BLM4021 Gömülü Sistemler
Laboratuvar Projesi Final Raporu

Grup No: 21

Proje Kategorisi: UYG1

Kişilerin Çalışma Yüzdesi:

Grup Sorumlusu:	**Burak Erdilli**	**%25**	**Rapor, Malzemelerin tespiti, Yazılmış kodun testi ve hataların düzeltimesi**
	Özkan Özeşme	%25	**Konu seçimi, Malzemelerin tespiti, Kodun hazırlanması ve testi, Devreye yardım**
	Mehmet Celal Keleş	%25	**Konu seçimi, Malzemelerin tespiti, Hazırlanmış kodun testi, Devrenin hazırlanması**
	Alperen Furkan Akkurt	%25	**Rapor, Kodun hazırlanmasına yardım, Devrenin hazırlanmasına yardım**

İçerik

Bu bölüme dokunmayınız. Sayfalar hiç kaymayacak ve ilgili bölüm hep aynı sayfada başlayacaktır. Kullanmadığınız sayfaları boş geçiniz.

$$\sum 18 SAYFA$$

I. Giriş ve Proje Tanıtımı

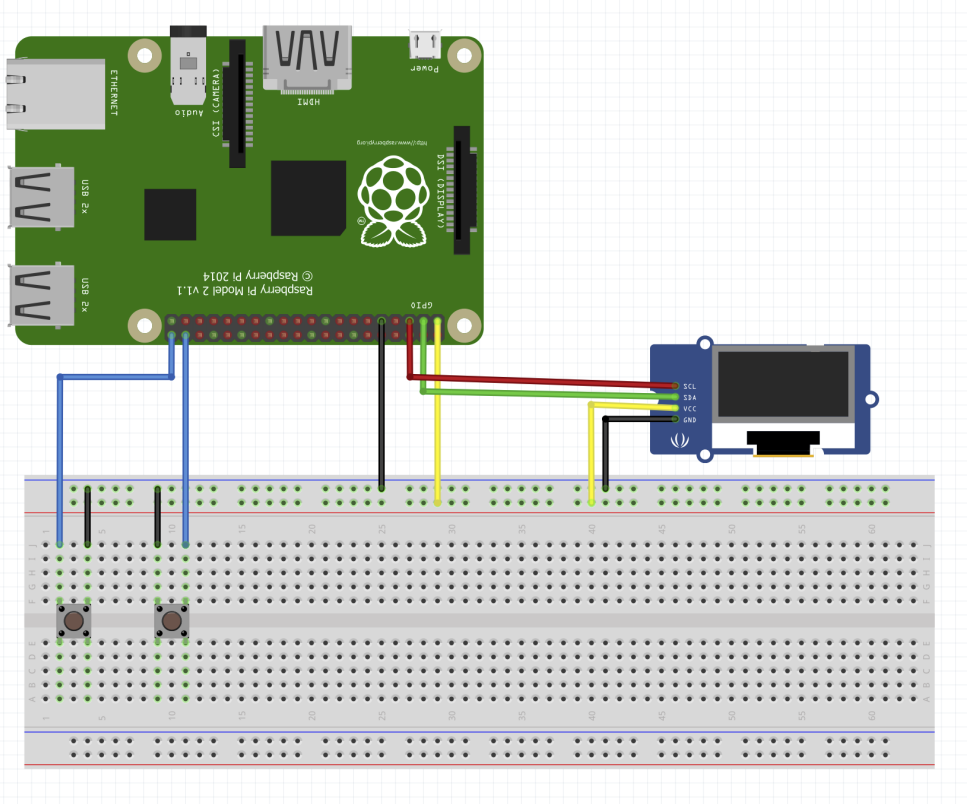
Proje, insan göz kırpmalarını Mors kodu olarak tanıyabilen ve karşılık gelen harfleri çıkarabilen bir sistem geliştirmek için bir Raspberry Pi 2 ve Python kullanacaktır.

Genel olarak, bu proje Raspberry Pi 2 ve Python kullanımını birleştirerek insan göz kırpmalarını tanımak ve Mors alfabesi harflerine çevirmek için benzersiz ve yenilikçi bir sistem oluşturmaktadır.

İnsan göz kırpmalarını Mors kodu olarak tanımlanması, kullanıcının yüzünün kamera yardımıyla çekilmesini gerektirmektedir. Python kodu daha sonra kameradan gelen görüntüyü analiz eder ve göz kırpmalarını tanımlar. Göz kırpmaları tanımlandıktan sonra Python kodu, Mors kodu çeviri, Mors kodu tablosunu kullanarak bunları uygun harflere dönüştürülür. Bunun için kullanıcıdan ek olarak butonlara basması istenmektedir. Bu sayede kullanıcı girdiği çizgi ve noktaları istediği zaman harfe dönüştürebilmekte, gerekirse de hatalı girdiği çizgi veya noktaları silebilmektedir. Elde edilen harfler ve kullanıcının girdiği çizgi ve noktalar daha sonra bir ekrana verilmektedir.

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-

II. *Fritzing* ile Ön Tasarım



Şekil 1: Fritzing devre şeması

Bu yapay görme projesi için devre, Raspberry Pi 2, bir kamera modülü, düğmeler, jumper kabloları ve bir ekran dahil olmak üzere çeşitli bileşenlerden oluşur.

Raspberry pi bir micro usb şarj cihazı ile çalışır, ekran gücünü Raspberry pi'nin 3.3 V pininden alır, 2 ve 3 pinleri de ekrana bağlanır. Kamera modülü USB üzerinden bağlanır. Düğmeler 20 ve 21 numaralı pinler aracılığıyla bağlanır. Ayrıca düğmeler ve ekran pi'nin GND pinine breadboard üzerinden bağlıdır.

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-

III. Kurulan Devre Detayları

Kullanılan malzemeler: 0.96 inch I2C oled ekran, büyük boy breadboard, 40pin ayrılabilen M-F ve M-M jumper kablo 200mm, 2 adet 4 pinli push button, usb kamera.

0.96 inch I2C oled ekran, 128x64 çözünürlükte olup mavi renkte görüntü vermektedir. 4 adet pini bulunmaktadır. Bunlar sırasıyla: GND, VCC, SCL, SDA'dır. Bu pinleri raspberry pi'deki 9(GND), 1(3V) , 5(GPIO3 - I2C SCL) ve 3(GPIO 2 - I2C SDA) pinlerine takılmıştır. Takıldıktan sonra doğru bağlanıp bağlanmadığını kontrol etmek için Adafruit kütüphanelerini kurup örnekler çalıştırılmış ve doğru bağlantı sağlandığı anlaşılmıştır. Adafruit kütüphanesinin dışında kendimizin nasıl yazı ya da resim gösterebileceğimizi test edebilmek için oled_text kütüphanesini de araştırmamız gerekti. Bu kütüphane ile resim yerine direkt olarak istediğimiz satıra yazı ekleyip daha yüksek performans elde edebildiğimiz anlaşılmıştır.

4 pinli push butonların ikisinin de bir ayağını ground amacıyla breadboard'umuz üzerinden raspberry pi'ye bağladığımız GND hattına bağladık. Butonların diğer ayaklarını sırası ile raspberry pi'deki 38 (GPIO20) ve 40 (GPIO21) portlarına bağlayarak pull up olması amacıyla ayarlamalarını yaptık.

Usb kamera olarak 2megapixel full hd kamera kullanılmıştır. kameramız 60 hertz görüntü alabilmesine rağmen raspberry pi'deki görüntü işlemesini de hesaba kattığımızda çözünürlüğü düşürmemiz gerekmekteydi. Çözünürlüğü düşürdüğümüze raspberry pi'den birim zamanda alınan frame sayısında ciddi artış gözlenmiştir. Eğer çözünürlük düşürme yapılmasaydı her bir göz kırpmanın algılanmasını için 1 saniyeden daha uzun süre gözün kapalı kalması gerekiyordu ki bu da projenin kullanışlılığı açısından elverişli değildi.

-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-O-

IV. Yazılım Tasarımı

```
GPIO.setup(21, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(20, GPIO.IN, pull_up_down=GPIO.PUD_UP)
def test_callback(channel):
    if (len(chars) > 0):
        chars.pop()
        oled.text("".join(chars), 1)
def test_callback_2(channel):
    chars.append(" ")
    oled.text("".join(chars), 1)
GPIO.add_event_detect(21, GPIO.FALLING, callback=test_callback)
GPIO.add_event_detect(20, GPIO.FALLING, callback=test_callback_2)
i2c = busio.I2C(SCL, SDA)
```

GPIO modülü Raspberry Pi üzerinde 21 ve 22 numaralı iki giriş pini kurar. Bu yapılandırma GPIO.PUD_UP giriş pinini varsayılan olarak yüksek ayarlar ve bir düğmeye veya anahtara basıldığında düşük çekilebilir.

Ardından, char boş değilse son öğeyi kaldıran test_callback() ve char'a bir boşluk karakteri ekleyen test_callback_2() işlevleri tanımlanır ve bunlar düşen kenar durumunda geri arama işlevleri olarak çağrılır.

```
def calculate_distance(x1, y1, x2, y2):
    return np.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
```

calculate_distance fonksiyonu, iki boyutlu Kartezyen düzlemde iki noktanın koordinatları olan x1, y1, x2 ve y2 olmak üzere dört bağımsız değişken alır. Bu iki nokta arasındaki Öklid mesafesini hesaplar ve döndürür.

```
def calculate_aspect_ratio(landmarks):
    x1, y1 = landmarks.part(36).x, landmarks.part(36).y
    x2, y2 = landmarks.part(39).x, landmarks.part(39).y
    left_eye_x = calculate_distance(x1, y1, x2, y2)
    x1, y1 = landmarks.part(42).x, landmarks.part(42).y
    x2, y2 = landmarks.part(45).x, landmarks.part(45).y
    right_eye_x = calculate_distance(x1, y1, x2, y2)
```

```
left_eye_y = (calculate_distance(landmarks.part(37).x,
              landmarks.part(37).y, landmarks.part(41).x,
              landmarks.part(41).y) + calculate_distance(landmarks.part(38).x,
              landmarks.part(38).y,landmarks.part(40).x,landmarks.part(40).y)) / 2
right_eye_y = (calculate_distance(landmarks.part(43).x, landmarks.part(43).y,
              landmarks.part(47).x,landmarks.part(47).y) + calculate_distance(
              landmarks.part(44).x,landmarks.part(44).y,landmarks.part(46).x,
              landmarks.part(46).y)) / 2
```

calculate_aspect_ratio fonksiyonu, yüzdeki işaretler hakkında bilgi içeren bir nesne olduğu varsayılan işaretler adlı bir argüman alır. Fonksiyon, önce her bir gözün iki dış köşesi arasındaki yatay mesafeyi (x-koordinat farkı) ve her bir gözün iki iç köşesi arasındaki dikey mesafeyi (y-koordinat farkı) hesaplayarak sol ve sağ gözlerin en-boy oranını hesaplar. Her bir gözün en-boy oranı daha sonra yatay mesafenin dikey mesafeye bölünmesiyle hesaplanır. Son olarak, fonksiyon sol ve sağ gözlerin en-boy oranlarının ortalamasını döndürür.

```
morse_code = {...}
```

Bu kod, alfabenin harflerini, 0-9 rakamlarını ve boşluğu karşılık gelen Mors kodu karakterlerine eşleyen morse_code adlı bir dictionary tabimladik.

```
cap = cv2.VideoCapture(0)
cap.set(3, 160)
cap.set(4, 120)
```

Buradaki İlk satırda kullanılacak kamerayı tanımlıyoruz, ardından VideoCapture nesnesinin set() yöntemini genişlik için 3 ve 160 ve yükseklik için 4 ve 120 argümanlarıyla çağırarak kamera tarafından yakalanan videonun çözünürlüğünü 160x120 piksel olarak ayarlıyoruz.

```
while True:
    ret, frame = cap.read()
    frame = cv2.resize(frame, (120, 90), interpolation=cv2.INTER_CUBIC)
    faces = detector(frame)
    for face in faces:
        if (seenface == 0):
            seenface = 1
            oled.text("detected face", 5)
        landmarks = predictor(frame, face)
        eye_aspect_ratio = calculate_aspect_ratio(landmarks)
        if eye_aspect_ratio > 5.1:
            last_classifications.append(1)
        else:
            last_classifications.append(0)
```

Her görüntü için, kod ilk olarak bir yüz dedektörü nesne dedektörü kullanarak görüntüdeki yüzleri tespit ediyoruz. Ardından, tespit edilen her yüz için, daha önce açıklanan calculate_aspect_ratio() işlevi çağrılır. Gözler kapalıysa, kod last_classifications listesine 1 değerini ekler. Aksi takdirde, 0 değeri eklenir.

```
    if len(last_classifications) > 35:
        if sum(last_classifications[-6:]) == 0:
            last_sum = sum(last_classifications)
            if last_sum > 5:
                chars.append("-")
                oled.text("".join(chars), 1)
            elif last_sum > 2:
                chars.append(".")
                oled.text("".join(chars), 1)
            last_classifications = []
    tempchars = chars.copy()
```

Ardından, kod last_classifications listesinin uzunluğunu kontrol eder. Eğer 35'ten fazla elemanı varsa, kod son altı elemanın toplamını kontrol eder. Toplam 0 ise, last_classifications listesinin tüm öğelerinin toplamı hesaplanır. Bu toplamın değerine bağlı olarak, kod aşağıdaki karakterlerden birini liste karakterlerine ekler:

eğer toplam 2'ten büyükse "."

Toplam 5'ten büyükse "-",

Kod daha sonra last_classifications listesini temizler ve chars listesinin tempchars adında bir kopyasını oluşturur

```
while i < len(tempchars):
    if tempchars[i] == " ":
        thischar = "".join(tempchars[:i])
        i += 1
        for key, value in morse_code.items():
            if (value == thischar):
                # print(key)
                written.append(key)
                break
        tempchars = tempchars[i:]
        i = 0
    else:
        i += 1
```

```
if (oldwritten != written):
    oldwritten = written.copy()
    oled.text("".join(written), 3)  # Line 3
```

Daha sonra tempchars listesini yineler ve her öğe için morse_code sözlüğündeki değerlerle karşılaştırır. Bir eşleşme bulunursa, sözlükteki ilgili anahtar yazılı listeye eklenir. Son olarak, kod yazılı listeyi çerçeve üzerinde görüntüler ve çerçevede tespit edilen yüz işaretlerini daire içine alır.

```
if (not faces):
    if (seenface == 1):
        seenface = 0
        oled.text("no face detected", 5)
```

Eğer bir yüz algılanmazsa Ekrana "yüz algılanmadı" yazar.

```
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

kullanıcı "q" harfine basarak loop dan çıkabilir.

Raspberry pi’ye bağlanıp projeyi çalıştırmak için ssh ve vnc kullanılmıştır. Projenin test kısımlarında kameradan gelen görüntüyü görmek ve görüntü işlemenin gözümüzü ne zaman algıladığını anlamak için vnc üzerinden bağlanmamız gerekti. Projenin test aşamaları bittiğinde ise vnc ile bağlanmaya ihtiyaç kalmadığı için sadece ssh kullanılmaya devam edilmiştir.

Vnc ile bağlamak için instructables.com[1] üzerinden yardım alınmıştır.

V. Sonuçlar, Demo Detayları ve Sunum Linki

Yazdığımız kodu kendi bilgisayarımızda test ederken işlem gücümüz daha yüksek olduğu için kameradan saniyede alınan frame sayısı daha yüksekti. Aynı kodu raspberry üzerinde çalıştırdığımız zaman beklediğimiz üzere saniyedeki frame sayısı düştü. Bu durumda kodumuzun daha düzgün çalışması için kameradan alınan verinin çözünürlüğünü düşürerek işleyebildiğimiz frame sayısını artırmayı hedefledik ve bu hedefimize başarıyla ulaştık. Çözünürlüğü düşürdüğümüz için ise kameraya daha düzgün bir şekilde yüzümüzü okutmamız gerekiyor.

Oled ekrana yazı yazdırmak ve aynı anda görüntü işlemeye devam etmek kodun performansını kısıtlamaya ve yavaş çalışıp göz kırpma süresini düzgün algılayamamasına sebep oluyordu. Bu sebepten dolayı öncelikle ekrana sadece okunan karakteri yazdırmayı düşündük ama bu da yeterli olmadı. Bunun üzerine sadece karakter değişikliklerinde ekrana yazı yazdırmaya karar verdik. Sadece karakter değişikliklerinde ekrana yazı yazdırmanın performansı yeterli görününce göz kırpma süresine göre oluşan nokta ya da çizgileri de ekrana aynı şekilde sadece değişiklik olduğunda bastırmaya karar verdik. Bu iki ekrana yazdırma işlemini optimize bir şekilde yaptığımız için performansta gözle görülür bir düşüş olmadı.

sunum linki: https://youtu.be/ZExXTV8DRgI

-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-

VI. Referanslar

[1] https://www.instructables.com/How-to-Setup-Raspberry-Pi-Without-Monitor-and-Keyb/

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-