

**Mekatronik Mühendisliği Bölümü**

GÖMÜLÜ SİSTEMLER

ve

UYGULAMALARI

FİNAL PROJE ÖDEVİ

MEHMET AKSU – 171214011 FEDİ SALHİ – 170214925

**VOICE2DISPLAY**

**İÇİNDEKİLER**

**Genel Bakış** 3

**PYGAME** 4

**GOOGLE SPEECH RECOGNITION** 4

**Donanım Alt Yapısı** 5

**Yazılım Alt Yapısı** 6

**Ses Tanımla Kod Dosyasının Oluşturulması** 6

**KAYNAKÇA** 14

**PROJENİN GİTHUB ADRESİ** 14

# **Genel Bakış**

Bir önce ki Vize projesinde konuşma engeli bulunan bir vatandaşın anlatmak istediğini karşısında ki kişiye iletebilmesi için bir kamera ile yapmış olduğu işaretleri algılayıp işaret dilini yazıya dönüştürmeyi başararak tek yönlü iletişimi sağlamıştık.

Şimdi yapmış olduğumuz Final projesinde ise iki yönlü iletişimi sağlamak için Konuşma engeli olan kişilerin %85-90 ‘ının aynı zamanda sağır da olduklarından hareketle yola çıkarak: Bir vatandaşın işaret dilini kullanan bir başka kişiyle iletişiminde kullanacağı sesi yazıya dönüştüren programı geliştirmiş olacağız.

# **PYGAME**

Pygame, Pete Shinners tarafından Python programlama dilinde etkileşimli oyunlar hazırlamak için SDL kütüphanesi üstünde kurulmuş bir Python kütüphanesidir.

Pygame ‘in en önemli özelliklerinden birisi, pek çok platformda (Microsoft Windows, Linux, Mac OS) çalışmasıdır. Pygame, kullanıcılara birçok medya türünün desteklendiği (.jpg, .gif, .png, .bitmap) bir arayüz sunar. Klavye ve fare için pek çok girdi fonksiyonuyla arayüzlerle etkileşimi kolaylaştırır. Pygame'de joystick çalıştırmak dahi mümkündür. .mp3, .wav ve .midi ses formatlarını destekler. Tkinter gibi yaygın bir grafiksel kullanıcı arayüzü aracından farklı olarak, görüntü döngüsü kullanıcı tarafından belirlenir. Bu özellik, geliştiricilerin oyun üzerindeki hakimiyetinin artmasını sağlar.

# **GOOGLE SPEECH RECOGNITION**

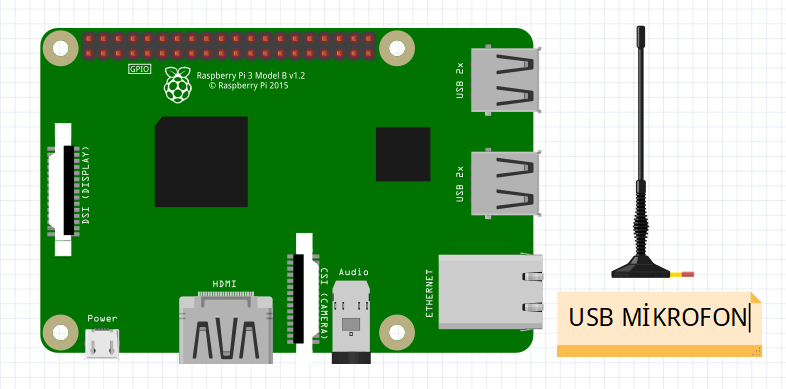
Konuşma metinleştirici, sesli konuşmayı yazıya çeviren bilgisayar yazılımlarına verilen genel addır.

Google Cloud Speech API, kullanımı kolay API ile güçlü sinir ağı modelleri uygulayarak geliştiricilerin sesi metne dönüştürmesini sağlar. API, global kullanıcı tabanınızı desteklemek için 80'den fazla dil ve lehçeyi tanır. Uygulama mikrofonuna konuşan kullanıcıların söylediklerini yazıya dökebilir

# **Donanım Alt Yapısı**

Projede kullanılan elemanlar aşağıda ki gibidir:

1. Raspberry Pi 3 Model B v1.2
2. Display
3. Klavye ve Fare
4. Mikrofon



**Devrenin Fritzing ile Modellenmesi.PNG**

# **Yazılım Alt Yapısı**

Bu projenin yazılım alt yapısı;

* Ses\_tanimla.py

Ve 1 klasörden;

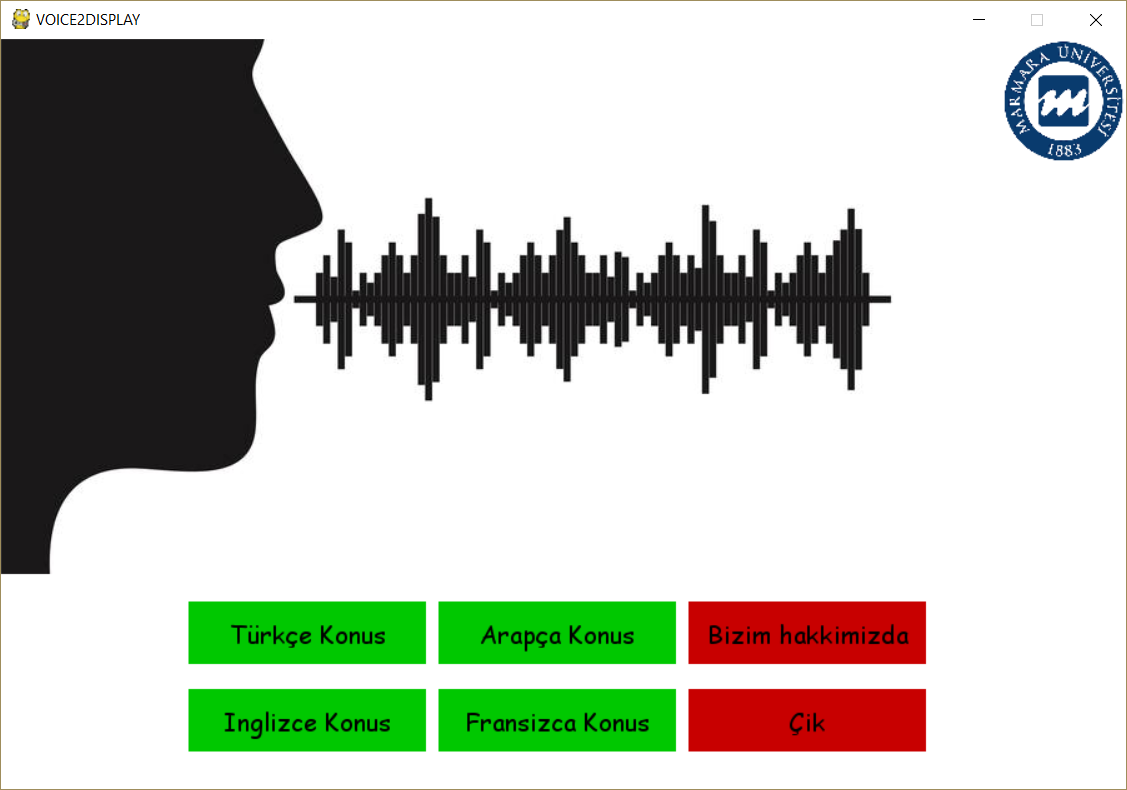
* Final Project

Oluşmaktadır.

## **Ses Tanımla Kod Dosyasının Oluşturulması**

**Ses\_tanimla.py** isimli kod dosyasında **pygame** modülünü kullanarak kullanıcı için bir arayüz oluşturduk. Arayüz içinde ki butonlar kullanarak 4 farklı dilde ses tanımlama işlemini yaptırdık.

Ses tanımlama işlemi Google tarafından geliştirilen **Speech Recognition API** ‘si kullanarak yaptık.



* Gerekli modülleri kod dosyamıza import edelim.

1. import pygame
2. import pyaudio
3. import time
4. import speech\_recognition **as** sr

* Kod dosyamıza başlayalım.

1. pygame.init() #pygame modülünü başlatıyoruz.
2. ekran\_genisligi = 900 #arayüz ekran genişliği
3. ekran\_yuksekligi = 600  #arayüz ekran yüksekliği

* Arayüz içinde kullanılacak renkleri **RGB** değerlerini tanımlayalım.

1. siyah = (0, 0, 0)
2. beyaz = (255, 255, 255)
3. kirmizi = (200, 0, 0)
4. yesil = (0, 200, 0)
5. acik\_kirmizi = (255, 0, 0)
6. acik\_yesil = (0, 255, 0)
7. gameDisplay = pygame.display.set\_mode((ekran\_genisligi, ekran\_yuksekligi))

**pygame.display.set\_mode**() fonksiyonuna parametre olarak verilen genişlik ve yükseklik değerline göre bir pencere oluşturuldu ve **gameDisplay** değişkenine atandı.

1. pygame.display.set\_caption('VOICE2DISPLAY')

**pygame.display.set\_caption()** fonksiyonu kullanılarak oluşturulan pencereye VOICE2DISPLAY ismi verildi.

1. gameDisplay.fill(beyaz) #Tüm çerçeveyi beyaz yapar.
2. img1 = pygame.image.load('img1.jpg')  #Pencereye img1 resmini ekler.
3. img1 = pygame.transform.scale(img1, (100, 100)

**pygame.transform.scale()** fonksiyonu, yüklenen fotoğrafın boyutlandırılmasını sağlar.

1. gameDisplay.blit(img1, (800, 0)) #Eklenen img1 fotoğrafının pencerede kaydırılarak konumlamasının yapılmasını sağlar.
2. img2 = pygame.image.load('img2.jpg')  #Pencereye img2 resmini ekler.
3. gameDisplay.blit(img2, (-70, -100))  #Eklenen img2 fotoğrafının pencerede kaydırılarak konumlamasının yapılmasını sağlar.

* Pencere üzerinde ki “**Çık”** butonuna basınca gerçekleştirilecek olan Close () fonksiyonu:

1. **def** **close**():
2. pygame.quit() #pygame modülünü sonlandırır.
3. **quit**()  #Pencereyi kapatır.

* Pencerede gösterilecek olan metni yazdıracak olan **message\_display ()** fonksiyonu:

1. **def** **message\_display**(text):
2. largeText = pygame.font.Font('freesansbold.ttf', 30)
3. TextSurf, TextRect = text\_objects(text, largeText)
4. TextRect.center = ((500), (350))
5. gameDisplay.blit(TextSurf, TextRect)
6. pygame.display.update()

largeText = pygame.font.Font('freesansbold.ttf', 30)

Pencerede gösterilecek mesaj için kullanılacak yazının karakteri ve boyutunu tanımlar.

TextSurf, TextRect = text\_objects(text, largeText)

text\_objects() fonksiyonuna parametre olarak ekrana yazdırılacak olan metin ve metnin formatı verilir. Bu fonksiyonun return ettiği metin alanı ve metin çerçevesi TextSurf ve TextRect değişkenlerine atanır.

TextRect.center = ((500), (350))

Metin çerçevesinin merkezini pencere üzerinde konumlandırılmasını sağlar.

gameDisplay.blit(TextSurf, TextRect)

Oluşturduğumuz metni (TextSurf) çerçeveye (TextRect) eklenir.

pygame.display.update()

Pencere güncellenir.

* Arayüze metin yazmak için kullanılacak fonksiyonu tanımlayalım.

1. **def** **text\_objects**(text, font):
2. textSurface = font.render(text, **True**, siyah)
3. return textSurface, textSurface.get\_rect()

text\_objects() fonksiyonu iki parametre alır. Birinci parametresi ekrana yazdırılacak olan metindir. İkinci parametresi ise metnin formatıdır.

Bu fonksiyonun return edeceği değişkenler ise metin alanı (textSurface) ve metin çerçevesi (textRect) olacaktır.

* Arayüzde ki butonlara ait fonksiyonları tanımlayalım.

1. **def** **button**(msg, x, y, w, h, ic, ac, action = **None**):
2. mouse = pygame.mouse.get\_pos()
3. click = pygame.mouse.get\_pressed()
4. if x + w > mouse[0] > x **and**y + h > mouse[1] > y:
5. pygame.draw.rect(gameDisplay, ac, (x, y, w, h))
6. if click[0] == 1 **and** action != **None**:
7. action()
8. else :
9. pygame.draw.rect(gameDisplay, ic, (x, y, w, h))
10. smallText = pygame.font.SysFont("comicsansms", 20)
11. textSurf, textRect = text\_objects(msg, smallText)
12. textRect.center = ((x + (w / 2)), (y + (h / 2)))
13. gameDisplay.blit(textSurf, textRect)

**button**(msg, x, y, w, h, ic, ac, action = **None**):

button fonksiyonun parametreleri;

msg: buton üzerinde ki metin,

x,y: butonun sol üst noktasının koordinatları,

w,h: butonun genişliği ve yüksekliği,

ic,ac: mouse butonun üzerindeyken ve mouse üzerinde değilken oluşan buton rengini ifade eder,

action: butona basınca gerçekleştirilecek olan olay (fonksiyon).

mouse = pygame.mouse.get\_pos()

mouse kursörünün bulunduğu noktanın koordinatları [x,y] şeklinde mouse değişkenine atanır.

click = pygame.mouse.get\_pressed()

mouse un basılı olup olmadığı kontrol edilir ve True yada False değeri click değişkenine atanır.

if x + w > mouse[0] > x **and**y + h > mouse[1] > y:

pygame.draw.rect(gameDisplay, ac, (x, y, w, h))

Mouse, buton alanında ise buton rengi açık renkli olacağını belirtir ve butonun sol üst noktası (x,y) koordinatlarında olacak şekilde buton çizdirilir.

if click[0] == 1 **and** action != **None**:

action()

Buton basılı ise ve gerçekleştirilecek olay None değilse, action() fonk. çalıştırılır.

else :

pygame.draw.rect(gameDisplay, ic, (x, y, w, h))

Mouse, buton alanında değil ise buton rengi koyu renkte olacağını belirtir ve butonun sol üst noktası (x,y) koordinatlarında olacak şekilde buton çizdirilir.

smallText = pygame.font.SysFont("comicsansms", 20)

textSurf, textRect = text\_objects(msg, smallText)

textRect.center = ((x + (w / 2)), (y + (h / 2)))

gameDisplay.blit(textSurf, textRect)

smallText, buton üzerine yazdırılacak metnin karakter tipi ve boyutunu belirtir.

textSurf, buton üzerinde ki metnin çerçevesini ve alanını belirtir.

textRect, metin çerçevesinin merkezinin konumunu belirtir.

gameDisplay.blit() fonksiyonu, metin alanını (TextSurf) çerçeveye (TextRect) ekler.

* Ses tanımlamak için kullanılacak fonksiyonları tanımlayalım.
* İlk olarak Türkçe ile başlayalım.

1. **def** **s2t\_tr**():
2. gameDisplay.blit(img1, (800, 0))
3. r = sr.Recognizer()
4. with sr.Microphone() **as** source:
5. **print**('Bir sey soyleyin!')
6. audio = r.listen(source)
7. **print**('okundu!')
8. text = r.recognize\_google(audio, language = 'tr-TR')
9. **print**(text)
10. message\_display(text)

r = sr.Recognizer()  #r isminde Speech\_recognition tipinde bir obje tanımladık.

with sr.Microphone() **as** source:

**print**('Bir sey soyleyin!')

audio = r.listen(source)

**print**('okundu!')

Dinleme işlemi gerçekleştiriliyor. Dinleme işlemini gerçekleştirdikten sonra audio değişkenine kaydediyor.

text = r.recognize\_google(audio, language = 'tr-TR')

**print**(text)

message\_display(text)

Dinlenilen ses (audio) tanımlanıp text değişkenine atanır. Daha sonra bu text değişkeni görüntülenmesi için hem komut satırına hem de arayüze yazdırılır.

Diğer diller içinde aynı işlemler gerçekleştirilecek fakat,

r.recognize\_google() fonksiyonunun language parametresi hangi için tanımlama yapılacaksa ona göre düzenlenmelidir.

Fransızca için: language = 'fr-FR'

Arapça için: language = 'ar-UE'

İngilizce için: language = 'en-US'

* Arayüzde ki bilgi butonuna basıldığında çalışacak olan fonksiyonu tanımlayalım.

1. **def** **info**():
2. **print**(""" Bu proje Fedi ve Mehmet tarafindan gömüllü sistemler  dersi finalödevi olarak tasarlamis """)
3. time.sleep(1)

time.sleep(1) : mouse a tıkladığımızda birden çok kez tıklamış oluruz ve böylece mesaj birkaç kez gösterilmiş olur. Bunu önlemek için tıkladıktan sonra 1 saniyelik bekleme süresi konulmuştur.

* Şimdi ana fonksiyonumuzu tanımlayalım.

1. **def** main():
2. while True:
3. for event in pygame.event.get():
4. if event.type == pygame.QUIT:
5. pygame.quit()
6. **quit**()
7. button("Türkçe Konus!", 150, 450, 190, 50,
8. yesil, acik\_yesil, s2t\_tr)
9. button("Inglizce Konus!", 150, 520, 190, 50,
10. yesil, acik\_yesil, s2t\_en)
11. button("Arapça Konus!", 350, 450, 190, 50,
12. yesil, acik\_yesil, s2t\_ar)
13. button("Fransizce Konus!", 350, 520, 190, 50,
14. yesil, acik\_yesil, s2t\_fr)
15. button("Bizim hakkimizda", 550, 450, 190, 50,
16. kirmizi, acik\_kirmizi, info)
17. button("Çik", 550, 520, 190, 50, kirmizi, acik\_kirmizi, close)
18. pygame.display.update()

main() fonksiyonu içerisinde, button() fonksiyonunu çağırarak farklı dillere ait butonlar eklenir. Aynı zamanda bizim hakkımızda ve programdan çıkmak için Çik butonu da eklenmiştir.

# **KAYNAKÇA**

* 1. <https://tr.wikipedia.org/wiki/Pygame>

# **PROJENİN GİTHUB ADRESİ**

<https://github.com/mmtaksuu/voice2display.git>

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| https://ci4.googleusercontent.com/proxy/za3R5VMRl7xRmYDEOhCbX2B1XmO81OSjamkEgnS4fOvaEaLM0raiFBNExv1vvzHW5l9caNlhisqzILG5fh_oOuyGUmbuDwwQhXGCbTHWUzs3yZOXHqnEty2ukj03Xns=s0-d-e1-ft#https://mysignature.io/images/photos/ecdd0ea1dc6f71e6dbe371f45bc74080.png | |  | | --- | | **Mehmet AKSU** | | Mechatronics Engineer Student | Maker | | **M:**[05347356989](tel:05347356989) | | **E:**[mmtaksu.25@gmail.com](mailto:mmtaksu.25@gmail.com) | | **A:**Kadıköy / İstanbul | | [www.mechatronictech.com](http://www.mechatronictech.com/) | | |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | | |

