



Teknoloji Fakültesi

T.C.
MARMARA ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

POLİS İÇİN GÖRÜNTÜ İŞLEME TABANLI YARDIMCI ASİSTAN

Raporu Hazırlayan:

YAHYA NAJIB ALSHAMERİ

YAHYA ABDULLAH ALAHNOMİ

POLİS İÇİN GÖRÜNTÜ İŞLEME TABANLI YARDIMCI ASİSTAN

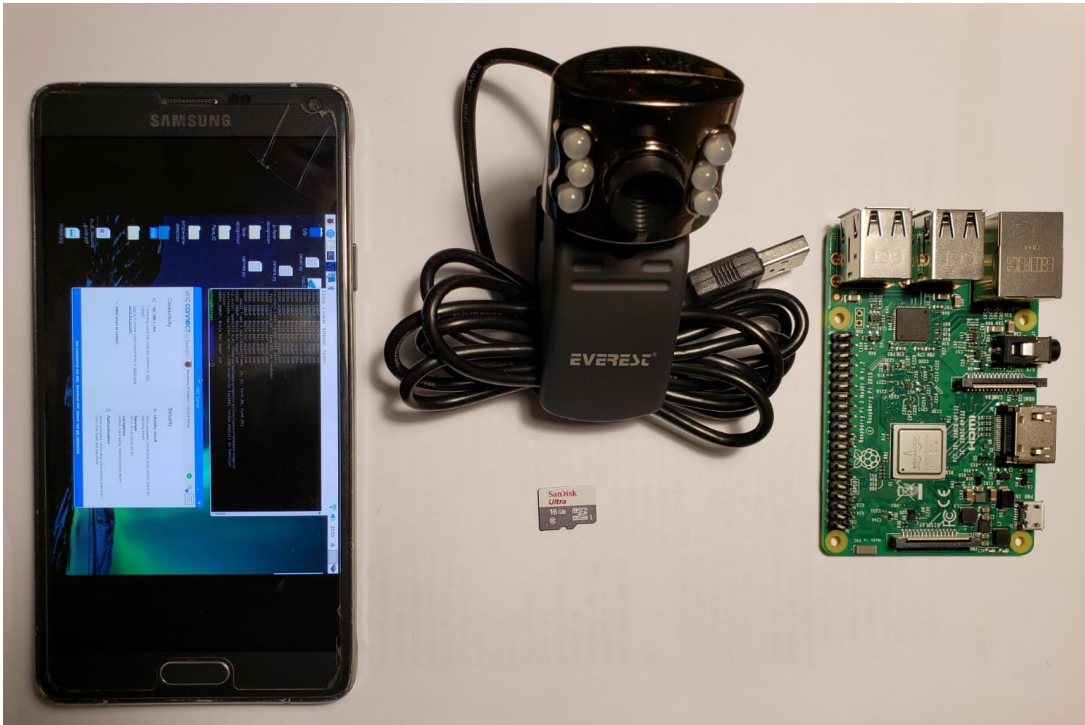
Giriş:

Bu proje polislerin GBT kontrolleri sırasında daha hızlı işlem yapabilmeleri için görüntü işleme yardımıyla bu sorunu çözebilmek. Görüntü işleme için kullanılacak olan kamera raspberry Pi ile bağlı olup alınan görüntülerin burada işlenip sonuç elde ederek polise gerekli bilgileri aktarabilecektir.

Projede raspberry e bağlı bir kamera olacaktır. Kameranın kullanım yeri ise toplu taşıma araçları, yoğun caddeler ve güvenliğin önem kazandığı herhangi bir yerde kullanılmak üzere amaçlanabilir. Bu kameralar insan yüzlerine odaklanıp buradaki görüntüyü bağlı olduğu raspberryye aktaracaktır. Bu aktarım cadde sokaklarda güvenlik kameralarından da olabilir GBT kontrolü yapacak olan herhangi bir polis üzerindeki mini kameradan da olabilir. Raspberrynin veri tabanında olan belli başlı isimleri Raspberryye aktarılan görüntülerle karşılaştırılıp bir suç teşkil edip etmediği tespit edilir. Sonuçlanan bu bilgi polise aktarılır.

Gerekli Donanım Bileşenleri:

1. 1 adet Rasperry pi Model B .
2. 1 adet WebCamera.
3. 1 adet bellek kartı.
4. Her hangi bir telefondaki VNC uygulamasıyla rasperry`i kontrol etmektedir.



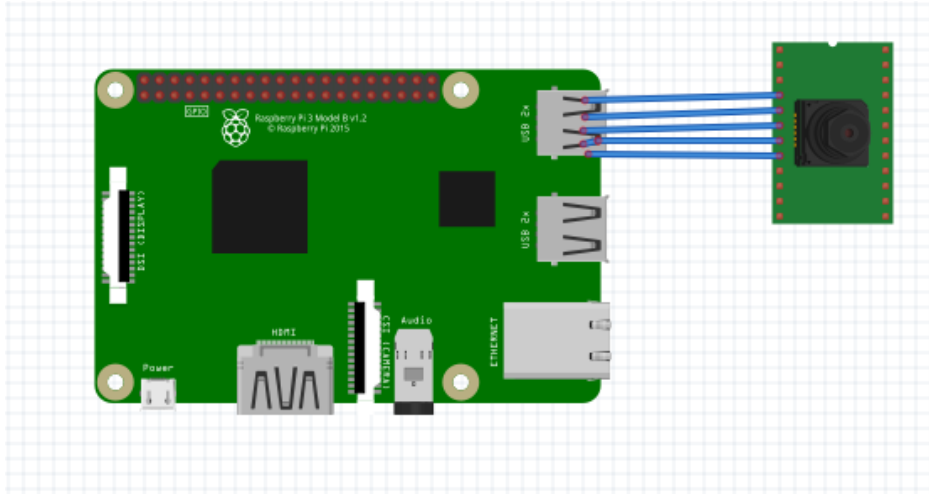
Gerekli Yazılım Bileşenleri

1. Python(<https://www.python.org/>)
2. openCV (<https://opencv.org/>)
3. Tensorflow (<https://www.tensorflow.org/>)

Kullanılan Bileşenlerin Özellikleri:

1. Rasperry pi 3;[Raspberry Pi](#), kredi kartı büyüklüğünde bir bilgisayardır.Günümüze kadar birçok modeli ortaya çıkmıştır. Bizim bu projede kullandığımız model B dir. Ürünü şuradaki adresten temin edebilirsiniz. (<https://www.direnc.net/>)
2. Web Camera; Bu ürün şuradan temin ettik vs. (www.urunadi.com/webcamera.htm)

Şematik Çizimi:



Resim 1 : Şematik Çizimi

Yapım Aşamaları:

Raspbian kurulumu :

- Raspbian kurulumu için <https://www.raspberrypi.org/downloads/raspbian/> adresini ziyaret ediyoruz.
- İndirdiğimiz raspbian dağıtımını bir yere açıyoruz.
- Daha sonra [buradan](#) sağ kısımdan Win32DiskImager'ı (binary) indiriyoruz
- Açılan sayfadan imajı seçip sağ kısımdan SD Kartın bölümünü seçiyoruz ve write'a basıyoruz (Not: Kartın içi sıfırlanıyor).
- Bir süre bekledikten sonra karta yükleme tamamlanıyor, ondan sonra kartı sökebiliriz.
- **Güç kaynağı** olarak 5Volt 1 Amper istemekte Raspberry Pi.

OpenCV kurulumu:

OpenCV'yi <http://opencv.org/downloads.html> adresine giriyoruz ve indirmek istediğimiz sürümün altındaki **OpenCVfor Windows** linkine tıklıyoruz.İndirme bağlantısı sourceforge.net sitesine yönlendirecek ve indirme işlemi başlayacak.İndirdiğinizde sıkıştırılmış olarak gelecektir, çalıştırdığınızda OpenCV dosyalarını çıkartmak için bir dizin isteyecektir, burada çıkartılmasını istediğiniz dosya dizini yolunu yazarak **Extract** butonuna tıklayın. Dosyaları çıkarttığınız dizinde OpenCV klasörü içerisinde **build** ve **sources** diye 2 adet klasör bulunmaktadır. Build klasörü içerisinde Windows platformu için derlenmiş olarak sistem native kütüphaneler ve programlama dilleri için kütüphaneler bulunmaktadır. Sources klasöründe ise OpenCV kaynak kodları ve örnek uygulamalar yer almaktadır. Buradaki kaynak kodlar ile OpenCV'yi tekrardan derleyebilirsiniz.Java ile OpenCV uygulaması geliştirmek için **build** içerisindeki **java** klasöründe yer alan jar dosyasını ve kullanacağımız işletim sistemi mimarisine göre OpenCV Windows sistem kütüphanelerini kullanacağız.

Tensorflow kurulumu:

Deeplearning hizmeti sunar.

Tensorflow Windows'ta 2 yükleme yöntemi sunuyor. Birincisi **pip** ile yükleme, ikincisi **Anaconda** içinde yükleme.

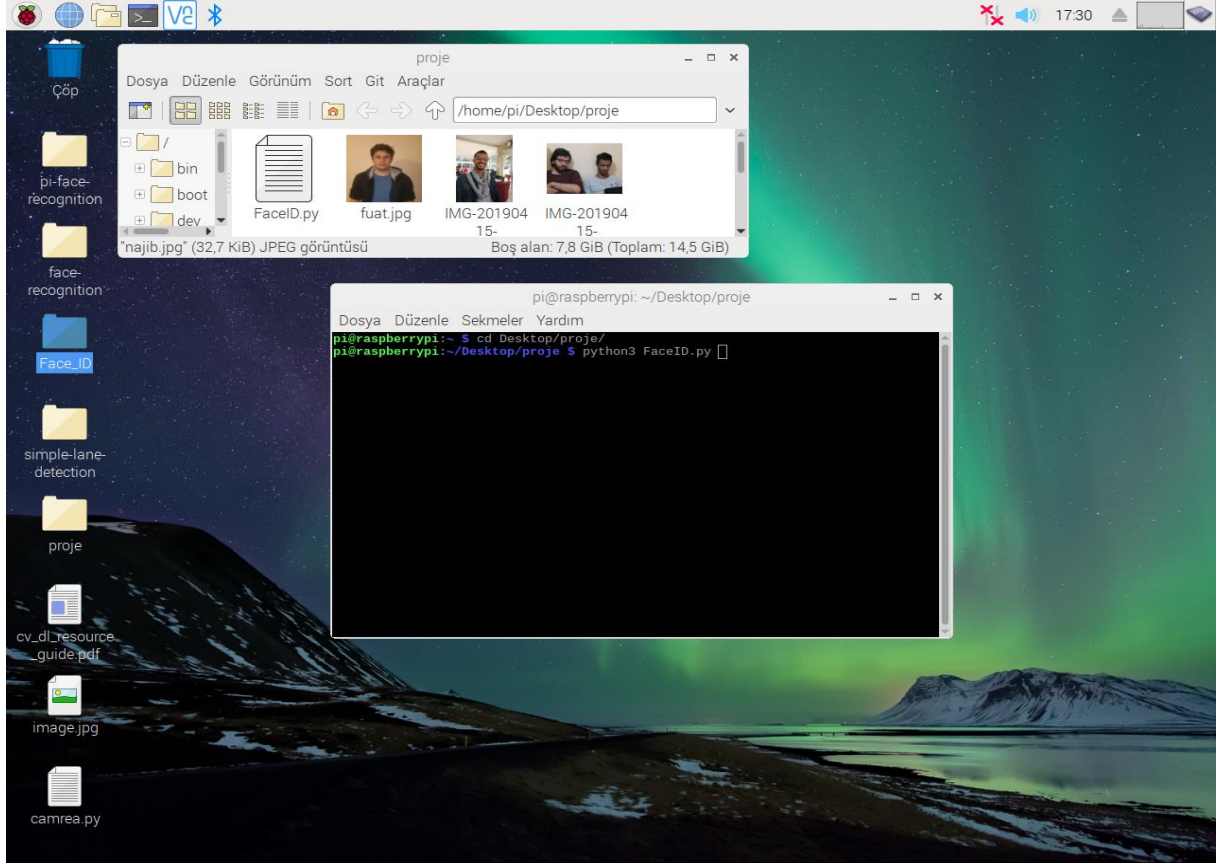
Tensorflowpip ile yüklendiğinde sanal bir ortam oluşturulmadan yüklenir, bu nedenle diğer Python modülleriyle karışıp diğer işlerinizin çalışmasını engelleyebilirler. Bu yüzden Anaconda ile yüklemek daha güvenli bir yöntem.

Ancak, Tensorflow Windows'ta şu an için Python 3.6 versiyonunu desteklemiyor. Bu yüzden Tensorflow'da sanal bir ortam oluşturmak, bunu da **Python 3.5** versiyonuyla yapmak gerekiyor.

NVIDIA ekran kartı varsa Tensorflow'unGPU'lu versiyonu yüklenebilir, yoksa sadece CPU üzerinde çalışan versiyon yüklenebilir.

Yapım Aşamaları:

- 1- Aşağıdaki **cd Desktop/proje2/** komut yardımıyla projenin yeri ulaştırır . Sonra **python3 FaceID.py** komutuyla Proje çalıştırmaktadır.



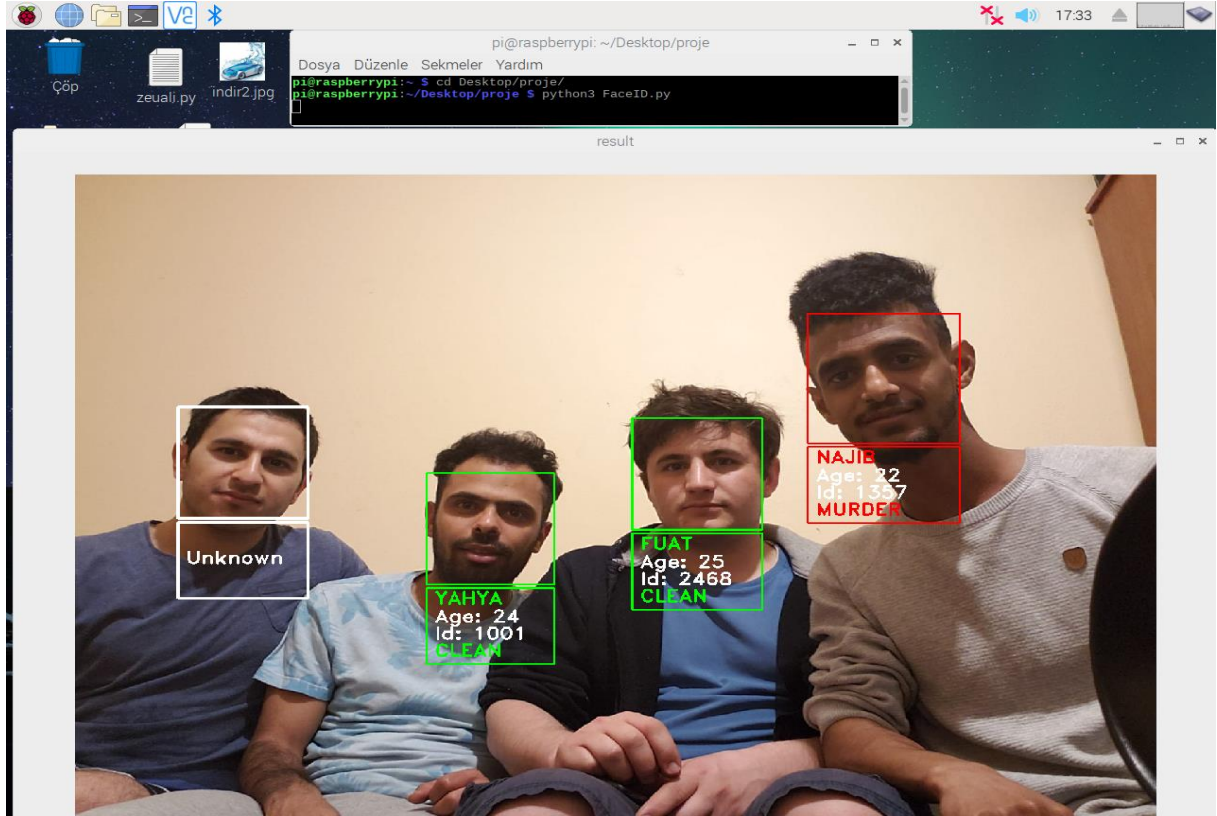
2- Aşağıdaki gibi Proje çalıştığı zaman kamera çalışıyor ve aşağıdaki görüntü gibi bir sonuç elde edilmektedir.

Farklı renk kullanarak kişilerin durumu tanımlandık:

Kırmızı ise suçlu kişidir.

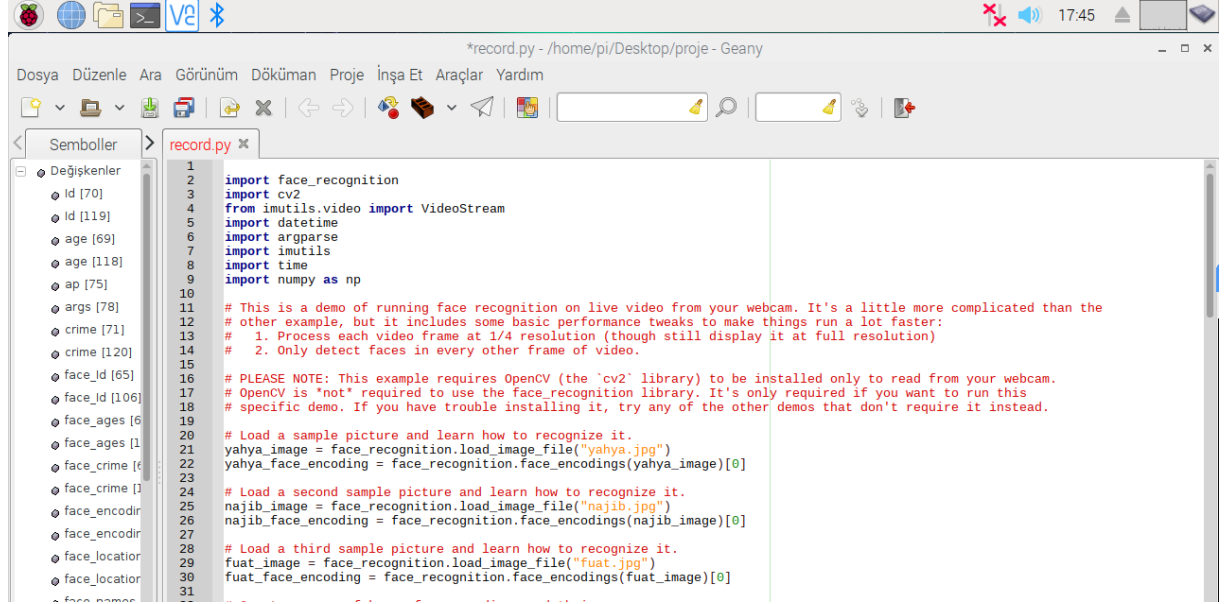
Yeşil ise temiz kişi demek.

Beyaz ise kayıtlı olmayan kişi demektir.



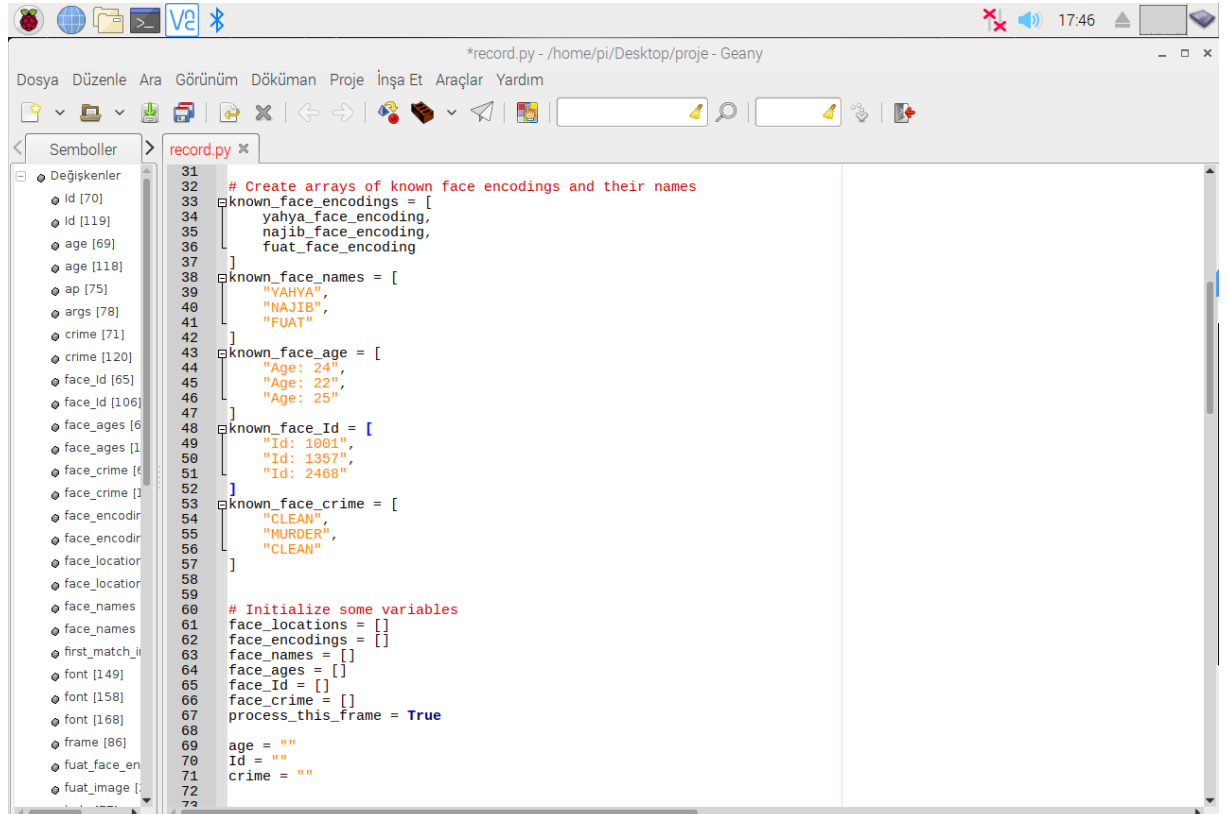
Projenin Kodları:

Kodların ilk kısmında kullanılan kütüphaneleri çağırdık .



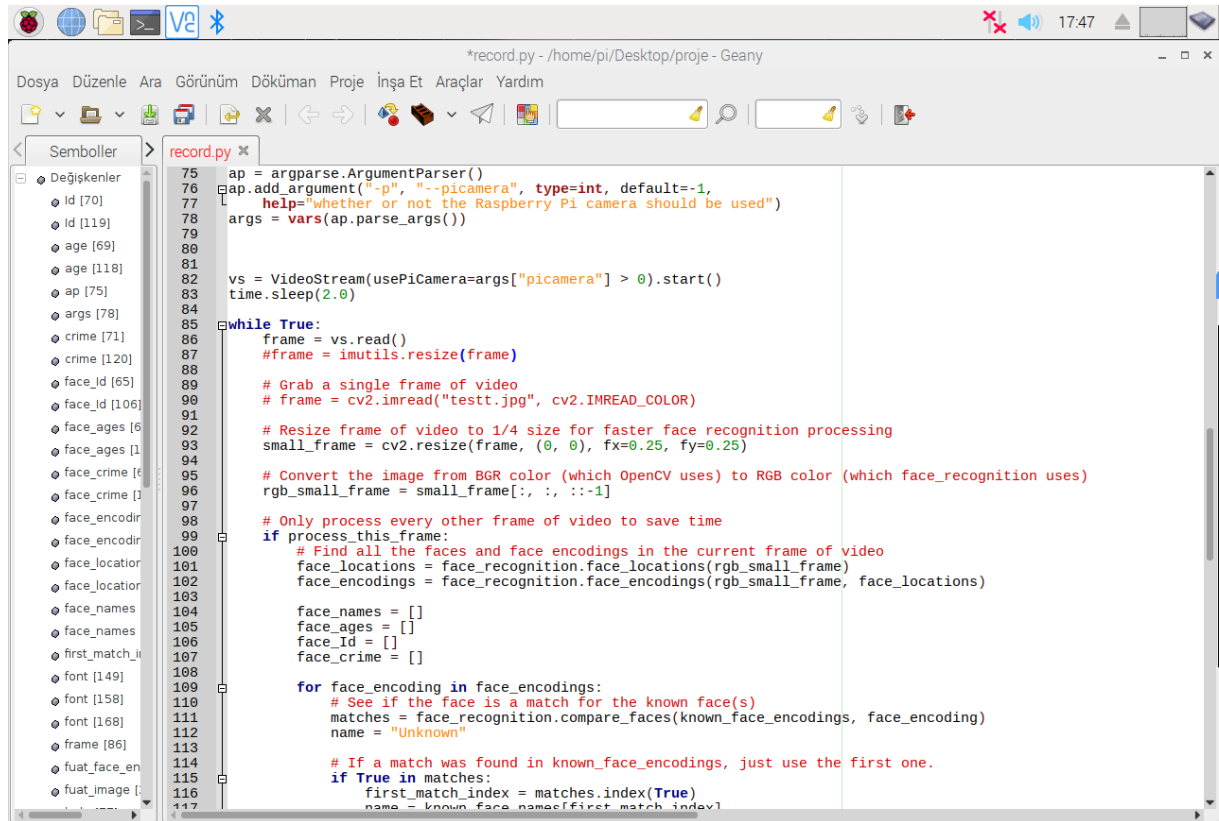
```
1 import face_recognition
2 import cv2
3 from imutils.video import VideoStream
4 import datetime
5 import argparse
6 import time
7 import numpy as np
8
9 # This is a demo of running face recognition on live video from your webcam. It's a little more complicated than the
10 # other example, but it includes some basic performance tweaks to make things run a lot faster:
11 # 1. Process each video frame at 1/4 resolution (though still display it at full resolution)
12 # 2. Only detect faces in every other frame of video.
13
14 # PLEASE NOTE: This example requires OpenCV (the `cv2` library) to be installed only to read from your webcam.
15 # OpenCV is 'not' required to use the face_recognition library. It's only required if you want to run this
16 # specific demo. If you have trouble installing it, try any of the other demos that don't require it instead.
17
18 # Load a sample picture and learn how to recognize it.
19 yahya_image = face_recognition.load_image_file("yahya.jpg")
20 yahya_face_encoding = face_recognition.face_encodings(yahya_image)[0]
21
22 # Load a second sample picture and learn how to recognize it.
23 najib_image = face_recognition.load_image_file("najib.jpg")
24 najib_face_encoding = face_recognition.face_encodings(najib_image)[0]
25
26 # Load a third sample picture and learn how to recognize it.
27 fuat_image = face_recognition.load_image_file("fuat.jpg")
28 fuat_face_encoding = face_recognition.face_encodings(fuat_image)[0]
```

Kodların ikinci kısmında GBT genel bilgiler tabanına kişilerinin bilgilerini kaydedilmektedir.

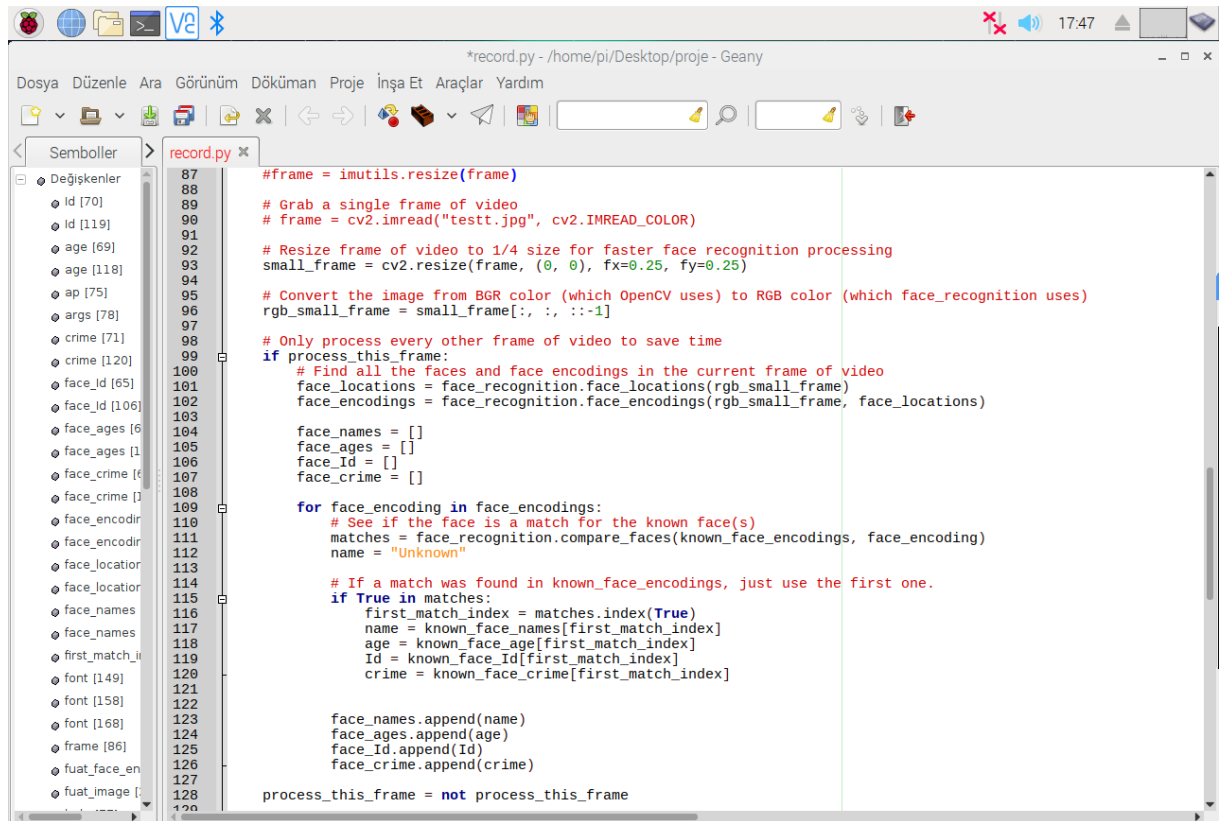


```
31 # Create arrays of known face encodings and their names
32 known_face_encodings = [
33     yahya_face_encoding,
34     najib_face_encoding,
35     fuat_face_encoding
36 ]
37 known_face_names = [
38     "YAHYA",
39     "NAJIB",
40     "FUAT"
41 ]
42
43 known_face_age = [
44     "Age: 24",
45     "Age: 22",
46     "Age: 25"
47 ]
48 known_face_Id = [
49     "Id: 1001",
50     "Id: 1357",
51     "Id: 2468"
52 ]
53 known_face_crime = [
54     "CLEAN",
55     "MURDER",
56     "CLEAN"
57 ]
58
59 # Initialize some variables
60 face_locations = []
61 face_encodings = []
62 face_names = []
63 face_ages = []
64 face_Id = []
65 face_crime = []
66 process_this_frame = True
67
68 age = ""
69 Id = ""
70 crime = ""
```


Kodların sonraki kısmında Görüntü işlemi ile yüz tanıma işlemi yapılacaktır.

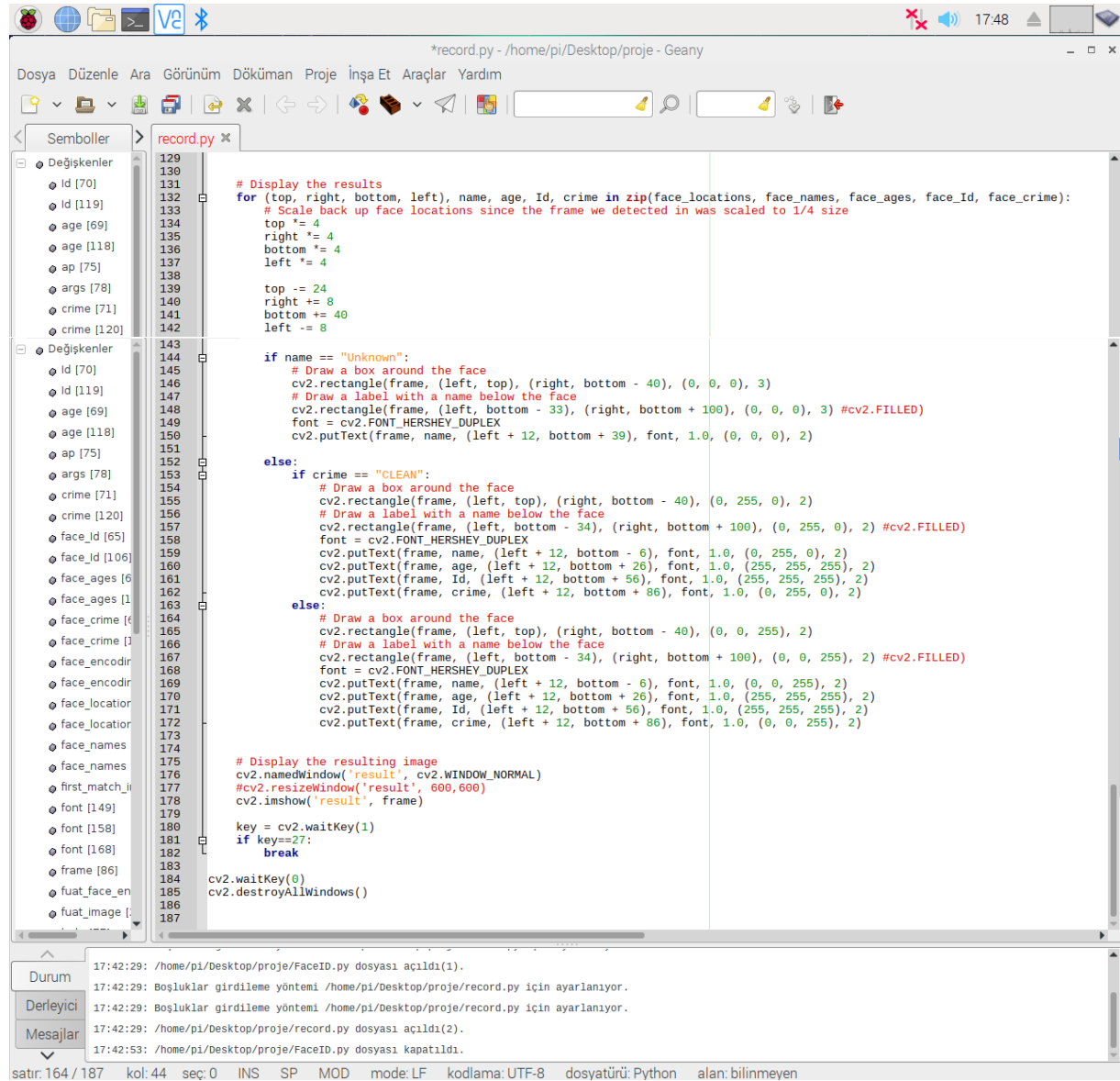


```
75 ap = argparse.ArgumentParser()
76 ap.add_argument("-p", "--picamera", type=int, default=-1,
77 help="whether or not the Raspberry Pi camera should be used")
78 args = vars(ap.parse_args())
79
80
81 vs = VideoStream(usePiCamera=args["picamera"] > 0).start()
82 time.sleep(2.0)
83
84
85 while True:
86     frame = vs.read()
87     #frame = imutils.resize(frame)
88
89     # Grab a single frame of video
90     # frame = cv2.imread("testt.jpg", cv2.IMREAD_COLOR)
91
92     # Resize frame of video to 1/4 size for faster face recognition processing
93     small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
94
95     # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
96     rgb_small_frame = small_frame[:, :, ::-1]
97
98     # Only process every other frame of video to save time
99     if process_this_frame:
100         # Find all the faces and face encodings in the current frame of video
101         face_locations = face_recognition.face_locations(rgb_small_frame)
102         face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
103
104         face_names = []
105         face_ages = []
106         face_id = []
107         face_crime = []
108
109         for face_encoding in face_encodings:
110             # See if the face is a match for the known face(s)
111             matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
112             name = "Unknown"
113
114             # If a match was found in known_face_encodings, just use the first one.
115             if True in matches:
116                 first_match_index = matches.index(True)
117                 name = known_face_names[first_match_index]
```



```
87     #frame = imutils.resize(frame)
88
89     # Grab a single frame of video
90     # frame = cv2.imread("testt.jpg", cv2.IMREAD_COLOR)
91
92     # Resize frame of video to 1/4 size for faster face recognition processing
93     small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
94
95     # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
96     rgb_small_frame = small_frame[:, :, ::-1]
97
98     # Only process every other frame of video to save time
99     if process_this_frame:
100         # Find all the faces and face encodings in the current frame of video
101         face_locations = face_recognition.face_locations(rgb_small_frame)
102         face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
103
104         face_names = []
105         face_ages = []
106         face_id = []
107         face_crime = []
108
109         for face_encoding in face_encodings:
110             # See if the face is a match for the known face(s)
111             matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
112             name = "Unknown"
113
114             # If a match was found in known_face_encodings, just use the first one.
115             if True in matches:
116                 first_match_index = matches.index(True)
117                 name = known_face_names[first_match_index]
118                 age = known_face_age[first_match_index]
119                 Id = known_face_Id[first_match_index]
120                 crime = known_face_crime[first_match_index]
121
122                 face_names.append(name)
123                 face_ages.append(age)
124                 face_id.append(Id)
125                 face_crime.append(crime)
126
127     process_this_frame = not process_this_frame
```


Kodların son kısmında ise asıl görüntü üzerine kişilerin GBT bilgileri yazılacaktır.



```
*record.py - /home/pi/Desktop/proje - Geany
Dosya Düzenle Ara Görünüm Döküman Proje İnşa Et Araçlar Yardım

# Display the results
for (top, right, bottom, left), name, age, Id, crime in zip(face_locations, face_names, face_ages, face_Id, face_crime):
    # Scale back up face locations since the frame we detected in was scaled to 1/4 size
    top *= 4
    right *= 4
    bottom *= 4
    left *= 4

    top -= 24
    right += 8
    bottom += 40
    left -= 8

    if name == "Unknown":
        # Draw a box around the face
        cv2.rectangle(frame, (left, top), (right, bottom - 40), (0, 0, 0), 3)
        # Draw a label with a name below the face
        cv2.rectangle(frame, (left, bottom - 34), (right, bottom + 100), (0, 0, 0), 3) #cv2.FILLED
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, name, (left + 12, bottom + 39), font, 1.0, (0, 0, 0), 2)

    else:
        if crime == "CLEAN":
            # Draw a box around the face
            cv2.rectangle(frame, (left, top), (right, bottom - 40), (0, 255, 0), 2)
            # Draw a label with a name below the face
            cv2.rectangle(frame, (left, bottom - 34), (right, bottom + 100), (0, 255, 0), 2) #cv2.FILLED
            font = cv2.FONT_HERSHEY_DUPLEX
            cv2.putText(frame, name, (left + 12, bottom - 6), font, 1.0, (0, 255, 0), 2)
            cv2.putText(frame, age, (left + 12, bottom + 26), font, 1.0, (255, 255, 255), 2)
            cv2.putText(frame, Id, (left + 12, bottom + 56), font, 1.0, (255, 255, 255), 2)
            cv2.putText(frame, crime, (left + 12, bottom + 86), font, 1.0, (0, 255, 0), 2)

        else:
            # Draw a box around the face
            cv2.rectangle(frame, (left, top), (right, bottom - 40), (0, 0, 255), 2)
            # Draw a label with a name below the face
            cv2.rectangle(frame, (left, bottom - 34), (right, bottom + 100), (0, 0, 255), 2) #cv2.FILLED
            font = cv2.FONT_HERSHEY_DUPLEX
            cv2.putText(frame, name, (left + 12, bottom - 6), font, 1.0, (0, 0, 255), 2)
            cv2.putText(frame, age, (left + 12, bottom + 26), font, 1.0, (255, 255, 255), 2)
            cv2.putText(frame, Id, (left + 12, bottom + 56), font, 1.0, (255, 255, 255), 2)
            cv2.putText(frame, crime, (left + 12, bottom + 86), font, 1.0, (0, 0, 255), 2)

    # Display the resulting image
    cv2.namedWindow('result', cv2.WINDOW_NORMAL)
    #cv2.resizeWindow('result', 600,600)
    cv2.imshow('result', frame)

    key = cv2.waitKey(1)
    if key==27:
        break

cv2.waitKey(0)
cv2.destroyAllWindows()
```

17:42:29: /home/pi/Desktop/proje/FaceID.py dosyası açıldı(1).
17:42:29: Boşluklar girdileme yöntemi /home/pi/Desktop/proje/record.py için ayarlanıyor.
17:42:29: Boşluklar girdileme yöntemi /home/pi/Desktop/proje/record.py için ayarlanıyor.
17:42:29: /home/pi/Desktop/proje/record.py dosyası açıldı(2).
17:42:53: /home/pi/Desktop/proje/FaceID.py dosyası kapatıldı.

satır: 164 / 187 kol: 44 seq: 0 INS SP MOD mode: LF kodlama: UTF-8 dosyatürü: Python alan: bilinmeyen

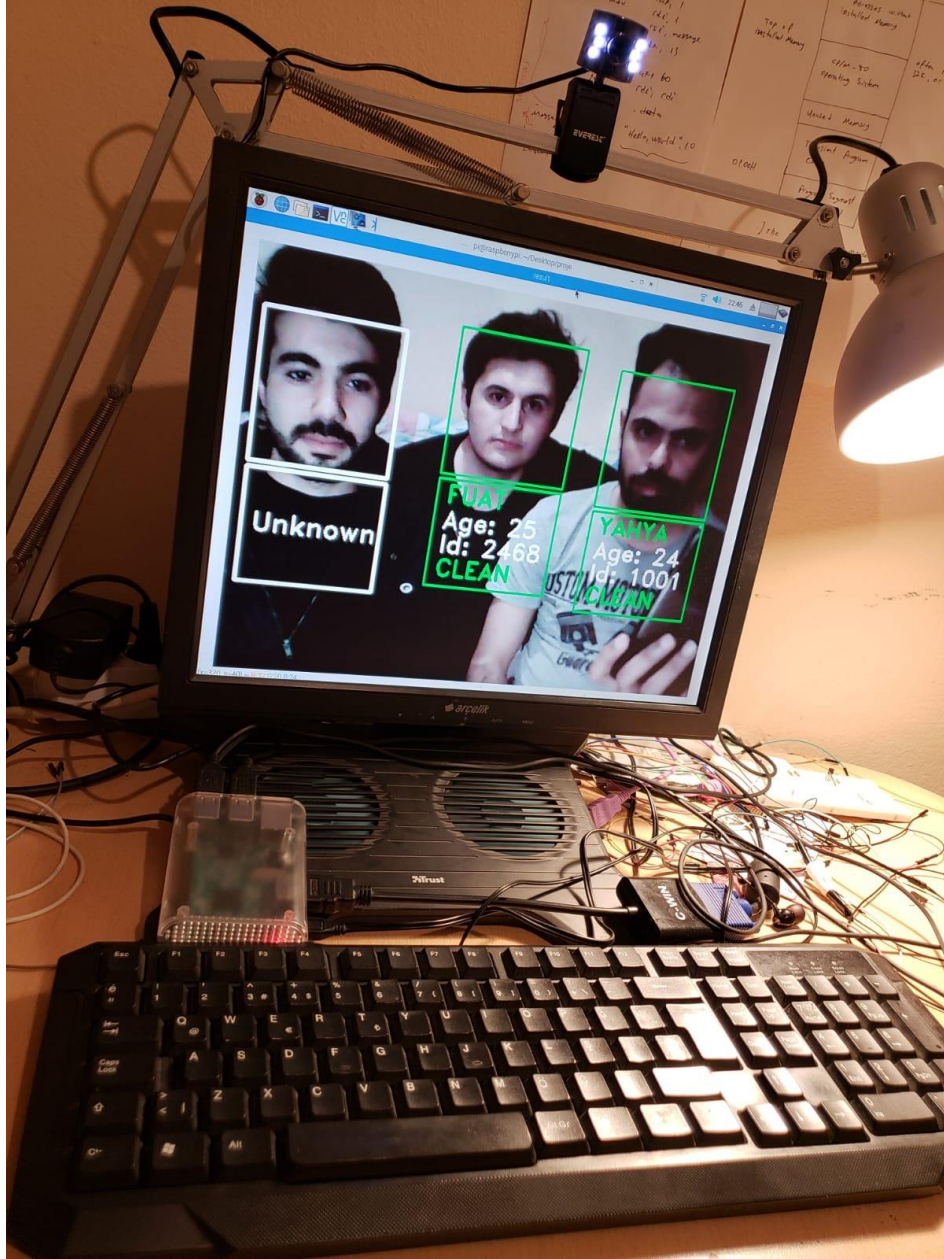
Kaynak Kodu :

Buradaki proje resimlerine, videolarına (kisa bir video koyunuz) ve kaynak koduna <https://github.com/hyuice> adresinden erişilebilir.

Nasıl Kullanılır:

Bu projenin çalıştırılması oldukça basittir. İlk olarak raspberry bağlı kamera ile görüntü elde edilir. Kamera ile alınan bu görüntülerraspberrynin veri tabanındaki görüntülerle belli bir algoritmaya göre karşılaştırılır ve bir sonuç elde edilir. Bu sonuç kullanıcıya bir arayüz aracılığıyla sunulur.

Proje Resimi :



Öneriler:

Projemizde kameranın aldığı görüntüler raspberryyesunulmaktadır bunun bir adım ötdesi olarak fotoğraf yerine video şeklinde olan canlı görüntüler raspberryye aktarılır böylece çok daha fazla insan taranmış olur

Video Linki:

YoutubeVideo Kaydını <https://youtu.be/afcbnwXuEPM> adresinden görüntüleyebilirsiniz.