

**Smart Doorbell**

**Mohammad Ahmad**

**Bachelor of Software & Electronic Engineering**

**Galway-Mayo Institute of Technology**

**2019/2020**

A screenshot of a cell phone

Description automatically generated

**Declaration**

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Software & Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

**Acknowledgements**

I am grateful to my family, friends, and the staff at GMIT for their help, support, and guidance throughout this project and my time in GMIT. I would like to thank my supervisor Paul Lennon for guidance from start to the end of the project.

**Table of Contents**

[1 Introduction 6](#_Toc40627122)

[2 The Internet of Things Overview 7](#_Toc40627123)

[3 Project Architecture Diagram 9](#_Toc40627124)

[4 Development Platform and Tools 10](#_Toc40627125)

[5 Interrupts 11](#_Toc40627126)

[6 Face Recognition 12](#_Toc40627127)

[7 Sending Mail with an image attached 14](#_Toc40627128)

[8 Sending Mail Only 15](#_Toc40627129)

[9 Problem Solving 16](#_Toc40627130)

[10. Conclusion 17](#_Toc40627131)

[11. References 18](#_Toc40627132)

[12. Code 19](#_Toc40627133)

# Introduction

During winter time of the year it becomes really hard to open the door if the bell rings especially for elderly and people leaving on their own as they don’t know if it is safe to open the door or not, the goal for this project is to create a Smart Bell system for the security of those and many others. This project will be a beneficial security system for the people out there to be safe and secure all the time through out the year. Security is very important for everyone out there these days as anything can happen at any time of day, month or year through out their life. This project will also be a cheap solution for everyone out there. It can also be used worldwide for their security.

The plan for this project is to create a face recognition system that will detect a person when they arrive at the door. When someone walks in front of the camera it sends an alert to the user's mail account with an image and stating the name of the recognised person or it will state unrecognised person outside if not recognised.

Technologies to be used for this project hardware wise are a Raspberry Pi, PIR Motion sensor and a camera. Software to be used is Python programming language, alongside learning useful libraries that will help me creating this.

The scope of this project involves researching and learning the required features that will be implemented as mentioned above, writing software prototypes, testing, debugging and integration of code to create a Smart Bell system for everyone out there. After completing this project, the skills gained from it will be applicable in future projects and will improve my software skills too.

This is a final year project for a Bachelor of Engineering Honours degree, the course is conducted at the Galway-Mayo Institute of Technology. The course is titled Software and Electronic Engineering. For this project, the goal is to implement knowledge gained from the course and research external topics for use in the project.

# The Internet of Things Overview

**Raspberry Pi Overview**

Raspberry Pi is a miniscule, extremely inexpensive and single-board computers with a size of a credit-card which essentially connects to a monitor via HDMI cable and any regular keyboard or mouse can be used. It has evolved throughout the years with variations in processors, memory and networks. It enables users from beginners to experts, to explore the vide variety of features in computing, especially for learning programming such as Python which this project is based on. Users may also browse the internet and expect a very similar experience to that of a desktop computer or laptop with the integration of Central Processing Unit (CPU) and Graphics Processing Unit (GPU). All Pi’s operate at 700 MHz as default but they may be clocked up to 1000 MHz offering a powerful experience to that of Intel processors. The OS (Operating System) used in this project is Raspbian which is the OS created by Raspberry Pi which is used for normal use on a Raspberry Pi. Raspbian is a free closed source OS based on Debian, made efficiently for the Raspberry Pi. Many other OS’s may also be used such as Windows and Ubuntu.

**Raspberry PI Specifications**

CPU – Broadcom BCM2711, Quad core Cortex-A72 64-bit SoC - 1.5GHz, RAM – 1GB, 2GB or 4GB LPDDR4-2400 SDRAM, Wi-Fi – 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE, Ethernet – Gigabit, USB – 2 USB 3.0 ports - 2 USB 2.0 ports, GPIO header – Raspberry Pi standard 40 pin, HDMI – 2 × micro-HDMI ports, Display port – 2-lane MIPI DSI, Camera port – 2-lane MIPI CSI, Audio – 4-pole stereo audio and composite video port, Storage – Micro-SD card slot for loading operating system and data storage.

**Raspberry Pi Camera**

Raspberry Pi v2 Camera was utilized in this project for facial recognition which has a high definition 8 megapixel Sony IMX219 image sensor and it’s integrated with Raspberry Pi consisting of a focused lens. It can capture high quality images with a resolution of 3280 x 2464 pixels. It may also support 1080p, 720p and 640 x 480p images with frame rates of 30, 60 and 90 per second. The camera is connected to Raspberry Pi via a devoted Camera Serial Interface (CSI) that provide an output that interfaces with multiple image sensors interfaces and it also connects also with a small socket on the board surface; the board itself is miniscule ,having an area of 575 mm weighing at approximately 4 g, thus making it a compact device for easy-use. Also, the camera is connected to the Raspberry Pi via a short ribbon cable. Owing to the use of Sony IMX219 image sensor which captures high quality images at 8 megapixels with a fixed focus lens, it’s the perfect go-to system in many scenarios. The camera may also capture still images at a resolution of 3280 x 2464p pixel static images and further supporting 1080p, 720p and 640 x 480p videos with frame rates of 30,60 and 90 per second.

**PIR Motion Sensor**

A passive infrared sensor is an electronic sensor that measures infrared light radiating from objects in its field of view. They are most often used in PIR-based motion detectors. PIR sensors are commonly used in security alarms and automatic lighting applications.  PIR can detect animal/human movement in a requirement range, which is determined by the spec of the specific sensor. The detector itself does not emit any energy but passively receives it, detects infrared radiation from the environment.

**Face Recognition**

Facial recognition is a way of recognizing a human face through technology. A facial recognition system uses biometrics to map facial features from a photograph or video. It compares the information with a database of known faces to find a match. Once it recognises the information from the database, it displays the result of the human stating who it is. It can be used for security for various reasons especially by the police in big countries that have a bigger population.

**Python**

Pythion is the main programming language used in Raspberry Pi along with Scratch. Python is optimized and designed for ease-of-readability. It’s mostly object orientated with a high-level and general-purpose use. Python has many applications and it’s mainly used for the development of Graphical User Interface (GUI) application which is based on this project but it may also be used for creating websites and other internet applications – allowing users to focus on key functions of the applications. It comprises of English keywords whereas, other languages use punctuations. It also contains fewer syntax constructions over other programming languages. Python is a very portable language hence the use of this specific language for this project, it can run on many hardware’s such as Raspberry Pi which is to be used for this particular project. It provides interfaces to commercially available databases, thus being compatible with the Raspberry Pi v2 Camera. Python supports many interesting systems and libraries of which include OCV (Open Computer Vision) for recognitions of specific objects based on imagery. It may also be used for AI (Artificial Intelligence) and machine learning.

**Haar Cascade**

Haar Cascade classifiers are an effective object detection approach which was proposed by Paul Viola and Michael Jones in 2001. It is used to detect an object in an image. It is a machine learning-based approach where positive and negative images are used to teach the classifier. Positive images contain the images which we want our classifier to identify. Negative images are images of everything else in an image, which do not contain the object we want to detect.

# Project Architecture Diagram

A close up of text on a white background

Description automatically generated

# Development Platform and Tools

**Thonny**

The main development tool used in this project was the Thonny. Thonny is an easier alternative to write and learn code with. It is a user-friendly Python system which helps writing code seamlessly – helping a user by expressing a value by assessment. Normally, Thonny is integrated with Python 3.6 making it an efficient process for installation for beginners. It comes pre-installed with a wide variety of features for learning programming. One feature that particularly stands out and easy-to-use is the debug mode which debugs the entire code on completion, line-by-line instead of running the program. The visualisation of objects and variables being produced becomes apparent and values inside the code are converted to functions. It’s a very straight forward and user-friendly process as other debuggers may require manual set breakpoints to assess code which may be time consuming. For that evident reasoning, Thonny was of particular interest in this project for a seamless experience.

# Interrupts

**PIR Motion Sensor**

Interrupt that I used in my project was the PIR motion sensor to detected movement. It can detect movement of human, animals and other objects. The detector itself does not emit any energy but passively receives it, detects infrared radiation from the environment. The detection distance can range from 25cm to 20m depending in which one you are using as there are indoor and outdoor PIR sensors. If the system goes to sleep mode and someone walks in front for the sensor it will bring the system back to live. It is one of the best sensor to bring the system back live.

# Face Recognition

The main feature in my project is face recognition to recognise humans and indicating who that person is. When someone walks in the distance of the camera and if there information is stored in the database, it pulls the information out of the database to recognise that person and display who it is. It draws a box around there face and indicates their name that you set to display underneath that box. If the person is not recognised it display’s ‘Unknown Person’ underneath the box.

The code below show’s that it is adding the person’s details that you want to recognise and you can add as many persons as you want to recognise them as well. At the end you have to give that person a name so when that person is recognised it will display that name beside their face.

A screenshot of a cell phone

Description automatically generated

The code below shows you if the face matches it displays the results of that person that matched. It draws a box around the face of that person and underneath that box it displays the name of that person that you set.



A screenshot of a cell phone

Description automatically generated

A picture containing person, indoor, man, front

Description automatically generated



# Sending Mail with an image attached

If the face is recognised it sends an mail with an image of the person to the owners email account stating who is outside. This is one quick way to look up who is outside. It requires Wi-Fi to send mail out. You are required to put your login details, the message you want to send and to whom you want to send it too. The code below states how that is done.

A screenshot of a cell phone

Description automatically generated

Once the mail is sent out it should look like this when you receive it.

A screenshot of a cell phone

Description automatically generated

# Sending Mail Only

If the face is not recognised it sends an mail only to the owners email account stating ‘Unknown Person Outside’. It requires Wi-Fi to send mail out. You are required to put your login details, the message you want to send and to whom you want to send it too. The code below states how that is done.

A screenshot of a cell phone

Description automatically generated

Once the mail is sent out it should look like this when you receive it.

A screenshot of a cell phone

Description automatically generated

# Problem Solving

One of the problems that I encountered was the Raspbian operating system giving up on me twice when I installed open CV on it. After installing open CV for the first time, the applications and the folders on the raspberry pi wouldn’t open. So I had to install the Raspbian operating system again as that was the only way of getting back to work quick. The second time when I installed open CV the whole system froze at the end. I did try rebooting it up again but no luck, so I had to the Raspbian operating system again and the third time it worked.

The other problem that I encountered was setting up face recognition. It wouldn’t draw the results around the face and then I did a lot of research and was able to fix it by adding a few lines of code to make it to work.

# Conclusion

The purpose of this project was to create Smart Bell system for elderly and people leaving on their own for there safety. Face recognition system that will detect a person when they arrive at the door. When someone walks in front of the camera it sends an alert to the user's mail account, stating the name of the recognised person or it will state unrecognised person outside if not recognised.

The future for security is very important for everyone out there as anything can happen at any time of day, month or year throughout their life. It is anticipated that this project with the help of this project it would ease a bit of security of them and keep them well and safe. It is a reliable way of recognising a person who would be at the door. This system can be used by anyone out there and it would be affordable as well.

Upon completion of the project, once the programme is started it detects a person when they arrive at the door. When the person comes in distance to the camera it send’s and email to the owner stating who is outside. So they can have a look at their emails to see who it is.

For the creator of this project, it is hoped to learn the fundamentals of programming in Python, learn about how open CV works and how to send out a email when required on the right time.

# References

[1] Projects.raspberrypi.org. 2020. [online] Available at: <https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/2> [Accessed 17 May 2020].

[2] Projects.raspberrypi.org. 2020. [online] Available at: <https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/2> [Accessed 17 May 2020].

[3] Projects.raspberrypi.org. 2020. [online] Available at: <https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/2> [Accessed 17 May 2020].

# Code

Source code can also be found here - <https://github.com/MohAhmad1/Smart_Bell_Project->

**Smart Bell**

import face\_recognition  
import cv2  
import numpy as np  
from espeak import espeak  
import smtplib  
from email.mime.multipart import MIMEMultipart    
from email.mime.base import MIMEBase    
from email.mime.text import MIMEText     
from email import encoders   
  
# Get a reference to webcam   
video\_capture = cv2.VideoCapture(0)   
  
mohammad\_pic = face\_recognition.load\_image\_file("moe.jpg")  
mohammad\_face\_encoding = face\_recognition.face\_encodings(mohammad\_pic)[0]  
  
# Add another person here  
  
# Create arrays of known face encodings and their names  
known\_face\_converts = [  
    mohammad\_face\_encoding  
]  
known\_face\_labels = [  
    "mohammad ahmad"  
]  
  
PIR\_input = 29 #read PIR Output           
  
# Initialize variables  
face\_positions = []  
face\_converts = []  
face\_labels = []  
process\_this\_frame = True  
  
while True:  
    # Grab a single frame of video  
    ret, frame = video\_capture.read()  
  
    # Resize frame of video to 1/4 size for faster face recognition processing  
    small\_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)  
  
    # Convert the image from BGR color (which OpenCV uses) to RGB color (which face\_recognition uses)  
    rgb\_small\_frame = small\_frame[:, :, ::-1]  
  
    # Only process every other frame of video to save time  
    if process\_this\_frame:  
        # Find all the faces and face encodings in the current frame of video  
        face\_positions = face\_recognition.face\_locations(rgb\_small\_frame)  
        face\_converts = face\_recognition.face\_encodings(rgb\_small\_frame, face\_positions)  
  
        face\_labels = []  
        for face\_encoding in face\_converts:  
            # See if the face is a match for the known face  
            matches = face\_recognition.compare\_faces(known\_face\_converts, face\_encoding)

# Detecting movement

  PIR\_input

print("Movement Detected!")  
            # Sending email with image  
            def send\_an\_email\_image():    
                toEmail = '[g00321094@gmail.com](mailto:g00321094@gmail.com)'      # Sending email to  
                fromEmail = '[g00321094@gmail.com](mailto:g00321094@gmail.com)'          # Sending from  
                subject = "Mohammad Ahmad is outside the Door"              # Subject/Message of Email  
   
                msg = MIMEMultipart()    
                msg['Subject'] = subject    
                msg['From'] = fromEmail    
                msg['To'] = toEmail    
   
                part = MIMEBase('application', "octet-stream")    
                part.set\_payload(open("moe.jpg", "rb").read())    
                encoders.encode\_base64(part)    
                part.add\_header('Content-Disposition', 'attachment; filename="image.jpg"')   # File name and format name  
                msg.attach(part)    
   
                try:    
                   send = smtplib.SMTP('[smtp.gmail.com](http://smtp.gmail.com)', 587)  # Protocol  
                   send.ehlo()    
                   send.starttls()    
                   send.ehlo()    
                   send.login(user = '[g00321094@gmail.com](mailto:g00321094@gmail.com)', password = 'gmit2020')  # User id & password   
                   send.sendmail(fromEmail, toEmail, msg.as\_string())    
                   send.quit()  
         
                except SMTPException as error:    
                    print ("Error")                # Exception  
   
            send\_an\_email\_image()  
            # if face unkown, print unkown  
            name = "Unknown"  
            def send\_an\_email():    
                toEmail = '[g00321094@gmail.com](mailto:g00321094@gmail.com)'      # Sending email to  
                fromEmail = '[g00321094@gmail.com](mailto:g00321094@gmail.com)'          # Sending from  
                subject = "Unkown person outside the door"              # Subject/Message of Email  
   
                msg = MIMEMultipart()    
                msg['Subject'] = subject    
                msg['From'] = fromEmail    
                msg['To'] = toEmail  
                try:    
                   send = smtplib.SMTP('[smtp.gmail.com](http://smtp.gmail.com)', 587)  # Protocol  
                   send.ehlo()    
                   send.starttls()    
                   send.ehlo()    
                   send.login(user = '[g00321094@gmail.com](mailto:g00321094@gmail.com)', password = 'gmit2020')  # User id & password   
                   send.sendmail(fromEmail, toEmail, msg.as\_string())    
                   send.quit()  
         
                except SMTPException as error:    
                    print ("Error")                # Exception  
   
            send\_an\_email()  
                          
              
          #  Or instead, use the known face with the smallest distance to the new face  
            face\_distances = face\_recognition.face\_distance(known\_face\_converts, face\_encoding)  
            best\_match\_index = np.argmin(face\_distances)  
            if matches[best\_match\_index]:  
                name = known\_face\_labels[best\_match\_index]  
                espeak.synth(name)  
                  
  
            face\_labels.append(name)  
            espeak.synth(name)  
  
    process\_this\_frame = not process\_this\_frame  
  
  
    # Display the results  
    for (top, right, bottom, left), name in zip(face\_positions, face\_labels):  
        # Scale back up face locations since the frame we detected in was scaled to 1/4 size  
        top \*= 4  
        right \*= 4  
        bottom \*= 4  
        left \*= 4  
  
        # Draw a box around the face  
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)  
  
        # Draw a label with a name below the face  
        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)  
        font = cv2.FONT\_HERSHEY\_DUPLEX  
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)  
          
  
    # Display the resulting image  
    cv2.imshow('Video', frame)  
  
    # Press 'q' on the keyboard to quit!  
    if cv2.waitKey(1) & 0xFF == ord('q'):  
        break  
  
# Release handle to the webcam  
video\_capture.release()  
cv2.destroyAllWindows()