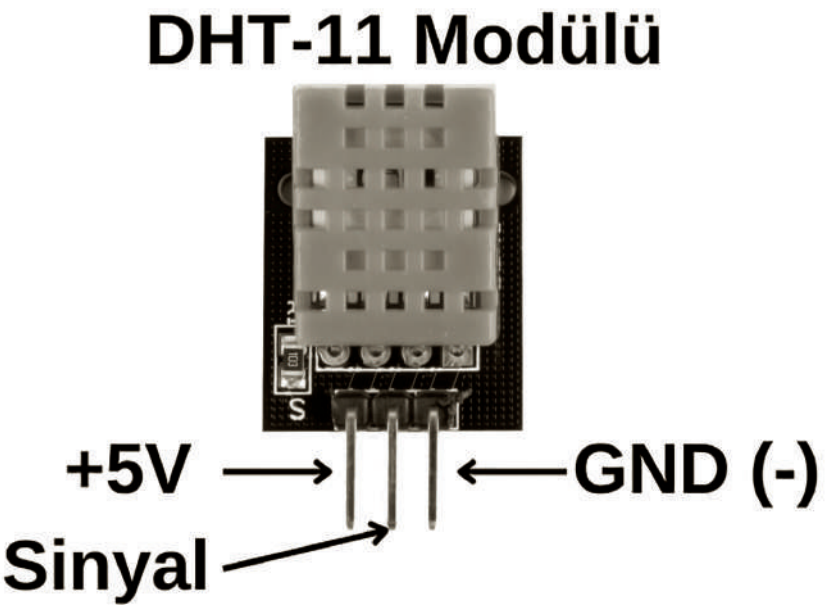
Bu bölümde, sizler ile birlikte ikinci ileri seviye projemizi geliştireceğiz. Gerçekleştireceğimiz proje ile birlikte DHT-11 ve yağmur sensörlerinin kullanımı hakkında bilgi sahibi olup, oluşturacağımız arayüzde sensör verilerini görüntüleyeceğiz.

Haydi başlayalım. :)

Bu başlığımızda 2 farklı sensördeki verileri okuyarak, oluşturduğumuz arayüzde bu sensörlerin değerlerini göstereceğiz. Projemize geçmeden önce sensörlerimiz hakkında kısaca bilgi sahibi olalım.

DHT-11 Sıcaklık ve Nem Sensörü Modülü

Günümüzde oldukça popüler olan bu sensör ile birlikte hem sıcaklık hem de ortamın nem oranı kolaylıkla ölçülebilmektedir. Bildiğiniz gibi ülkemizde sıcaklık derece(celcius) cinsinden gösterilmektedir. DHT-11 sensörü de bu ölçüm için uygun bir sensördür. Ayrıca, bu sensör ile birlikte harici modül olmadan ortamın nem oranını da hesaplayabiliriz. DHT-11 modülünü kullanabilmemiz için Arduino IDE'sine bu modülün kütüphanesini eklememiz gerekmektedir. Kütüphane ismi, **dht11.h** kütüphanesidir. Görüntümüzden de anlaşılacağı üzere modülün 3 farklı hattı bulunmaktadır.



Bu hatlar sırasıyla 5V, sinyal ve GND hatlarıdır. 5V hattımız Arduino kartımızdaki 5V pin hattına bağlanır. Sinyal hattımız ise Arduino kartımızdaki dijital pin hattına bağlanır. Genel olarak 2 numaralı dijital pin hattına bağlamamız tavsiye edilir. GND hattı ise Arduino'daki GND hatlarından bir tanesine bağlanmalıdır. Modülün hatları hakkında bilgi sahibi olduğumuza göre şimdi bağlantımıza geçiş yapalım ve DHT-11 kullanarak, ortamdaki hem nem hem de sıcaklık ölçümü örneği yapalım.

Malzeme Listesi

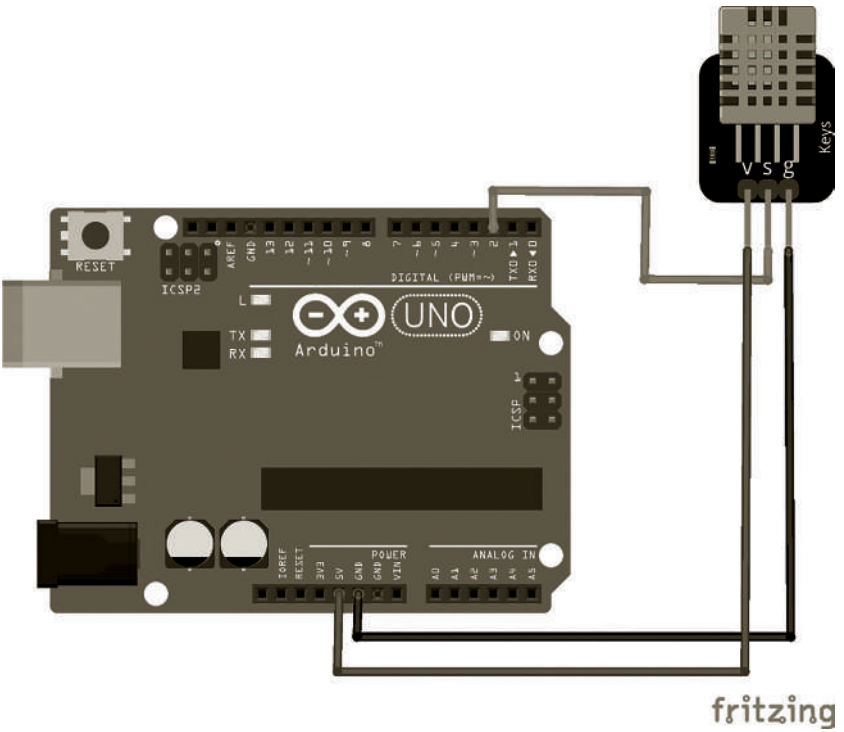
Örneğimizde kullanacağımız malzemeler;

» 1 x Adet Arduino UNO

» 1 x Adet DHT-11 Modülü

» 3 x Adet erkek-dişi Jumper Kablo

Bağlanti Şemasi ve Açiklamasi



Bağlantı şemamızı incelediğimizde oldukça kolay bir şema ile karşılaşıyoruz. Modülümüzün 5V hattını Arduino'daki 5V hattına, sinyal hattını 2 numaralı dijital pin hattına ve GND hattını ise Arduino kartımızdaki GND hattına bağladık. Bağlantımızı gerçekleştirdiğimize göre Arduino kodlarımızı oluşturalım ve modülümüz için gerekli olan sınıf ve fonksiyonların kullanımını da kodlarımız üzerinden öğrenelim.

6

Firmata Protokolü

Bu Bölümde

Bu bölümde, sizler ile birlikte Firmata protokolü hakkında bilgi sahibi olacağız. Python ile Arduino'nun pinlerini kontrol edebilmeyi, ayrıca Arduino'nun analog pinlerini Python'da okuyabilmeyi öğreneceğiz. Haydi başlayalım. :)

GİRİŞ

Firmata protokolü, Arduino gibi mikrodenetleyicileri bilgisayar ile kontrol etmemize olanak sağlayan protokoldür. Bizler de oluşturacağımız üçüncü projede Python ve Arduino'nun haberleşmesi için Firmata protokolünden yararlanacağız. Firmata, aynı zamanda farklı mikrodenetleyiciler ile de çalışabilmektedir. Firmata protokolünü kullanabilmemiz için Python tarafında pyfirmata modülünden yararlanmamız gereklidir. Bunun için aşağıdaki kodu terminal üzerinde çalıştırmalısınız;

```
pip install pyfirmata
```

Bazı durumlarda yukarıda belirtilen komut çalışmayabilmektedir. Bu sebeple aşağıdaki kodu da kullanabilirsiniz;

```
python3 -m pip install pyfirmata
```

Kurulumu gerçekleştirdiğimize göre şimdi `pyfirmata` modülünü kullanarak Arduino kartımızı kontrol etmeyi öğrenelim. Eğer Arduino kartımızı kontrol etmek istiyorsak, öncelikle pyfirmata modülünü eklemeli Arduino sınıfından yararlanmalıyız. Bu sınıf, Arduino kartımızı kontrol etmemize olanak sağlar. Tek bir parametre almaktadır. Bu parametre Arduino'nun bağlı olduğu portun string halinde tanımlanmasıdır. Kullanımı, `pyfirmata.Arduino("port")` şeklindedir.

```
import pyfirmata
# Port tanımladık
port=pyfirmata.Arduino("com5")
```

Görüldüğü üzere modülümüzü tanımladık. Devamında ise port isminde nesne oluşturup, modül içerisinde bulunan Arduino sınıfımızı çağırdık. İlk parametre olarak, Arduino'nun bağlı olduğu portu tanımladık. Arduino UNO kartının 13 numaralı dijital pinini açıp, kapattığımızda gömülü LED yanıp, sönmektedir. Haydi şimdi pyfirmata modülüyle bu gömülü LED'i kontrol edelim.

Proje 6: Arduino'da Bulunan Gömülü LED'in Kontrolü

Bu projemizde pyfirmata modülünü kullanarak, Arduino'daki 13 numaralı dijital pin hattını kontrol edeceğiz. Kullanacağımız malzeme sadece Arduino UNO kartı. Haydi Python kodumuzu inceleyelim ve yorumlayalım.

Python Kodları ve Açıklaması

```
import pyfirmata
import time
# Port tanımladık
arduino=pyfirmata.Arduino('com5')
# 13 Numaralı Dijital Pinin Kontrolü
while True:
    # Ledi Yaktık
    arduino.digital[13].write(1)
    time.sleep(3)
    # Ledi Söndürdük
    arduino.digital[13].write(0)
    time.sleep(3)
```

Kodlarımızı incelediğimizde ilk olarak `pyfirmata` modülümüzü tanımladık. Devamında ise LED'i belirli bir süre yakıp, söndürebilmek için de `time` modülünü kodlarımıza ekledik. Görüldüğü üzere `arduino` isminde nesne oluşturduk ve `pyfirmata` içerisindeki Arduino sınıfını nesnemize atadık. Portunu belirttikten sonra döngü oluşturduğumuzu gözlemliyoruz. Eğer dijital pinleri kontrol etmek istiyorsak, nesnemiz içerisinde bulunan digital listesini çağırmalı ve pin numaramızı indeks olarak tanımlamalıyız. Bizde `digital[13]` şeklinde tanımladık.

Dijital pin hattımızı tanımladıktan sonra yapılması gereken `write()` metodunun çağırılması ve kontrol değerinin girilmesi gereklidir. `write` metodu 1 adet parametre alır ve alacağı değerler bellidir. Bu değerler `0` veya 1'dir. `0` değeri `LOW`, 1 değeri ise `HIGH` komutuna eşittir. Böylelikle biz de dijital pinimizi tanımladıktan sonra `write` metodunu çağırdık ve ilk olarak parametresini 1 olarak ayarladık. Böylelikle gömülü ledimizi 3 saniye boyunca yakacağız. Devamında ise tekrardan nesnemizle 13 numaralı dijital pinimize ulaştık ve bu sefer write metoduyla `0` değerini tanımladık. Böylelikle ledimiz 3 saniye boyunca da sönük kalacak. Python kodlarımızın ne işe yaradığını öğrendiğimize göre şimdi Arduino tarafına odaklanalım.

ARDUINO KODLARI VE AÇIKLAMASI

Arduino tarafında herhangi bir kod yazmamıza gerek yoktur. Ancak, Arduino IDE'sinde hazır bulunan StandardFirmata örneğini açmalı ve Arduino kartımıza kodları yüklemeliyiz. Bu işlem için şu adımları izlememiz gereklidir;

1. Adım

Arduino IDE'sini açın ve Dosya seçeneğine tıklayın.

2. Adım

Örnekler seçeneğine tıklayın ve Firmata'yı seçin.

3. Adım

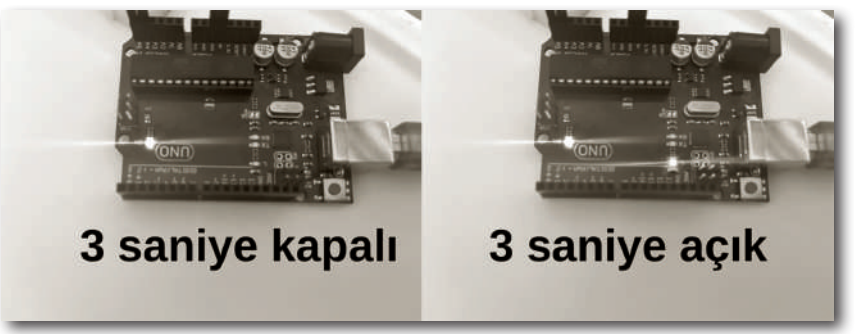
Firmata üzerinden **StandardFirmata** dosyasını seçin ve kartınıza kodları yükleyin.

4. Adım

Tebrikler. İşlerimiz buraya kadar :).

Bu şekilde işlemlerimizi gerçekleştirip, Python kodumuzu çalıştırdığımızda görüntüdeki gibi sonuçla karşılaşacağız.

SONUÇ



7

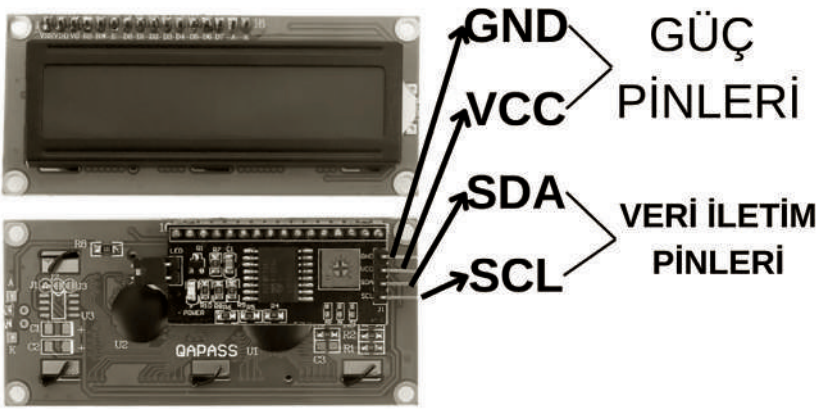
I2C LCD Ekran Modülü

Bu Bölümde

Bu bölümde sizler ile birlikte I2C destekli LCD ekranın kullanımı hakkında bilgi sahibi olacağız. LCD ekranın Arduino ile bağlantısının nasıl yapılabildiğini, bir yazının nasıl yazdırılabildiğini, konumlama işleminin nasıl yapılabildiğini ve kaydırma işleminin nasıl yapıldığını sizlerle birlikte öğreneceğiz.

Haydi başlayalım. :)

Bu konu başlığımızda Arduino kartı ile birlikte I2C destekli LCD ekran modülünü nasıl kullanabileceğimiz hakkında bilgi sahibi olacağız. LCD ekran, açılımı **Liquid Crystal Display**, yani Sıvı Kristal Ekran olarak Türkçe ismi bulunmaktadır. Günümüzde Arduino dahil olmak üzere birçok mikrodenetleyici ile kullanılmaktadır. Genel olarak menü, bilgi monitörü veya kontrol monitörü olarak elektronik projelerde kullanılmaktadır. Biz de bir sonraki genel projemizde LCD ekranda yazı yazdırabilen arayüz geliştireceğimiz için LCD ekran kullanımı hakkında bilgi sahibi olmalıyız. Bu sebeple özel bir başlıkta LCD ekranı sizlere tanıtmak istiyorum. Öğreneceğimiz LCD ekran I2C desteklidir. Normal şartlarda LCD ekran kontrolü yaparken Arduino ile arasında çok fazla bağlantı yapmamız gerekir. Ancak, I2C destekli LCD ekranda bağlantı sayısı, güç pinleri dahil olmak üzere 4'tür. Kullanacağımız LCD ekranın 4 farklı hattı bulunuyor.



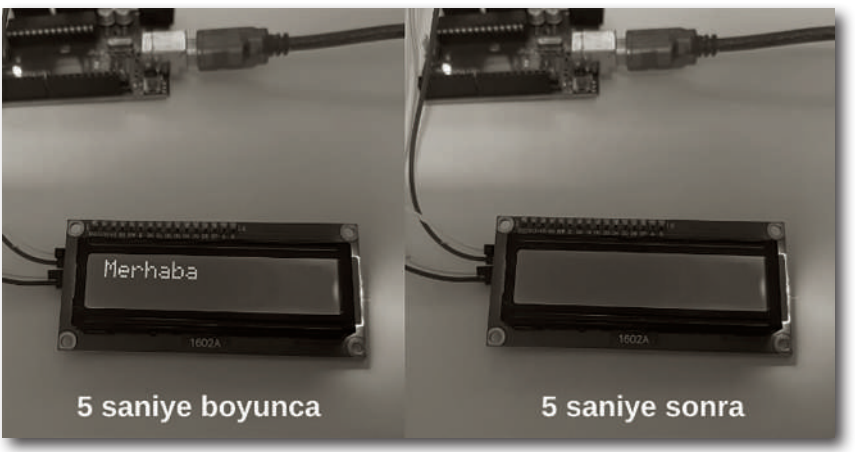
Görselden de anlaşılacağı üzere 2 hat güç için, diğer 2 hat ise veri iletim pinleridir. LCD ekranımız 5V gerilim ile çalışmaktadır. Bu sebeple, Arduino bağlantısını gerçekleştirirken, LCD ekranın VCC hattını Arduino kartımızda ki 5V hattına bağlamamız gereklidir. Eğer I2C destekli LCD ekranı Arduino ile kullanmak istiyorsak, SDA ve SCL hatlarını özel olarak belirlenmiş hatlara bağlamamız gereklidir. Bu hatlar Arduino modelleri arasında farklılık göstermektedir. Tablo üzerinden Arduino modellerine göre bağlantı yerlerini görüntüleyebilirsiniz.

Kodlarımızı incelediğimizde ilk olarak LCD ekran kütüphanesini Arduino kodlarımıza ekledik. Devamında ise `LiquidCrystal_I2C` sınıfını çağırdık ve `lcd` isminde nesne oluşturduk. Görüldüğü üzere 3 farklı parametre tanımladık. İlk olarak adres bilgisini tanımladık ve benim kullandığım LCD ekran adresi 0x27'dir.

Devamında ise hepimiz için ortak olan sütun ve satır sayılarını tanımladık. Bu parametreler sırasıyla 16 ve 2'dir. LCD ekranımızı kullanabilmemiz için oluşturduğumuz nesne ile başlatmamız gereklidir. Bunun için begin() metodu kullanılır. Kullanımı, `lcd_ekran_nesnesi.begin()` şeklindedir ve herhangi bir parametre almaz. Görüldüğü üzere biz de kodlarımızda LCD ekranımızı başlattık. Elbette LCD ekranda **Merhaba** çıktısı alacağımız için, çıktı alma metodumuzu da kullanmamız gerekir. Biz de öyle yaptık. Bu işlem için `print()` metodu kullanılır.

Bu metot, LCD ekranda çıktı almamıza olanak sağlar. Kullanımı `lcd_ekran_nesnesi.print(veri)` şeklindedir. Bu veri sayı ya da string şeklinde tanımlanabilir. Örneğin 27 sayısını ekranda çıktı alacaksak, `lcd_ekran_nesnesi.print(27)` şeklinde, **Merhaba** şeklinde çıktı alacaksak da `lcd_ekran_nesnesi.print("Merhaba")` şeklinde string olarak tanımlama yapmalıyız. Görüldüğü üzere biz de string olarak **Merhaba** çıktısını, çıktı alma metodumuzda belirttik. Merhaba çıktısı ekranda 5 saniye boyunca gözükecek. Devamında ise ekranımızı temizledik. Bunun için de `clear()` metodu kullanılır. Bu metot, LCD ekranımızı temizlememize olanak sağlar. Kullanımı, `lcd_ekran_nesnesi.clear()` şeklindedir ve herhangi bir parametre almaz. Bu şekilde kodlarımızı Arduino kartımıza yüklediğimizde, görüntüde ki gibi sonuçla karşılaşacağız.

Sonuç



ÖRNEK (LCD EKRAN KONUM AYARLAMA)

Bu örneğimizde LCD ekranda aldığımız çıktıların konumunu nasıl ayarlayabileceğimiz hakkında bilgi sahibi olacağız. Bu işlem için bağlantı şemamız ve kullanacağımız malzemelerimiz önceki örnek ile aynı. Haydi örnek Arduino kodumuz üzerinden bu işlemin nasıl yapıldığını öğrenelim.

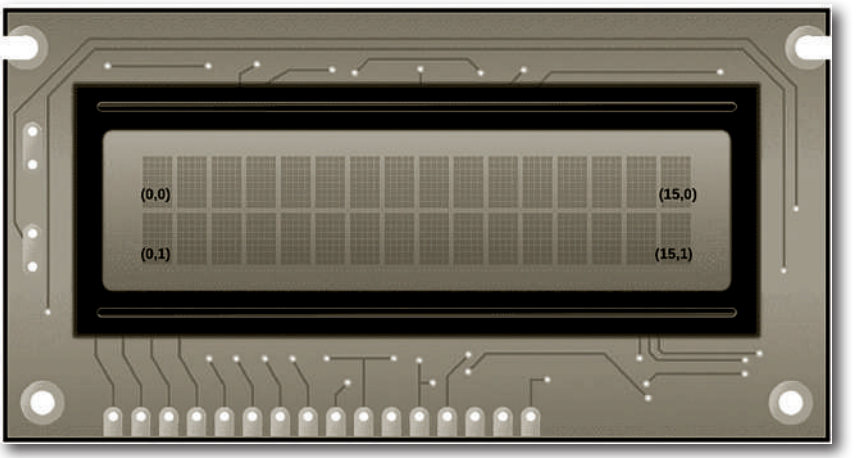
ARDUINO KODLARI VE AÇIKLAMASI

```
#include <LiquidCrystal_I2C.h>
// LCD Ekran nesnemizi oluşturduk
LiquidCrystal_I2C lcd(0x27,16,2);
void setup() {
// Ekranın çalışması için başlattık
lcd.begin();
}
void loop() {
  // Başlangıç konumunda çıktı aldık
  lcd.setCursor(0,0);
  lcd.print("Yilmaz");
  // Alt satırda ve 3. sütundan itibaren çıktı aldık
  lcd.setCursor(2,1);
  lcd.print("ALACA");
}
```

Kodlarımızı incelediğimizde ilk olarak LCD ekran kütüphanemizi tanımladık ve gerekli olan sınıfımızı çağırıp, `lcd` isminde nesne oluşturduk. Devamında ise nesnemiz için adres numarası, sütun sayısı ve satır sayısı parametrelerini tanımladık. `setup` fonksiyonumuzda ise LCD ekranımızı başlattık. Amacımız LCD ekranda alacağımız çıktıların konumunu ayarlamak olduğu için döngü fonksiyonumuzda bu işlemi gerçekleştirdik. Bunun için, `setCursor()` metodu kullanılmaktadır. Bu metot 2 farklı parametre almaktadır. Bu parametreler başlangıç sütun değeri ve başlangıç satır değeridir.

Kullanımı, `lcd_ekran_nesnesi.setCursor(baslangic_sutun_no,baslangic_satir_no)` şeklindedir. Bu 2 parametre de tam sayı şeklinde tanımlanır. Döngü fonksiyonumuzu incelediğimizde ilk olarak parametreyi `(0,0)` olarak tanımladık. Böylelikle Yilmaz çıktısı LCD ekranın sol üst köşesinde gözükecektir. LCD ekranın konumlama işlemi her iki parametre için 0'dan başlamaktadır. Devamında ise `setCursor` metodunu tekrar çağırdık ve bu sefer `setCursor(2,1)`

şeklinde tanımladık. Bu parametreye göre alacağımız ALACA çıktısı, alt satırda ve 3. sütundan itibaren görüntülenmeye başlayacak. Konumlama işlemi için görüntüyü inceleyerek daha iyi anlayabilirsiniz.



Bu şekilde kodumuzu Arduino kartımıza yüklediğimizde görüntüdeki gibi çıktı alacağız.

Sonuç

8

İleri Seviye Proje

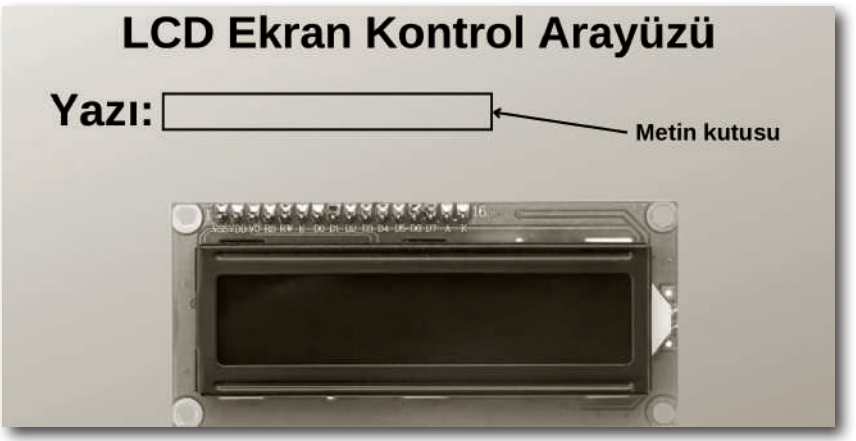
Bu Bölümde

Bu bölümde sizler ile birlikte üçüncü ileri seviye projemizi geliştireceğiz.

Gerçekleştireceğimiz proje ile birlikte LCD ekranda yazı yazdırma işlemi yapabilen arayüz oluşturacağız. Bu arayüzde klavyeden girdiğimiz yazıları yazıp, enter tuşuna bastığımızda yazdığımız yazıyı LCD ekranda görüntüleyeceğiz. Haydi başlayalım. :)

GENEL PROJE 3: LCD EKRAN KONTROL ARAYÜZÜ

Üçüncü projemiz ile birlikte LCD ekranda yazı yazdırabileceğimiz arayüz tasarlayacağız ve Arduino'ya bağlı LCD ekran üzerinden çıktı alacağız. Gerçekleştireceğimiz proje sayesinde hem klavyeden yazdığımız verileri arayüzde görüntüleyebileceğiz hem de yazdığımız veriyi tamamlayıp, **Enter** tuşuna bastıktan sonra LCD ekranımızda çıktı alacağız. Bu bizim son geliştireceğimiz proje olacak. Buraya kadar 3'ü kapsamlı olmak üzere çok fazla proje ve örnek geliştirdik. Bunun için sizi tebrik ederim. Haydi şimdi son projemize odaklanalım.



LCD ekran arayüzümüzü incelediğimizde görüldüğü üzere metin kutumuz mevcut. Buraya yazı yazacağız. Ancak, yazdığımız yazının etkili olabilmesi için öncelikle kutumuza tıklamamız gerekiyor. Bu işlemi de yazılım ile ayarlayacağız. Tabii ki projenin birçok faktörü bulunuyor.

Örneğin metin kutusunun taşması, yani 16x2'lik LCD ekranda 16 karakterden fazla harfin belirtilmesi. Bunun önüne geçmek amacıyla da koşullu yapımızı oluşturacağız buna bağlı olarak eğer girilen karakter sayısı 16'dan fazla olursa kullanıcıya uyarı verdirteceğiz. Diğer yandan kullanıcı yazmak istediği yazıyı yazdıktan sonra silmek isteyebilir. Bunun için de ayrıca yazdığımız yazıların silinmesini sağlayacağız. Tabii ki metin kutumuzda yazdığımız yazıyı bir de gözlemlemek amacıyla arayüzde bulunan LCD ekranda da yazımızı göstereceğiz. Eğer kullanıcı yazdığı yazıyı LCD ekranda görüntülemek isterse enter tuşuna basacak biz de verileri Arduino'ya göndereceğiz. Tabii ki Arduino ile de LCD ekranı kontrol edeceğiz. Ek olarak, kullanıcı enter tuşuna bastıktan sonra da metin kutusunu temizleyeceğiz. Projemizin tüm kaynaklarına aynı şekilde Drive üzerinden ulaşabilirsiniz.

Malzeme Listesi

Projemizde kullanacağımız malzemeler;

» 1 x Adet Arduino UNO

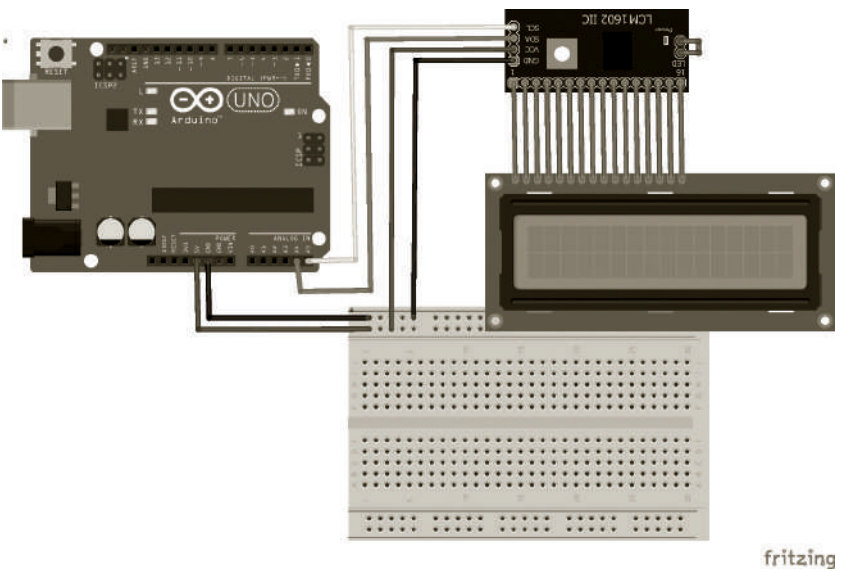
» 1 x Adet I2C destekli 16x2 LCD Ekran

» 4 x Adet erkek-dişi Jumper Kablo

» 1 x Adet Breadboard

Kullanacağımız malzemeler hakkında bilgi sahibi olduğumuza göre bağlantı şemasına geçelim ve yorumlayalım.

Bağlanti Şemasi ve Açiklamasi



Bağlantı şemamızı incelediğimizde LCD ekranımızın SDA hattını Arduino kartımızdaki A4 hattına bağladık. LCD ekrandaki SCL hattını ise Arduino kartımızdaki A5 hattına bağladık. LCD ekranımıza güç verebilmek için bir de güç hatlarını bağlamamız gerekiyor. LCD ekranımız 5V gerilim ile çalıştığı için, Arduino'daki 5V hattına LCD ekranın VCC hattını bağladık. Son olarak LCD ekranın GND hattını Arduino'daki GND hattına bağlamamız yeterli olacaktır. Bağlantı şemamızı incelediğimize göre şimdi Python kodlarına geçiş yapalım ve inceleyelim.

Python Kodları ve Açıklaması

```
import pygame
from pyfirmata import Arduino, util, STRING_DATA
# Modülü Başlattık
pygame.init()
# Port ve Kart Tanımladık
port='com5'
kart=Arduino(port)
# Pencere Oluşturduk
GENISLIK,YUKSEKLIK=1280,720
pencere=pygame.display.set_mode((GENISLIK,YUKSEKLIK))
# Arayüz Arka Plan
arayuz_goruntu=pygame.image.load("lcd_arayuz.jpg")
# Boş Metin Oluşturduk
metin=""
aktif=False
# Metin font
font=pygame.font.SysFont("arial",48)
# Pencere Döngüsü
durum=True
while durum:
    pencere.blit(arayuz_goruntu,(0,0))
    for etkinlik in pygame.event.get():
        if etkinlik.type==pygame.QUIT:
            durum=False
        if etkinlik.type==pygame.MOUSEBUTTONDOWN:
            konum=pygame.mouse.get_pos()
            if 250<=konum[0]<=710 and 200<=konum[1]<=260:
                aktif= not aktif
            else:
                aktif=False
        if etkinlik.type==pygame.KEYDOWN:
            if aktif:
                if etkinlik.key==pygame.K_RETURN:
                 kart.send_sysex(STRING_DATA,util.str_to_two_byte_iter(metin))
                    metin=""
                    pass
                elif etkinlik.key==pygame.K_BACKSPACE:
                    metin=metin[:-1]
```

Yılmaz ALACA

instagram.com/yilmazalacaa

twitter.com/alacayilmaz01

linkedin.com/in/yilmazalaca

udemy.com/user/yilmaz-alaca

youtube.com/@yilmazalaca

yilmazalaca01@gmail.com