VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM 59014

Internet of Things Project Report on

"Office entry authentication using facial recognition"

By

Nikhil S Shastry(1BM17CS053)
Mohammed Omar Khan (1BM17CS138)
Meghana Mukunda Joshi (1BM17CS146)
Niyathi Srinivasan Kumbale (1BM17CS147)

Under the Guidance of

Antara Roy Choudhury
Assistant Professor, Department of CSE
BMS College of Engineering

IoT Application Development carried out at B.M.S. College of Engineering

Department of Computer Science and Engineering
BMS College of Engineering
(Autonomous college under VTU)
P.O. Box No.: 1908, Bull Temple Road, Bangalore-560 019
2019-2020

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Internet of Things project titled "**Office entry authentication using facial recognition**" has been carried out by **Niyathi Srinivasan Kumbale (1BM17CS147), Meghana Mukunda Joshi (1BM17CS146), Nikhil S Shastry(1BM17CS053) Mohammed Omar Khan (1BM17CS138)** during the academic year 2019-2020.

Signature of the guide
Antara Roy Choudhury
Assistant Professor
Department of Computer Science and Engineering
BMS College of Engineering, Bangalore

Examiners

Name	**Signature**
1.	
2.	

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, **Niyathi Srinivasan Kumbale (1BM17CS147), Meghana Mukunda Joshi (1BM17CS146), Nikhil S Shastry(1BM17CS053), Mohammed Omar Khan (1BM17CS138)** students of 5th Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that this IoT Application development work entitled **"Office entry authentication using facial recognition"** has been carried out by us under the guidance of Antara Roy Choudhury, Assistant Professor, Department of CSE, BMS College of Engineering, Bangalore during the academic semester Aug-Dec 2019. We also declare that to the best of our knowledge and belief, the development reported here is not from a part of any other report by any other student.

Signature

Niyathi Srinivasan Kumbale (1BM17CS147)

Meghana Mukunda Joshi (1BM17CS146)

Mohammed Omar Khan (1BM17CS138)

Nikhil S Shastry(1BM17CS053)

Objective of the project

The office entry authorization system either allows or denies access to employees, based on facial recognition. At the end of each day, the floor manager receives an email consisting of a log of each present employee's name and the time at which they have entered the building.

The project consists of four phases:

1. Face detection and data gathering
2. Training recognizer
3. Facial recognition
4. Sending email to floor manager

Literature Survey

Sl.No	Name of the Project or Product (Existing)	Commercial or Non-Commercial	Features
1	**FaceFirst**	**Commercial**	**Scalable, accurate**
2	**Nest Hello**	**Commercial**	**Versatile, Cost eff.**
3	**Tend Secure Lync**	**Commercial**	**Database storage**
4	**Nest Cam IQ Indoor**	**Commercial**	**Google Home Asst.**

FaceFirst- It is a global patented facial recognition software platform designed to be scalable, fast and accurate while maintaining the highest levels of security, privacy, and convenience. Using artificial intelligence and machine learning, the FaceFirst system can be used as a one-to-one access control solution or can be used to establish a physical perimeter to keep out unwanted or potentially dangerous people.

Nest Hello- The product is amongst the most versatile facial recognition products available in the market. Nest's IQ can tell who's already inside your house but also who is outside and duration of the stay within the house. It is also fairly cheap compared to its competitors. It comes with face-tracking features for which the users have to subscribe to the Nest Aware cloud.

Tend Secure Lynx- The Tend Secure Lynx has multiple features which includes seven-day event-based storage of clips. It has facial recognition free of charge, unlike the Nest Aware service. It comes for a price as low as $60. However, it's performance isn't as good as the Nest Hello. It only serves as an option if someone wants an inexpensive indoor home security camera with decent facial recognition features.

Nest Cam IQ Indoor- It is similar to the Nest Hello Doorbell and lets you have a constant view of the front with consistent accuracy. It has several benefits that include assistance from Google assistant. The camera essentially doubles as a Google Home speaker and can answer basic questions about weather and traffic while also controlling several smart home appliances.

Proposed Project

The product we want to develop centers around automating the ask of attendance with added security, be it in an office or school or any other work environment. The project's main goal is to keep track of people who enter the building while ensuring only those authorized to enter the facility of the building do so. The platform will record the time of the entry of the person which can be stored as records for later uses. Unrecognized people will not be allowed to enter and must first be added to the system's database. Through this project, we can automate two laborious processes of security and attendance.

Features and Advantages

By automating the process of security, fewer people need to be employed for security reasons. This can lead to fewer workers which can benefit the organization immensely. If the person is not authorized by the system, they will not be allowed to enter the facility. This would eliminate the cost of having Multiple Security guards at the entrance of a building.

The product is also of a reasonable price as it comes with multiple features. With the record-keeping system of the project, it can keep track of when a person enters and leaves the facility. This way, the manual process of attendance is eliminated.

The product is easy to use with a basic setup phase. If one wishes to add their identity to the system, he/she can easily do so within a matter of few seconds. With such ease of use, the product also offers high accuracy of facial recognition. This ensures high security and prevents a chance for any fraudulent activities.

Hardware and Software Requirements

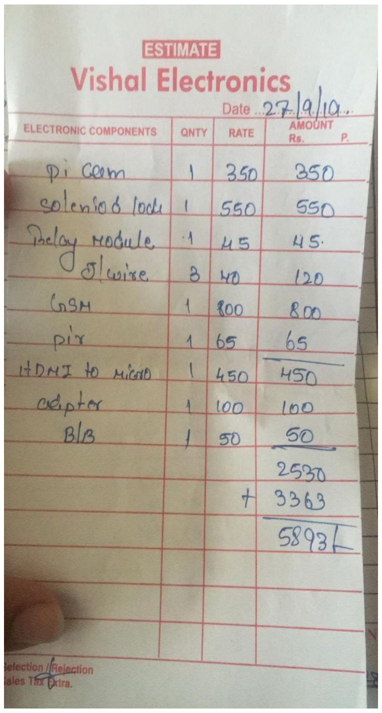
Hardware requirements-

1. Raspberry Pi 4
2. Raspberry Pi Camera
3. 2 Channel-Relay
4. 12V Solenoid Lock
5. Jumper Wires
6. 12V Power Supply

Software Requirements-

1. OpenCV
2. OS- Raspbian Buster
3. Pi Camera driver for Raspberry Pi
4. NumPy

Total cost incurred



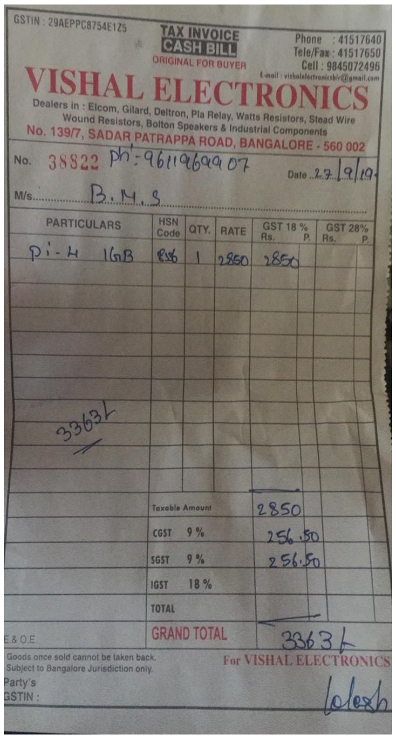
ESTIMATE

Vishal Electronics

Date 27/9/19

ELECTRONIC COMPONENTS	QNTY	RATE	AMOUNT Rs. P.
pi cam	1	350	350
solenoid lock	1	550	550
Relay module	1	45	45
J/wire	3	40	120
GSM	1	800	800
pir	1	65	65
HDMI to micro	1	450	450
adpter	1	100	100
B/B	1	50	50
			2530
		+	3363
			5893/-

Selection/Rejection
Sales Tax Extra.



GSTIN : 29AEPPC8754E1Z5

TAX INVOICE
CASH BILL
ORIGINAL FOR BUYER

Phone : 41517640
Tele/Fax : 41517650
Cell : 9845072496
E-mail : vishalelectronicsblr@gmail.com

VISHAL ELECTRONICS

Dealers in : Elcom, Gilard, Deltron, Pla Relay, Watts Resistors, Stead Wire Wound Resistors, Bolton Speakers & Industrial Components
No. 139/7, SADAR PATRAPPA ROAD, BANGALORE - 560 002

No. 38822 ph:-9611969907 Date 27/9/19

M/s. B.M.S

PARTICULARS	HSN Code	QTY.	RATE	GST 18 % Rs. P.	GST 28% Rs. P.
pi-4 1GB	8536	1	2850	2850	

3363/-

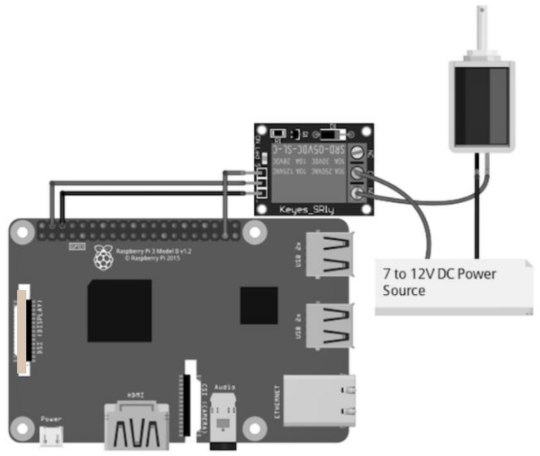
Taxable Amount	2850
CGST 9 %	256.50
SGST 9 %	256.50
IGST 18 %	
TOTAL	
GRAND TOTAL	3363/-

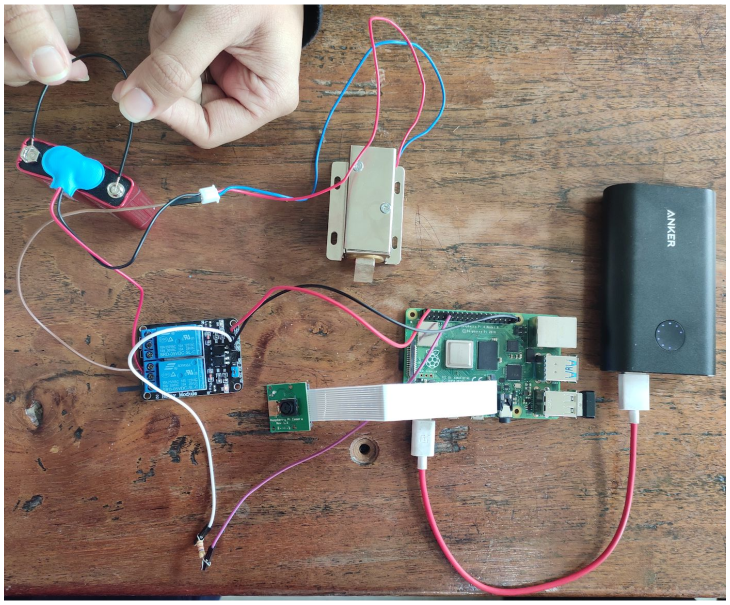
E.& O.E.
Goods once sold cannot be taken back.
Subject to Bangalore Jurisdiction only.
Party's GSTIN :

For VISHAL ELECTRONICS

[illegible]

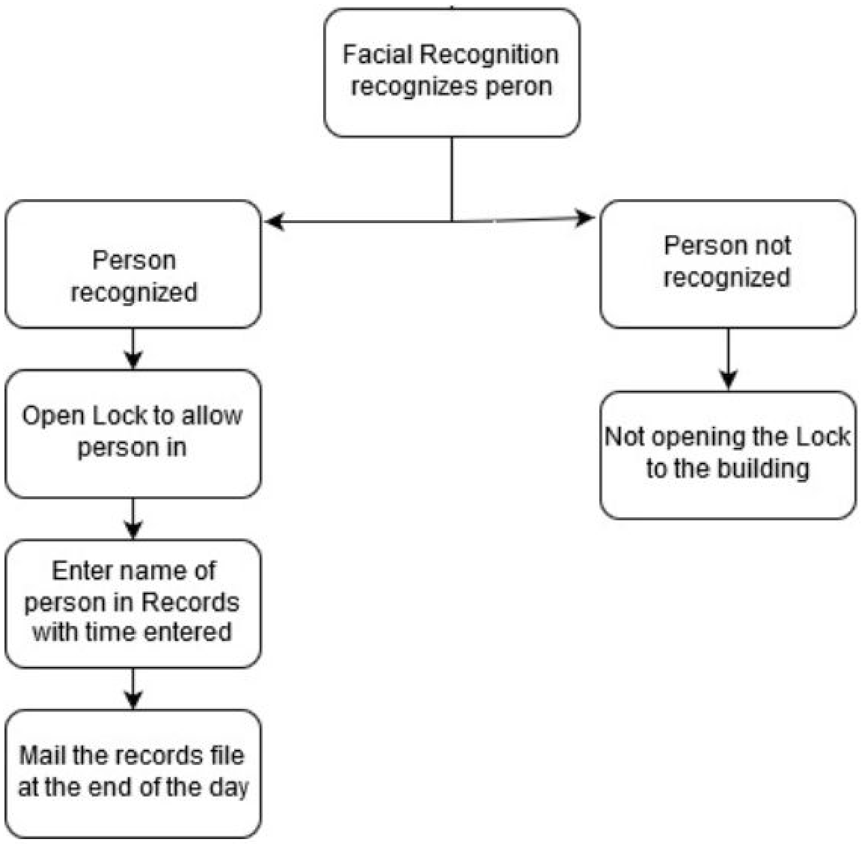
Design and Connections





The circuit above illustrates the connections made for the functioning of the project. The Raspberry-Pi camera is directly connected to the Raspberry-Pi. Its function is to scan for any frames with faces in them and then pass these frames to the Raspberry-Pi where the facial recognition service takes care of recognizing the faces. To detect the faces a cascade classifier is used. Once the faces have been detected and recognized, the Raspberry-Pi sends a GPIO high to the pin to which the 2-Channel Relay is connected to. Upon receiving the high output from the Raspberry-Pi, the relay switches on. The 12V Solenoid Lock is connected to the relay and a 12V power source. This makes the Solenoid Lock activate thus allowing a recognized face to enter the door. However, had an unknown face been detected, the Raspberry-Pi would not send a GPIO high to the relay which would keep the lock on, hence the door closed.

Implementation


Facial Recognition recognizes peron
Person recognized
Person not recognized
Open Lock to allow person in
Not opening the Lock to the building
Enter name of person in Records with time entered
Mail the records file at the end of the day

The Raspberry-Pi camera is the senor in this project which is used to detect and recognize faces of the people in front of it. The sensor i.e. the camera is always on to ensure that if anyone stands before it, the face of the person is detected and then checked to see if the model recognizes the face. The Raspberry-Pi is the microcontroller in charge of deciding if the face of the person is recognized or not. It does so through the python scripts that run as the camera is on. Each frame is captured and inspected to determine if the person is known. The actuator is the Solenoid Lock which opens when a person is recognized by the model. However, if an unknown person is in front of the camera, the Raspberry-pi does not allow the lock to open.

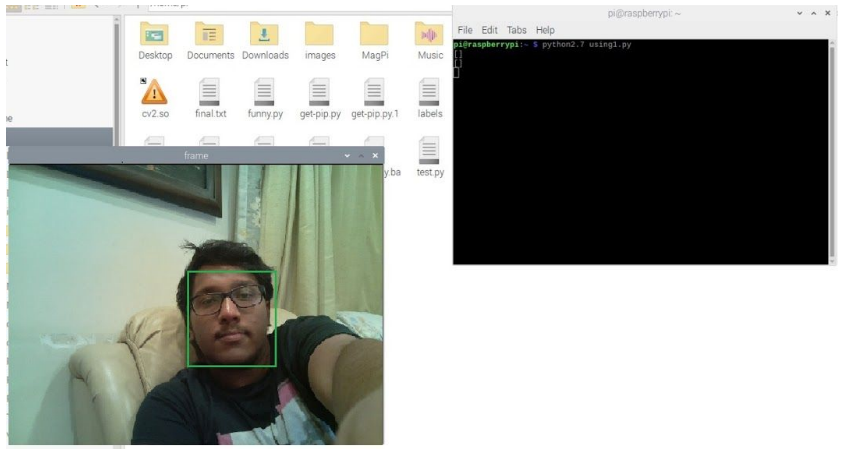
There are four phases to this system, which are:

1. Data Gathering for Face Detection-

The first task is to gather the data to train the classifier. A Raspberry Pi camera will capture 30 images. A python code will take 30 faces of each person using OpenCV pre-trained classifier. OpenCV already contains many pre-trained classifiers for face, eyes, smile, etc. The resolution is set at (640, 480) and frame rate at 30 fps. *PiRGBArray()* gives us a 3-dimensional RGB array organized(rows, columns, colors) from an unencoded RGB capture. PiRGBArray's advantage is its ability to read the frames from Raspberry Pi camera as NumPy arrays making it compatible with OpenCV. It avoids the conversion from JPEG format to OpenCV format which would slow the process. Next, the user is asked for a name. If a directory with that name is already there, it will respond with "Name already exists" and will exit the code. If a directory with this name isn't there, it will create the directory and images will be saved with this name. After that, the *capture_continuous* function will be used to start reading the frames from the Raspberry Pi camera module.

The *capture_continuous* function takes three arguments:

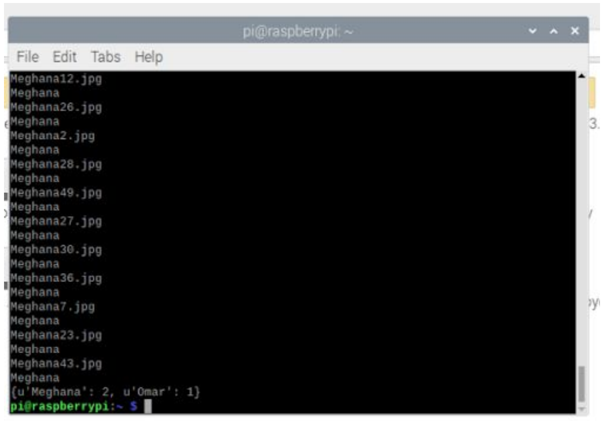
1. rawCapture
2. The format in which each frame should be read since OpenCV expects the image to be in the BGR format rather than the RGB so we specify the format to be BGR.
3. The use_video_port boolean, making it true means that a stream is being treated as a video.



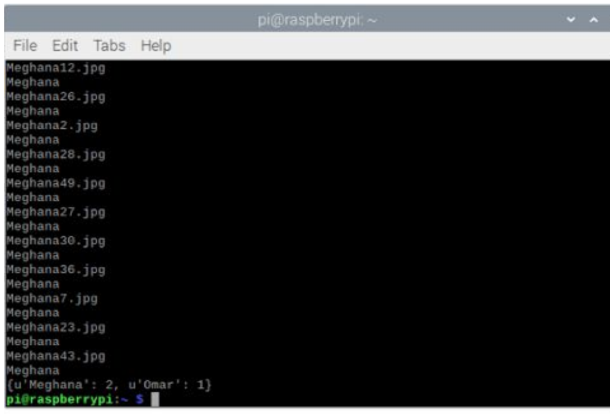
```
pi@raspberrypi:~ $ python2.7 using1.py
[]
[]
```

2. Training the recognizer

Now the recognizer can be trained according to the data gathered in the previous step. The LBPH (LOCAL BINARY PATTERNS HISTOGRAMS) face recognizer will be used, which is included on the OpenCV package. Finally, the dictionary which contains the directory names and label IDs will be stored.



```
Meghana12.jpg
Meghana
Meghana26.jpg
Meghana
Meghana2.jpg
Meghana
Meghana28.jpg
Meghana
Meghana49.jpg
Meghana
Meghana27.jpg
Meghana
Meghana30.jpg
Meghana
Meghana36.jpg
Meghana
Meghana7.jpg
Meghana
Meghana23.jpg
Meghana
Meghana43.jpg
Meghana
{u'Meghana': 2, u'Omar': 1}
pi@raspberrypi:~ $
```

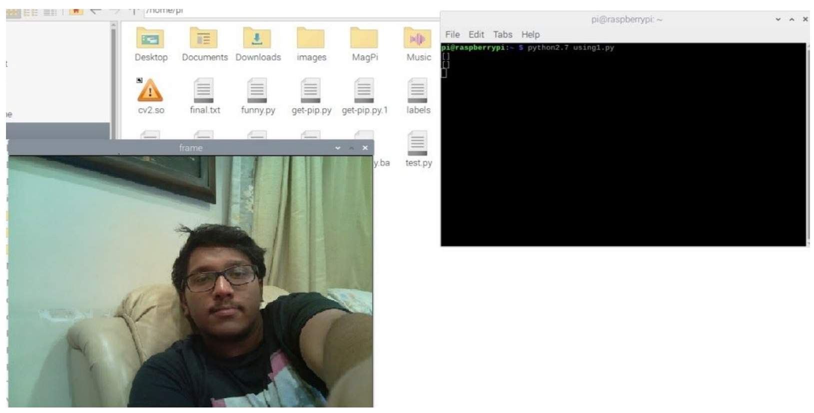


```
Meghana12.jpg
Meghana
Meghana26.jpg
Meghana
Meghana2.jpg
Meghana
Meghana28.jpg
Meghana
Meghana49.jpg
Meghana
Meghana27.jpg
Meghana
Meghana30.jpg
Meghana
Meghana36.jpg
Meghana
Meghana7.jpg
Meghana
Meghana23.jpg
Meghana
Meghana43.jpg
Meghana
{u'Meghana': 2, u'Omar': 1}
pi@raspberrypi:~ $
```

3. Using the Recognizer for Facial Recognition

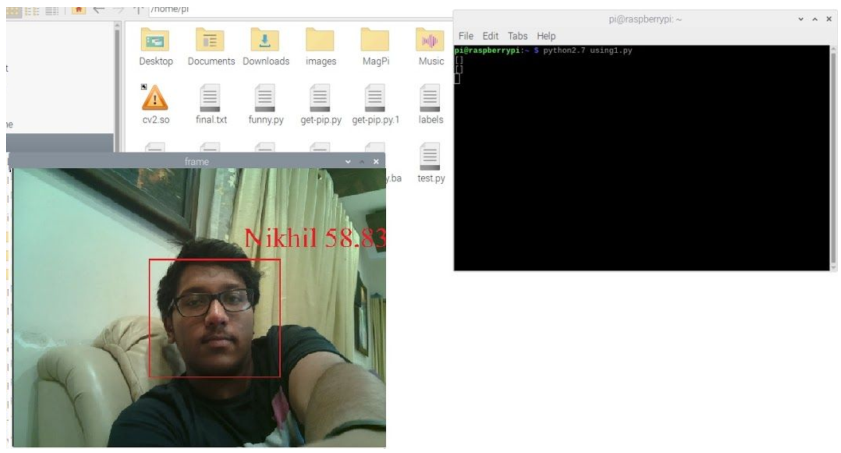
The recognizer set up in the previous section can now be used to recognize the faces. It will give a confidence level and label ID. If the face matches, the relay will turn on.

NOT RECOGNIZED



```
pi@raspberrypi:~ $ python2.7 using1.py
[]
[]
```

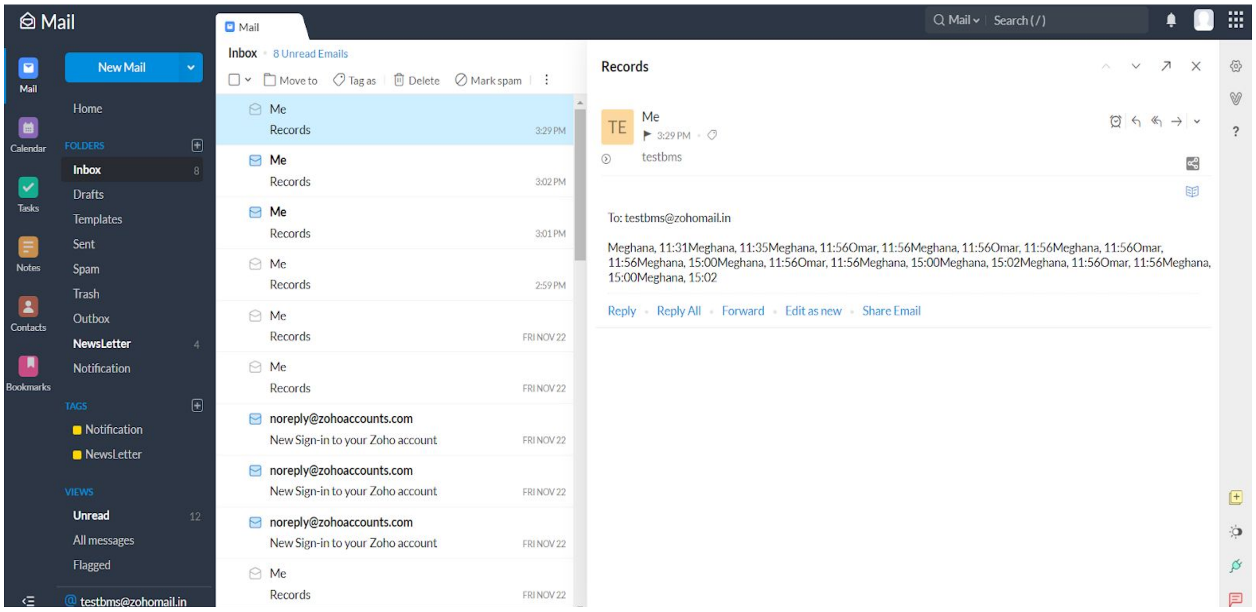
RECOGNIZED



```
pi@raspberrypi:~ $ python2.7 using1.py
[]
[]
```

4. Send email to floor manager

At the end of each day, the floor manager receives an email consisting of a log of each present employee’s name and the time at which they have entered the building.



Me	Records	3:29 PM
Me	Records	3:02 PM
Me	Records	3:01 PM
Me	Records	2:59 PM
Me	Records	FRI NOV 22
Me	Records	FRI NOV 22
noreply@zohoaccounts.com	New Sign-in to your Zoho account	FRI NOV 22
noreply@zohoaccounts.com	New Sign-in to your Zoho account	FRI NOV 22
noreply@zohoaccounts.com	New Sign-in to your Zoho account	FRI NOV 22
Me	Records	FRI NOV 22

Records

TE Me
3:29 PM
testbms

To: testbms@zohomail.in

Meghana, 11:31Meghana, 11:35Meghana, 11:56Omar, 11:56Meghana, 11:56Omar, 11:56Meghana, 11:56Omar, 11:56Meghana, 15:00Meghana, 11:56Omar, 11:56Meghana, 15:00Meghana, 15:02Meghana, 11:56Omar, 11:56Meghana, 15:00Meghana, 15:02

1. Code to add a new person to the facial database

```
from __future__ import absolute_import
import cv2
from picamera.array import PiRGBArray
from picamera import PiCamera
import numpy as np
import os
import sys
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 30
rawCapture = PiRGBArray(camera, size=(640, 480))
faceCascade =
cv2.CascadeClassifier("/home/pi/opencv/data/haarcascades_cuda/haarcascade_frontalface_defau
lt.xml")
name = raw_input("What's his/her Name? ")
dirName = "./images/" + name
print dirName
if not os.path.exists(dirName):
   os.makedirs(dirName)
   print "Directory Created"
else:
   print "Name already exists"
   sys.exit()
count = 1
for frame in camera.capture_continuous(rawCapture, format="bgr",use_video_port=True):
   if count > 50:
      break
   frame = frame.array
   gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
   faces = faceCascade.detectMultiScale(gray, scaleFactor = 1.5, minNeighbors = 5)
   for (x, y, w, h) in faces:
      roiGray = gray[y:y+h, x:x+w]
      fileName = dirName + "/" + name + unicode(count) + ".jpg"
      cv2.imwrite(fileName, roiGray)
      cv2.imshow("face", roiGray)
      cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
      count += 1
   cv2.imshow('frame', frame)
   key = cv2.waitKey(1)
   rawCapture.truncate(0)
   if key == 27:
      break
cv2.destroyAllWindows()
```

2. Code to train the recognizer

```
from __future__ import with_statement
from __future__ import absolute_import
import os
import numpy as np
from PIL import Image
import cv2
import pickle
from io import open
faceCascade =
cv2.CascadeClassifier(u"/home/pi/opencv/data/haarcascades_cuda/haarcascade_frontalface_defa
ult.xml")
recognizer = cv2.face.LBPHFaceRecognizer_create()
baseDir = os.path.dirname(os.path.abspath(__file__))
imageDir = os.path.join(baseDir, u"images")
currentId = 1
labelIds = {}
yLabels = []
xTrain = []
for root, dirs, files in os.walk(imageDir):
    print root, dirs, files
    for file in files:
        print file
        if file.endswith(u"png") or file.endswith(u"jpg"):
            path = os.path.join(root, file)
            label = os.path.basename(root)
            print label
            if not label in labelIds:
                labelIds[label] = currentId
                print labelIds
                currentId += 1

            id_ = labelIds[label]
            pilImage = Image.open(path).convert(u"L")
            imageArray = np.array(pilImage, u"uint8")
            faces = faceCascade.detectMultiScale(imageArray, scaleFactor=1.1, minNeighbors=5)
            for (x, y, w, h) in faces:
                roi = imageArray[y:y+h, x:x+w]
                xTrain.append(roi)
        yLabels.append(id_)

with open(u"labels", u"wb") as f:
  pickle.dump(labelIds, f)
  f.close()
recognizer.train(xTrain, np.array(yLabels))
recognizer.save(u"trainer.yml")
print labelIds
```

3. Code to run the facial recognition system and to allow access to registered faces

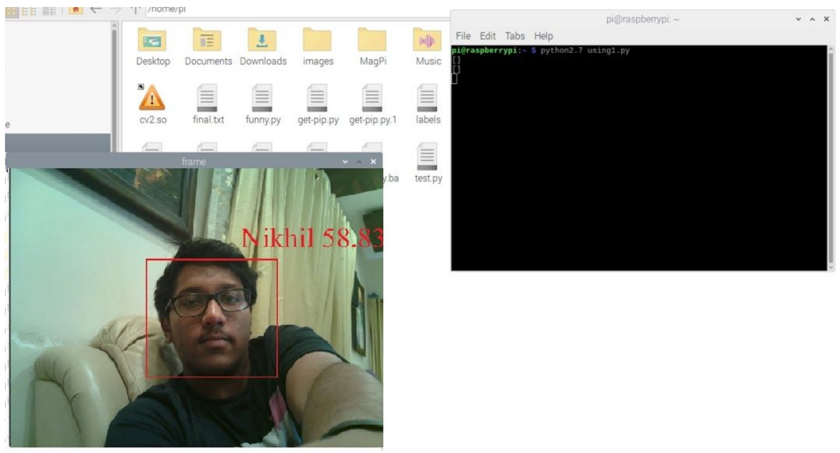
```
from __future__ import with_statement
from __future__ import absolute_import
import cv2
from picamera.array import PiRGBArray
from picamera import PiCamera
import numpy as np
import pickle
import RPi.GPIO as GPIO
from time import sleep
from datetime import datetime
from io import open
relay_pin = [26]
GPIO.setmode(GPIO.BCM)
GPIO.setup(relay_pin, GPIO.OUT)
GPIO.output(relay_pin, 0)
with open(u'labels', u'rb') as f:
  dict = pickle.load(f)
  f.close()
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 30
rawCapture = PiRGBArray(camera, size=(640, 480))
faceCascade =
cv2.CascadeClassifier(u"/home/pi/opencv/data/haarcascades_cuda/haarcascade_frontalface_defa
ult.xml")
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read(u"trainer.yml")
font = cv2.FONT_HERSHEY_SIMPLEX
lst = []
```

```
for frame in camera.capture_continuous(rawCapture, format=u"bgr", use_video_port=True):
    frame = frame.array
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray, scaleFactor = 1.5, minNeighbors = 5)
    for (x, y, w, h) in faces:
        roiGray = gray[y:y+h, x:x+w]
        id_, conf = recognizer.predict(roiGray)
        for name, value in dict.items():
            if value == id_ and conf<=100:
                print lst
                if (name in lst or len(lst)==0) and conf<=70:
                    lst.append(name)
                else:
                    lst=[]
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
                cv2.putText(frame, name + unicode(conf), (x, y), font, 2, (0, 0 ,255), 2,cv2.LINE_AA)
                if len(lst)>10:
                    print name,conf
                    now = datetime.now()
                    curr = now.strftime("%H:%M")
                    ent = name+"\t"+curr+"\n"
                    file=open("records.txt","a")
                    file.write(ent)
                    print curr
                    file.close()
                    GPIO.output(relay_pin, 1)
                    lst=[]

            else:
                GPIO.output(relay_pin, 0)
    cv2.imshow(u'frame', frame)
    key = cv2.waitKey(1)
    rawCapture.truncate(0)
    if key == 27:
        print "HEYYYYY"
        file = open("records.txt","r")
        contents = file.read()
        y=[]
        i=contents.split("\n")
        for x in i:
            y.append(x.split("\t"))
        i=0
        nf = open("final.txt","a")
        for i in range(0,len(y)-1):
            if y[i] != y[i+1]:
                n1 = (str(y[i])+"\n").decode('utf8')
                #n1=str(y[i])+"\n"
                nf.write(n1)
            else:
                pass
        nf.close()
        break
cv2.destroyAllWindows()
```

Result

After the model has been trained and deployed, we can use it for facial recognition and unlocking the Solenoid lock. When a person who is recognized is present in front of the camera, the Cascade classifier will detect a face and then begin checking if the face is recognizable. If it is, a box with the name of the person will be displayed and the Solenoid lock will open for a few seconds allowing the person to enter.



```
pi@raspberrypi:~ $ python2.7 using1.py
```

However, if the face of the person is not recognized, the camera will detect the face of this unrecognized person and but since the program is unaware of who it is, the lock is not opened. The person can add his face to the already existing model if he/she wishes to which can grant him access too.

Conclusion

The project's main goal is to automate the process that requires people working. The project eliminates the need for multiple security guards by allowing or preventing access to authorized people. It also comes with the added benefit of maintaining logs of when people enter the building. It records the time of the person's entry. The project is cost-effective and easy to use for anyone. It does have some drawbacks that include a low-quality camera, high power consumption, and non-compact design. There is room for improvement in areas like face detection, face recognition, and energy saving. Face detection can be made better by coding a better cascade classifier. Facial recognition can be improved by training better-angled images of the person and using a better code. Energy is drastically consumed as the camera is always on. We can avoid this by using a PIR sensor or which can detect the movement of a person and only then switch the camera on to detect faces. PIR sensor will consume less energy as compared to the Raspberry-pi.