Door Unlock System using Facial Recognition

Abrstract: *This project presents a low cost and flexible Door unlock system using face identity. Simple and easy hardware implementation of face detection is done using Raspberry Pi, which itself is a minicomputer that uses Linux OS. Programming is done using python. Human face recognition procedure basically consists four phases: namely face detection, face normalization, face representation and face identification.*

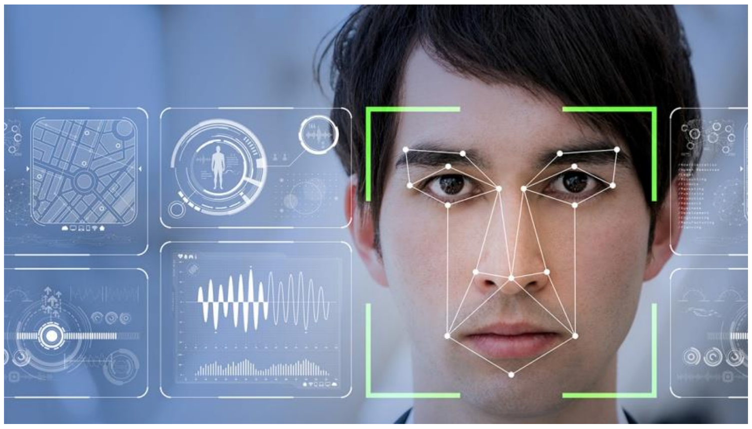
Chapter 1: Introduction:

Facial recognition is an advanced technology that helps in discerning and identifying human faces from an image or video. A system employed to perform facial recognition uses biometrics to map facial features from the photo or video. It compares this information with a large database of recorded faces to find a correct match.

Facial recognition is touted to be one of the top 3 methods of biometric recognition to identify people by measuring some aspect of individual physiology or anatomy. Facial recognition is the fastest-growing biometric technology and is expected to grow to $7.7 billion by 2022. This is because facial recognition has a wide range of commercial applications and is relatively simple to set up. It can be used for everything from surveillance to targeted marketing.

Facial recognition technology gained popularity in the early 1990s when the United States Department of Defense was seeking a technology that could spot criminals who furtively crossed borders. The Defense Department roped in eminent university scientists and experts in the field of facial recognition for this purpose by providing them with research financing.

Facial recognition made bold headlines in early 2001 immediately after it was first used in a public space—at Super Bowl XXXV in Tampa—by the law enforcement authorities to search for criminals and terrorists among the crowd of thousands of spectators. Soon after that, facial recognition systems were installed in other sensitive parts of the US to keep track of felonious activities.



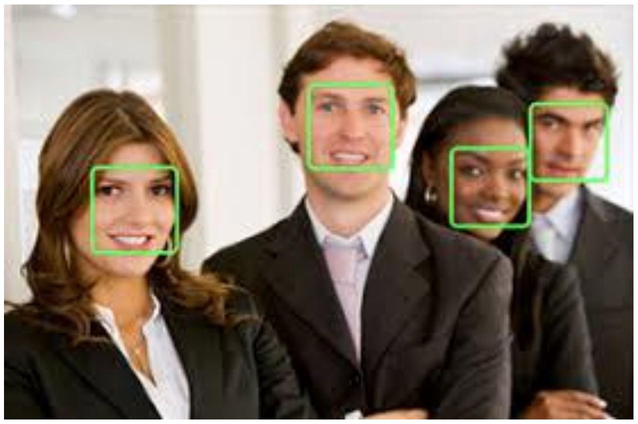
Chapter 2: How Facial Recognition Works

A facial recognition setup consists of advanced cameras that capture photos of people who pose or simply walk by, and sophisticated software working on those pictures will attempt to find the right match from the vast database to identify the person(s) in the image. Now, let’s take a closer look at the technical details of how these systems work.

Basic steps in facial identification

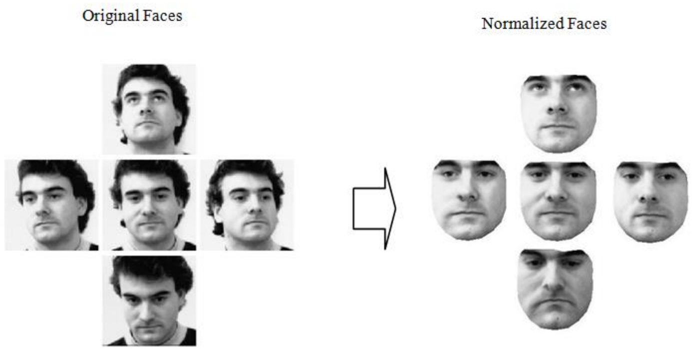
As mentioned earlier, facial recognition methods vary slightly, depending on the application and manufacturer, but they generally involve a series of steps that serve to capture, process, analyze and match the captured face to a database of recorded images. These basic steps are:

1. **Detection**: When the facial recognition system is attached to a video surveillance system, the recognition software scans the field of view of the camera for what it detects as faces. Upon the detection of each face-like image on a head-shaped form, it sends the face to the system to process it further. The system then estimates the head's position, orientation, and size. Generally, a face needs to be turned at least 35 degrees toward the camera for the camera to detect it.



Face detection

2. **Normalization**: The image of the captured face is scaled and rotated so that it can be registered and mapped into an appropriate pose and size. This is called normalization. After normalization, the software reads the geometry of the face by determining key factors, include the distance between the eyes, the thickness of the lips, the distance between the chin and the forehead, and many others. Some advanced face recognition systems use hundreds of such factors. The result of this processing leads to the generation of what is called a facial signature.



3. **Representation**: After forming the facial signature, the system converts it into a unique code. This coding facilitates easier computational comparison of the newly acquired facial data to stored databases of previously recorded facial data.

4. **Matching**: This is the final stage in which newly acquired facial data is compared to the stored data; if it matches with one of the images in the database, the software returns the details of the matched face and notifies the end user.

Chapter 3: Hardware description:

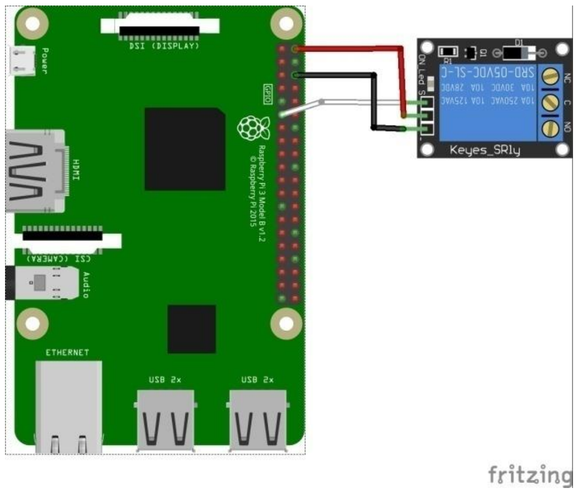
a. Raspberry PI: The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.

The Raspberry Pi 3 Model B+ is the final revision in the Raspberry Pi 3 range.

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- Extended 40-pin GPIO header
- Full-size HDMI
- 4 USB 2.0 ports
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- 4-pole stereo output and composite video port
- Micro SD port for loading your operating system and storing data
- 5V/2.5A DC power input
- Power-over-Ethernet (PoE) support (requires separate PoE HAT)

b. Relay : A relay is an electrically operated switch. It consists of a set of input terminals for a single or multiple control signals, and a set of operating contact terminals. The switch may have any number of contacts in multiple contact forms, such as make contacts, break contacts, or combinations thereof.

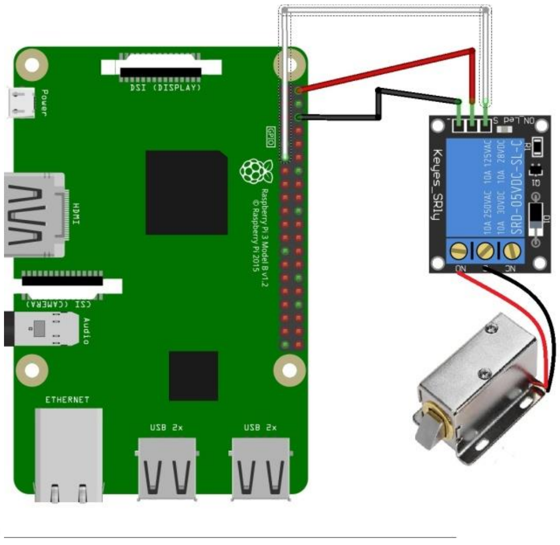
Relays are used where it is necessary to control a circuit by an independent low-power signal, or where several circuits must be controlled by one signal.



Connect Relay to Raspberry PI.

- Connect GND pin of the Relay with GND pin of the Raspberry PI
- Connect VCC pin of the Relay with 5V pin of the Raspberry PI
- Connect S pin of the Relay with GPIO17 pin of the Raspberry PI

c. **Electric Door Lock :** An electronic lock (or electric lock) is a locking device which operates by means of electric current. Electric locks are sometimes stand-alone with an electronic control assembly mounted directly to the lock. ... Electronic locks can also be remotely monitored and controlled, both to lock and to unlock.

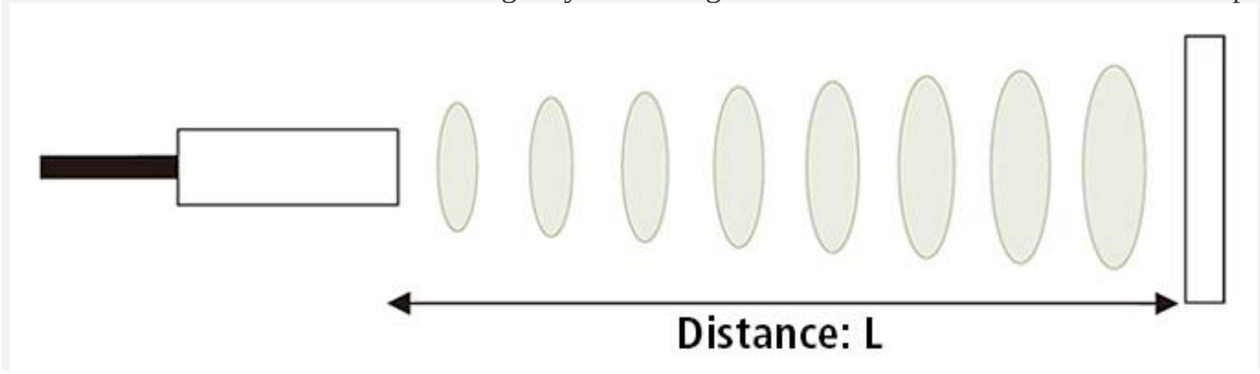


Connect Relay to Raspberry PI and electric door lock

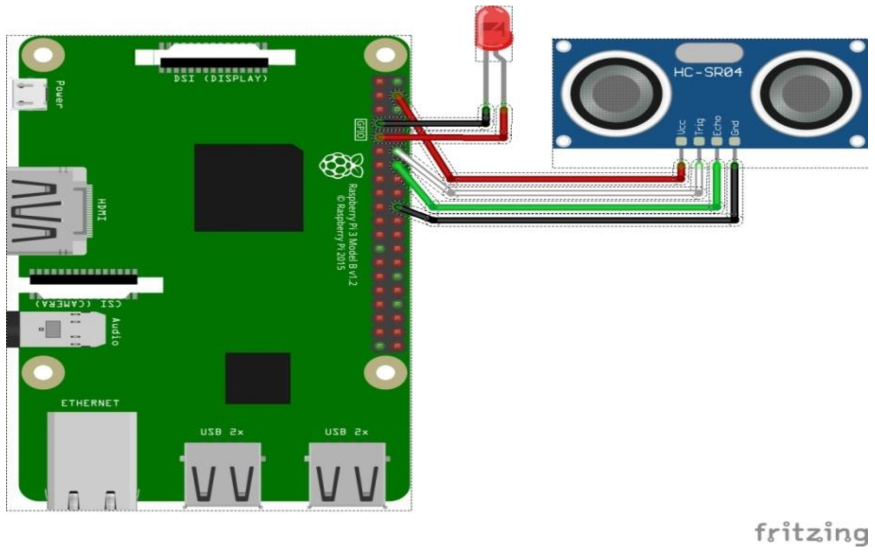
- Connect GND pin of the Relay with GND pin of the Raspberry PI
- Connect VCC pin of the Relay with 5V pin of the Raspberry PI
- Connect S pin of the Relay with GPIO17 pin of the Raspberry PI
- Connect Red wire of the Lock with C pin of the Relay.
- Connect Black wire of the Lock with ON pin of the Relay.

Ultrasonic Sensor: As the name indicates, ultrasonic sensors measure distance by using ultrasonic waves. The sensor head emits an ultrasonic wave and receives the wave reflected back from the target. Ultrasonic

Sensors measure the distance to the target by measuring the time between the emission and reception.



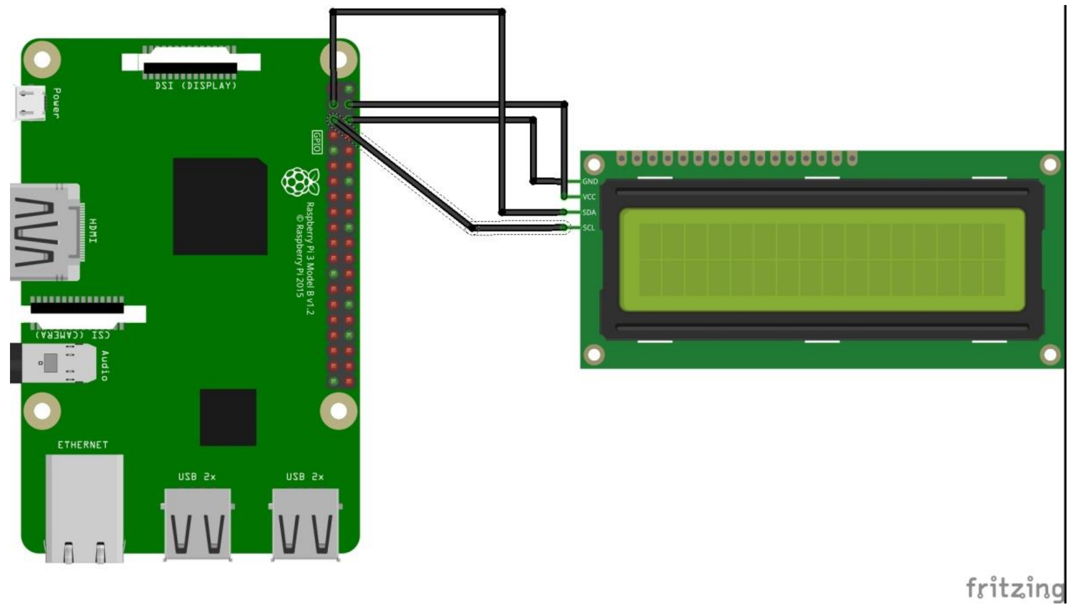
An optical sensor has a transmitter and receiver, whereas an ultrasonic sensor uses a single ultrasonic element for both emission and reception. In a reflective model ultrasonic sensor, a single oscillator emits and receives ultrasonic waves alternately. This enables miniaturization of the sensor head.



Connect Ultrasonic Sensor to Raspberry PI

- Connect GND pin of the Relay with GND pin of the Raspberry PI
- Connect VCC pin of the Relay with 5V pin of the Raspberry PI
- Connect Trig pin with GPIO pins 18 .
- Connect Echo pin with GPIO pin 24
- Connect Anode pin of LED with 4
- Connect Negetive pin of Led with GND

d. **LCD Display :** character LCD 16x4 display which is built in with ST7066 controller IC; its default interface is 6800 4/8-bit parallel, 5V power supply. These LCD display 16x4 modules are also available in SPI and I2C interface by using RW1063 controller IC.



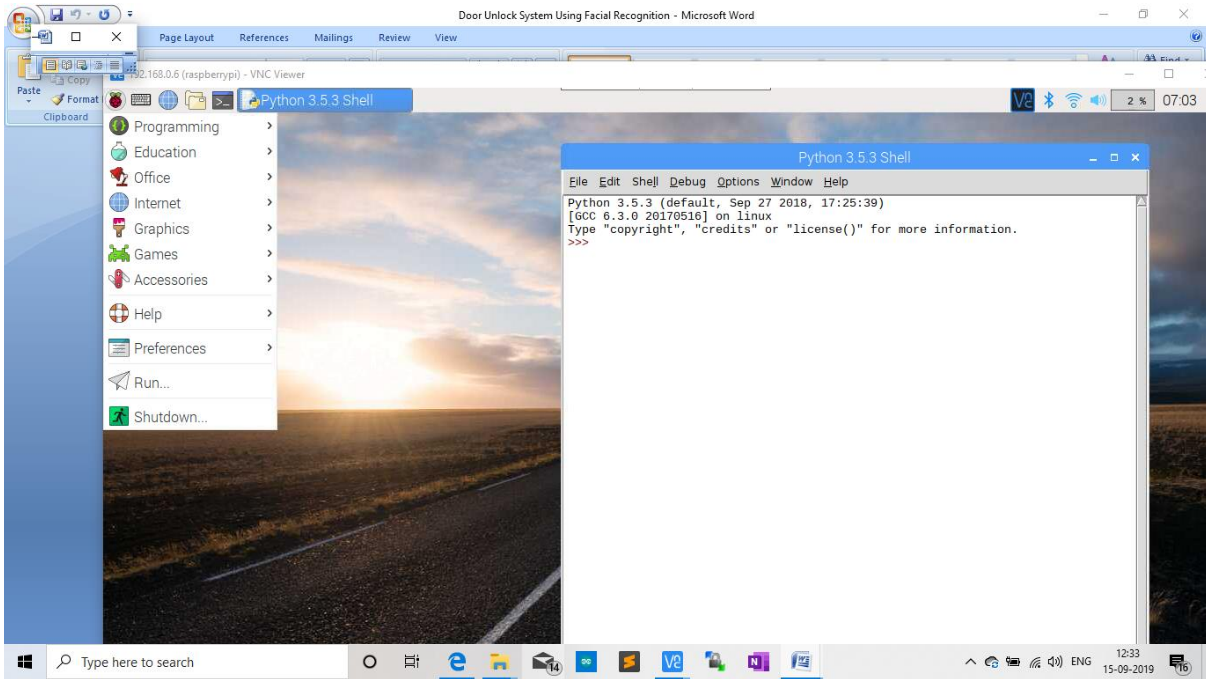
Connect LCD to Raspberry PI

- Connect GND pin of the Relay with GND pin of the Raspberry PI
- Connect VCC pin of the Relay with 5V pin of the Raspberry PI
- Connect SDA with GPIO pin SDA of the Raspberry PI
- Connect SCL with GPIO pin SCL of the Raspberry PI

e. **Camera Module :** This small camera is an accessory that has specifically been designed for Raspberry pi. The camera connects to the Raspberry Pi board using one of the two CSI sockets that are on the Raspberry Pi. The CSI interface has been specifically created to transfer data coming from a camera and has therefore a large bandwidth. The difference with the standard camera is that the Pi Noir does not use an infrared filter (NOIR = No Infrared).he camera board has a size of 25mm x 20mm x 9mm and a weight of 3g, making this camera module for Raspberry Pi a solution particularly suitable for embedded robotics. The camera has a resolution of 2592 x 1944 pixels for a static image and also supports the following videos resolutions: 1080p30, 720p60 and 640x480p60/90.The camera is supported by the last versions of Raspbian, the operating system of the Raspberry Pi.

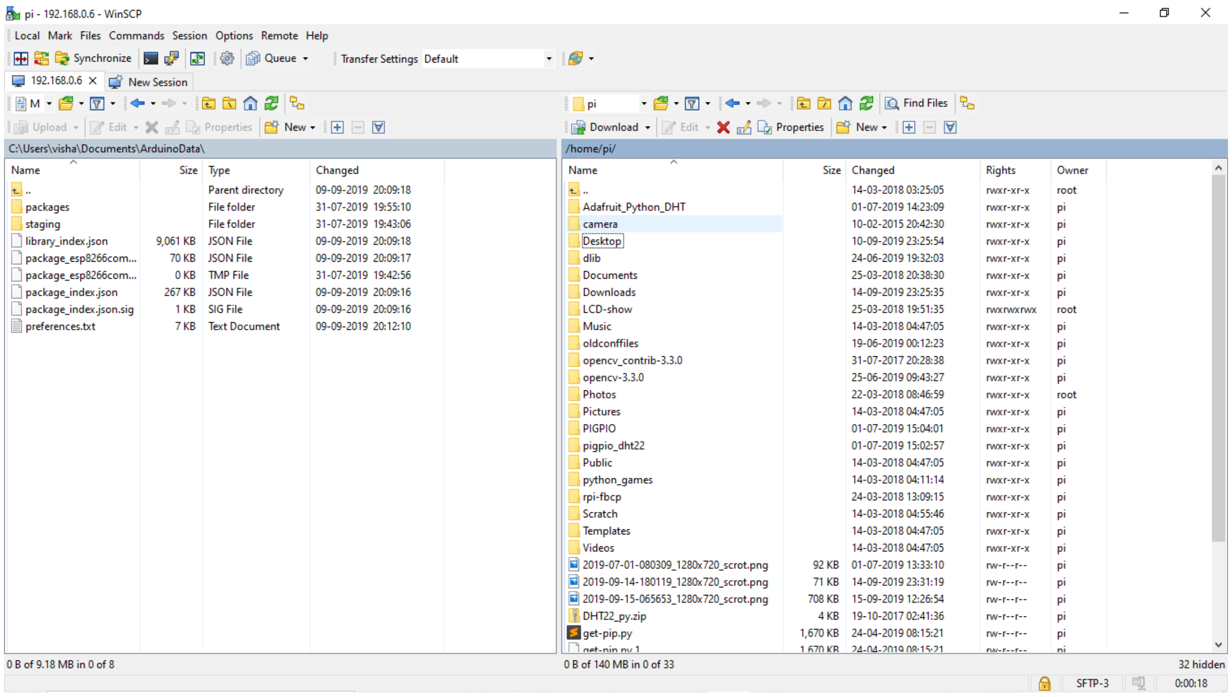
Chapter 4: **Software description:**

a. **VNC Viewer:** In computing, Virtual Network Computing is a graphical desktop-sharing system that uses the Remote Frame Buffer protocol to remotely control another computer. It transmits the keyboard and mouse events from one computer to another, relaying the graphical-screen updates back in the other direction, over a network.In this project VNC Viewer is used to control raspberry pi remotely and perform some operation.



```
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
```

b. **WinSCP :** WinSCP is a free and open-source SFTP, FTP, WebDAV, Amazon S3 and SCP client for Microsoft Windows. Its main function is secure file transfer between a local and a remote computer. Beyond this, WinSCP offers basic file manager and file synchronization functionality. WinSCP is used to transfer data from PC to raspberry pi.


pi - 192.168.0.6 - WinSCP
Local Mark Files Commands Session Options Remote Help
Synchronize
Queue
Transfer Settings Default
192.168.0.6
New Session
Upload
Edit
Properties
New
Download
Find Files

C:\Users\visha\Documents\ArduinoData\

Name	Size	Type	Changed
..		Parent directory	09-09-2019 20:09:18
packages		File folder	31-07-2019 19:55:10
staging		File folder	31-07-2019 19:43:06
library_index.json	9,061 KB	JSON File	09-09-2019 20:09:18
package_esp8266com...	70 KB	JSON File	09-09-2019 20:09:17
package_esp8266com...	0 KB	TMP File	31-07-2019 19:42:56
package_index.json	267 KB	JSON File	09-09-2019 20:09:16
package_index.json.sig	1 KB	SIG File	09-09-2019 20:09:16
preferences.txt	7 KB	Text Document	09-09-2019 20:12:10

0 B of 9.18 MB in 0 of 8

/home/pi/

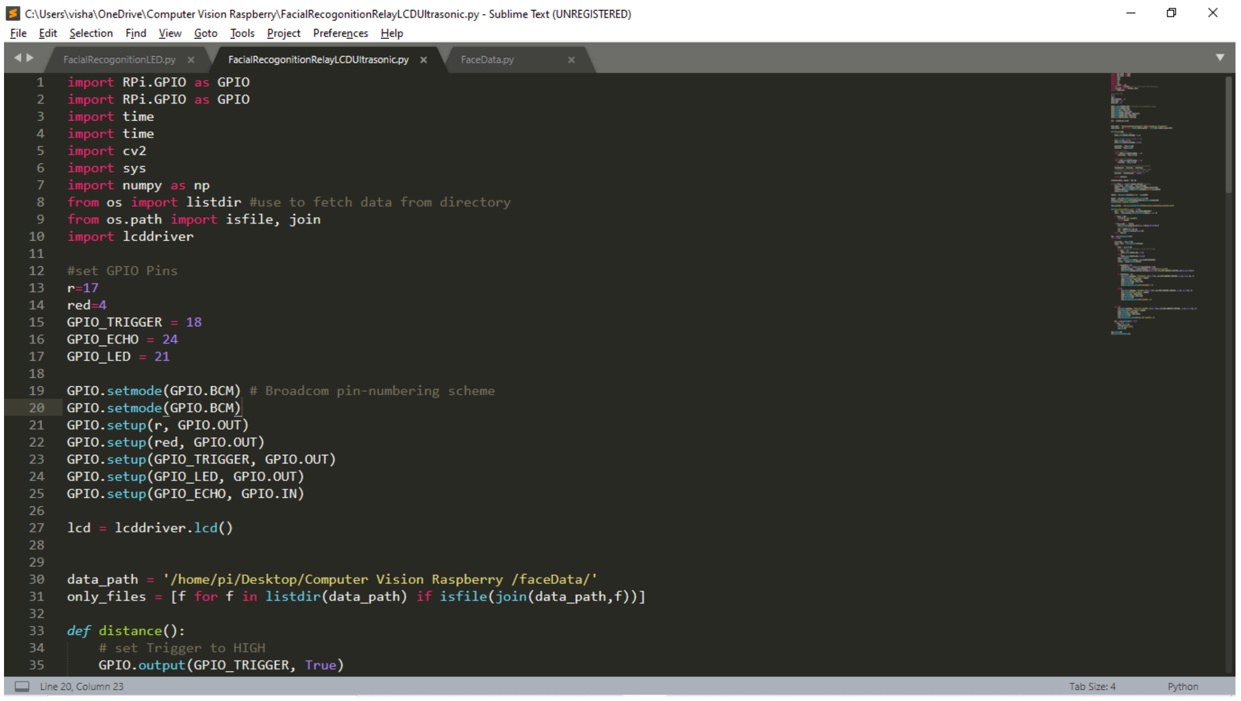
Name	Size	Changed	Rights	Owner
..		14-03-2018 03:25:05	rwxr-xr-x	root
Adafruit_Python_DHT		01-07-2019 14:23:09	rwxr-xr-x	pi
camera		10-02-2015 20:42:30	rwxr-xr-x	pi
Desktop		10-09-2019 23:25:54	rwxr-xr-x	pi
dlib		24-06-2019 19:32:03	rwxr-xr-x	pi
Documents		25-03-2018 20:38:30	rwxr-xr-x	pi
Downloads		14-09-2019 23:25:35	rwxr-xr-x	pi
LCD-show		25-03-2018 19:51:35	rwxrwxrwx	root
Music		14-03-2018 04:47:05	rwxr-xr-x	pi
oldconffiles		19-06-2019 00:12:23	rwxr-xr-x	pi
opencv_contrib-3.3.0		31-07-2017 20:28:38	rwxr-xr-x	pi
opencv-3.3.0		25-06-2019 09:43:27	rwxr-xr-x	pi
Photos		22-03-2018 08:46:59	rwxr-xr-x	root
Pictures		14-03-2018 04:47:05	rwxr-xr-x	pi
PIGPIO		01-07-2019 15:04:01	rwxr-xr-x	pi
pigpio_dht22		01-07-2019 15:02:57	rwxr-xr-x	pi
Public		14-03-2018 04:47:05	rwxr-xr-x	pi
python_games		14-03-2018 04:11:14	rwxr-xr-x	pi
rpi-fbcp		24-03-2018 13:09:15	rwxr-xr-x	pi
Scratch		14-03-2018 04:55:46	rwxr-xr-x	pi
Templates		14-03-2018 04:47:05	rwxr-xr-x	pi
Videos		14-03-2018 04:47:05	rwxr-xr-x	pi
2019-07-01-080309_1280x720_scrot.png	92 KB	01-07-2019 13:33:10	rw-r--r--	pi
2019-09-14-180119_1280x720_scrot.png	71 KB	14-09-2019 23:31:19	rw-r--r--	pi
2019-09-15-065653_1280x720_scrot.png	708 KB	15-09-2019 12:26:54	rw-r--r--	pi
DHT22_py.zip	4 KB	19-10-2017 02:41:36	rw-r--r--	pi
get-pip.py	1,670 KB	24-04-2019 08:15:21	rw-r--r--	pi
get-pip.py.1	1,670 KB	24-04-2019 08:15:21	rw-r--r--	pi

0 B of 140 MB in 0 of 33

32 hidden

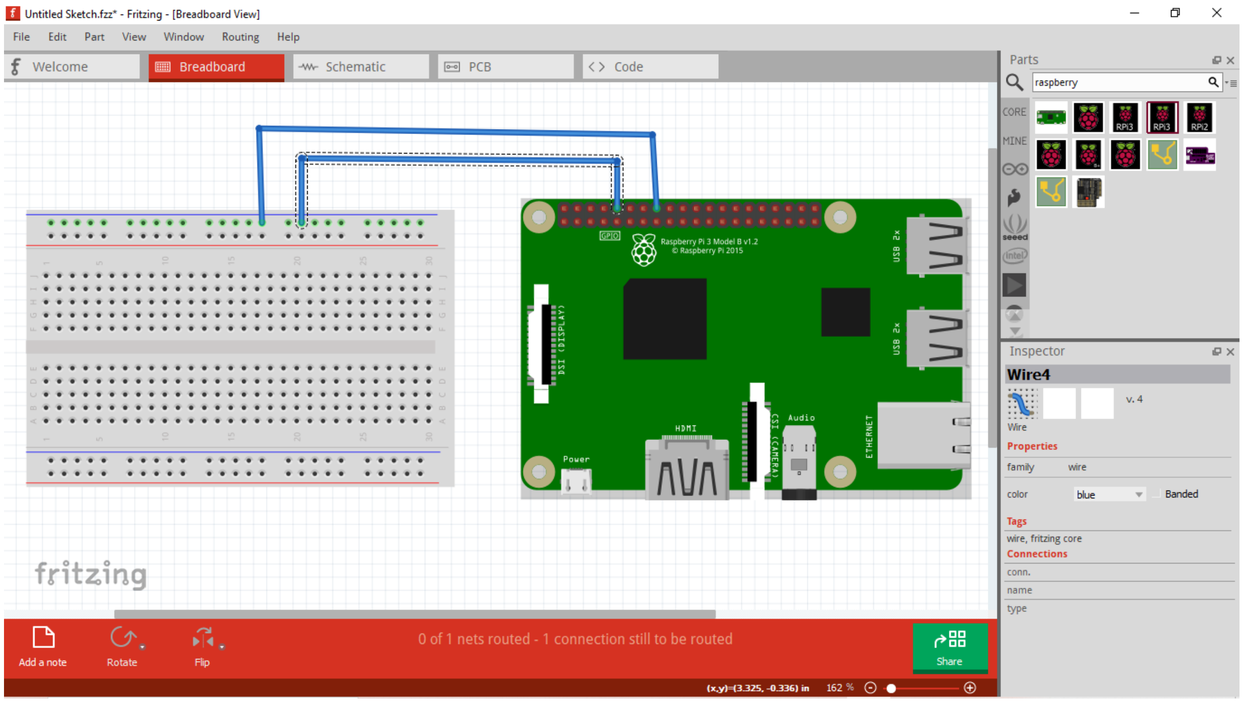
SFTP-3 0:00:18

c. **Sublime Text:** Sublime Text is a text editor written in C++ and Python available on windows, Mac and Linux. A text editor is a program developers write their code. In my project I used Sublime Text to write code on PC and then transfer code from my PC to the raspberry pi using WinSCP.


C:\Users\visha\OneDrive\Computer Vision Raspberry\FacialRecogonitionRelayLCDUltrasonic.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FacialRecogonitionLED.py
FacialRecogonitionRelayLCDUltrasonic.py
FaceData.py

```
import RPi.GPIO as GPIO
import RPi.GPIO as GPIO
import time
import time
import cv2
import sys
import numpy as np
from os import listdir #use to fetch data from directory
from os.path import isfile, join
import lcddriver

#set GPIO Pins
r=17
red=4
GPIO_TRIGGER = 18
GPIO_ECHO = 24
GPIO_LED = 21

GPIO.setmode(GPIO.BCM) # Broadcom pin-numbering scheme
GPIO.setmode(GPIO.BCM)
GPIO.setup(r, GPIO.OUT)
GPIO.setup(red, GPIO.OUT)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_LED, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)

lcd = lcddriver.lcd()


data_path = '/home/pi/Desktop/Computer Vision Raspberry /faceData/'
only_files = [f for f in listdir(data_path) if isfile(join(data_path,f))]

def distance():
    # set Trigger to HIGH
    GPIO.output(GPIO_TRIGGER, True)
```

d. **Fritzing :** Fritzing is an open-source initiative to develop amateur or hobby CAD software for the design of electronics hardware, to support designers and artists ready to move from experimenting with a prototype to building a more permanent circuit. All the project design and Schematics are shown using Fritzing.

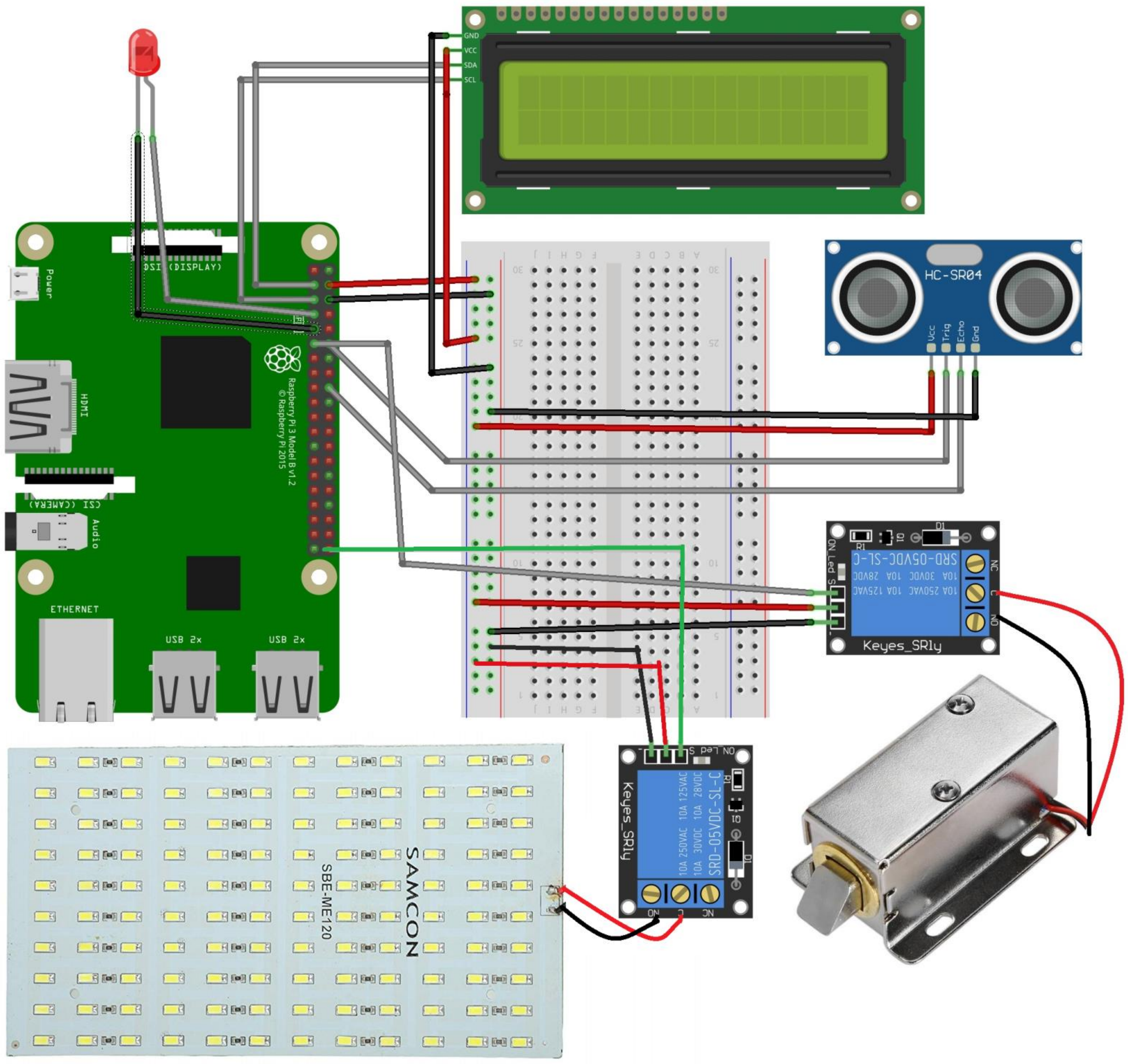


Properties
family wire
color blue Banded
Tags
wire, fritzing core
Connections
conn.
name
type

0 of 1 nets routed - 1 connection still to be routed

(x,y)=(3.325, -0.336) in 162 %

Chapter 5: Project with Source Code & Schematics:



Working Principle:

In this project, i have used Linux OS based Raspberry pi as my main component to running the facial recognition program. I have chosen this component because it is compact user friendly and very helpful in managing sensor, actuator and other IOT devices.

In this system a pie camera has been install into the slot provided in the board for taking images, videos and face detection to perform some specific task.

The complete system start working when image capture and it matches the golden image then the raspberry pi gives voltage output of 3volt GPIO17 which start a relay and that operate the electric door lock.

There is a 20x4 Character LCD Display which show the output of facial identities depending on the condition. For example, if face is match to golden image then it show “Door unlock” , else it show “Door lock” or if image cannot read buy camera it show “Image not match”.

This System work fine at day light, but find some problem in detecting at low light, so to encounter this problem, I have used an Ultrasonic sensor to overcome low light problem firstly, set the range of 120 cm and if the user be in this range then it turn GPIO4 to HIGH and then LED light which is connected to GPIO4 turn ON and camera get a clean image.

Project Code:

Part 1: Collecting the data Set first

- Step 1- Import OpenCV libarary
- Step 2- Capture video feed and resize it to 200x200
- Step 3- Detect and draw rectangle around face.
- Step 4- Convert Image from BGR2GRAY
- Step 5- Using imwrite() collect face sample to train our model
- Step 6- Break the loop after collecting 150 gray scale face sample.

```python
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)

def face_extractor(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    if faces is():
        print('Face NOt Found!')
        return None

    for(x,y,w,h) in  faces:
        cropped_face = img[y:y+h, x:x+w]

    return cropped_face

count = 0
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 = 'F:/Artificial Intelligence/Computer Vision/Computer Vision Project/face data/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 cropped',face)
    else:
        print("Face Not Found!")
        pass

    if cv2.waitKey(1) == 13 or count == 150:
        break
cap.release()
cv2.destroyAllWindows()
print('Collecting Sample Completed!!!')
```

Part 2: Train a Model to predict our face

- Step 1- Import OS, Numpy & OpenCV libarary
- Step 2- Store face sample in the 'only_files' list
- Step 3- iterate through 'only_files' and read as a gray scale.
- Step 4- Store face data as an array
- Step 5- Create face recognition model using LBPHFaceRecognizer
- Step 6- Train the model by providing training data and label and print after completion.

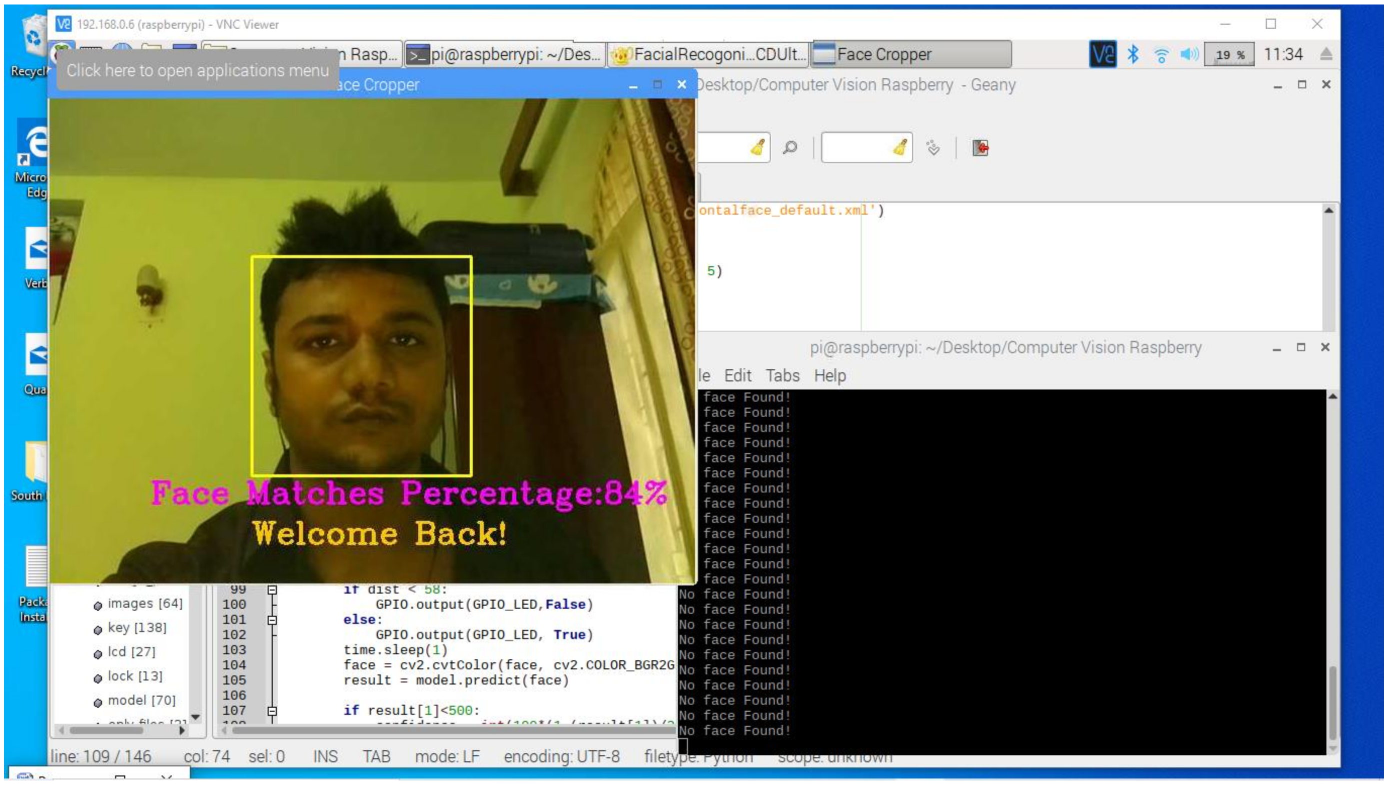
```python
import cv2
import numpy as np
from os import listdir #use to fetch data from directory
from os.path import isfile, join

data_path = 'F:/Artificial Intelligence/Computer Vision/Computer Vision Project/face data/'
only_files = [f for f in listdir(data_path) if isfile(join(data_path,f))]

Training_Data, Labels = [], []

for i, files in enumerate(only_files): #itterate
    image_path = data_path + only_files[i]
    images = cv2.imread(image_path,cv2.IMREAD_GRAYSCALE)
    Training_Data.append(np.asarray(images,dtype=np.uint8))
    Labels.append(i)

Labels = np.asarray(Labels,dtype = np.int32)

model = cv2.face.LBPHFaceRecognizer_create()
model.train(np.asarray(Training_Data),np.asarray(Labels))
print("Model Training Complete")
```

Part 3: Test the model:

- Step 1- Import Import RPi libarary for working on Raspberry PI GPIO.
- Step 2- Define GPIO used and set Input or Output for pins.
- Step 3- Read the Ultrasonic Sensor and set the range to turn on LED.
- Step 4- find out confident & if confident greater then 80% unlock door.
- Step 5- Show the output of Status of door on LCD Display.
- Step 6- perform GPIO clean up, camera release and close all window.

```python
import RPi.GPIO as GPIO
import time
import cv2
import sys
import numpy as np
from os import listdir #use to fetch data from directory
from os.path import isfile, join
import lcddriver

#set GPIO Pins
lock=17
red=4
GPIO_TRIGGER = 18
GPIO_ECHO = 24
GPIO_LED = 21
GPIO.setmode(GPIO.BCM) # Broadcom pin-numbering scheme
GPIO.setmode(GPIO.BCM)
GPIO.setup(lock, GPIO.OUT)
GPIO.setup(red, GPIO.OUT)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_LED, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)
lcd = lcddriver.lcd()

data_path = '/home/pi/Desktop/Computer Vision Raspberry /faceData/'
only_files = [f for f in listdir(data_path) if isfile(join(data_path,f))]

def distance():
    # set Trigger to HIGH
    GPIO.output(GPIO_TRIGGER, True)
    # set Trigger after 0.01ms to LOW
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)
    StartTime = time.time()
    StopTime = time.time()

    # save StartTime
    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()

    # save time of arrival
    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()

    # time difference between start and arrival
    TimeElapsed = StopTime - StartTime
    # multiply with the sonic speed (34300 cm/s)
    # and divide by 2, because there and back
    distance = (TimeElapsed * 34300) / 2
    return distance

Training_Data, Labels = [], []

for i, files in enumerate(only_files): #itterate
    image_path = data_path + only_files[i]
    images = cv2.imread(image_path,cv2.IMREAD_GRAYSCALE)
    Training_Data.append(np.asarray(images,dtype=np.uint8))
    Labels.append(i)

Labels = np.asarray(Labels,dtype = np.int32)

model = cv2.face.LBPHFaceRecognizer_create()
model.train(np.asarray(Training_Data),np.asarray(Labels))
print("Model Training Complete")

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

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

    if faces is():
        print("No face Found!")
        return img,[]

    for(x,y,w,h) in  faces:
        cv2.rectangle(img,(x,y),(x+w, y+h),(0,255,255),2)
        #region of interest
        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:
        dist = distance()
        #print ("Measured Distance = %.1f cm" % dist)
        if dist < 58:
            GPIO.output(GPIO_LED,False)
        else:
            GPIO.output(GPIO_LED, True)
        time.sleep(1)
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        result = model.predict(face)

        if result[1]<500:
            confidence = int(100*(1-(result[1])/300))
            display_string = 'Face Matches Percentage:'+str(confidence)+"%"
                cv2.putText(image,display_string,(100, 400), cv2.FONT_HERSHEY_COMPLEX,1,(255,0,255),2)

            if confidence >75:
                cv2.putText(image, "Welcome Back!", (200, 440), cv2.FONT_HERSHEY_COMPLEX, 1, (0,196, 255), 2)
                cv2.imshow('Face Cropper', image)
                GPIO.output(lock, GPIO.LOW)
                GPIO.output(red, GPIO.LOW)
                lcd.lcd_clear()
                lcd.lcd_display_string("Door Unlock:", 1)

            else:
                cv2.putText(image, "Ring the Door Bell!", (200, 440), cv2.FONT_HERSHEY_COMPLEX, 1, (155, 87, 255), 2)
                cv2.imshow('Face Cropper', image)
                GPIO.output(lock, GPIO.HIGH)
                GPIO.output(red, GPIO.LOW)
                lcd.lcd_clear()
                lcd.lcd_display_string("Ring the Door Bell!", 1)
        except:
            cv2.putText(image, "Face Not Found!", (200, 440), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2)
            cv2.imshow('Face Cropper', image)
            GPIO.output(lock, GPIO.HIGH)
            GPIO.output(red, GPIO.HIGH)
            lcd.lcd_clear()
            lcd.lcd_display_string("Face Not Found:", 1)

        key = cv2.waitKey(1) & 0xFF
        if key==27:  # ESC
            GPIO.cleanup()
            print("Clean Up!")
            sys.exit()

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

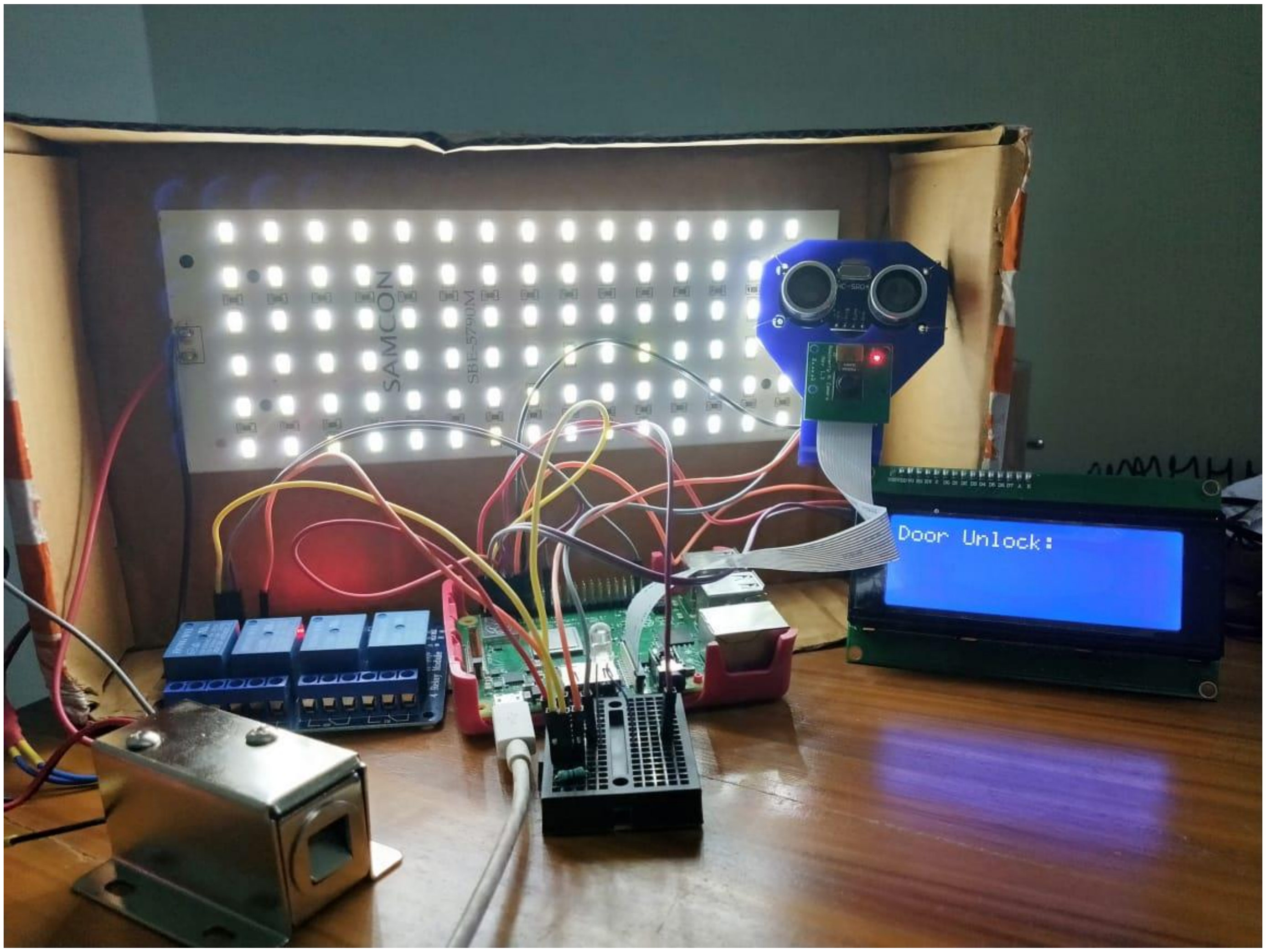
Chapter 6: Screenshot and output :

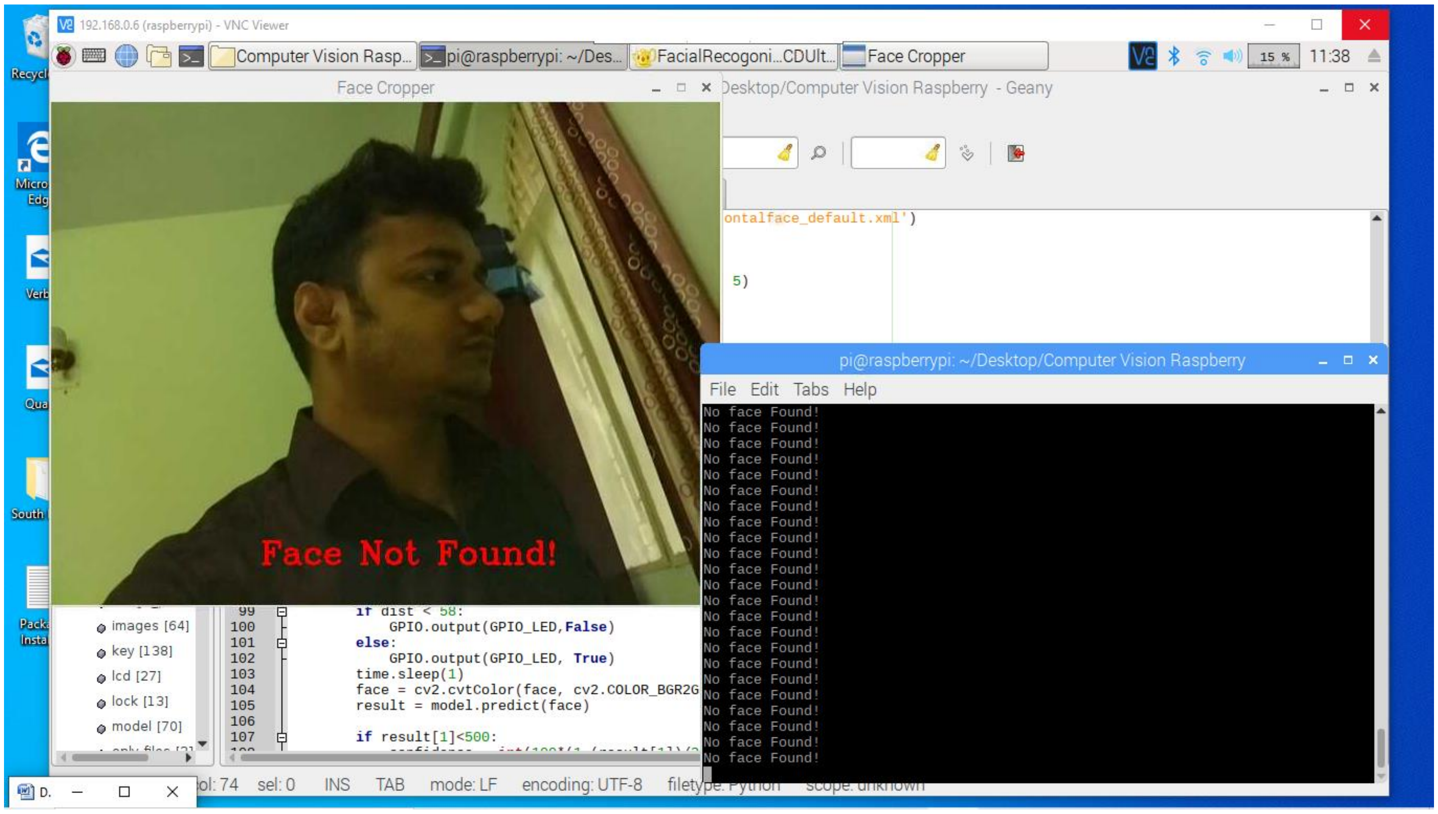

192.168.0.6 (raspberrypi) - VNC Viewer
Click here to open applications menu
pi@raspberrypi: ~/Des...
FacialRecogoni...CDUlt...
Face Cropper
11:34
Desktop/Computer Vision Raspberry - Geany
pi@raspberrypi: ~/Desktop/Computer Vision Raspberry
Face Matches Percentage:84%
Welcome Back!
images [64]
key [138]
lcd [27]
lock [13]
model [70]
line: 109 / 146 col: 74 sel: 0 INS TAB mode: LF encoding: UTF-8 filetype: Python scope: unknown

```
ontalface_default.xml')

5)
```

```
    if dist < 58:
        GPIO.output(GPIO_LED,False)
    else:
        GPIO.output(GPIO_LED, True)
    time.sleep(1)
    face = cv2.cvtColor(face, cv2.COLOR_BGR2G
    result = model.predict(face)

    if result[1]<500:
```

```
face Found!
face Found!
face Found!
face Found!
face Found!
face Found!
face Found!
face Found!
face Found!
face Found!
face Found!
face Found!
face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
```

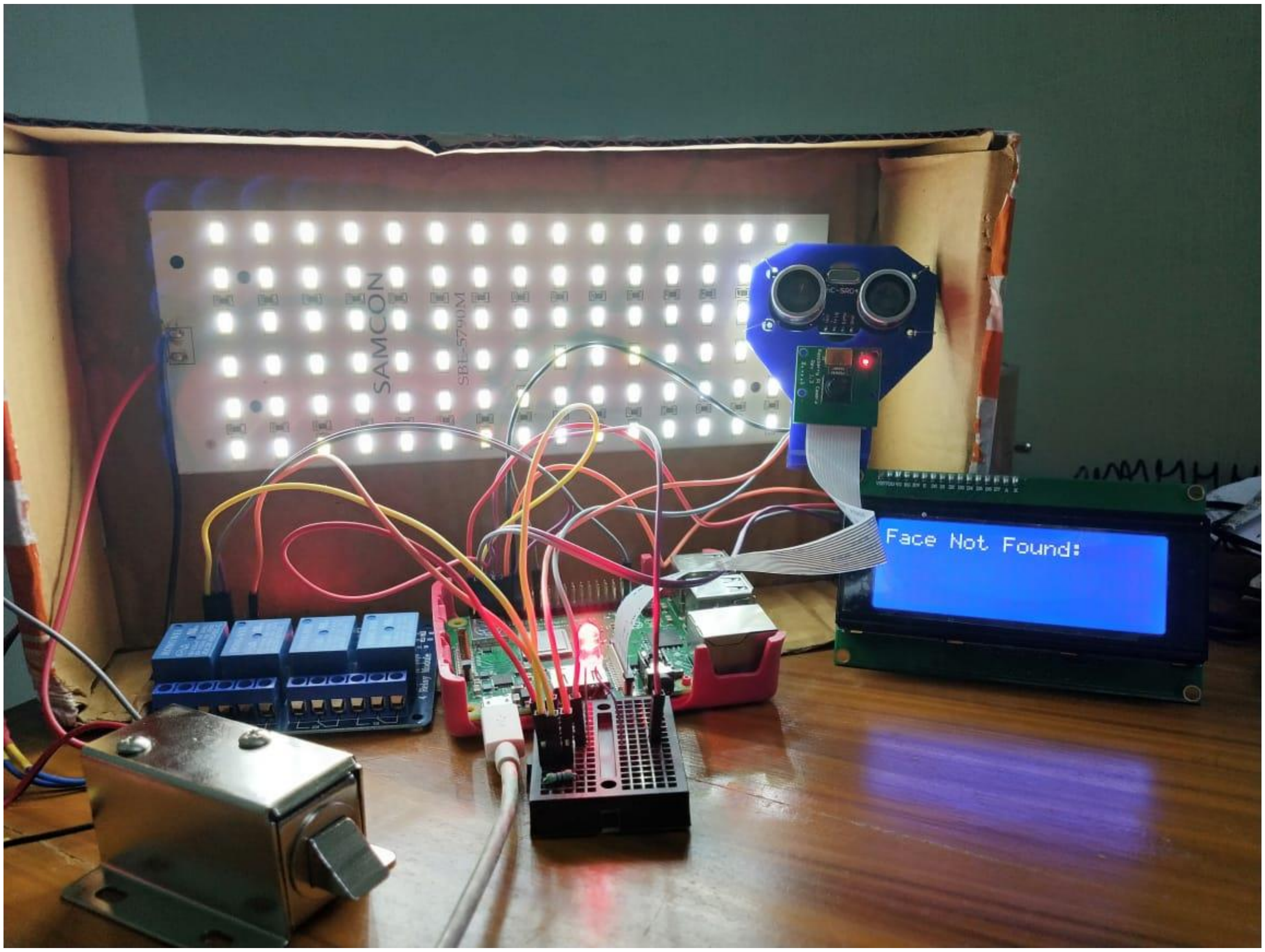


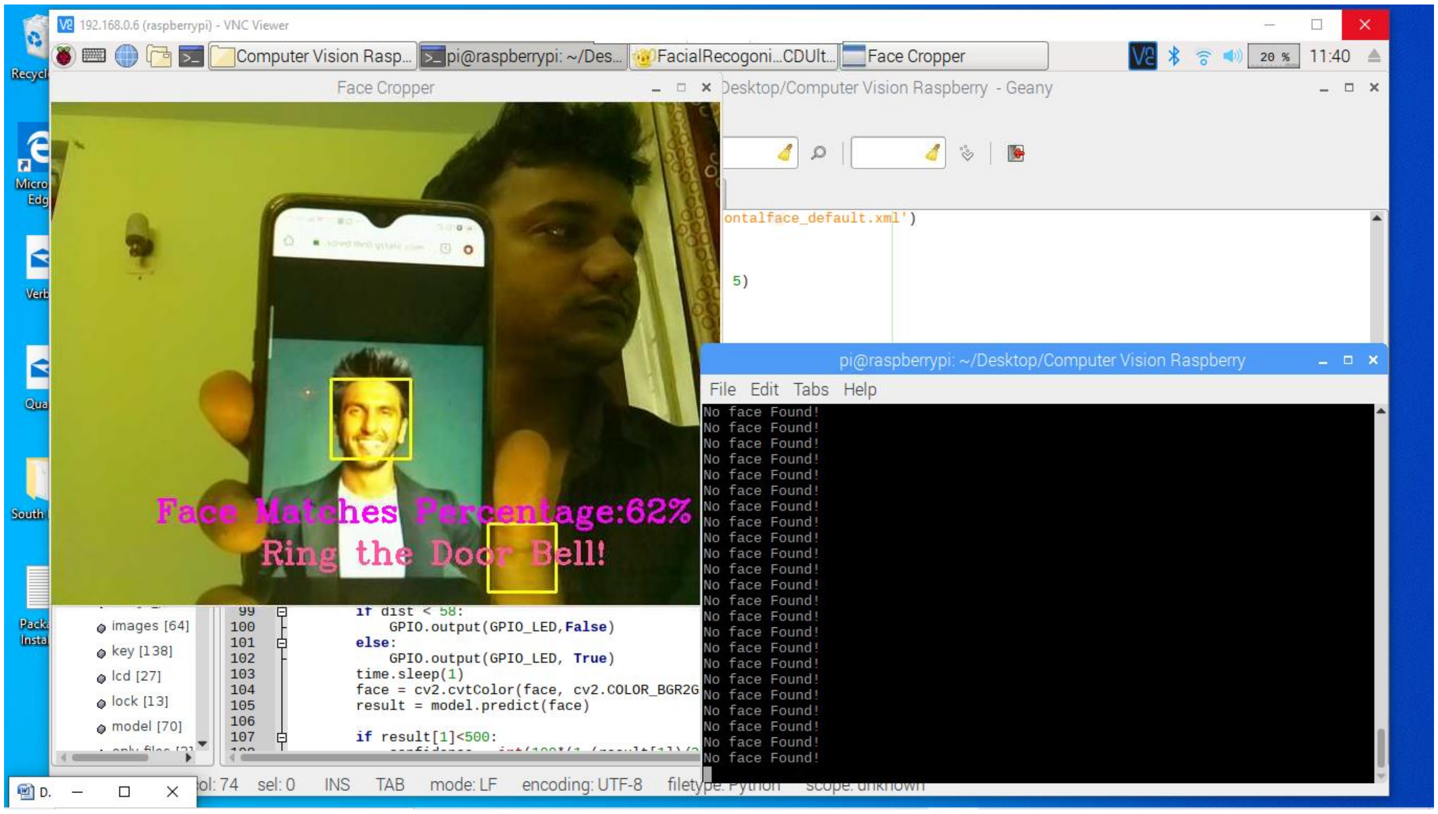

192.168.0.6 (raspberrypi) - VNC Viewer
Computer Vision Rasp...
pi@raspberrypi: ~/Des...
FacialRecogoni...CDUlt...
Face Cropper
Face Cropper
Face Not Found!
Desktop/Computer Vision Raspberry - Geany
pi@raspberrypi: ~/Desktop/Computer Vision Raspberry
File Edit Tabs Help
images [64]
key [138]
lcd [27]
lock [13]
model [70]
ol: 74 sel: 0 INS TAB mode: LF encoding: UTF-8 filetype: Python scope: unknown

```
ontalface_default.xml')

 5)
```

```
        if dist < 58:
            GPIO.output(GPIO_LED,False)
        else:
            GPIO.output(GPIO_LED, True)
        time.sleep(1)
        face = cv2.cvtColor(face, cv2.COLOR_BGR2G
        result = model.predict(face)

        if result[1]<500:
```

```
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
```




192.168.0.6 (raspberrypi) - VNC Viewer
Computer Vision Rasp...
pi@raspberrypi: ~/Des...
FacialRecogoni...CDUlt...
Face Cropper
Desktop/Computer Vision Raspberry - Geany
pi@raspberrypi: ~/Desktop/Computer Vision Raspberry
File Edit Tabs Help
images [64]
key [138]
lcd [27]
lock [13]
model [70]

Face Matches Percentage:62%

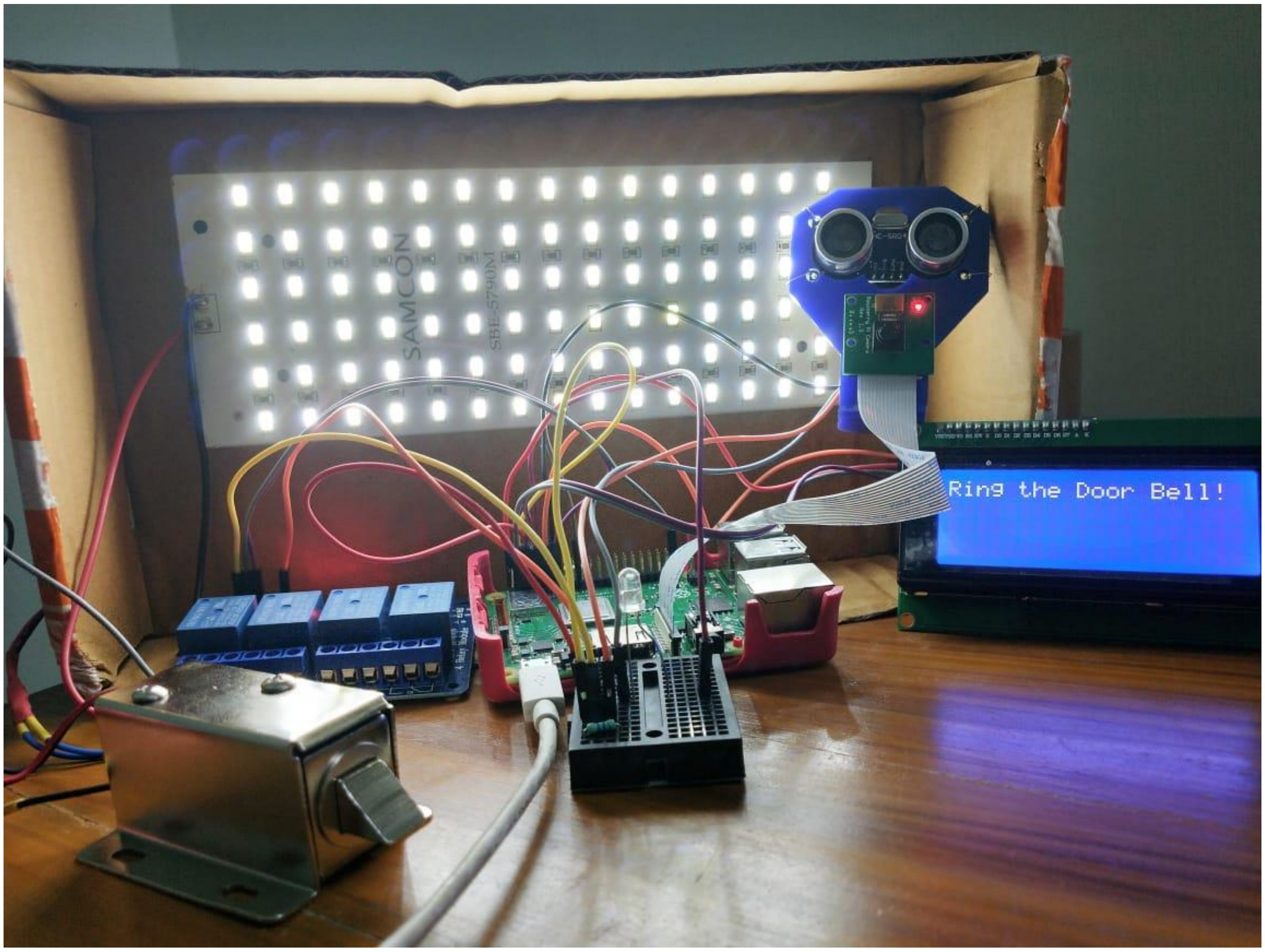
Ring the Door Bell!

```
ontalface_default.xml')

 5)
```

```
        if dist < 58:
            GPIO.output(GPIO_LED,False)
        else:
            GPIO.output(GPIO_LED, True)
        time.sleep(1)
        face = cv2.cvtColor(face, cv2.COLOR_BGR2G
        result = model.predict(face)

        if result[1]<500:
```

```
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
No face Found!
```

```
col: 74   sel: 0   INS   TAB   mode: LF   encoding: UTF-8   filetype: Python   scope: unknown
```



Chapter 7: Future Work:

My future enhancement for this project:

If someone forcefully tries to open the door/break in.

- Step 1- A buzzer/siren comes into play.
- Step 2- Immediately Send a SMS to the registered mobile number.
- Step 3 -By using same camera it will start recording the break in activity.
- Step 4 -Advance Lock down Protocol will initiate after break in.