11.IMAGE PROCESSING

```
import cv2     //I used cv2 library


image=cv2.imread("image.png")
//I added “image.png” in file for showing on the screen that is named as “frame”.
cv2.imshow("frame",image)
cv2.waitKey(5000)
cv2.destroyAllWindows()
```

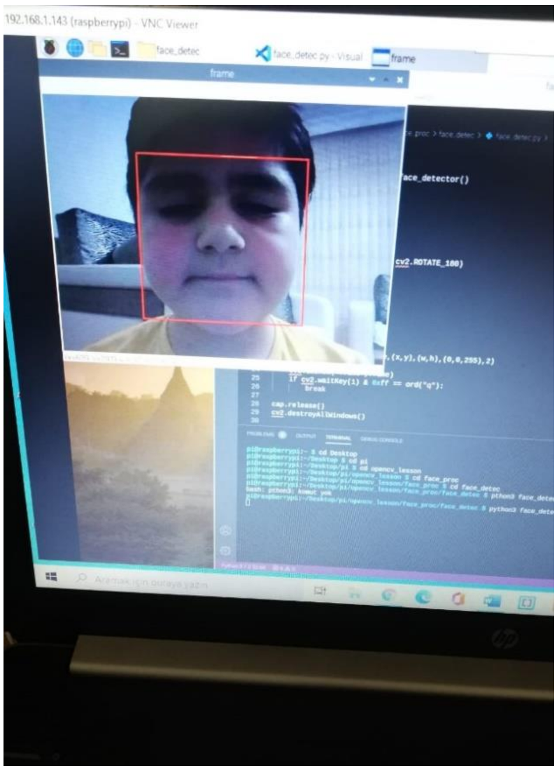
11.1. Reading Images From The Camera

```
 cap=cv2.VideoCapture(0)
//0 in here indicates the id of the camera installed on the module.
//If we had attached another camera with usb, we would number it as 1 2 3.
while true:
    _,frame=cap.read()
//We read the image from the camera and transfer it to the frame. If the sign of _; is whether the camera is turned
on.
if cv2.waitKey(1) & 0xff==ord("q"):
//// so when q is clicked, close the window.
    break
cap.release()
cv2.destroyAllWindows()
```

11.2. Face Recognition And Matrix Converting Process

```
import cv2
import dlib
//dlib is machine language library.

detector=dlib.get_frontal_face_detector()

cap = cv2.VideoCapture(0)
```

```
while True:
    _,frame = cap.read()
    faces=detector(frame)
    print(faces)

    cv2.imshow("frame",frame)
    if cv2.waitKey(1) & 0xff == ord("q"):
        break

cap.release()
cv2.destroyAllWindows()
```



11.3.face_detec.py

```
import cv2
import dlib

detector = dlib.get_frontal_face_detector()
cap = cv2.VideoCapture(0)

while True:
    _, frame = cap.read()
    frame = cv2.rotate(frame, cv2.ROTATE_180)
    faces = detector(frame)
    for face in faces:
        x = face.left()
    y = face.top()
    w = face.right()
    h = face.bottom()
    cv2.rectangle(frame, (x, y), (w, h), (0, 0, 255), 2)
cv2.imshow("frame", frame)
if cv2.waitKey(1) & 0xff == ord("q"):
    break

cap.release()
cv2.destroyAllWindows()
```

11.4. Face_reg.py

```
import cv2
import face_recognition
import dlib

detector = dlib.get_frontal_face_detector()

efe_image = face_recognition.load_image_file("efe.jpg")
efe_image_encoding = face_recognition.face_encodings(efe_image)[0]

cap = cv2.VideoCapture(0)

while True:
    _, frame = cap.read()
    frame = cv2.rotate(frame, cv2.ROTATE_180)
    face_locations = []

    faces = detector(frame)
    for face in faces:
        x = face.left()
        y = face.top()
        w = face.right()
        h = face.bottom()
        face_locations.append((y, w, h, x))

    # faces_locations = face_recognition.face_locations(frame)

    faces_encodings = face_recognition.face_encodings(frame, face_locations)
    i = 0
for face in faces_encodigs:
    y, w, h, x = face_locations[i]
    i += 1
    result = face_recognition.compare_faces([efe_image_encoding], face)
    if result[0] == True:
        cv2.rectangle(frame, (x, y), (w, h), (0, 0, 255), 2)
        cv2.rectangle(frame, (x, h), (w, h + 30), (0, 0, 255), -1)
        cv2.putText(frame, "efe", (x, h + 25), cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)
    else:
        cv2.rectangle(frame, (x, y), (w, h), (0, 0, 255), 2)
        cv2.rectangle(frame, (x, h), (w, h + 30), (0, 0, 255), -1)
        cv2.putText(frame, "bilinmiyor", (x, h + 25), cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)

cv2.imshow("frame", frame)
if cv2.waitKey(1) & 0xff == ord("q"):
    break

cap.release()
cv2.destroyAllWindows()
```

11.5.facepoint.py // Identify Face Points

```
 import cv2
import dlib

detector = dlib.get_frontal_face_detector()
model = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

cap = cv2.VideoCapture(0)

while True:
    _, frame = cap.read()

frame = cv2.rotate(frame, cv2.ROTATE_180)

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

faces = detector(gray)

for face in faces:
    points = model(gray, face)
    Image
    30

for i in range(0, 68):
    x, y = (points.part(i).x, points.part(i).y)
    cv2.circle(frame, (x, y), 5, (0, 0, 255), -1)
```

```
cv2.imshow("frame", frame)
if cv2.waitKey(1) & 0xff == ord("q"):
    break

cap.release()
cv2.destroyAllWindows()
```

NOTE: The purpose of determining the points on the face; For example, when logging in, foreign persons can also enter by showing the registered persons' faces. For this reason, it would be more correct to confirm the opening of the door, for example after blinking 3 times.

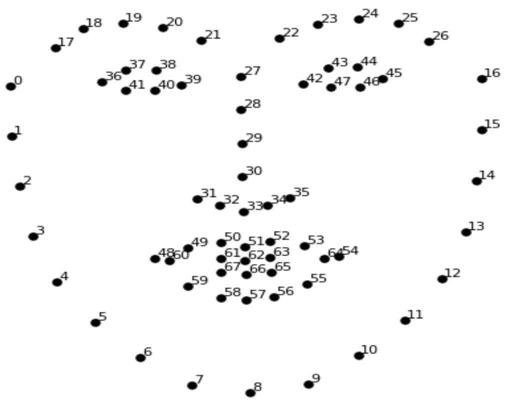


Figure 11.5: Shape predictor 68 face landmarks

11.6. Eye Project

For the blinking process; eye_project.py.

For the eye on the top, we need to find between 37 and 38, for example. Then we need to find between 40-41 for the lower eye. Then we need to measure the distance between these two midpoints. If it is explained step by step in the codes; At first, the code below is written to find the middle point of the eyes.

```
import cv2
import dlib

detector = dlib.get_frontal_face_detector()
model = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

cap = cv2.VideoCapture(0)



def mid(p1, p2):
    return (int((p1[0] + p2[0]) / 2), int((p1[1] + p2[1]) / 2))



while True:
    _, frame = cap.read()

frame = cv2.rotate(frame, cv2.ROTATE_180)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faces = detector(frame)

for face in faces:
    points = model(gray, face)
    points_list = [(p.x, p.y) for p in points.parts()]

    # right eye
    p1_ust_sag, p2_ust_sag = points_list[37], points_list[38]
    p1_alt_sag, p2_alt_sag = points_list[41], points_list[40]

    po_ust_sag = mid(p1_ust_sag, p2_ust_sag)
    po_alt_sag = mid(p1_alt_sag, p2_alt_sag)

    # left eye
    p1_ust_sol, p2_ust_sol = points_list[43], points_list[44]
    p1_alt_sol, p2_alt_sol = points_list[47], points_list[46]

    po_ust_sol = mid(p1_ust_sol, p2_ust_sol)
    po_alt_sol = mid(p1_alt_sol, p2_alt_sol)

    print(po_alt_sag[1] - po_ust_sag[1])
    cv2.circle(frame, (po_alt_sag[0], po_alt_sag[1]), 3, (255, 0, 0),
-1)
cv2.imshow("frame", frame)
if cv2.waitKey(1) & 0xff == ord("q"):
    break

cap.release()
cv2.destroyAllWindows()
```

11.6.a. main.py

At this step, eye distance is measured and recorded in proportion to the nose distance.

A file is created by taking the data from the eyes and stored there.

f = open ("dataset.csv", "a")

0, for closed eye.

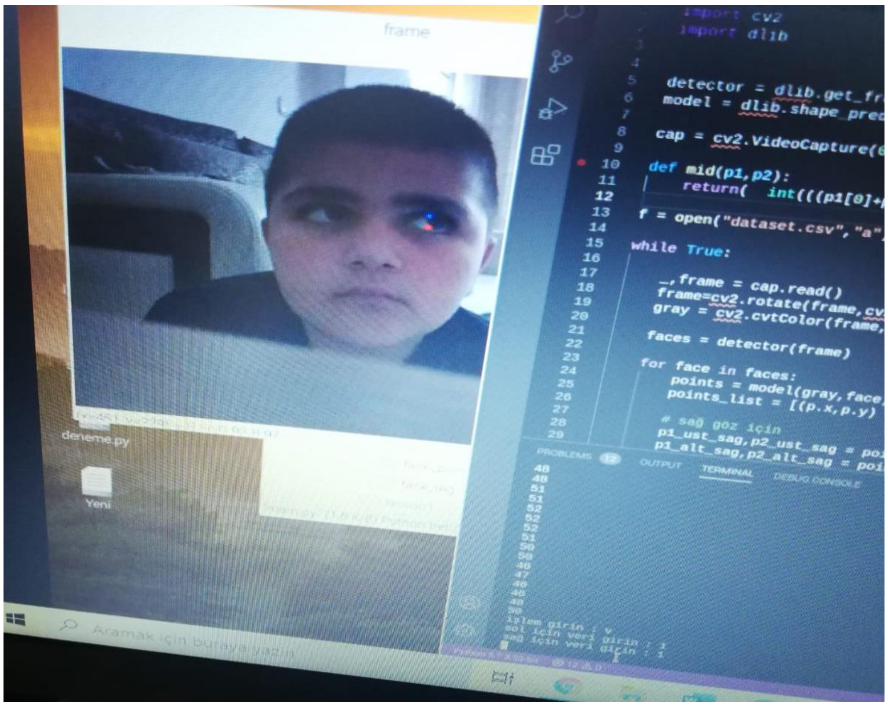
1, for open eye.

```
 import cv2
import dlib

detector = dlib.get_frontal_face_detector()
model = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

cap = cv2.VideoCapture(0)


def mid(p1, p2):
    return (int(((p1[0] + p2[0]) / 2)), int((p1[1] + p2[1]) / 2))


f = open("dataset.csv", "a")

while True:
```

```
_, frame = cap.read()
frame = cv2.rotate(frame, cv2.ROTATE_180)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

faces = detector(frame)

for face in faces:
    points = model(gray, face)
    points_list = [(p.x, p.y) for p in points.parts()]

    # sağ göz için
    p1_ust_sag, p2_ust_sag = points_list[37], points_list[38]
    p1_alt_sag, p2_alt_sag = points_list[41], points_list[40]

    po_ust_sag = mid(p1_ust_sag, p2_ust_sag)
    po_alt_sag = mid(p1_alt_sag, p2_alt_sag)

    sag_mesafe = po_alt_sag[1] - po_ust_sag[1]

    # sol göz için
    p1_ust_sol, p2_ust_sol = points_list[43], points_list[44]
    p1_alt_sol, p2_alt_sol = points_list[47], points_list[46]

    po_ust_sol = mid(p1_ust_sol, p2_ust_sol)
    po_alt_sol = mid(p1_alt_sol, p2_alt_sol)

    sol_mesafe = po_alt_sol[1] - po_ust_sol[1]

    # burun için
    mburun = points_list[30][1] - points_list[27][1]
    print(mburun)

    cv2.circle(frame, (po_ust_sol[0], po_ust_sol[1]), 3, (255, 0, 0), -1)
    cv2.circle(frame, (po_alt_sol[0], po_alt_sol[1]), 3, (0, 0, 255), -1)

cv2.imshow("frame", frame)
if cv2.waitKey(1) & 0xff == ord("q"):
    r = input("işlem girin : ")
    if r == "q":
        break
        elif r == "v":
            sol = input("sol için veri girin : ")
            sag = input("sağ için veri girin : ")
            f.write(f"{sol_mesafe},{sag_mesafe},{mburun},{sol},{sag}\n")

cap.release()
cv2.destroyAllWindows()
```

The numbers written on the screen are the nose distance. If the eye is open according to the nose distance, it gives the distance to the eye in proportion to the nose distance. We store these numbers in the dataset according to the distances. Then we classify it as classification (Figure 11.6.a.1).

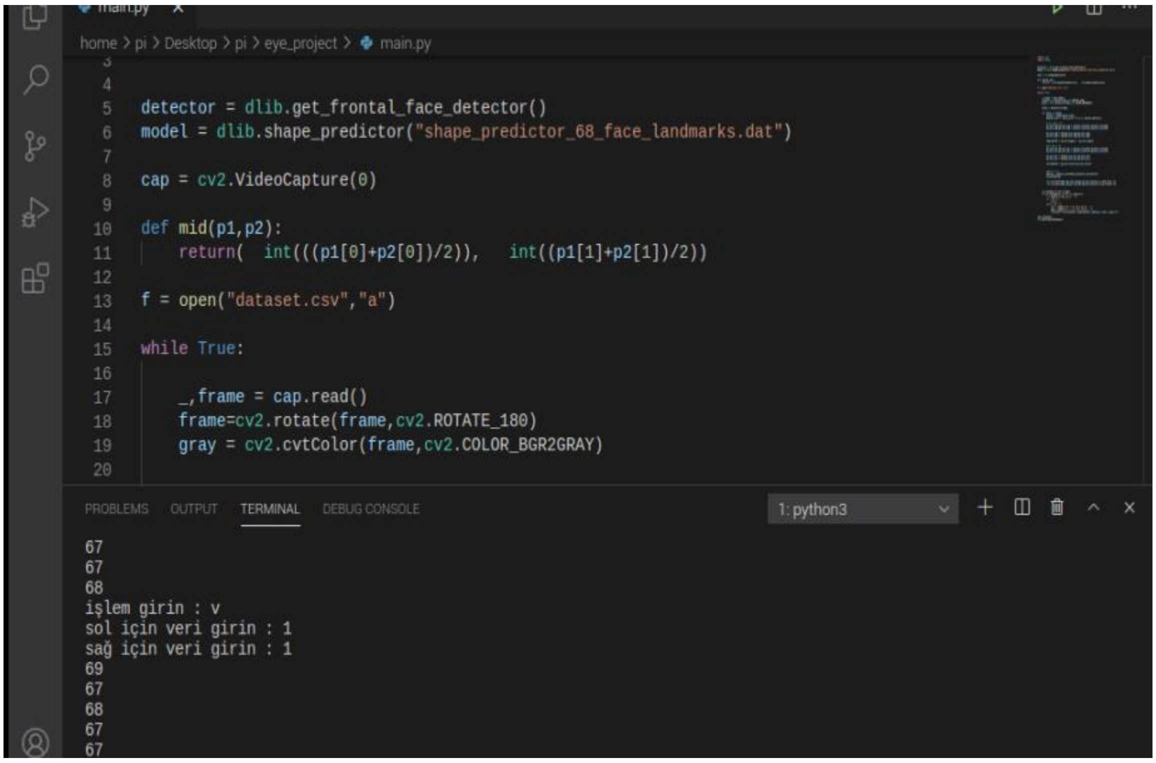
Once we press q, the screen stops and we say v to save. At that time, we indicate whether the eye can be open or not as 1 and 0. Then, when we come to the window, ie flame, and press q, we exit the window (Figure 11.6.a.2).



```
import cv2
import dlib

detector = dlib.get_fr
model = dlib.shape_pred

cap = cv2.VideoCapture(

def mid(p1,p2):
    return(  int(((p1[0]+

f = open("dataset.csv","a"

while True:

    _,frame = cap.read()
    frame=cv2.rotate(frame,cv
    gray = cv2.cvtColor(frame,

    faces = detector(frame)

    for face in faces:
        points = model(gray,face
        points_list = [(p.x,p.y)

        # sağ göz için
        p1_ust_sag,p2_ust_sag = poi
        p1_alt_sag,p2_alt_sag = poi
```

```
48
48
51
51
52
52
52
51
50
50
46
47
46
46
48
50
işlem girin : v
sol için veri girin : 1
sağ için veri girin : 1
```

Figure 11.6.a.1



```
detector = dlib.get_frontal_face_detector()
model = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

cap = cv2.VideoCapture(0)

def mid(p1,p2):
    return(  int(((p1[0]+p2[0])/2)),   int((p1[1]+p2[1])/2))

f = open("dataset.csv","a")

while True:

    _,frame = cap.read()
    frame=cv2.rotate(frame,cv2.ROTATE_180)
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)

```

```
67
67
68
işlem girin : v
sol için veri girin : 1
sağ için veri girin : 1
69
67
68
67
67
```

Figure 11.6.a.2

11.6.b. dataset.csv

Classified as left,right,nose,left_label,right_label. Recorded data samples;

15,16,61,1,1

29,29,119,1,1

22,22,84,1,1

18,18,69,1,1

9,10,68,0,0

11,12,90,0,0

12,12,76,0,0

7,7,62,0,0

17,16,65,1,1

18,19,74,1,1

11,11,77,1,1

12,12,77,0,0

8,8,65,0,0

11.6.c. Classification

Linear regression is preferred for classification..

```
import pandas as pd
from sklearn.linear_model import LinearRegression
```

```
dataset=pd.read_csv("dataset.csv")

x = dataset.iloc[:,:1].values
y = dataset.iloc[:,1:].values

lr=LinearRegression()
lr.fit(x,y)



pred=lr.predict([[20]])

print(pred)
```

In linear regression, we train our dataset with fit. By writing x and y, it is provided with fit to establish a relationship between it and give us predictions.

It is provided to be using “Predict”. For example, given a value of 20 for x, it will predict Y.

11.6.d. Lr_train.py

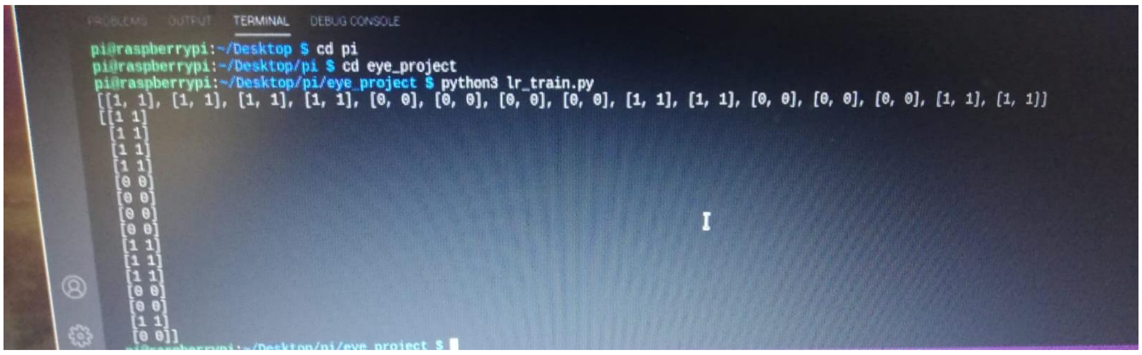
```
 import pandas as pd
from sklearn.linear_model import LinearRegression


dataset = pd.read_csv("dataset.csv")


x = dataset.iloc[:,:3].values
y = dataset.iloc[:,3:].values


lr = LinearRegression()
lr.fit(x,y)
```

```
pred = lr.predict(x)

pred_list = []

for i in pred:
    b = 0
    ik = 0
    if i[0] > 0.5:
        b =1
    else:
        b=0
    if i[1] > 0.5:
        ik=1
    else:
        ik=0
    pred_list.append([b,ik])
print(pred_list)
print(y)
```

It will predict the x's. Values greater than 0.5 are written as 1, and values smaller than 0 are written as 0.

B = first data = left eye

ik = second data = right eye



```
pi@raspberrypi:~/Desktop $ cd pi
pi@raspberrypi:~/Desktop/pi $ cd eye_project
pi@raspberrypi:~/Desktop/pi/eye_project $ python3 lr_train.py
[[1, 1], [1, 1], [1, 1], [1, 1], [0, 0], [0, 0], [0, 0], [0, 0], [1, 1], [1, 1], [0, 0], [0, 0], [0, 0], [1, 1], [1, 1]]
[[1 1]
 [1 1]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]]
pi@raspberrypi:~/Desktop/pi/eye_project $
```

Figure 11.6.d

Here we combine the main file and the lr_train file that we will classify. Eventually all.py is created.

When merging the file, we add the libraries first. Then we add the commands in lr_train so that you can train the dataset firstly. Then we add main.py.

11.6.e. butun.py

In this study, it seems clear that; It reads 1 when the eye is open and 0 when it is closed. (Figure 11.6.e)

```
import cv2
import dlib
import pandas as pd
from sklearn.linear_model import LinearRegression
import time


dataset = pd.read_csv("dataset.csv")


x = dataset.iloc[:,:3].values
y = dataset.iloc[:,3:].values


lr = LinearRegression()
lr.fit(x,y)


detector = dlib.get_frontal_face_detector()
model = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

cap = cv2.VideoCapture(0)
```

```
def mid(p1,p2):
    return(  int(((p1[0]+p2[0])/2)),   int((p1[1]+p2[1])/2))



while True:
    _,frame = cap.read()
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)

    faces = detector(frame)

    for face in faces:
        points = model(gray,face)
        points_list = [(p.x,p.y) for p in points.parts()]

        # sağ göz için
        p1_ust_sag,p2_ust_sag = points_list[37],points_list[38]
        p1_alt_sag,p2_alt_sag = points_list[41],points_list[40]

        po_ust_sag = mid(p1_ust_sag,p2_ust_sag)
        po_alt_sag = mid(p1_alt_sag,p2_alt_sag)

        sag_mesafe = po_alt_sag[1] - po_ust_sag[1]

        #sol göz için
        p1_ust_sol,p2_ust_sol = points_list[43],points_list[44]
        p1_alt_sol,p2_alt_sol = points_list[47],points_list[46]

        po_ust_sol = mid(p1_ust_sol,p2_ust_sol)
        po_alt_sol = mid(p1_alt_sol,p2_alt_sol)

        sol_mesafe = po_alt_sol[1]-po_ust_sol[1]



        #burun için
        mburun = points_list[30][1]-points_list[27][1]



        pred =  lr.predict([[sol_mesafe,sag_mesafe,mburun]])
        pred_list = []
        for i in pred:
            b = 0
            ik = 0
            if i[0] > 0.5:
                b =1
            else:
                b=0
            if i[1] > 0.5:
                ik=1
            else:
                ik=0
            pred_list.append([b,ik])
        print(pred_list)
        if pred_list == []:
            print("boş")
        else:
            print("boş değil")
            cv2.putText(frame,str(pred_list[0][0]),po_alt_sol,cv2.FONT_HERSHEY_COMPLEX,2,(0,0,255),1)
            cv2.putText(frame,str(pred_list[0][1]),po_alt_sag,cv2.FONT_HERSHEY_COMPLEX,2,(0,0,255),1)



    cv2.imshow("frame",frame)
    if cv2.waitKey(1) & 0xff == ord("q"):
        break



cap.release()
cv2.destroyAllWindows()
```

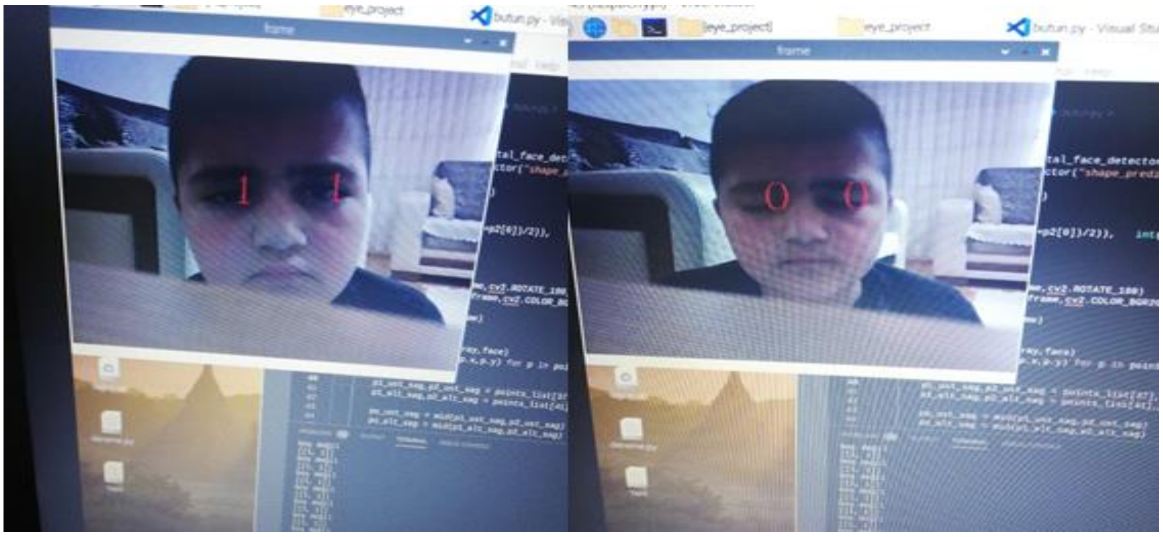


Figure 11.6.e

12.CODES OF PROJECT

12.1. main.py

```
import pandas as pd
import cv2
import dlib
import face_recognition
from sklearn.linear_model import LinearRegression

kubra_image = face_recognition.load_image_file("kubra.jpg")
kubra_face_encoding = face_recognition.face_encodings(kubra_image)[0]

efe_image = face_recognition.load_image_file("efe.jpg")
efe_face_encoding = face_recognition.face_encodings(efe_image)[0]

biden_image = face_recognition.load_image_file("biden.jpg")
biden_face_encoding = face_recognition.face_encodings(biden_image)[0]

known_face_encodings = [
    kubra_face_encoding,
    efe_face_encoding,
    biden_face_encoding
]
known_face_names = [
    "kubra",
    "efe",
    "Joe Biden"
]

detector = dlib.get_frontal_face_detector()
model = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

dataset = pd.read_csv("dataset.csv")

x = dataset.iloc[:, :3].values
y = dataset.iloc[:, 3:].values

lr = LinearRegression()
lr.fit(x, y)



def mid(p1, p2):
    return (int((p1[0] + p2[0]) / 2), int((p1[1] + p2[1]) / 2))



cap = cv2.VideoCapture(0)
say = 0
while True:
    _, frame = cap.read()
    frame = cv2.rotate(frame, cv2.ROTATE_180)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    face_locations = []
    # yuz tanma için istenilen format ve hızlandırma
    faces = detector(frame)
    for face in faces:
        x = face.left()
        y = face.top()
        w = face.right()
        h = face.bottom()
        face_locations.append((y, w, h, x))

    # göz durumu
    for face in faces:
        points = model(gray, face)
        point_list = [(p.x, p.y) for p in points.parts()]

        # sag goz için
        sag_ust = mid(point_list[37], point_list[38])
        sag_alt = mid(point_list[41], point_list[40])
        # göz görselleştirme
        cv2.circle(frame, sag_ust, 3, (0, 0, 255), -1)
        cv2.circle(frame, sag_alt, 3, (0, 0, 255), -1)
        sag_goz_mesafe = sag_alt[1] - sag_ust[1]

        # sol göz için
        sol_ust = mid(point_list[43], point_list[44])
        sol_alt = mid(point_list[47], point_list[46])
        # göz görselleştirme
        cv2.circle(frame, sol_ust, 3, (0, 0, 255), -1)
        cv2.circle(frame, sol_alt, 3, (0, 0, 255), -1)
        sol_goz_mesafe = sol_alt[1] - sol_ust[1]

        burun_mesafe = point_list[30][1] - point_list[27][1]

        pred = lr.predict([[sol_goz_mesafe, sag_goz_mesafe, burun_mesafe]])
        pred_list = []
        for p in pred:
            sol = 0
            sag = 0
            if p[0] > 0.5:
                sol = 1
            else:
                sol = 0
            if p[1] > 0.5:
                sag = 1
            else:
                sag = 0
            pred_list.append([sol, sag])
        print(pred_list)

        if pred_list[0][0] == 0 and pred_list[0][1] == 0:
            say += 1

            # yuz tanıma işlemi
    faces_encodings = face_recognition.face_encodings(frame, face_locations)

    for face in faces_encodings:

        result = face_recognition.compare_faces(known_face_encodings, face)

        if result[0] == True:
            cv2.rectangle(frame, (x, y), (w, h), (0, 0, 255), 2)
            cv2.rectangle(frame, (x, h), (w, h + 30), (0, 0, 255), -1)
            cv2.putText(frame, "kubra", (x, h + 25), cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)
            if pred_list[0][0] == 0 and pred_list[0][1] == 0:
                print("kapı açılıyor...")
                say = 0
        elif result[1] == True:
            cv2.rectangle(frame, (x, y), (w, h), (0, 0, 255), 2)
            cv2.rectangle(frame, (x, h), (w, h + 30), (0, 0, 255), -1)
            cv2.putText(frame, "efe", (x, h + 25), cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)
            if pred_list[0][0] == 0 and pred_list[0][1] == 0:
                print("kapı açılıyor...")
                say = 0
        elif result[2] == True:
            cv2.rectangle(frame, (x, y), (w, h), (0, 0, 255), 2)
            cv2.rectangle(frame, (x, h), (w, h + 30), (0, 0, 255), -1)
            cv2.putText(frame, "joe", (x, h + 25), cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)
            if pred_list[0][0] == 0 and pred_list[0][1] == 0:
                print("kapı açılıyor...")
                say = 0
        else:
            cv2.rectangle(frame, (x, y), (w, h), (0, 0, 255), 2)
            cv2.rectangle(frame, (x, h), (w, h + 30), (0, 0, 255), -1)
            cv2.putText(frame, "bilinmiyor", (x, h + 25), cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)

            if pred_list[0][0] == 0 and pred_list[0][1] == 0:
                print("yüzünüz tanınmadı için kapı açılmıyor...")
            say = 0

    cv2.imshow("frame", frame)
    if cv2.waitKey(1) & 0xff == ord("q"):
        break

cap.release()
cv2.destroyAllWindows()
```

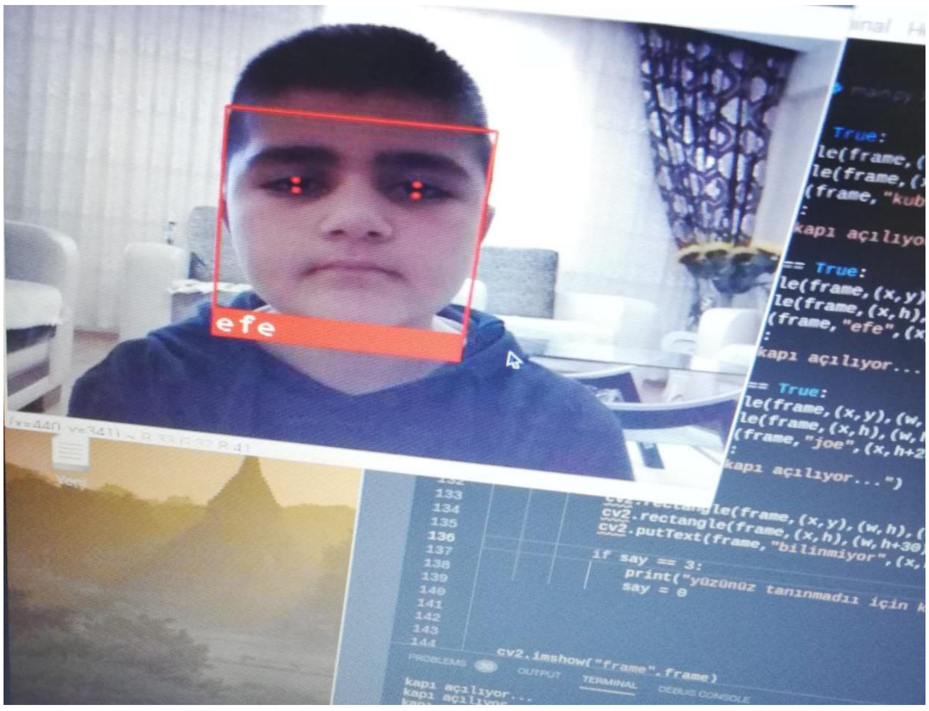


Figure 12.1

12.2. tez.py

This section is the final version of the arduino and python code that will work on the project.

After the eye is closed and opened 3 times, the door will open and the LCD screen will write "door is opening" and the blue LED will light.

```
import serial
from time import sleep
import pandas as pd
import cv2
import dlib
import face_recognition
import time
from sklearn.linear_model import LinearRegression

ser = serial.Serial("/dev/ttyUSB0", baudrate=9600, timeout=1)
ser.flush()

kubra_image = face_recognition.load_image_file("kubra.jpg")
kubra_face_encoding = face_recognition.face_encodings(kubra_image)[0]

efe_image = face_recognition.load_image_file("efe.jpg")
efe_face_encoding = face_recognition.face_encodings(efe_image)[0]

biden_image = face_recognition.load_image_file("biden.jpg")
biden_face_encoding = face_recognition.face_encodings(biden_image)[0]

umitas_image = face_recognition.load_image_file("umitas.jpg")
umitas_face_encoding = face_recognition.face_encodings(umitas_image)[0]

known_face_encodings = [
    kubra_face_encoding,
    efe_face_encoding,
    biden_face_encoding,
    umitas_face_encoding
]
known_face_names = [
    "kubra",
    "efe",
    "Joe Biden",
    "UMIT TAS"
]

detector = dlib.get_frontal_face_detector()
model = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

dataset = pd.read_csv("dataset.csv")

x = dataset.iloc[:, :3].values
y = dataset.iloc[:, 3:].values

lr = LinearRegression()
lr.fit(x, y)


def mid(p1, p2):
    return (int((p1[0] + p2[0]) / 2), int((p1[1] + p2[1]) / 2))


cap = cv2.VideoCapture(0)
say = 0
temp1 = False
temp2 = False
loop_ = 10
```

```
lock1 = 0
while True:
    _, frame = cap.read()
    frame = cv2.rotate(frame, cv2.ROTATE_180)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    face_locations = []
    # yuz tanma için istenilen format ve hızlandırma
    faces = detector(frame)
    for face in faces:
        x = face.left()
        y = face.top()
        w = face.right()
        h = face.bottom()
        face_locations.append((y, w, h, x))
    pred_list = []
    # göz durumu
    for face in faces:
        points = model(gray, face)
        point_list = [(p.x, p.y) for p in points.parts()]

        # sag goz için
        sag_ust = mid(point_list[37], point_list[38])
        sag_alt = mid(point_list[41], point_list[40])
        # göz görselleştirme
        cv2.circle(frame, sag_ust, 3, (0, 0, 255), -1)
        cv2.circle(frame, sag_alt, 3, (0, 0, 255), -1)
        sag_goz_mesafe = sag_alt[1] - sag_ust[1]

        # sol göz için
        sol_ust = mid(point_list[43], point_list[44])
        sol_alt = mid(point_list[47], point_list[46])
        # göz görselleştirme
        cv2.circle(frame, sol_ust, 3, (0, 0, 255), -1)
        cv2.circle(frame, sol_alt, 3, (0, 0, 255), -1)
        sol_goz_mesafe = sol_alt[1] - sol_ust[1]

        burun_mesafe = point_list[30][1] - point_list[27][1]

        pred = lr.predict([[sol_goz_mesafe, sag_goz_mesafe, burun_mesafe]])

        for p in pred:
            sol = 0
            sag = 0
            if p[0] > 0.5:
                sol = 1
            else:
                sol = 0
            if p[1] > 0.5:
                sag = 1
            else:
                sag = 0
            pred_list.append([sol, sag])
        print(pred_list)

    # yuz tanıma işlemi
    faces_encodings = face_recognition.face_encodings(frame, face_locations)

    for face in faces_encodings:
```

```
result = face_recognition.compare_faces(known_face_encodings, face, tolerance=0.5)

if result[0] == True:
    cv2.rectangle(frame, (x, y), (w, h), (0, 0, 255), 2)
    cv2.rectangle(frame, (x, h), (w, h + 30), (0, 0, 255), -1)
    cv2.putText(frame, "kubra", (x, h + 25), cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)
    if pred_list[0][0] == 0 and pred_list[0][1] == 0 and temp1 != True:
        temp1 = True
    elif pred_list[0][0] == 1 and pred_list[0][1] == 1 and temp2 != True:
        temp2 = True
    if temp1 == True and temp2 == True:
        say += 1
        temp1 = False
        temp2 = False
        print(say)
        print(temp1)
        print(temp2)

elif result[1] == True:
    cv2.rectangle(frame, (x, y), (w, h), (0, 0, 255), 2)
    cv2.rectangle(frame, (x, h), (w, h + 30), (0, 0, 255), -1)
    cv2.putText(frame, "efe", (x, h + 25), cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)
    if pred_list[0][0] == 0 and pred_list[0][1] == 0 and temp1 != True:
        temp1 = True
    elif pred_list[0][0] == 1 and pred_list[0][1] == 1 and temp2 != True:
        temp2 = True
    if temp1 == True and temp2 == True:
        say += 1
        temp1 = False
        temp2 = False

        print(say)
        print(temp1)
        print(temp2)



elif result[2] == True:
    cv2.rectangle(frame, (x, y), (w, h), (0, 0, 255), 2)
    cv2.rectangle(frame, (x, h), (w, h + 30), (0, 0, 255), -1)
    cv2.putText(frame, "joe", (x, h + 25), cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)
    if pred_list[0][0] == 0 and pred_list[0][1] == 0 and temp1 != True:
        temp1 = True
    elif pred_list[0][0] == 1 and pred_list[0][1] == 1 and temp2 != True:
        temp2 = True
    if temp1 == True and temp2 == True:
        say += 1
        temp1 = False
        temp2 = False
        print(say)
        print(temp1)
        print(temp2)



elif result[3] == True:
    cv2.rectangle(frame, (x, y), (w, h), (0, 0, 255), 2)
    cv2.rectangle(frame, (x, h), (w, h + 30), (0, 0, 255), -1)
    cv2.putText(frame, "UMIT TAS", (x, h + 25), cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)
    if pred_list[0][0] == 0 and pred_list[0][1] == 0 and temp1 != True:
        temp1 = True
    elif pred_list[0][0] == 1 and pred_list[0][1] == 1 and temp2 != True:
```

```
                temp2 = True
            if temp1 == True and temp2 == True:
                say += 1
                temp1 = False
                temp2 = False

                print(say)
                print(temp1)
                print(temp2)

        else:
            cv2.rectangle(frame, (x, y), (w, h), (0, 0, 255), 2)
            cv2.rectangle(frame, (x, h), (w, h + 30), (0, 0, 255), -1)
            cv2.putText(frame, "bilinmiyor", (x, h + 25), cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 2)

            print(say)
            print(temp1)
            print(temp2)

    if lock1 == 1:
        say = 3

    if say == 3:
        lock1 = 1
        loop_ -= 1
        print(loop_ * 0.5, ' saniye kaldı ')

        ser.write(b'1,1\n')
        if loop_ == 0:
            say = 0
            lock1 = 0

    else:
        loop_ = 10
        lock1 = 0
        ser.write(b'0,0\n')

    cv2.imshow("frame", frame)
    if cv2.waitKey(1) & 0xff == ord("q"):
        break

cap.release()
cv2.destroyAllWindows()
```

12.3. Arduino Codes

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>




LiquidCrystal_I2C lcd(0x27, 16, 2);
```

```
int led1=6;
int led2=7;

int array[2]={0,0};
String data;

int temp1;
int temp2;

void setup() {

   Serial.begin(9600);
   lcd.init();
   lcd.backlight();

pinMode(led1,OUTPUT);
pinMode(led2,OUTPUT);

}

void loop() {



if(Serial.available()>0){

   String data = Serial.readStringUntil('\n');
   array[0] = getValue(data,',',0).toInt();
   array[1] = getValue(data,',',1).toInt();
   temp1 = array[0];

   temp2 = array[1];

}
else{
   array[0] = temp1;
   array[1] = temp2;

}

if (array[0] == 1 && array[1] == 1){

   lcd.setCursor(0,0);
   lcd.print("");
   lcd.setCursor(0,1);
   lcd.print("");
digitalWrite(led1 , LOW);
digitalWrite(led2, HIGH);


   lcd.setCursor(0,0);
   lcd.print("YUZ TANINDI  ");
   lcd.setCursor(0,1);
   lcd.print("KAPI ACILIYOR   ");

//delay(50);

}
else  {
//kırmızı led , yanmama durumu

lcd.setCursor(0,0);
    lcd.print("");
    lcd.setCursor(0,1);
    lcd.print("");

digitalWrite(led1 , HIGH);
digitalWrite(led2, LOW);

    lcd.setCursor(0,0);
    lcd.print("YUZ TANINMADI");
    lcd.setCursor(0,1);
    lcd.print("KAPI ACILMAYACAK");
//delay(50);


}






}

String getValue(String data, char separator, int index)
{
    int found = 0;
int strIndex[] = { 0, -1 };
int maxIndex = data.length() - 1;

for (int i = 0; i <= maxIndex && found <= index; i++) {
if (data.charAt(i) == separator || i == maxIndex) {
    found++;
    strIndex[0] = strIndex[1] + 1;
    strIndex[1] = (i == maxIndex) ? i+1 : i;
}
}
return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}
```

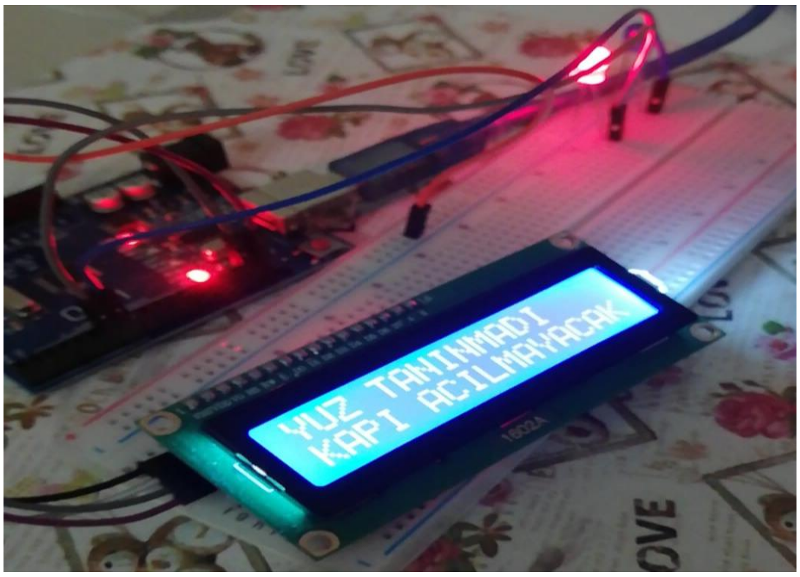


Figure 12.3 : Situation that not opening of the door