Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component report

Programme	: B.Tech(CSE)
Course Title	: ARTIFICIAL INTELLIGENCE
Course Code	: CSE3013
Slot	: B2

Title: Face Recognition enabled Door Lock Mechanism

Team Members: Rishank Pratik | 19BCE1606

Wilson Vidyut Doloy | 19BCE1603

Faculty: Dr.Padmavathy T V

Sign: [illegible]

Date: 26|4|2022

INDEX

ACKNOWLEDGMENT

Primarily, we would like to thank the almighty for all the blessings he showered over us to complete this project without any flaws.

The success and final outcome of this assignment required a lot of guidance and assistance from many people and we are extremely fortunate to have got this all along with the completion of our project. Whatever we have done is only due to such guidance and assistance by our faculty, Dr. PADMAVATHY T V, to whom we are really thankful for giving us an opportunity to do this project.

Last but not the least, we are grateful to all our fellow classmates and our friends for the suggestions and support given to us throughout the completion of our project.

ABSTRACT

Face Recognition has always been one of the most fascinating and intriguing technologies as it deals with human faces. Covid-19 outbreak has propelled the world to move towards touchless facial recognition technology. It is gaining huge traction worldwide owing to its contactless biometric features.
Companies are getting rid of traditional fingerprinting scanners and creating massive business opportunities by adopting AI-based facial recognition technology. Some of the applications where its usage has become crucial are security & surveillance, authentication/access control systems, digital healthcare, photo retrieval, etc.

Our objective was to create a much-needed device for current scenario, a face detection enabled door lock mechanism which will prevent spread of infection through door knobs, door- bells and also provide stronger security system.

Our project is **Multi User**, that is, it will work for every person whose face was added into to the database.

Our algorithm will also be **Fool-proof**, i.e., a real face is needed. It will not work on a digital photo or a printed photo.

INTRODUCTION

This system is based on AI, to make the door only accessible when your face is recognized by the recognition algorithms and then only you are allowed in by the house.
When a person comes at the door and if the face is verified then only door will open else it will not unlock.

The PEAS Description is as follows:

- **Performance Measure:-** Fast, Secure, Comparatively cheap
- **Environment:-** At the main door
- **Actuator:-** Capture Images
- **Sensors:-** Camera

The Environment Type here is **Fully Observable**
The agent sensor(camera) is capable to sense or access the complete state of an agent at each point in time.
We are also planning to add an Infra-red sensor so that camera can switch on when someone is near to door's proximity.

In setup mode, a picture of our face will be taken by the camera.
The image will then be converted into greyscale for faster processing.

Now, the Input will be the face, detected by the peek hole camera in real time.
The picture of the face will then be compared with store image in the database and subsequently door will unlock or remain locked

Fig.1- Existing Systems

Many such systems already exist in the market
What makes our system stand apart is:

1. Fool-proof,
2. Faster,
3. Reduced costs,
4. No physical assistance required

ARCHITECTURE

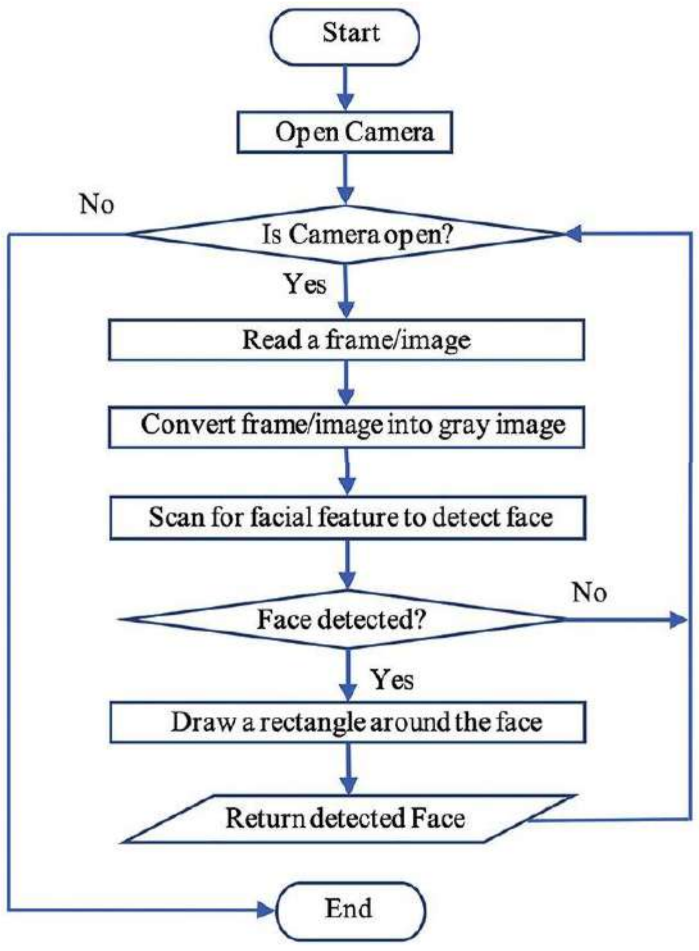

Start
Open Camera
No
Is Camera open?
Yes
Read a frame/image
Convert frame/image into gray image
Scan for facial feature to detect face
No
Face detected?
Yes
Draw a rectangle around the face
Return detected Face
End

Fig.2- Architecture diagram

In setup mode, a picture of our face will be taken by the camera.
The image will then be converted into greyscale for faster processing.
Now, the Input will be the face, detected by the peek hole camera in real time.
The picture of the face will then be compared with store image in the database and subsequently door will unlock or remain locked

MODULES

1. Setting up the Hardware:

To our Arduino board we have connected the servo motor for the door lock. And, instead of an external 2MP door cam, we are using the laptop’s webcam. The Arduino UNO is now connected to PC.

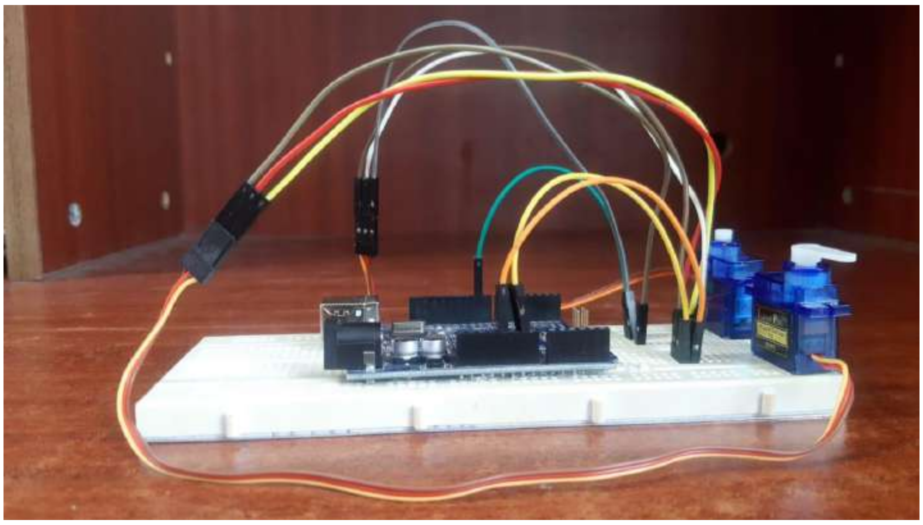


Fig.3- Hardware Setup

2. Face Sample Collection:

The images of the face will be taken by the camera (we are using laptop’s webcam in our case) and will be sequentially cropped and edited. This will be done by the Open CV package. They will then be stored in the database. Original Picture is cropped, edited and converted from RGB to Grayscale

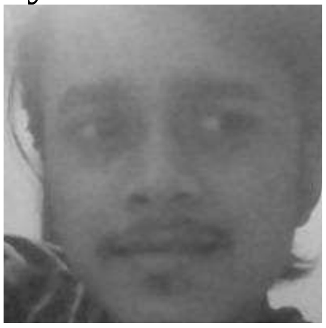


Fig.4- Wilson’s Picture

It is for Multi user as well, Face will be matched with all of the stored faces and if it matches 83%, then door will unlock or else not.

3. Dataset Training:
The samples will then be used to train our algorithm and once complete will show training completed.

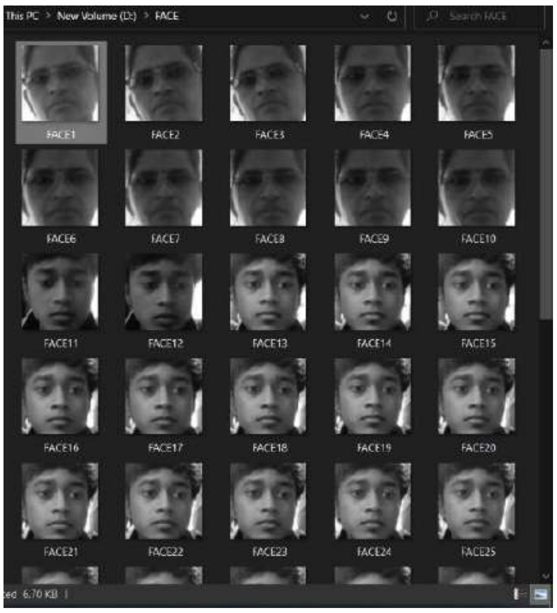


Fig.5- Program is trained to recognize each face.

Then when a face is detected at camera, program scans and authenticates it with the database. On successful recognition door will unlock and on failed recognition, relevant message will be said and displayed.

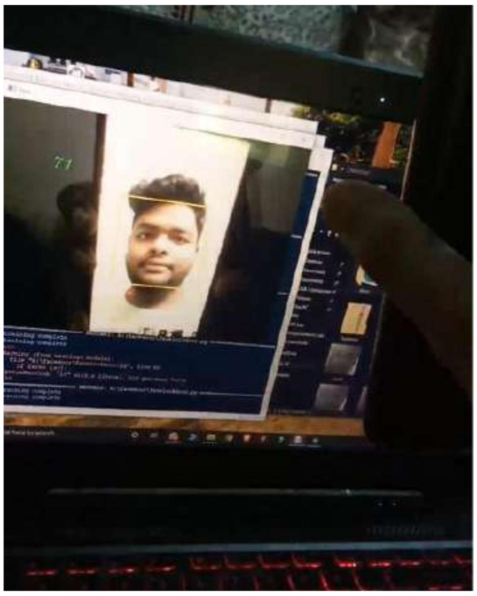


Fig.6- Digital Images, i.e., images shown via phone mobile will not be accepted, as confidence values is low (<83)

4. Door Lock/Unlock Mechanism:
Now when a person shows up, the camera will scan his image and verify with the database.
If it matches on a greater percentage, the servo motor will run and door will unlock for 5 secs and a welcome voice can be heard.
Else if authentication fails or face was not detected then door will remain lock and a voice to notify this will be heard.

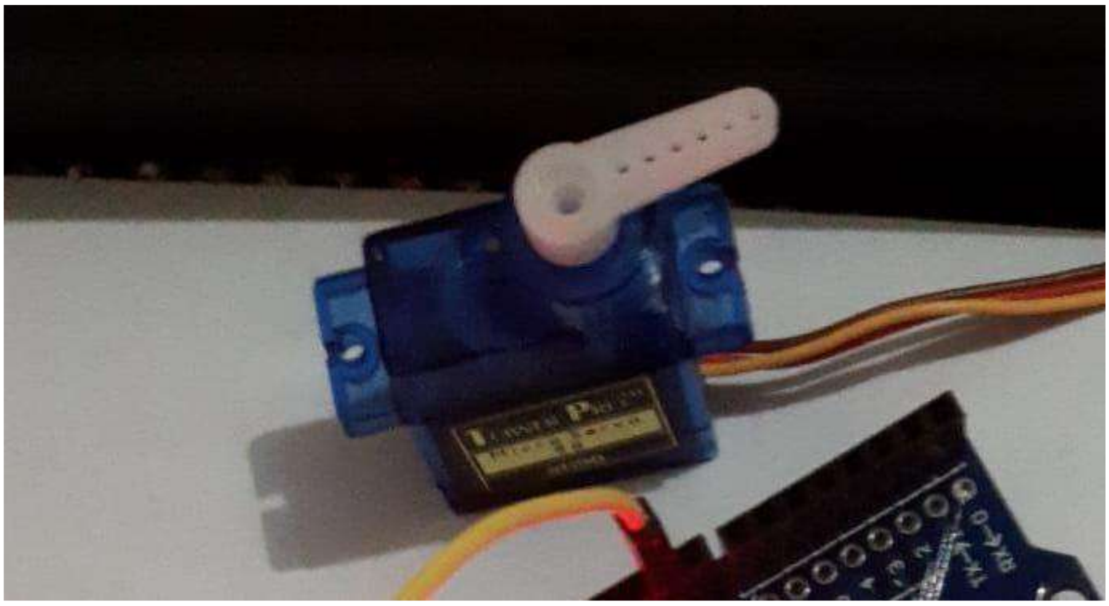


Fig.7- Servo motor for locking/unlocking door

TECHNOLOGIES AND INTEGRATION

Hardware:
- Arduino UNO
- 2 Mega Pixel VGA Camera (Laptop Webcam)
- Servo Motor
- Jumper Wires
- Door Lock

Software:
- Python IDE
- Bitcraze Crazyflie Python Client
- OpenCV
- TinkerCad for Simulation

IMPLEMENTATION DETAILS

First pictures of the person are clicked and the images are transformed from RGB to Grayscale, because it is easy to detect faces in the grayscale. This is where OpenCV comes into play. After that, the image manipulation used, in which the resizing, cropping, blurring and sharpening of the images done if needed.

The next step is image segmentation, which is used for contour detection or segments the multiple objects in a single image so that the classifier can quickly detect the objects and faces in the picture. Now we use Haar-Like features algorithm. This algorithm used for finding the location of the human faces in a frame or image with the help of edge detection, line detection, center detection for detecting eyes, nose, mouth, etc. in the picture.

The next step is to give the coordinates of x, y, w, h which makes a rectangle box in the picture to show the location of the face. There are also many other detection techniques that are used together for detection such as smile detection, eye detection, blink detection, etc.
A scale is selected smaller than the target image.
It is then placed on the image, and the average of the values of pixels in each section is taken.

If the difference between two values pass a given threshold, it is considered a match.

Face detection on a human face is performed by matching a combination of different Haar-like-features.

There are three stages for the face recognition as follows:
1. Collecting images (which we have already done)
2. Extracting unique features, classifying them.
3. Matching features of an input image to the features in the saved images and predict identity.

Face detector is used to detect faces in the image, cropped and transferred to be recognised.

This is done using the same technique used for the image capture application.

For face to be detected, a prediction is made using FaceRecognizer.predict() which compares the input image with each image in the dataset and checks for confidence.

If the confidence is higher than the set threshold (83% in our case), the door will be unlocked.

Circuit

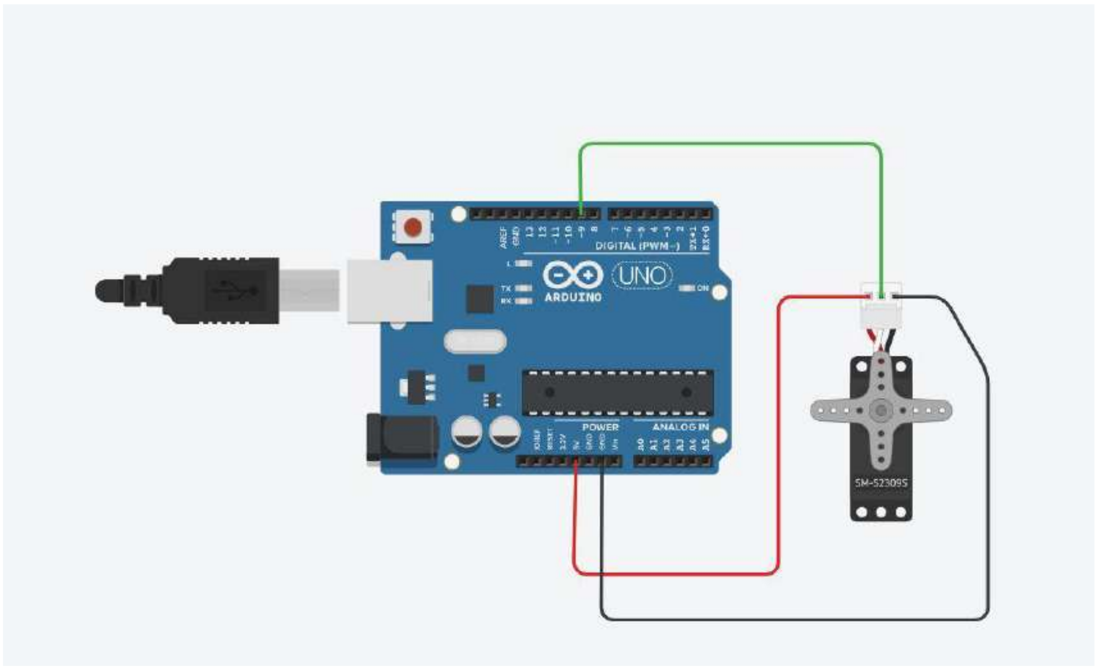
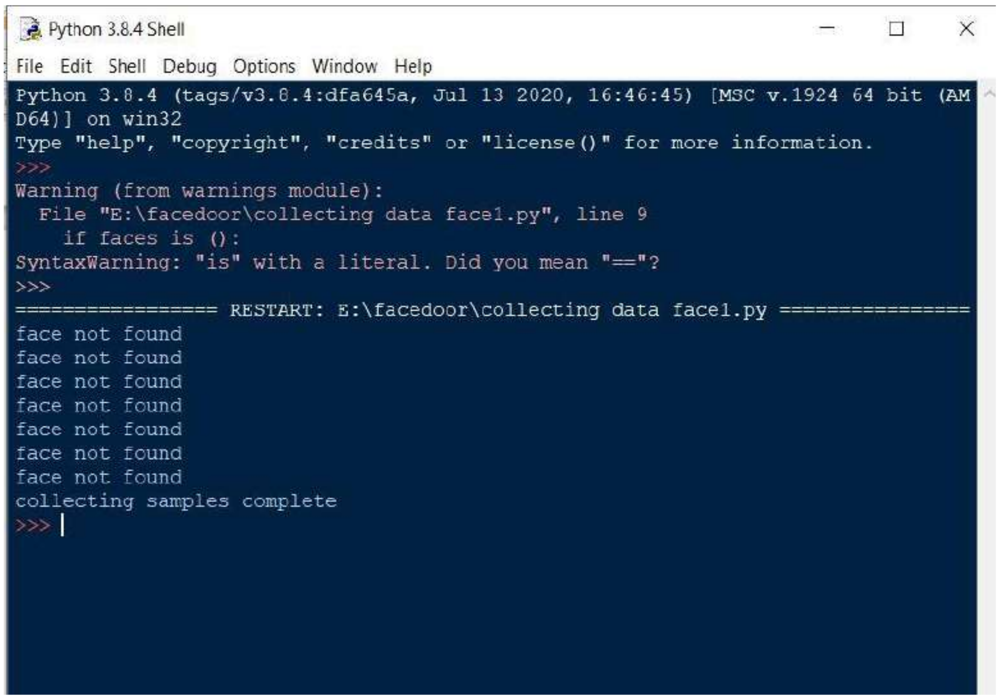


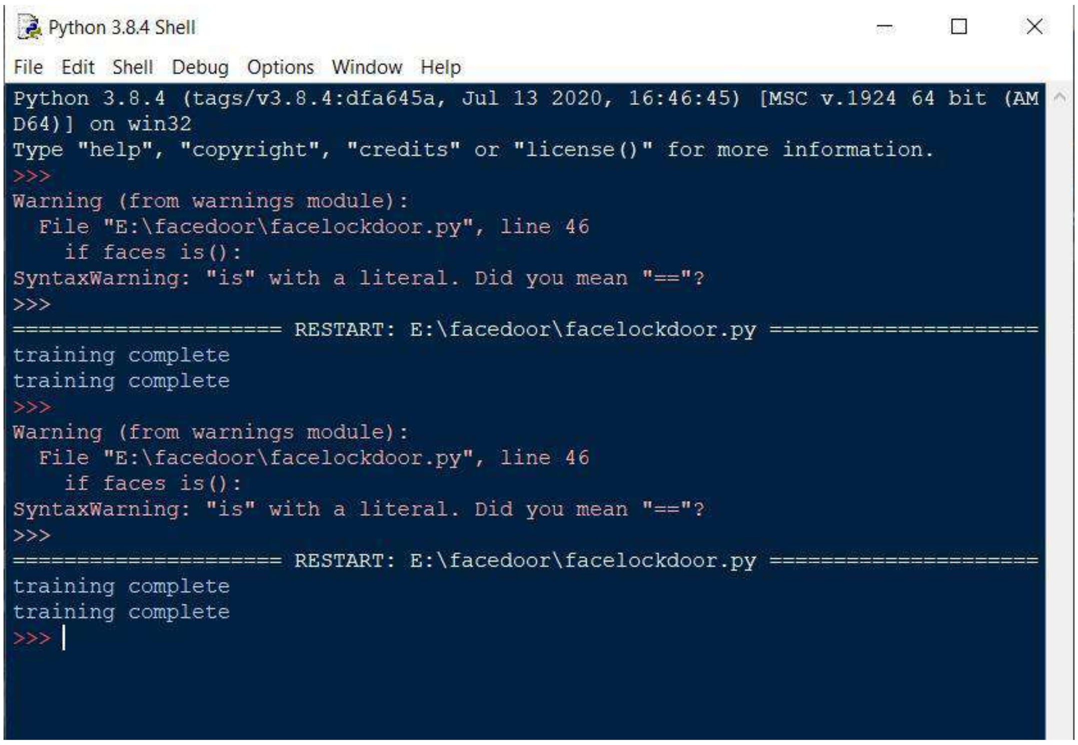
Fig.8- Circuit Diagram

RESULTS AND DISCUSSIONS



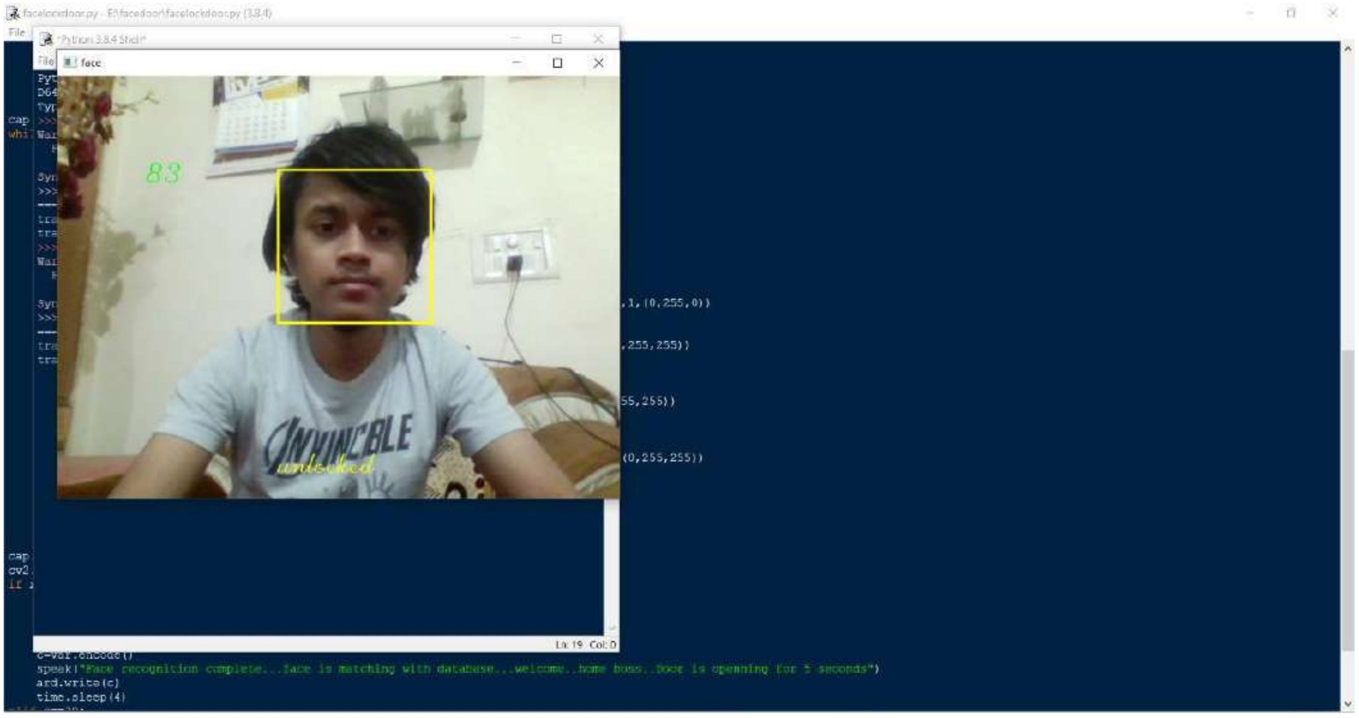
```
Python 3.8.4 (tags/v3.8.4:dfa645a, Jul 13 2020, 16:46:45) [MSC v.1924 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
Warning (from warnings module):
  File "E:\facedoor\collecting data face1.py", line 9
    if faces is ():
SyntaxWarning: "is" with a literal. Did you mean "=="?
>>>
================= RESTART: E:\facedoor\collecting data face1.py ================
face not found
face not found
face not found
face not found
face not found
face not found
face not found
collecting samples complete
>>> 
```

Fig.9- Collecting face data



```
Python 3.8.4 (tags/v3.8.4:dfa645a, Jul 13 2020, 16:46:45) [MSC v.1924 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
Warning (from warnings module):
  File "E:\facedoor\facelockdoor.py", line 46
    if faces is():
SyntaxWarning: "is" with a literal. Did you mean "=="?
>>>
===================== RESTART: E:\facedoor\facelockdoor.py =====================
training complete
training complete
>>>
Warning (from warnings module):
  File "E:\facedoor\facelockdoor.py", line 46
    if faces is():
SyntaxWarning: "is" with a literal. Did you mean "=="?
>>>
===================== RESTART: E:\facedoor\facelockdoor.py =====================
training complete
training complete
>>> 
```

Fig.10- Training



```
,1,(0,255,0))
,255,255))
55,255))
(0,255,255))
```

```
c=var.encode()
speak("Face recognition complete...face is matching with database...welcome..home boss..Door is openning for 5 seconds")
ard.write(c)
time.sleep(4)
```

Fig.10- Face Recognition in action

CONCLUSION AND FUTURE ENHANCEMENTS

In this proposed face recognition door lock/unlock system using Arduino UNO. Python and OpenCV was used to implement the feature extraction and classifier, in which we used LBPH (Local Binary Patterns Histograms). To add the person's image in database First the persons images are added to the database and then the prototype design contains Arduino UNO and servo motor which is our door, in which the output of face recognition algorithm will lock or unlock the door if the face matches more than 83% from the database.

It goals at creating minimal human intervention which has been achieved. It combines two modern technologies IoT and face recognition. It is also low cost and power efficient system.

Future work includes optimization of hierarchical image processing, use different features extraction and classifier.

REFERENCES

1. Multi-faces recognition process using Haar cascades and eigenface method T Mantoro, MA Ayu - 2018
2. A Survey of Face Recognition Techniques Rabia Jafri* and Hamid R. Arabnia* Journal of Information Processing Systems, Vol.5, No.2
3. https://arsfutura.com/magazine/building-a-face-recognition-powered-door-lock/
4. https://research.vit.ac.in/publication/facial-recognition-enabled-smart-door-unlock
5. Arduino UNO user manual https://www.arduino.cc/en/Guide/ArduinoUno

SAMPLE CODE

collecting data face.py

```
import cv2
import pyttsx3
import numpy as np

face_classifier = cv2.CascadeClassifier('D:\\Downloads\\Face Recognition Door
Lock/haarcascade_frontalface_default.xml')

def speak(audio):
    engine.say(audio)
    engine.runAndWait()
engine = pyttsx3.init('sapi5')
voices=engine.getProperty('voices')
engine.setProperty("voice",voices[0].id)
engine.setProperty("rate",140)
engine.setProperty("volume",1000)


def face_extractor(img):
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray,1.3,5)
    if faces is ():
        return None
    for (x,y,w,h) in faces:
        cropped_face = img[y:y+h,x:x+w]
    return cropped_face
cap = cv2.VideoCapture(0)
count = 0
speak("please look into the camera..")
while True:
    ret,frame = cap.read()
    if face_extractor(frame) is not None:
        count += 1
        face = cv2.resize(face_extractor(frame),(200,200))
        face = cv2.cvtColor(face,cv2.COLOR_BGR2GRAY)

        file_name_path = 'D:\\FACE\\FACE'+str(count)+'.jpg'
        cv2.imwrite(file_name_path,face)

        cv2.putText(face,str(count),(50,50),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
        cv2.imshow("face cropper",face)
    else:
        print('face not found')
        speak("face not found..")
        pass
    if cv2.waitKey(1)==13 or count==10:
        break
cap.release()
cv2.destroyAllWindows()
print("collecting samples complete")
```

facelockdoor.py

```
import cv2
import numpy as np
from os import listdir
from os.path import isfile,join
import serial
import time
import pyttsx3
q=1
x=0
c=0
m=0
d=0
while q<=2:
    data_path = 'D:/FACE/'
    onlyfiles = [f for f in listdir(data_path) if isfile(join(data_path,f))]
    Training_data, Lebels = [],[]
    for i , files in enumerate(onlyfiles):
        image_path = data_path + onlyfiles[i]
        images = cv2.imread(image_path,cv2.IMREAD_GRAYSCALE)
        Training_data.append(np.asarray(images, dtype = np.uint8))
        Lebels.append(i)

    Lebels = np.asarray(Lebels, dtype = np.int32)
    model = cv2.face.LBPHFaceRecognizer_create()
    model.train(np.asarray(Training_data),np.asarray(Lebels))
    print("training complete")
    q+=1
face_classifier = cv2.CascadeClassifier('D:\\Downloads\\Face Recognition Door
Lock/haarcascade_frontalface_default.xml')


def speak(audio):
    engine.say(audio)
    engine.runAndWait()
engine = pyttsx3.init('sapi5')
voices=engine.getProperty('voices')
engine.setProperty("voice",voices[0].id)
engine.setProperty("rate",140)
engine.setProperty("volume",1000)


def face_detector(img, size= 0.5):
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray,1.3,5)

    if faces is():
        return img,[]
    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y),(x+w,y+h),(0,255,255),2)
        roi = img[y:y+h, x:x+w]
        roi = cv2.resize(roi,(200,200))

    return img,roi

cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()

    image, face = face_detector(frame)
    try:
        face = cv2.cvtColor(face,cv2.COLOR_BGR2GRAY)
        result= model.predict(face)
        if result[1]<500:

            confidence = int((1-(result[1])/300)*100)
            display_string = str(confidence)
            cv2.putText(image,
display_string,(100,120),cv2.FONT_HERSHEY_SCRIPT_COMPLEX,1,(0,255,0))

        if confidence>=83:

cv2.putText(image,"unlocked",(250,450),cv2.FONT_HERSHEY_SCRIPT_COMPLEX,1,(0,255,255)
)
            cv2.imshow('face',image)
            x+=1
        else:

cv2.putText(image,"locked",(250,450),cv2.FONT_HERSHEY_SCRIPT_COMPLEX,1,(0,255,255))
            cv2.imshow('face',image)
            c+=1
    except:
        cv2.putText(image,"Face not
found",(250,450),cv2.FONT_HERSHEY_SCRIPT_COMPLEX,1,(0,255,255))
        cv2.imshow('face',image)
        d+=1
        pass

    if cv2.waitKey(1)==13 or x==10 or c==30 or d==20:
        break
cap.release()
cv2.destroyAllWindows()
if x>=5:
    m=1
    ard = serial.Serial('com9' ,9600)
    time.sleep(2)
    var = 'a'
    c=var.encode()
    speak("Face recognition complete..")
    speak("Face is matching with database...")
    speak("Welcome home..boss..")
    speak("Door is openning for 5 seconds")
    ard.write(c)
    time.sleep(4)
elif c==30:
    speak("face is not matching..please try again")
elif d==20:
    speak("face is not found please try again ")
if m==1:
    speak("door is closing")
```

door.ino

```
#include<Servo.h>
Servo myservo;
char d;
int pos;
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
pinMode(5,OUTPUT);
myservo.attach(9);
  myservo.write(0);
}

void loop() {
  // put your main code here, to run repeatedly:
if(Serial.available())
{
  d=Serial.read();
}
if(d=='a')
{
  Serial.print(d);
  delay(300);
  for(pos=0;pos<=180;pos+=5)
   { myservo.write(pos);
   delay(20);
   }
   delay(5000);
   for(pos=180;pos>=0;pos-=5)
   {
   myservo.write(pos);
   delay(20);
   }
   }
   d="";

}
```

****