

Face Detection Using Raspberry pi for Auto Lock-Unlock Operation.

Abdullah Jiroudi.

Smart security system is based on machine learning abilities for face detection and recognition. The user has full control besides instant notification. When unknown person is approaching user's property, recognition identity mechanism operates. This mechanism is performed by integrating hardware gadgets such as microcontroller, camera and motion lock with high-level programing language embedded in RASPBERRY PI.

CONTENTS	Pages
Header	1
Abstract	2
Contents.....	3
Project Goals.....	4
Equipment and Platforms	5
Part1: face recognition	6
1.1 detecting.....	6
1.2 Recognition.....	8
1.2.1 Data set creator.....	9
1.2.2 Traning.....	10
1.2.3 Identifying.....	12
Part 2 : Lunching the lock	13
Part 3 :Notification	14
Cashier Part.....	15
Chat Part.....	16
Settings Part.....	17
Email Part	21
Report Part.....	22
Dashboard Part.....	23
Database Design.....	24
Project Files And Folders	25
User Front End.....	27
Code Example Laravel	31
Code Example JS	36
Address Information.....	47

Project Goals

Combining the concept of face recognition with control systems, launch gates or trigger any physical machine with face is not enough, we wanted to give full control for owner.

Sometimes you may have a problem with machine learning process in your device so errors may occur, to avoid this issue you can have control by giving privileges to who you want.

Equipment and Platforms

Hardware:

1. **Microcontroller (Raspberry pi 3):** It is a tiny computer board that comes with CPU, GPU, USB ports, I/O pins, WiFi, Bluetooth, USB and network boot and is capable of doing some functions like a regular computer.
2. **Camera:** The Raspberry Pi Camera v2 is a high quality 5 megapixel Sony IMX219 image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It's capable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60 and 640x480p60/90 video. It attaches to Pi by way of one of the small sockets on the board upper surface and uses the dedicated CSI interface, designed especially for interfacing to cameras.
3. 12 Volts electric lock.
4. Electric switch: Transport 12 volts when 5 volts input is connected.
5. 12 volts charger.

Software:

1. **Python programming language:** Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace.
2. **OpenCV:** OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source BSD license.

3. **Bluetooth:** Bluetooth is a wireless technology standard for exchanging data between fixed and mobile devices over short distances using short-wavelength UHF radio waves in the industrial, scientific and medical radio bands, from 2.400 to 2.485 GHz, and building personal area networks.
4. **Ubuntu Mate.**

Part1

Face recognition

Detecting

If we want to recognize an object in an image we need to be able to detecting it in our frame, from hardware perspective this can be done by feeding or teaching your machine to what type of figures it must detect. In our case our object is human so we need deep learning course to what humans look like given to our machine

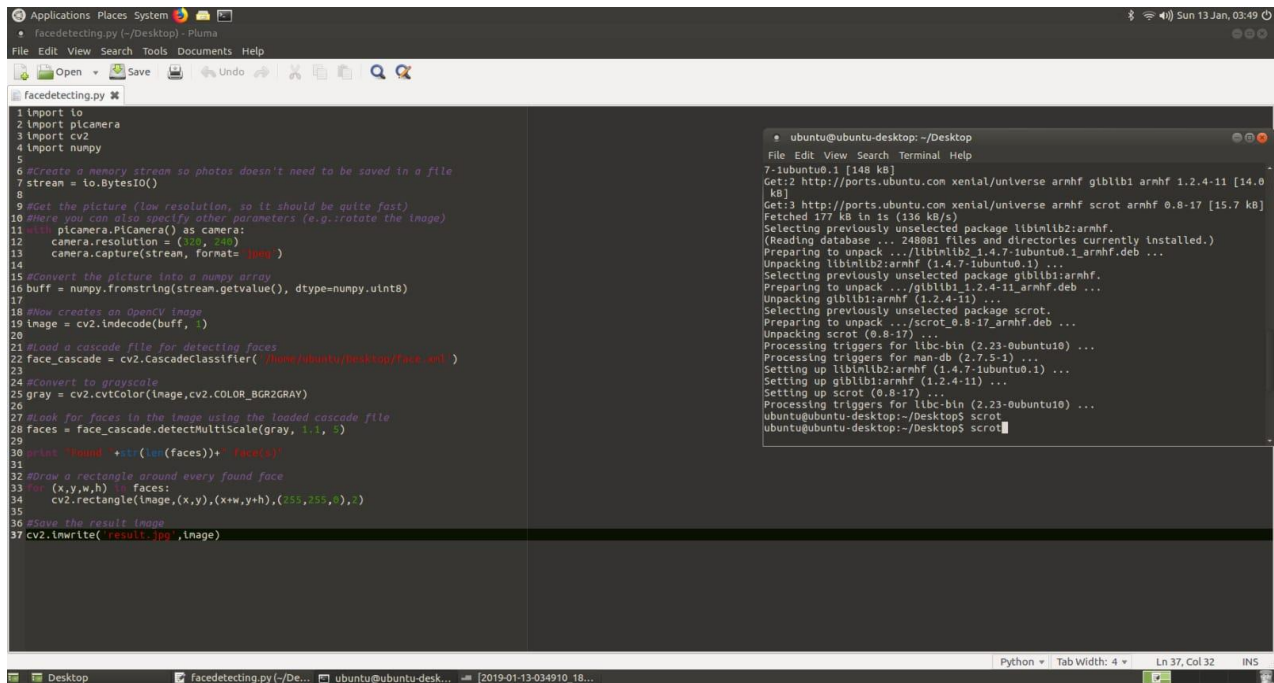
Intel provided many files and documents with thousands of lines of codes (faces) so they can be used in your code as deep learning step to many types of objects. Here is a part of the file

```
<opencv_storage>
<cascade type_id="opencv-cascade-classifier"><stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>24</height>
  <width>24</width>
  <stageParams>
    <maxWeakCount>211</maxWeakCount></stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount></featureParams>
  <stageNum>25</stageNum>
  <stages>
    <_>
      <maxWeakCount>9</maxWeakCount>
      <stageThreshold>-5.0425500869750977e+00</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 0 -3.1511999666690826e-02</internalNodes>
          <leafValues>
            2.0875380039215088e+00 -2.2172100543975830e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 1 1.2396000325679779e-02</internalNodes>
          <leafValues>
            -1.8633940219879150e+00 1.3272049427032471e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 2 2.1927999332547188e-02</internalNodes>
          <leafValues>
            -1.5105249881744385e+00 1.0625729560852051e+00</leafValues></_>
      </weakClassifiers>
    </_>
  </stages>
</cascade>
```

After feeding this file to our machine it is now aware to the objects it will detect.

Our code's function is to determine the wanted objects (human faces) in the frame, at the moment it detects an image it will capture and save it with **jpg** format. Detected faces will be surrounded with rectangular as emphasizing for detected faces.

We use Python programming language in coding.



Recognition

Recognition step has three sub-steps as follow:

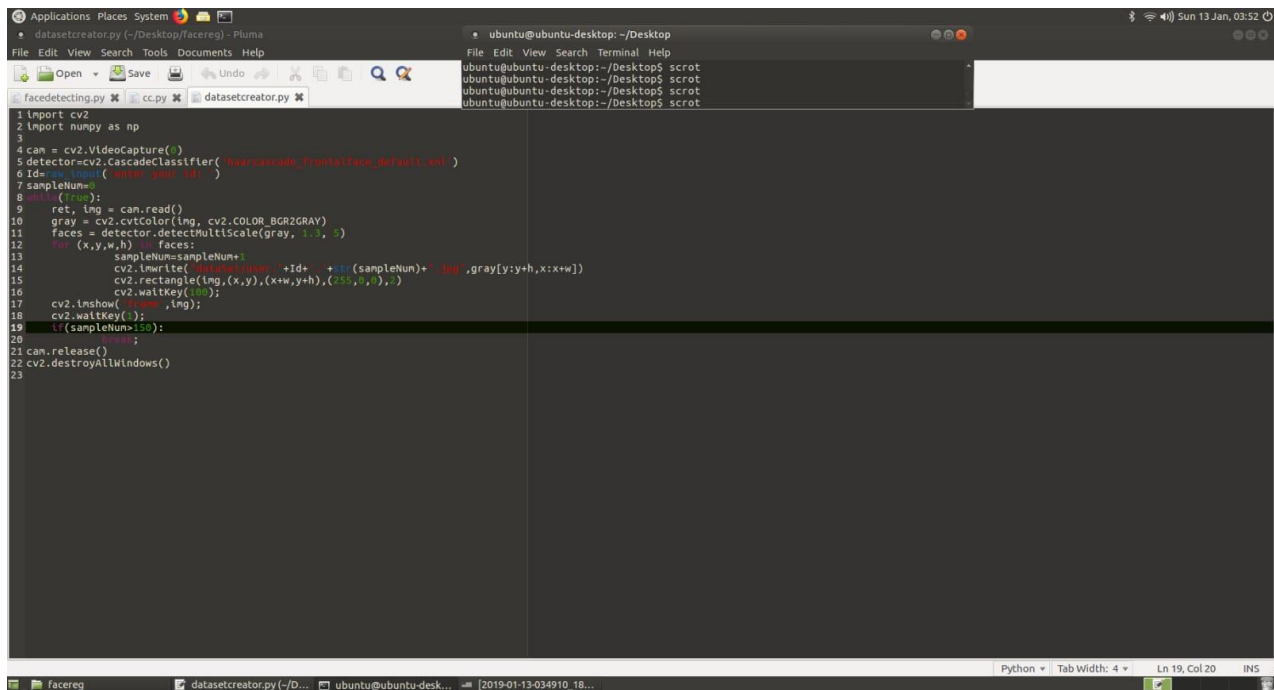
1-Dataset creator.

2-Training.

3-Identifier.

Dataset creator

Here we must have samples of the face we want to recognize later. samples are a group of images captured and saved in a file, more images will lead to more accuracy and speed in recognition.

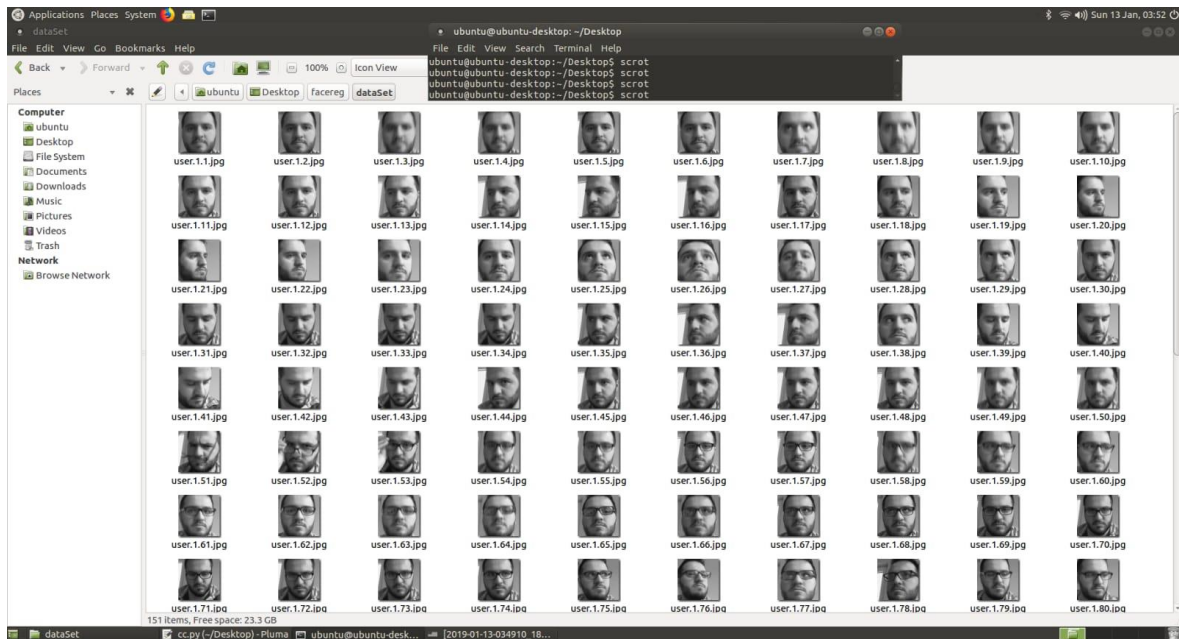


```

1 import cv2
2 import numpy as np
3
4 cam = cv2.VideoCapture(0)
5 detector=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
6 id=0
7 sampleNum=0
8 while(True):
9     ret, img = cam.read()
10    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
11    faces = detector.detectMultiScale(gray, 1.1, 5)
12    for (x,y,w,h) in faces:
13        sampleNum=sampleNum+1
14        cv2.imwrite("dataset/gray"+id+" "+(sampleNum)+".jpg",gray[y:y+h,x:x+w])
15        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
16        cv2.waitKey(100);
17    cv2.imshow('Frame',img);
18    cv2.waitKey(1);
19    id=(sampleNum%150);
20
21 cam.release()
22 cv2.destroyAllWindows()
23

```

Note: The samples are taken in gray scale range to detect the face then have samples to it. As shown below:

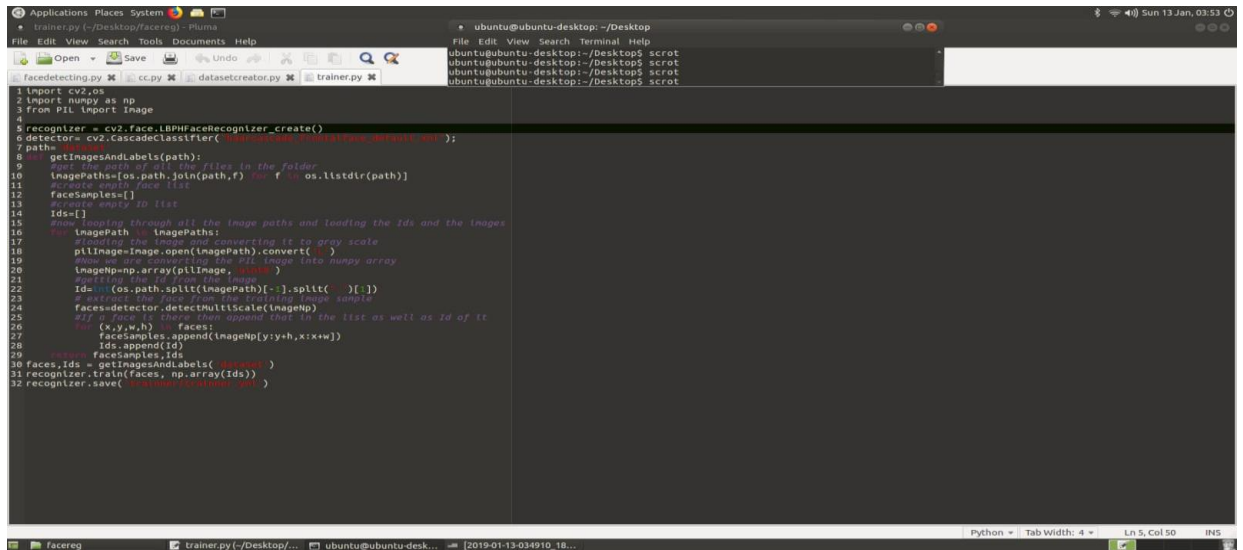


Training

As we noticed above, each set of samples has its own ID. In training, we save the samples in an array with ID for comparing in the last phase. Then, training is also saved in a **YML** file as a huge array in a gray scale range.

What is a YML file?

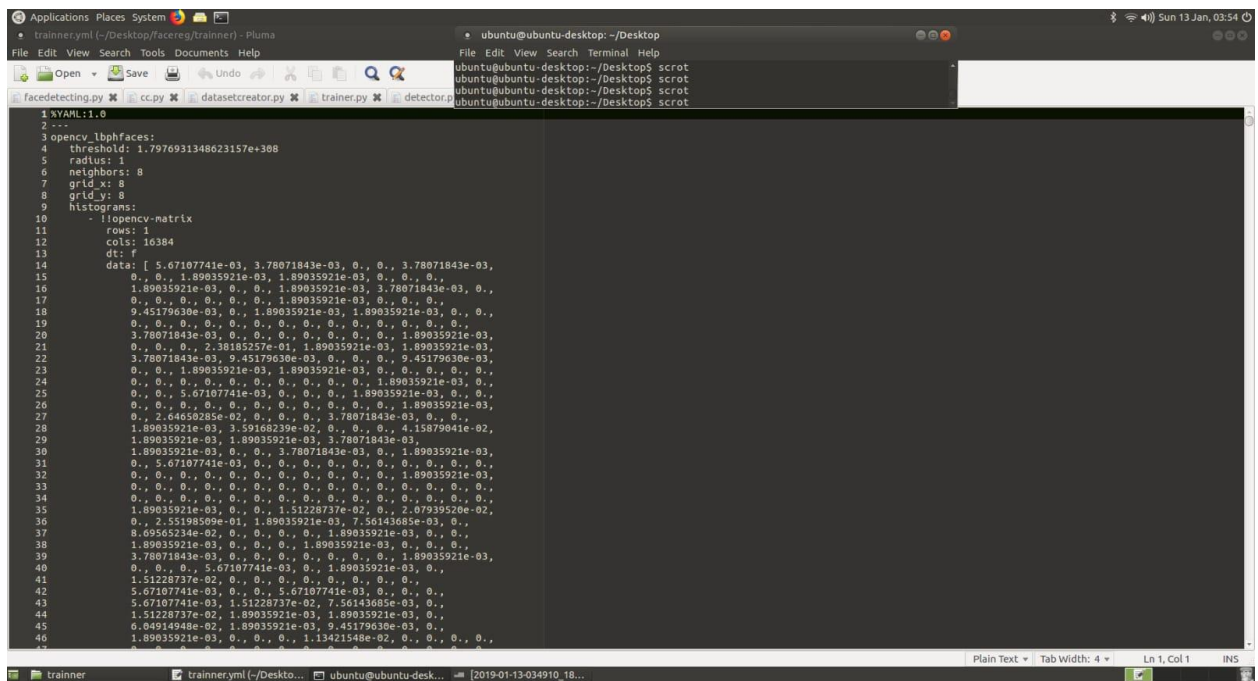
A file created in the YAML format, a human-readable data format used for data serialization, allows data to be written and read independent of any particular language. It can be integrated into many different programming languages using supporting YAML libraries, including C/C++, Ruby, Python, Java, Perl, C#, PHP, and others.



```

1 import cv2, os
2 import numpy as np
3 from PIL import Image
4
5 recognizer = cv2.FaceRecognizer_create()
6 detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
7
8 def getImagesAndLabels(path):
9     # Getting the image paths and labels in the folder
10    imagePath = os.path.join(path, '*')
11    # Getting the image paths
12    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
13    # Getting the image labels
14    faceSamples = []
15    # Getting the image IDs
16    ids = []
17    # Looping through all the image paths and loading the IDs and the images
18    for imagePath in imagePaths:
19        # Loading the image and converting it to gray scale
20        pilImage = Image.open(imagePath).convert('L')
21        # Converting the image into numpy array
22        imageNp = np.array(pilImage, dtype=np.uint8)
23        # Getting the image ID
24        id = os.path.splitext(imagePath)[-1].split('.')[1]
25        # Detecting the face in the image
26        faces = cv2.detectMultiScale(imageNp)
27        # If a face is there then append that to the list as well as ID of it
28        for (x,y,w,h) in faces:
29            faceSamples.append(imageNp[y:y+h,x:x+w])
30            ids.append(id)
31    return faceSamples, ids
32
33 faces, ids = getImagesAndLabels('dataset')
34 recognizer.train(faces, np.array(ids))
35 recognizer.save('trainer.yml')

```



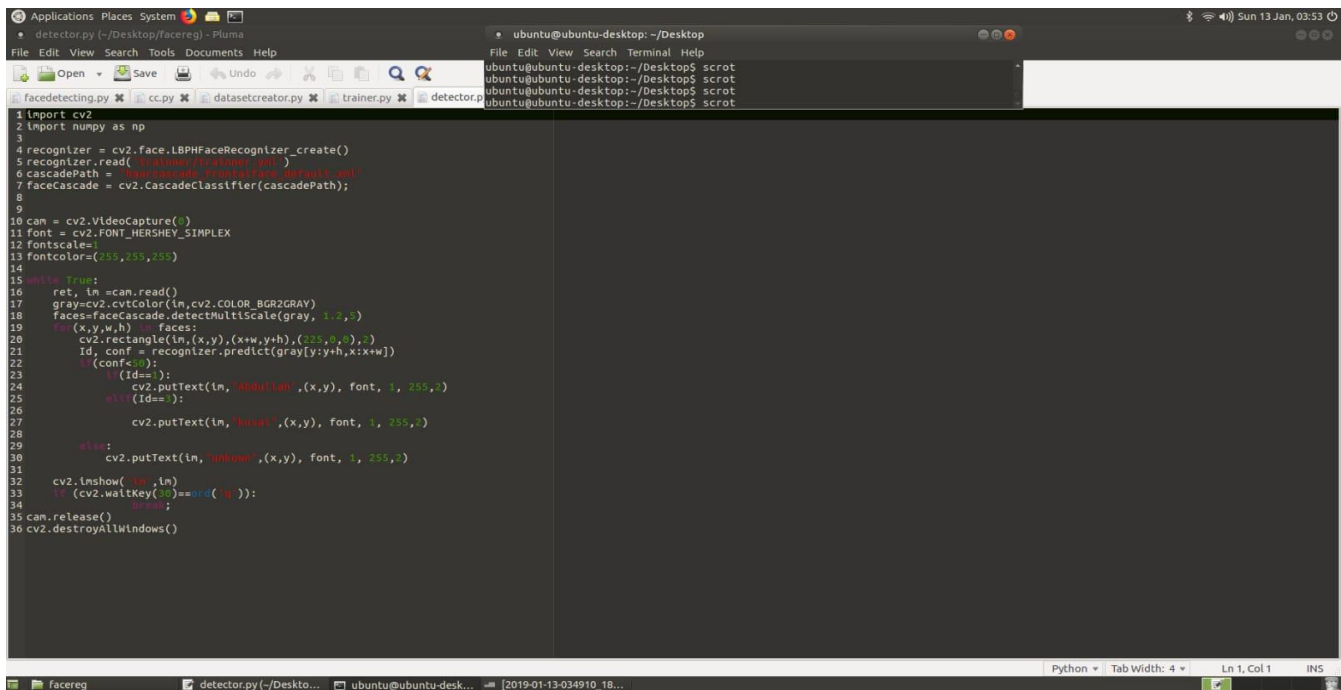
```

1 # YOLOv3
2
3 opencv_lbpFaces:
4   threshold: 1.7976931348623157e+308
5   radius: 1
6   neighbors: 8
7   grid_x: 8
8   grid_y: 8
9   histogram:
10    - llopencv-matrix
11      rows: 1
12      cols: 16384
13      data: [ 5.67107741e-03, 3.78071843e-03, 0., 0., 3.78071843e-03,
14             0., 0., 1.89035921e-03, 1.89035921e-03, 0., 0., 0.,
15             1.89035921e-03, 0., 0., 1.89035921e-03, 3.78071843e-03, 0.,
16             0., 0., 0., 0., 0., 1.89035921e-03, 0., 0., 0.,
17             9.45179630e-03, 0., 1.89035921e-03, 1.89035921e-03, 0., 0.,
18             0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
19             3.78071843e-03, 0., 0., 0., 0., 0., 1.89035921e-03,
20             0., 0., 0., 2.38185257e-01, 1.89035921e-03, 1.89035921e-03,
21             3.78071843e-03, 9.45179630e-03, 0., 0., 9.45179630e-03,
22             0., 0., 1.89035921e-03, 1.89035921e-03, 0., 0., 0., 0.,
23             0., 0., 0., 0., 0., 0., 0., 0., 1.89035921e-03, 0.,
24             0., 0., 5.67107741e-03, 0., 0., 0., 1.89035921e-03, 0.,
25             0., 0., 0., 0., 0., 0., 0., 0., 1.89035921e-03,
26             0., 2.44589285e-02, 0., 0., 0., 3.78071843e-03, 0., 0.,
27             1.89035921e-03, 1.59168239e-02, 0., 0., 4.15879841e-02,
28             1.89035921e-03, 1.89035921e-03, 3.78071843e-03,
29             1.89035921e-03, 0., 0., 3.78071843e-03, 0., 1.89035921e-03,
30             0., 5.67107741e-03, 0., 0., 0., 0., 0., 0., 0., 0.,
31             0., 0., 0., 0., 0., 0., 0., 0., 1.89035921e-03,
32             0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
33             0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
34             0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
35             1.89035921e-03, 0., 0., 1.51228737e-02, 0., 2.07939520e-02,
36             0., 2.55198509e-01, 1.89035921e-03, 7.56143685e-03, 0.,
37             8.69565234e-02, 0., 0., 0., 1.89035921e-03, 0., 0.,
38             1.89035921e-03, 0., 0., 1.89035921e-03, 0., 0., 0.,
39             3.78071843e-03, 0., 0., 0., 0., 0., 0., 1.89035921e-03,
40             0., 0., 5.67107741e-03, 0., 1.89035921e-03, 0.,
41             1.51228737e-02, 0., 0., 0., 0., 0., 0., 0.,
42             5.67107741e-03, 0., 0., 5.67107741e-03, 0., 0., 0.,
43             5.67107741e-03, 1.51228737e-02, 7.56143685e-03, 0.,
44             1.51228737e-02, 1.89035921e-03, 1.89035921e-03, 0.,
45             0.04914948e-02, 1.89035921e-03, 9.45179630e-03, 0.,
46             1.89035921e-03, 0., 0., 1.13421540e-02, 0., 0., 0.,
47             0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,

```

Identifying

After taking the samples and having training on them we will start our identifying process which simply having stream in the camera and determine the faces in the frame by surrounding them with rectangular and the person name .So simply it is comparing step between current stream faces and saved samples with Id's.



```

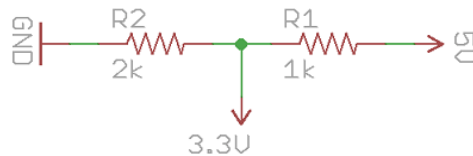
1 import cv2
2 import numpy as np
3
4 recognizer = cv2.face.LBPHFaceRecognizer_create()
5 recognizer.read('trainer/trainer.yml')
6 cascadePath = 'haarcascade_frontalface_default.xml'
7 faceCascade = cv2.CascadeClassifier(cascadePath);
8
9
10 cam = cv2.VideoCapture(0)
11 font = cv2.FONT_HERSHEY_SIMPLEX
12 fontscale=1
13 fontcolor=(255,255,255)
14
15 while True:
16     ret, im =cam.read()
17     gray=cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
18     faces=faceCascade.detectMultiScale(gray, 1.2,5)
19     for (x,y,w,h) in faces:
20         cv2.rectangle(im,(x,y),(x+w,y+h),(255,0,0),2)
21         Id, conf = recognizer.predict(gray[y:y+h,x:x+w])
22         if(conf<5):
23             if(Id==0):
24                 cv2.putText(im,'Abdulhamid',(x,y), font, 1, 255,2)
25             elif(Id==1):
26                 cv2.putText(im,'hassan',(x,y), font, 1, 255,2)
27
28         else:
29             cv2.putText(im,'unknown',(x,y), font, 1, 255,2)
30
31     cv2.imshow('cam',im)
32     if (cv2.waitKey(30) ==ord('q')):
33         break
34
35 cam.release()
36 cv2.destroyAllWindows()
  
```

Launching the Lock

In this part the camera is stand by to detect any object in the frame and apply comparison process with the database to output either matching or not matching result.

Everything starts when recognition occurs, this will trigger the pin responsible for launching the switch which takes 5 volts input and gives 12 volts output to unlock the electronic lock. Since Raspberry pi's pins can only give 3.3 volts we needed to create circuit of voltage difference from 3.3 to 5 volts as in figure 2.1:

Fig 2.1



This output will be delivered to switch to supply 12 volts for the lock.

Code for first mode(outside)

```
import cv2
import numpy as np
import RPi.GPIO as GPIO
import os
GPIO.setmode(GPIO.BCM)
GPIO.setup(13, GPIO.OUT)
from time import sleep

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('trainer/trainer.yml')
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);

cam = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_SIMPLEX
fontscale = 1
fontcolor = (255, 255, 255)
GPIO.output(13, GPIO.LOW)
head="new visitor came!!"
```

```

exit_yes=-1

GPIO.output(13, GPIO.LOW)

while exit_yes!=1:
    ret, im = cam.read()
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray, 1.2, 5)
    for (x, y, w, h) in faces:
        cv2.rectangle(im, (x, y), (x + w, y + h), (0, 0, 225), 2)
        Id, conf = recognizer.predict(gray[y:y + h, x:x + w])
        if (conf < 50):
            if (Id == 2):
                "The ID for the dataset was taken before"
                cv2.putText(im, "Abdullah", (x, y), font, 1, 255, 2)
                GPIO.output(13, GPIO.HIGH) "To trigger output pin"
                exit_yes=1
                sleep(5)
                GPIO.output(13, GPIO.LOW)
            else:
                GPIO.output(13, GPIO.LOW)
                cv2.putText(im, "unknown,Call owner please!!", (x, y), font, 1, 255, 2)
                cv2.imshow('im', im)
                if (cv2.waitKey(30) == ord('q')):
                    break
        if(exit_yes==1):
            break

cam.release()
cv2.destroyAllWindows()

```

In this mode user or owner has no control it just set the system to this mode, then system will wait for any visitor to check for matching or not matching .When matching detected it automatically open the door of the facility for five seconds then unlock it again for another visitor.

Notification

Second mode of our project, similar to mode 1 but here full control is given to the owner to allow to whoever is available to enter the facility.

Instead of launching the lock directly if match found, using gmail notification will be sent to the owner as message with name of the visitor, then via Bluetooth-based application installed on any smart device is utilized to launch the door ,with 1 to unlock and 0 to lock.

Following code explains the procedures step by step :

```
import cv2
import bluetooth
import numpy as np
import RPi.GPIO as GPIO
import os
from time import sleep

LOCK = 13
GPIO.setmode(GPIO.BCM) # programming the GPIO by BCM pin numbers. (like PIN40 as GPIO21)
GPIO.setwarnings(False)
GPIO.setup(LOCK, GPIO.OUT) # initialize GPIO21 (LOCK) as an output Pin
GPIO.output(LOCK, 0)
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('trainer/trainer.yml')
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);

cam = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_SIMPLEX
fontscale = 1
fontcolor = (255, 255, 255)
GPIO.output(13, GPIO.LOW)
head = "new visitor came!!!"
exit_yes = -1

while exit_yes != 1:
    ret, im = cam.read()
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray, 1.2, 5)
    for (x, y, w, h) in faces:
        cv2.rectangle(im, (x, y), (x + w, y + h), (0, 0, 225), 2)
        Id, conf = recognizer.predict(gray[y:y + h, x:x + w])
```

```

if (conf < 50):
if (Id == 2):
cv2.putText(im, "Abdullah", (x, y), font, 1, 255, 2)
body = "abdullah is ringing!!!"
os.system(" echo " + body + " | mail -s " + head + " gusai582@gmail.com") "Using Ubuntu's terminal
command syntax this code is responsible for sending the e-mail"

exit_yes = 1
else:
GPIO.output(13, GPIO.LOW)
cv2.putText(im, "unknown,Call owner please!!", (x, y), font, 1, 255, 2)

cv2.imshow('im', im)
if (cv2.waitKey(30) == ord('q')):
break
if (exit_yes == 1):
break
cam.release()
os.system("python blue.py") "Another python file is run ,which responsible for Bluetooth stage"
cv2.destroyAllWindows()

```

blue.py file:

```

import bluetooth
import RPi.GPIO as GPIO #calling for header file which helps in using GPIOs of PI
LOCK=13

GPIO.setmode(GPIO.BCM) #programming the GPIO by BCM pin numbers. (like PIN40 as GPIO21)
GPIO.setwarnings(False)
GPIO.setup(LOCK,GPIO.OUT) #initialize GPIO21 (LOCK) as an output Pin
GPIO.output(LOCK,0)

server_socket=bluetooth.BluetoothSocket( bluetooth.RFCOMM )

port = 1
server_socket.bind(("",port))
server_socket.listen(1)

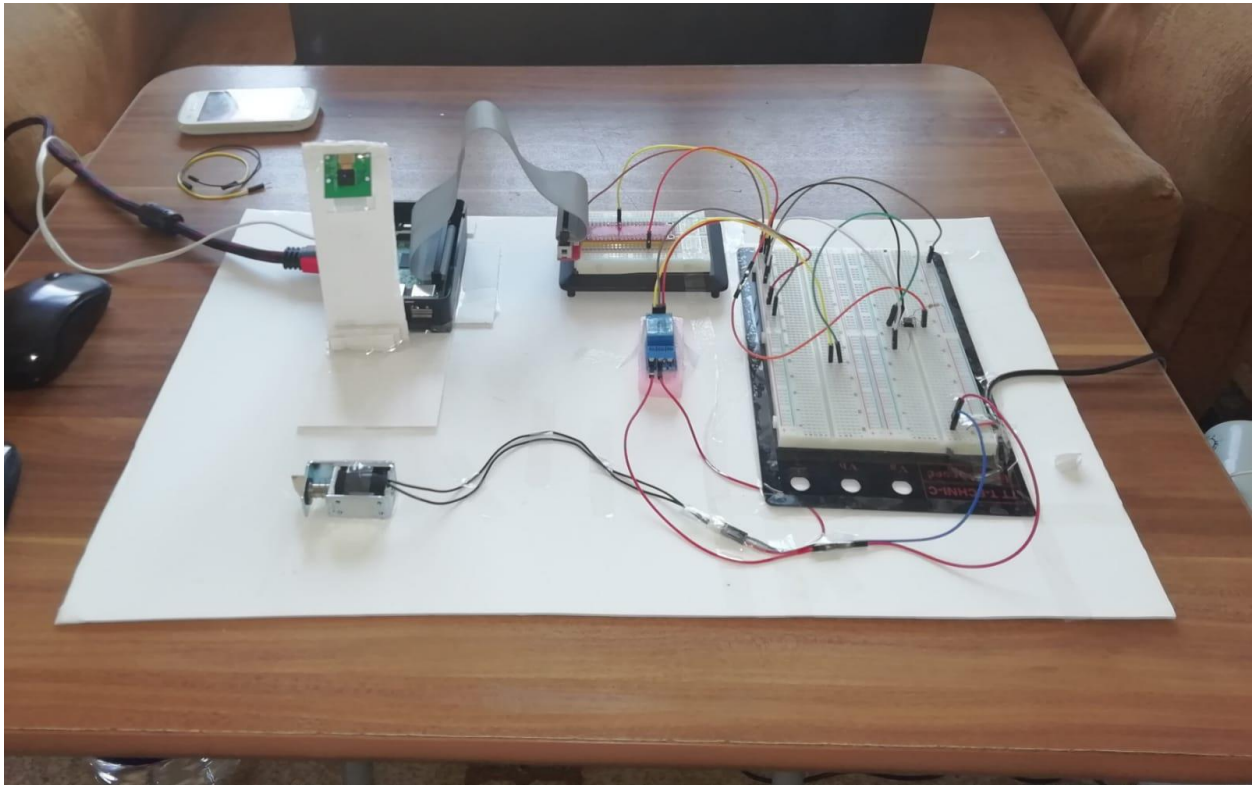
client_socket,address = server_socket.accept()
print "Accepted connection from ",address
while 1:

data = client_socket.recv(1024) " Variable has the received command from the app"
print "Received: %s" % data
if (data == "0"): #if '0' is sent from the Android App, turn OFF the LOCK
print ("GPIO 21 LOW, LOCK OFF")
GPIO.output(LOCK,0)
if (data == "1"): #if '1' is sent from the Android App, turn OFF the LOCK

```



```
print ("GPIO 21 HIGH, LOCK ON")
GPIO.output(LOCK,1)
if (data == "q"):
print ("Quit")
break
```



References:

- 1) Davies, Ellis, and Shepherd (eds.), *Perceiving and Remembering Faces*, Academic Press, London, 1981
- 2) M. Frodigh, P. Johansson and P. Larsson, “Wireless ad hoc networking- The art of networking without a network”, *Ericsson Review*, pp. 10-14, 2000
- 3) Raspberry Pi User Guide, 2nd Edition
[Eben Upton](#), [Gareth Halfacree](#)
ISBN: 978-1-118-79548-4
- 4) Girish Birajda, Shrikant Mahindrakar, “Embedded webserver based home automation using raspberry pi”, *International Journal of Modern Trends in Engineering and Research*, vol. 1, no.5
2014, India
- 5) <https://www.raspberrypi.org>