

Smart Doorbell-System Using Computer Vision and IoT

Senior Project

by

Wisam Ramadan - 21830418,

Samer Hammoud – 21830931

Submitted to the School of Engineering of the

Lebanese International University

Bekaa, Lebanon

in partial fulfillment of the requirements for the degree of

**BACHELOR OF SCIENCE IN COMMUNICATION AND COMPUTER
ENGINEERING**

Spring 2021

Approved by:

Supervisor

Dr. Abdel-Mehsen Ahmad

Committee Member

Dr. Zouhair El Bazzal

Dr. Samir Omar

DEDICATION

To my father, for his scarification of his comfort zone to provide decent and stable education for me, and my family who has encouraged and supported me to complete my education.

Wisam A. Ramadan

To my father brother and my family, especially my brother for the motivation and the support I got, under circumstances. Greeting to my father for the sacrifices he made to enable us continuously staying on the right side of life. Special thanks to the Lebanese international university for support in this economic crisis.

Samer H. Hammoud

ACKNOWLEDGMENT

Despite this challenging time, which is full of diseases and economic crisis, during which we have lost beloved ones, education has been pursued, teachers, and behind screens, have remained resistant for the goal of letting education reach every student. They believed in the holiness of education and trusted in us to continue what they started.

This project has required huge efforts of assistance and motivation to be completed. We would like to express our deep gratitude to our supervisor Dr. Abdel-Mehsen Ahmad, who has supported and guided us to a solution for every problem during the whole time to have this project finished. This could not have been completed successfully without such support.

We would also like to thank our faculty dean and instructors for being very generous to answer all our questions and share with us their knowledge. We are also appreciated for our university, LIU, for providing us with many opportunities to improve our practical and professional knowledge.

ABSTRACT

Nowadays, almost every home, institution, or governmental department is equipped with security systems which attempt to detect any unauthorized access to the private boundaries of the entity. The most common types of these systems are Security Cameras and Alarming Systems. However, there are many drawbacks which decrease the sufficiency of these systems in case of the occurrence of an incident. Security cameras are meant to record videos continuously to be used later in case a breakthrough occurs but do not stop them. While alarming systems go off in case a motion is detected only, not necessarily a human motion. For this purpose, we developed a system that connects between security cameras, alarming systems, and the administrator using the technology of Computer Vision and IoT to achieve the best solution for these drawbacks. This system is mainly placed at the door, and it notifies the administrator every time a human (visitor) is detected through an android application, which is dedicated to control the whole system, including the alarming system. Moreover, when the system recognizes a person authorized to access, it opens the door automatically, and notifies the administrator otherwise. Also, the system involves motion sensors placed around the targeted space which are activated when the place is empty. These sensors alarm the admin of any event through the system.

TABLE OF CONTENTS

ACKNOWLEDGMENT	III
ABSTRACT.....	IV
TABLE OF CONTENTS	V
LIST OF FIGURES	VIII
LIST OF TABLES	X
LIST OF SYMBOLS	XI
CHAPTER 1 INTRODUCTION	1
1.1 Background	1
1.2 Problem Statement	2
1.3 General overview of the project.....	3
1.4 Thesis Outline	3
CHAPTER 2 SURVEY OF EXISTING METHODS AND SIMILAR SYSTEMS	5
2.1 Introduction.....	5
2.2 Google Nest Hello.....	5
2.3 Arlo Essential Wireless Video Doorbell.....	7
2.4 AI Face Recognition Security Camera Integrated with Access Control System.....	8
2.5 Conclusion and Motivation.....	9
CHAPTER 3 SYSTEM DESIGN	11
3.1 Introduction.....	11
3.2 Requirements and Specification Analysis	11

3.2.1	Use Cases:	11
3.3	Financial Viability	12
3.4	Project Management Aspects.....	13
3.4.1	Stakeholders	13
3.4.2	Scope.....	13
3.4.3	Risks.....	14
3.4.4	Schedule and Milestones.....	14
3.5	Ethical and Social Considerations	16
3.6	Environmental and Sustainability Considerations	16
3.7	System Architecture.....	16
3.7.1	Hardware-based Part.....	18
3.7.2	Software-based Part	20
3.8	Relevant Standards.....	20
3.9	Class Diagrams	20
3.10	Sequence Diagrams.....	22
3.11	State Diagrams	23
3.12	Conclusion	24
CHAPTER 4 IMPLEMENTATION/SIMULATION AND TESTING		25
4.1	Introduction.....	25
4.2	Implementation Tools	25
4.2.1	Raspberry Pi 4 Model B:.....	25
4.2.2	Camera Module:.....	27
4.2.3	PIR Motion Sensor:.....	27
4.2.4	Android Application:	28

4.2.5	Python:	30
4.2.6	Face Detection	34
4.2.7	Blink Detection	35
4.2.8	Facial Recognition:	36
4.3	Implementation Summary	37
4.3.1	Main System	37
4.3.2	Android Application	48
4.4	Test Cases and Acceptance Criteria	53
4.5	Conclusion	55
CHAPTER 5 CONCLUSION AND FUTURE WORK.....		57
5.1	Conclusion	57
5.2	Future Work	58
APPENDIX A: IMPLEMENTATION DETAILS.....		59
APPENDIXB: USER MANUAL		60
APPENDIX C: DEPLOYMENT AND CONFIGURATION MANUAL		61
REFERENCES.....		62

LIST OF FIGURES

Figure 2-1 Google Nest Hello Camera [1].....	6
Figure 2-2 Arlo Doorbell System [2].....	7
Figure 3-1 System Architecture	17
Figure 3-2 System Software Model	18
Figure 3-3 Class Diagram	21
Figure 3-4 Sequence Diagram	22
Figure 3-5 State Diagram.....	24
Figure 4-1 Raspberry Pi [6]	26
Figure 4-2 Raspberry Pi Camera [18]	27
Figure 4-3 RIP Motion Sensor.....	28
Figure 4-4 Linked Database.....	29
Figure 4-5 CNN Architecture [14].....	33
Figure 4-6 Convolution Operation [14]	34
Figure 4-7 Haar Features [19].....	35
Figure 4-8 Raspbian OS.....	38
Figure 4-9 Database E/R Diagram.....	39
Figure 4-10 Liveness Detection Directory	41
Figure 4-11 Building Dataset Command	42
Figure 4-12 Model Training Command.....	43
Figure 4-13 Convolutional Neural Network Implementation.....	44
Figure 4-14 Facial Recognition Method Call	45
Figure 4-15 Notification-Command Code	46

Figure 4-16 Program Running.	47
Figure 4-17 Main Activity	48
Figure 4-18 Recent Visits Activity	49
Figure 4-19 Add Guest Activity	50
Figure 4-20 the Service Class	51
Figure 4-21 Notification	52
Figure 4-22 PHP Getting the Images	52
Figure 4-23 System Circuit.....	54

LIST OF TABLES

Table 2-1: Related Work.....	8
Table 3-1 Financial Viability	12
Table 3-2 Milestone Schedules	15

LIST OF SYMBOLS

ARM: Advanced RISC Machines.

CNN: Convolutional Neural Network.

CV: Computer Vision.

EAR: Eye Aspect Ratio.

GPU: Graphics Processing Unit

IP: Internet Protocol

LAN: Local Area Network.

OpenCV: Open-Source Computer Vision

PHP: Hypertext Preprocessor.

PIR: Passive Infrared.

PnP: Plug and Play.

PyPi: Python Package Index.

ROI: Region of Interest.

Wi-Fi: Wireless Fidelity.

XML: Extensible Markup Language.

CHAPTER 1

INTRODUCTION

1.1 Background

There has always been a concern by people towards their own security and the available practices for that. In today's world, security practices have expanded to involve the employment of hardware systems to decrease the risk of security invasions.

These systems include Door and Window Sensors, Presence Simulation, High-decibel alarms, Surveillance Cameras, ...etc. [3]. However, cameras are among the most spread and trusted means of home security as they provide photo captures and live video streaming of the spotted area. These cameras vary between being IP cameras, which connect on the internet and stream to users' devices using Wi-Fi connection, and CCTV cameras which stream through wired or wireless links. Also, some people use alarming systems which, after being triggered by some event, produce a high-decibel sound as a practice to alert others.

In contrast, the Internet of Things – IoT has recently become a promising technology which will add more improvements to the overall internet by extending it to connect and exchange data among different devices and systems over the internet. This is achieved by embedding devices and systems with sensors, software, and other technologies for the purpose of creating a presence for them on the internet [4].

Smart Doorbell system is a system dedicated to improving the overall security of home and institutions by integrating some exist practices, such as cameras and alarming systems, with the Internet of Things in order to provide real-time visual data about any human presence in the spotted area and take instant actions on some incidents.

1.2 Problem Statement

Best security practices are used to help prevent a robbery, break-in, entity invasion, ...etc., not only documenting them. Many security systems available suffer from a downside related to the inability of trying to stop an incident from happening.

On the other hand, traditional doorbell systems are only capable of manually and remotely opening a door, and in many models, the owner is not able to watch who is at the door. In this case, the only actions possible to take are either opening the door or ignoring it.

Most of nowadays security camera systems are only meant to record and document an incident that can be later used by authorities during investigation. However, these systems are not able to help prevent these incidents from happening. The best practice of using a security camera is to have someone sitting in front of the screen, watching the spotted area, and reporting any abnormal behavior. This is a time, money, and effort consuming task. Moreover, it is not efficiently capable of reporting every single incident 24/7 due to its human dependency. Also, these systems are not able to take automatic actions in case of any abnormal human presence. This would, in many cases, slow down taking preventive measures.

The purpose of Smart Doorbell Systems is to integrate many of the current existent security systems into one device that could be placed at the door or any different area. This device provides users with visual data about any human presence in the spotted area in the form of a mobile app notification. This alerts users only and every time a human is near their door/spotted area. Moreover, the device control can be both, automated and manual. Users can, through their mobile app, take an action, e.g., turning alarming system on, which would fasten responding to an incident. Also, to achieve a fast doorbell experience, this device is capable of automatically opening the door for visitors who are set as authorized people.

1.3 General overview of the project

A Raspberry Pi is connected to a camera and placed in front of a door. The Pi is connected to a database, so each captured photo is uploaded to the database for the administrator to check. Also, every time a photo is captured for someone, it is compared against all the pre-stored photos in a directory within the Pi's file system to determine identity if matched to someone.

- Property owner has the authorization to let in or reject a visitor approaching and turn on/off an alarming system.
- This system will be connected to an Android Application installed on the admins' phones through a database.
- The system sends an app notification every time a human and/or a motion is detected.
- The system is supported with technologies required to take an automated action for letting in an authorized visitor.
- Each time an unknown visitor approaches the spotted area, the user is notified through the android app with a message containing the visitor's face.

1.4 Thesis Outline

This report explains the project in full details. Chapter 2 introduces some related work of already existing methods, which provide similar solutions for the same problem discussed in this project. It also compares them with this solution, highlighting the motivation behind this project after shedding the light on the previous works' limitations.

Chapter 3 explains the system design and architecture. It demonstrates using class, state, sequence diagrams and others. It also discusses the needs and requirements of the system.

Chapter 4 includes the implementation summary and tools. It specifies which hardware and software tools were used and gives a little introduction about each one. Also, it provides detailed steps of how the system is implemented.

Chapter 5, finally, includes what was learned from this system and it mentions the future works that might improve it.

CHAPTER 2

SURVEY OF EXISTING METHODS AND SIMILAR SYSTEMS

2.1 Introduction

This chapter introduces the already existing systems that are dedicated to the field of our project. These systems show exactly the transformation of security systems from traditional technology, e.g., 24/7 monitoring of streaming, to the modern one which is mainly achieved by analyzing the captured images and/or videos and behaving accordingly. These systems are Google Nest Hello, Arlo Doorbell, and AI & Access Control Security Camera.

Each of these systems has its own strengths and limitations; However, in our project we collected the highest demanding features of all the three systems and implemented them in our system to have the best option for people in the region.

2.2 Google Nest Hello

Google Nest Hello doorbell system, Figure 2-1, is basically a camera integrated with a doorbell dedicated to let users monitor more easily and effectively most of the events happening near their main doors. The system's main function is to operate 24/7 and captures an image for anybody who either rings the bell or simply moves near the system. It then sends a notification to the user's device through a mobile application containing the captured image. The user then has several options whose s/he can decide to do. S/he can upload the image to their dedicated space in cloud along with the name of the person inside it. This enables the automatic voice announcement unit to tell who is at the door instead of saying "someone" by using the facial recognition module supported by the system. Also, the user

can choose to chat with a person at the door and s/he can choose to open the door remotely through the mobile app.



Figure 2-1 Google Nest Hello Camera [1]

The system also offers different features such as video recordings, zone monitoring, ...etc. However, it still suffers from drawbacks related to the fact that this system is not autonomous. Users are still required to physically and/or manually open the door for someone because visitors are only classified into familiar and unfamiliar. This will make authenticated people such as family members wait at the door for someone to manually open it. Moreover, the system is not linked with any warning services (e.g., alarming system) and does not enable users to turn on their alarming systems in case they recognize a person at the door to be a risk to their property.

2.3 Arlo Essential Wireless Video Doorbell

The Arlo Essential Wireless Video Doorbell, Figure 2-2, works in a similar manner to Google Nest Hello but with a slight difference. The system captures a photo each time it detects an object moving in the front yard and notifies the user through the mobile application. The user can open a real-time video stream and act based on the detected object. S/he can communicate with visitors through two-way Audio module, open the door remotely, or take a deterring action by either calling the police or triggering the built-in alarm.

However, the system still suffers from a drawback as it does not support a facial recognition service but rather an object-detection one. The system can only determine whether a moving object is a human, vehicle, or an animal but cannot identify the visitors. This prevents the system from taking an automatic and fast action based on the status of the visitor.



Figure 2-2 Arlo Doorbell System [2]

2.4 AI Face Recognition Security Camera Integrated with Access Control System

This institution security system is produced by CCTV Camera Pros company. The system enables various features such as Face Detection, Facial Recognition, Vehicle Detection, and License Plate Detection to support all events that can happen around the institution office. The administrator uploads manually the authorized people images in the Camera NVR so that the system can match the captured images with the pre-stored ones. When the camera detects a face, it runs the facial recognition module and compares the captured face with all the faces stored in the database. If no match occurs, the system does not trigger the access control module so that the door is not opened. However, if a match occurs, the system triggers the access control module causing the door to open. Only those authorized people's faces are stored in the database [5]. However, an important drawback associated with this system is the fact that it does not confirm that a person is real before recognizing him/her. This poses the system to serious security bugs such as spoofing (e.g., using a photo of an authorized person to gain access). Table 2-1 shows the advantages and disadvantages of the various works discussed in this section.

Table 2-1: Related Work

Method	Advantages	Disadvantages
Google Nest Hello	<ul style="list-style-type: none">• Detects people and motion and sends notifications.• Chatting service.• Remote control	<ul style="list-style-type: none">• Not autonomous.• No deterrence measures supported.• Objects can trigger the system and sends

	through app.	notifications.
	<ul style="list-style-type: none"> • Flexible upload of images. 	
Arlo Doorbell System.	<ul style="list-style-type: none"> • Detects people, objects, and motion and sends notification. • Chatting Service. • Live video stream. • Deterrence measures taken. • Facial Recognition. 	<ul style="list-style-type: none"> • Not autonomous. • No Liveness Detection Check.
AI Facial Recognition Camera/Access Control System.	<ul style="list-style-type: none"> • People and Object Detection. • Facial Recognition. • Live Video Stream. • Mobile App. 	<ul style="list-style-type: none"> • No Liveness Detection Check.

2.5 Conclusion and Motivation

As introduced above, all the three methods make use of computer vision and IoT in order to analyze the pictures and provide real-time data to users. Each method is featured with a slightly different technology that aims to improve home security. However, there is no one system that collects all the demanding and essential features that can help in the prevention of burglaries. This motivated us to work on a system that collects the most essential features supported by the above systems, along with our new improvements.

Our system can prevent potential thefts by notifying users about any unknown visitors around his/her property, detecting all system spoofing attempts, and turning the alarming system on upon user's request. The system proposed can be improved by using more

advanced cameras that support HDR and night vision. Moreover, using more advanced computing capabilities makes the process of image processing much faster.

CHAPTER 3

SYSTEM DESIGN

3.1 Introduction

This chapter explains the details of the system's design and its architecture. It introduces the different components used as well as their utilities in this project and communication links among them. Also, a comprehensive discussion is presented regarding the class, sequence and state diagrams including the user's interaction with the system and the software used for this purpose.

3.2 Requirements and Specification Analysis

This project is a doorbell security system dedicated to improving the experience of users in securing their desired entities. It delivers an efficient and easy control and monitoring of the system through an android application at the administrator's device. Both, the android application and the system, have access to a database where they can interact together and share images captured by the camera. The user can control the home's door lock and alarming system based on the content of the notifications received by the app. S/he can either opens the door and updates authorized people dataset or turns the alarming system on.

3.2.1 Use Cases:

The user makes use of the system in the following scenario:

- Place the doorbell system at the desired area.
- Keep the mobile phone close for any potential notifications.
- Open the application when a notification is received.
- Check the picture sent by the system to see the visitor at the door.

- If the visitor is authorized, but not yet included in the authorized people dataset, click “open door” button to let him/her in and then add his/her photo to the authorized people dataset.
- Click “Add Person” to add someone to the authorized people dataset.
- Fill the form and then click “upload”.
- If the visitor is not relevant to you, do nothing.
- If the visitor found to be a threat to your entity, click the “turn alarming system on” button to let the alarming system goes off.

3.3 Financial Viability

Table 3-1 Financial Viability

Component	Quantity	Price
Raspberry pi 4 model B	1	62 \$
Arduino Uno	1	10 \$
Motion sensors	1	2.25 \$
Raspberry pi camera RPI 8MP	1	38 \$
12V Solenoid Lock	1	7 \$
Jumper wires	60	5 \$

Table 3-1 shows the financial viability of this system. Note that the quantity specified is for the project simulation, for real time implementation quantities of sensors and cameras may increase depending on the property’s area. Also, different components models and computing capabilities may be used to decrease the cost.

3.4 Project Management Aspects

3.4.1 Stakeholders

This project benefits property owners (big area properties), it best fits homes and villas, the system can act as a concierge, so that it will monitor all house corners with the cameras and the motion sensors.

Moreover, anybody who is interested in maintaining home security can benefit from this project. It is small and primarily placed at the apartment's outer door; consequently, everybody can make a physical location for it.

3.4.2 Scope

Project Scope Description

The purpose of this project is to improve safety and security at private entities and provide a notification center for users. Also, it enables users to control the entry to their entities based on visitor's identity, and it provides autonomous entry for authorized ones.

Project Activities

- The system notifies users for any human presence,
- It performs automatic door opening for authorized visitors.
- The system confirms that the visitors are real (e.g., not fake people on phone screen) and takes anti-spoofing measures otherwise.
- It enables users to control house's door lock and alarming system,
- It also enables them to add more people to the authorized people dataset.

Project Deliverables

- Main unit which consists of computer, camera, and motion sensor.
- Android application.

Project Exclusions

- The system does not support audio communication.
- It does not provide single administrator (all users have the same privileges).
- There is no constant video recording and live streaming.

Operation Criteria

- The system must have 24 hours electric power, or
- Its battery must be charged frequently.
- Authorized access list may require to be updated frequently.

3.4.3 Risks

There are a few situations in which the system may not function efficiently as expected. First, the camera operates under a predetermined temperature, and therefore, temperatures lower or higher than the recommended range may affect the quality of the camera or its life. Also, the camera utilized in this project is not infrared night vision, it means that it does not provide a clear night footage. Moreover, foggy, raining, and snowing conditions may also affect the quality of the camera.

In addition, the camera may start failing to recognize authorized people after a while, and this may happen due to the slight physical change of visitors. This is solved by uploading a new image to the unauthorized visitor.

3.4.4 Schedule and Milestones

Table 3-2 presents estimated schedules during which the system's parts and modules are expected to be implemented.

Table 3-2 Milestone Schedules

Milestone Schedules of the Project				
Task	March	April	May	June
Facial Recognition Module	20			
Database Communication	25			
Liveness Detection Method 1		15		
Android Application			10	
Liveness Detection method 1 Validation			20	
Liveness Detection Method 2			31	
Report Submission			31	
System-App Communication				2
Project Demonstration and Presentation				17

3.5 Ethical and Social Considerations

An ethical consideration involved with surveillance systems is that they violate the privacy of other people. This is because the fact that these systems provide visual data about the spotted area.

Our system may be posed to some kind of this ethical consideration as it, as for now, does not support video recording. However, if placed in an area where the neighboring entities/homes are revealed to the scope of the camera, the system will capture a picture for every scene in which a human face is detected. This means that this system may record every human activity in the neighboring areas. Consequently, the system will create a threat to privacy which is of a considerable value to the society and individuals.

3.6 Environmental and Sustainability Considerations

This kind of systems may have an impact on the electromagnetic environment in the surrounded areas of the operating spot. This may lead to security complications with these devices. The complications are related to false alarms, image defects, loss of communication, control errors, ...etc. [16]

3.7 System Architecture

The system mainly consists of two parts, a hardware-based part and a software-based one. Both parts communicate together through a database stored on Apache server hosted on a laptop. The database contains data about the activities which happen around the doorbell as well as the user's commands. This data are mainly photos of visitors and of people to be added to the authorized people dataset.

The Apache server is hosted on a Raspberry Pi 4 Model B which is the core of this system. There are many other peripherals connected to the database which will be explored in the following part.

Figure 3-1 below demonstrates the whole system:

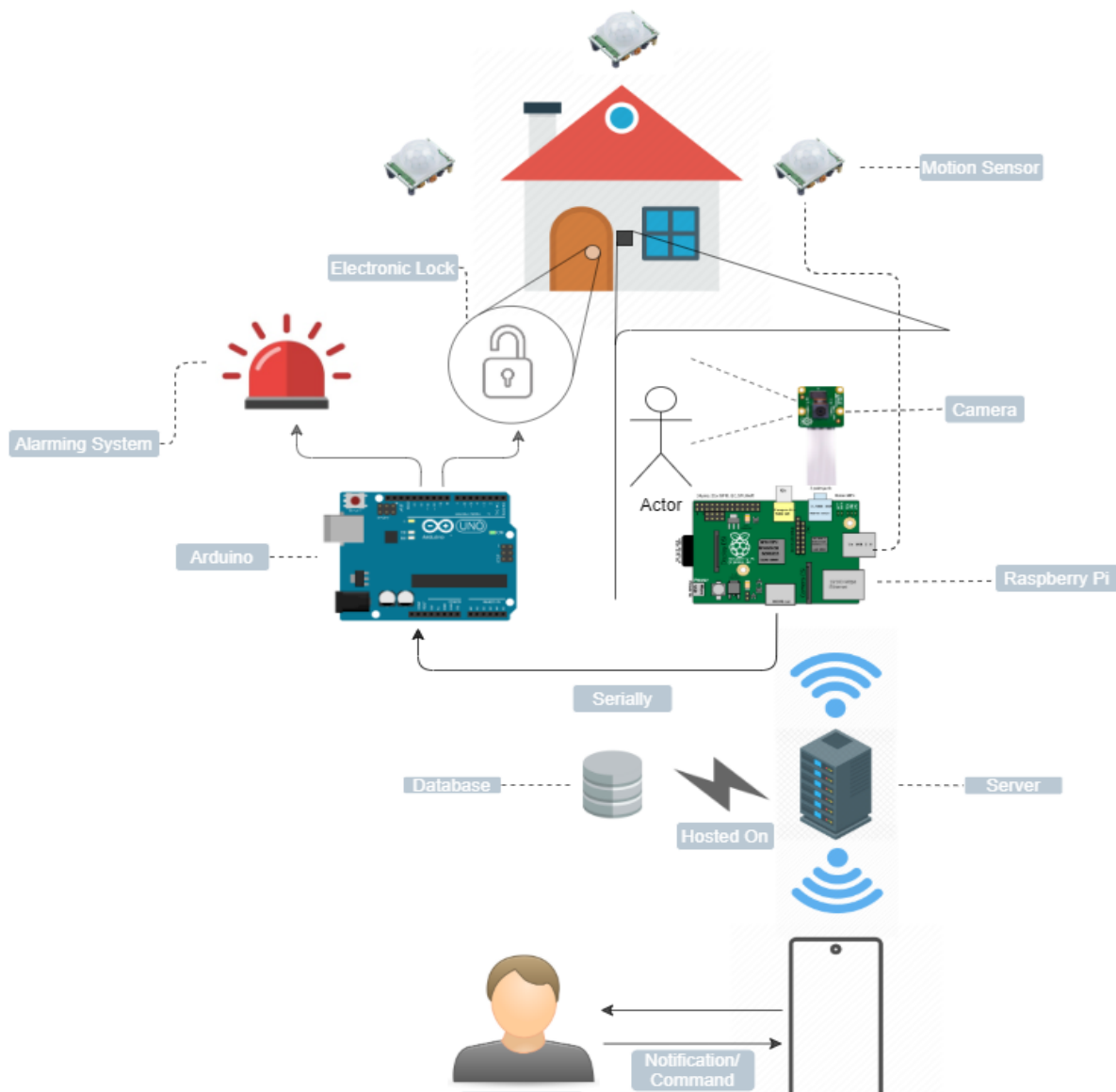


Figure 3-1 System Architecture

As show in, **Error! Reference source not found.**, the raspberry pi is connected to 3 different peripherals which differ in purpose.

As for the software used in this system, many languages and libraries have been utilized to implement the different parts and modules of the system. However, the main module and most important features of this system could not be completed efficiently without the use of Python language and its libraries.

Figure 3-2 shows all the software which have been used in this system.

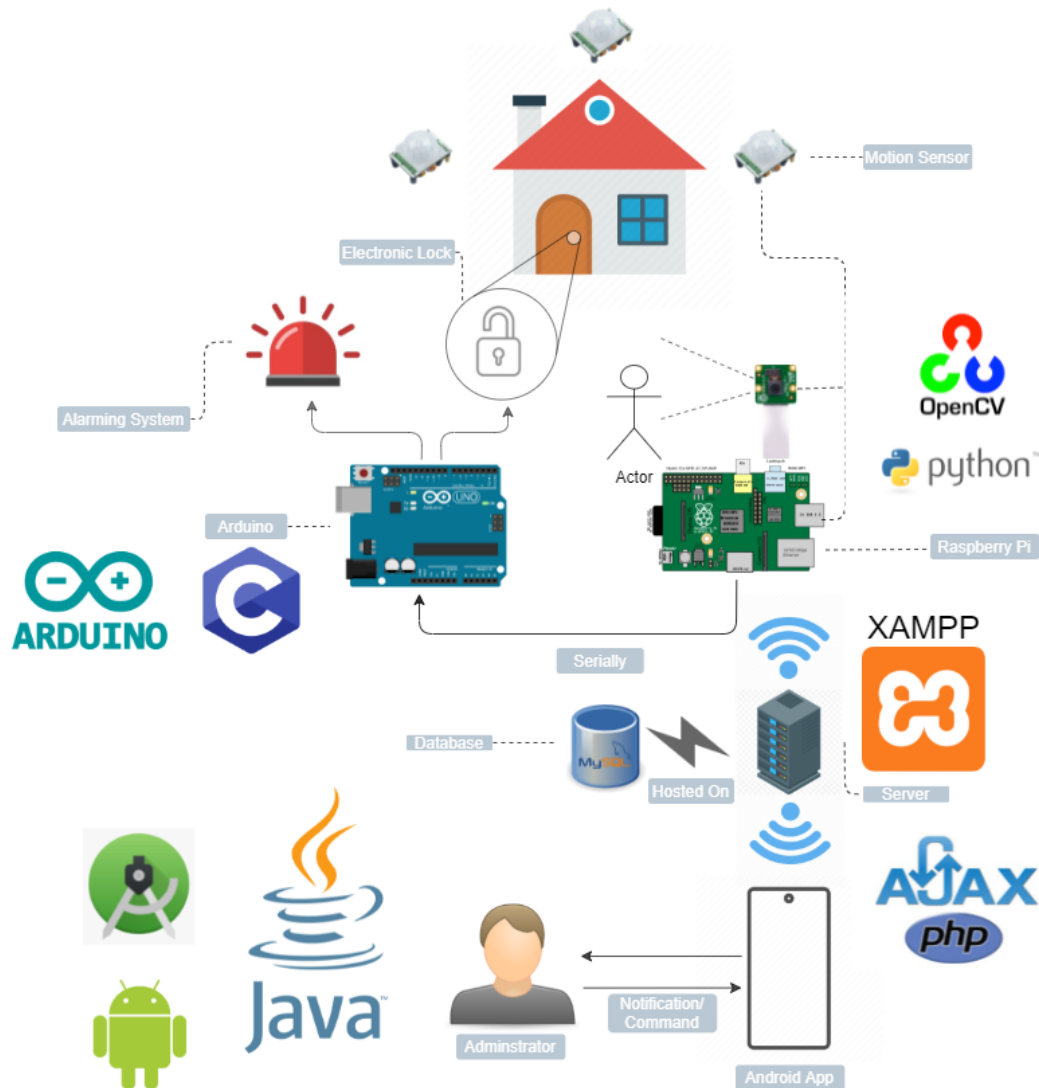


Figure 3-2 System Software Model

3.7.1 Hardware-based Part

- ## 1. Motion Sensor

This sensor can be more than one piece placed all around the targeted area by the system. When an object is detected by the sensor, it triggers the system causing it to send a notification to the user through the android application.

2. Raspberry Pi Camera Module V2

The camera supports face detection and facial recognition technologies to better improve the performance of the system.

When the camera is triggered by the Raspberry Pi, it turns on and captures the surrounding area of the system for any human face. After a picture is taken, the raspberry pi performs several tasks to determines the action to be performed.

3. Arduino Uno

The Arduino Uno Microcontroller is an execution unit that can be used by the raspberry pi. The Arduino is connected to two modules. The first module is Door Lock, while the second is Alarming System.

The Arduino can receive a signal from the raspberry pi and, depending on the signal, it either opens the door lock or turns the alarming system on.

4. Electronic Door Lock

5. Alarming System

6. Raspberry Pi 4 Model B

This unit is the core part of our system. It interconnects and coordinates all the above-mentioned parts in order to achieve the purpose of the Doorbell system. It receives a signal from the motion sensor when an object is detected, receives the camera footage for further analysis every time it detects a human face. After that, it processes the image for liveness detection and facial recognition and, based on the result, it either sends a command to the Arduino board directly or pushes a notification to the user and waits for his/her command.

3.7.2 Software-based Part

This part is mainly consisted of a monitoring and control unit of the system. This unit is an android application installed on the user's (owner) device in order to remotely observes and controls the security of his/her entity.

This application is user-friendly and easy-to-use for all users. It is consisted of one page, which is directly accessed when the user clicks on the app's icon or on the notification message. The page works as a notification center which shows the history of visitors' images with the last visitor's image being at the bottom of the page. Also, it contains a small control panel where the user can control the system.

1. Open Door Button.

This button is used in order to open the lock of the door in case the visitor is identified but he/she is not already stored in the database.

2. Turn on Alarming System Button

This button is used in order to turn the alarming system on when the user finds the visitor to be a threat to his/her entity.

3. Add to Database.

This button enables the user to add the authorized visitor's image and name to database so that the next time he/she visits the place, the door opens automatically.

3.8 Relevant Standards

3.9 Class Diagrams

The below diagram, Figure 3-3, shows the system's underlying classes and their hierarchy. First, the database class. This class is considered a connection channel between Raspberry Pi and user's application classes. In this class, the attributes: image, id, name, and

status are the main attributes of the tables constituting the database. While the methods to be used in this class are the MySQL methods: Select, Insert, and Update.

The second class is the Raspberry Pi class. This is the brain of our system. It performs most of the duties associated with the system. It is responsible for analyzing the input of the camera, applying face detection, liveness detection, and facial recognition, inserting data to the database, and executing the commands sent by the user's app.

Another class is the Android App class. It is considered as a notification center and control unit of the system. It shows notifications about visitors to the user, enables the user to add new authorized people to the database, and sends user's commands to the Raspberry Pi class.

The final class, Arduino, represents the execution unit of the Raspberry Pi class. It receives signals from the Raspberry Pi and depending on its type, it either opens the lock or turns on the alarming system.

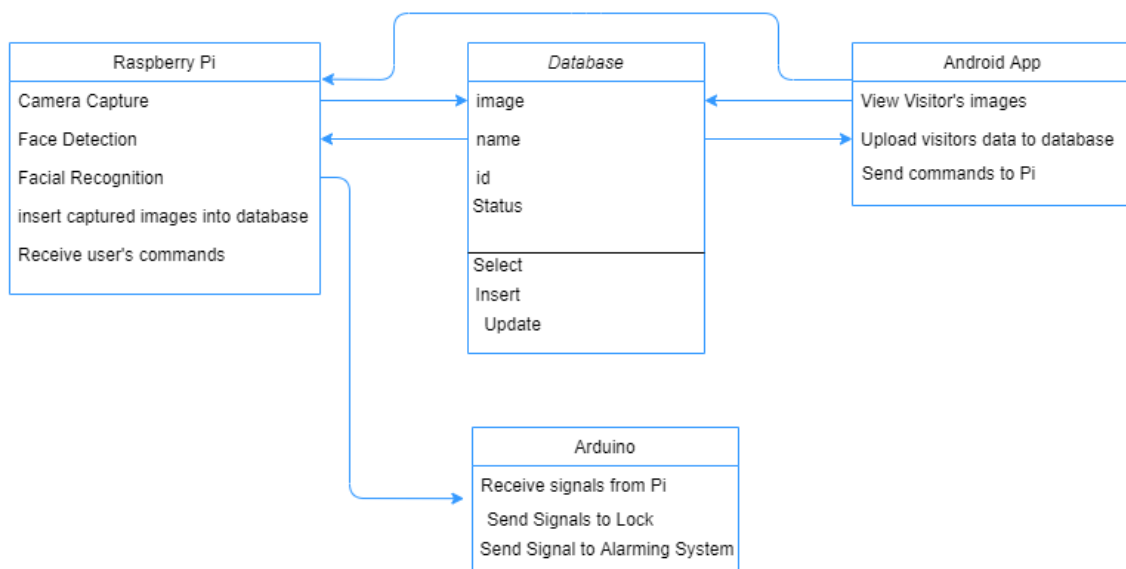


Figure 3-3 Class Diagram

3.10 Sequence Diagrams

Figure 3-4 shows the sequence diagram, the states followed from the beginning to the end of the real time monitoring process. As shown the sequence starts by the user himself, he/she decides to turn on/off the system.

Second the pi will turn on the sensors and remain idle until a signal is sent from the sensor, immediately the camera will be turned on.

The pi then captures the face of the target and then starts a process of comparison with the images stored inside a directory, it updates an attribute or insert a new item to the database.

The database state's change results in a notification on the users' side. In turn, the user can either sends a command to the Pi or ignores the notification.

According to the action, or the decision to be taken the pi will send the signal to the Arduino while the Arduino controls the doors lock.

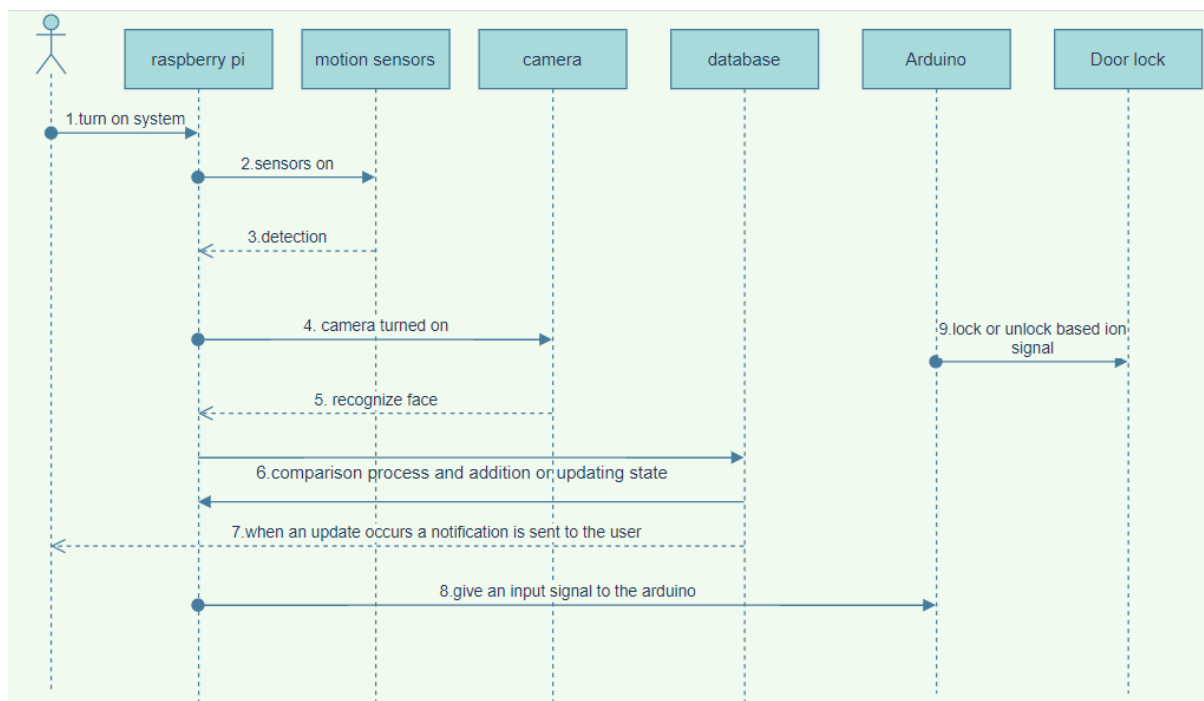


Figure 3-4 Sequence Diagram

3.11 State Diagrams

Figure 3-5 shows the states that the system passes through when it detects any object. The operation starts when the motion sensor detects an object. After that, in case the Pi detects a human's face during the second state, the Liveness Detection and Facial Recognition modules process the captured image, in case it is found to be real, it is compared against all the pre-stored images inside a directory in the Pi to determine whether it is authorized or not.

In the next state, the raspberry pi either opens the door's lock directly in case the visitor is found to be authorized after s/he is recognized or pushes a notification to the user's android application by inserting the image of the visitor or a message in the database.

During the fourth state, the control is at the user's side. The android application receives the notification sent by the raspberry pi which includes the visitor's image. The user then determines whether the visitor is recognized by him/her or not. In this state the user is also able to send commands remotely to the system based on his/her review of the visitor. S/he either sends a signal to the Raspberry Pi to open the door's lock or to turns the alarming system on. S/he is also able to add the visitor's image in the authorized people dataset in case s/he is found to be authorized.

In the fifth and final state, the Raspberry Pi receives the user's command sent by the android application, analyzes it, and sends a signal to the Arduino board accordingly. The Arduino either opens the door's lock or turns the alarming system on.

After the system passes through all the mentioned above states, and determines the proper action to be taken, it ends the operation.

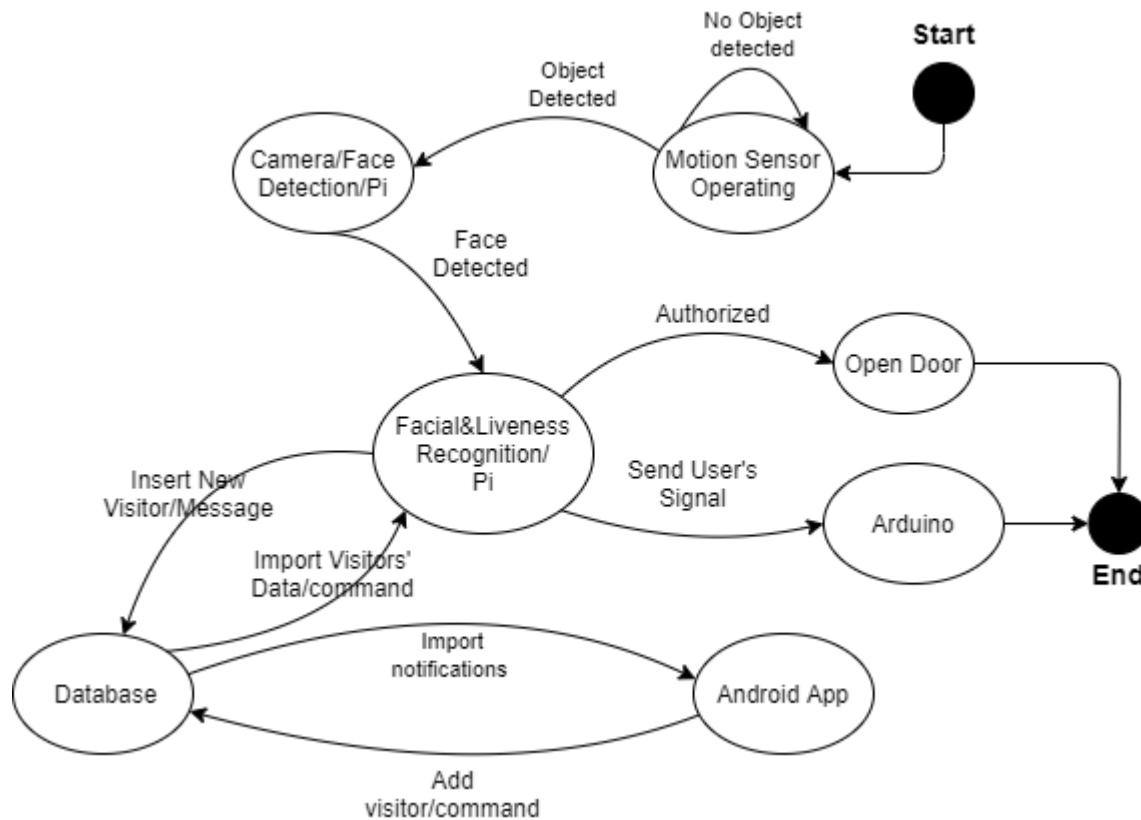


Figure 3-5 State Diagram

3.12 Conclusion

In this chapter several diagrams related to providing a deeper understanding of the system were introduced. Also, the various requirements and components were explained with covering both perspectives, the systems and user perspectives. Furthermore, the system architecture was introduced thoroughly to show the complete picture of the system.

CHAPTER 4

IMPLEMENTATION/SIMULATION AND TESTING

4.1 Introduction

This chapter represents the implementation and demonstration of the system including both parts, the hardware part (Raspberry Pi) and the software part (Android application). It also provides an introduction and usage for all the tools utilized in the system. Also, it shows the communication method between the Pi and the Android app using a database hosted on Laptop server.

4.2 Implementation Tools

The system's main processing unit is Raspberry Pi 4 Model B. It is connected to another minor microprocessor, Arduino, for execution of commands. However, there are various modules and sensors used to create the final prototype of the system.

4.2.1 Raspberry Pi 4 Model B:

The Raspberry Pi, Figure 4-1, is a small on-board computing system that offers many computer processing capabilities used for various kinds of applications. This board has many built-in ports, such as USB and Ethernet, which enables it to connect to different peripherals. Moreover, the device features 40 GPIO pins which enable the board to connect and communicate with a variety of sensors as well as exploring the Internet of Things (IoT).

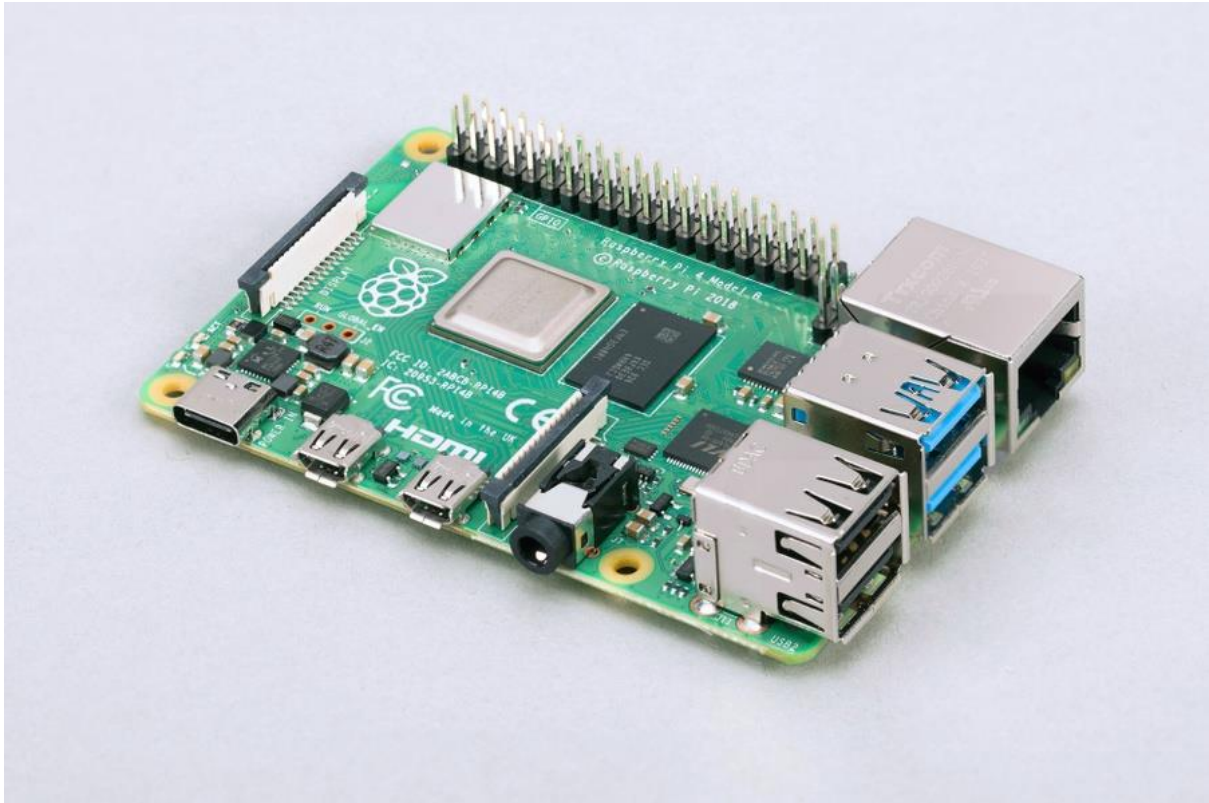


Figure 4-1 Raspberry Pi [6]

The primary operating system that runs on the Raspberry Pi is the Debian-based Linux-based Raspbian. Raspbian is the perfect general purpose-OS for Raspberry Pi users and can accomplish any task you throw at it. [7]

The Raspberry Pi is embedded with different hardware modules including ARM Based Broadcom Processor along with an on-chip GPU, and therefore, it is popularly used for real time image/video processing, IoT based applications and Robotics applications. [8]

Furthermore, the Raspberry Pi supports programming in many languages, such as Python, efficiently through the tools provided by Raspbian OS, and that is why it was chosen as the main board used in our project.

4.2.2 Camera Module:

The camera module used in this project is the Raspberry Pi camera V2.1, Figure 4-2. This camera is a high-definition camera that is compatible with all Raspberry models, an 8-megapixel camera that offers a low noise image capture for accurate face capturing, Raspberry Pi has camera port that connects the camera. Python offers library for the camera module, which is the Pi camera, this should be imported at the very beginning of the code to ensure using the camera.



Figure 4-2 Raspberry Pi Camera [18]

4.2.3 PIR Motion Sensor:

The PIR motion sensor, Figure 4-3, is an electronic sensor used to detect any motion in the targeted area. It allows you to sense motion, almost always used to detect whether has moved in or out of the sensors range. [9]



Figure 4-3 RIP Motion Sensor

These sensors are used in projects and products that detect when a person/object has approached the area in which the sensor is placed in. They are low power and low cost, rugged, have wide lens range, and are easy to interface with other modules (e.g., MCU). [9]

This sensor has been chosen in our project in order to be easily placed in the targeted area, for its small size, and interfaced with the Raspberry Pi.

4.2.4 Android Application:

4.2.4.1 Android Studio:

The Android studio IDE is used in our application to offer a Graphical User Interface between the administrator and the pi. Android studio is the official IDE for Google's android application designed specifically for Android Developments. The programming language used in this studio is a mixture of XML and JAVA.

4.2.4.2 XML:

XML (Extensible Markup Language) is a Markup language used to define a set of rules for encoding documents in a format that is readable for both human and machine, here xml is implemented to make the app friendlier to the user.

4.2.4.3 JAVA:

While the JAVA is a high-level language, class-based, object-oriented programming language, it is a general-purpose programming language. In the Android application, JAVA is used to program buttons and do services, so the App meets the users' functioning. Useful classes and libraries that are offered by JAVA are in use in this project, imported at the beginning of the coding.

4.2.4.4 Notepad ++:

Notepad ++ is a free source code editor and notepad replacement that supports several Languages, one of them is the PHP that is used as a server-side language, Figure 4-4, a general-purpose Scripting language. The PHP script written is to monitor changes in the database and add people to it, PHP here represents the link between the app, the pi, and the database. As a server-side language, the script is executed at the server side so no user interface is required.



Figure 4-4 Linked Database.

4.2.5 Python:

Python is a popular general-purpose programming language that can be used with a variety of applications. It includes dozens of third-party modules that can be found in the python package index (PyPi). Also, python is a “glue language” that can connect many incompatible components together. Due to its ability to run on many system architectures, python has gained a universalizability by contributing to different types of applications.

Python was designed to be a flexible programming language by enabling it to use extensions (through modules), and therefore Python does not have all its functionality built into its core. This design has scaled its popularity for adding programmable interfaces to existing applications. [10]

Furthermore, Python’s large library set available has been cited as one of the greatest strengths of the language. These libraries provide tools which can be used for a various of missions. Many protocols are supported such as HTTP for internet-facing applications, it also includes tools for graphical user interfaces, connecting to databases, ...etc. [10]

Python is widely used for many purposes in different tracks. In the area of Machine Learning, it is currently the most prevalent, mature, and well-supported among programming languages making it suitable for many developers working on Computer Vision. Moreover, implementing CV through Python enables developers to automate their applications which involve visualization. [11] This project employs Python as the main programming language in the Raspberry Pi board. Many libraries were used such as OpenCV, TensorFlow, face-recognition, ...etc.

4.2.5.1 OpenCV:

Implementing Computer Vision in Python has come true by using OpenCV library. It is an open-source computer vision and machine learning software library. OpenCV's primary purpose of building was to create a common ground for computer vision applications and to improve the use of machine visualization in different applications.

The library has more than 2500 optimized algorithms which involve computer vision and machine learning algorithms. There are various applications for these algorithms which include, but not limited to, face detection and recognition, object identification, human actions classification, finding similar images from database, ...etc. [12]

As a doorbell system project, there are many tasks which have been implemented using OpenCV library. OpenCV has made it possible and easy for us to process images for Face Detection which determines a human presence, Liveness Detection using images classification and blink detection, and lastly Facial Recognition. All these tasks benefited from the capability of OpenCV to operate in real-time.

4.2.5.2 TensorFlow:

TensorFlow is a python open-source library used for numerical computation dedicated to making machine learning faster and easier. Machine Learning is a complex task that enables computer programs to access stored data and use it in order to learn how to behave for a specific activity. However, by using some frameworks, such as TensorFlow, machine learning models have been less difficult and challenging to implement. TensorFlow integrates a plenty of machine learning and deep learning models and algorithms and makes them useful for many applications such as acquiring data, training models, and serving predictions. [13]

TensorFlow library was mainly used to achieve the Liveness Detection model in our project which is applied on images' faces to check whether they are real or fake. It is utilized

in the training and running of a deep neural network on a dataset of images to create image recognition making it able to classify images into different categories.

4.2.5.3 Convolutional Neural Networks (Deep Learning):

CNN is a network model which employs a dataset of pictures in order to visualize and predict the object in an image which has never been seen before.

CNN is composed of a series of layers with the “Convolution” layer being the central of the network. Each layer is made from a set of neurons which do not connect to all of the neurons in the next layer but rather to a region of it. The output layer of the network is a 3D data which holds several values each indicating a probability for one object. Therefore, the class of the object is predicted.

CNN is composed of two main parts: The Hidden Layers/Feature Extraction part, and The Classification part, Figure 4-5. In the first part, each separate layer extracts the features of the input data and send them as an input to a region of the next layer in which the same operation is repeated. In this part, the features of the image are detected. In the second part, all of the layers operate as a whole on the extracted features to assign a probability for the object on the image predicted by the algorithm. [14]

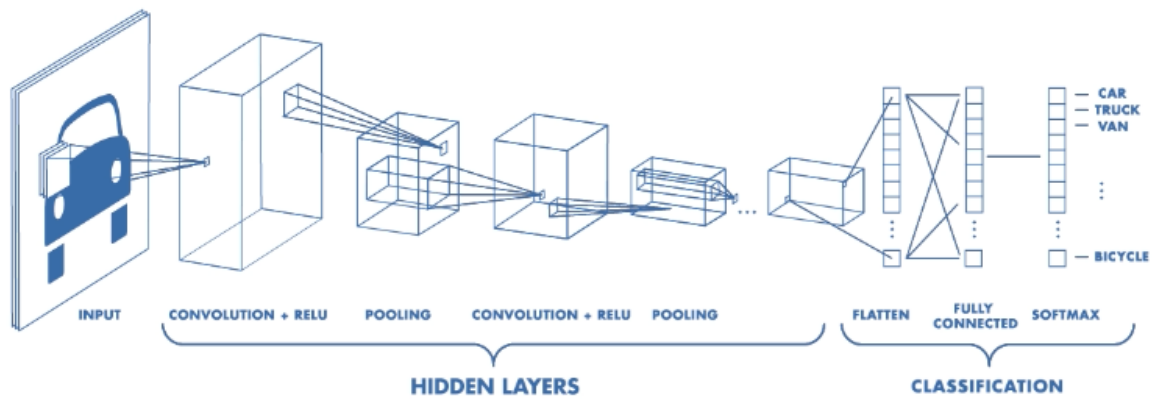


Figure 4-5 CNN Architecture [14]

During the first part, the Feature Extraction, the convolution is applied on the input data (image) along with a set of weights called “Filter”. The convoluted output is called a feature map, which is a data type that merges both, the input data and the filter.

Different convolution operations are applied on the input using different filters and therefore creating many feature maps. Features maps are then inserted to a Pooling layer, after being applied to a non-linearity function such as ReLU, in which their dimensionalities are reduced in order to reduce the number of parameters and computations in the network and therefore reducing training time, Figure 4-5.

Figure 4-6 shows the filter, in green, is sliding over the blue square, which is the input data. The convolution is done as a dot product by multiplying individual cells and then adding the sum of the cells of the result matrix to one cell in the red square, which is the feature map. [14]

Filters are 2-dimensional array of data (set of weights) which are designed carefully to detect a specific pattern/feature in an image. When applied to an input data larger in size, it

slides across the entire image enabling it to discover that specific feature all around the image. [15]

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

Figure 4-6 Convolution Operation [14]

4.2.6 Face Detection

Face detection is the approach used to detect faces given by an image, after detecting a face in an image, and knowing the exact location or the coordinates of that face, we extract the face for further processing which will be applied in facial recognition ahead.

Face detection is a machine learning approach, where a cascade function is created from a lot of positive and negative images. This algorithm needs many positive images (images with faces) and negative ones (images without faces) to train the classifier. Then the features are extracted from it. Figure 4-7 shows that each feature is a single value obtained by subtracting sum of pixels under white rectangle from those under black rectangle. [19]

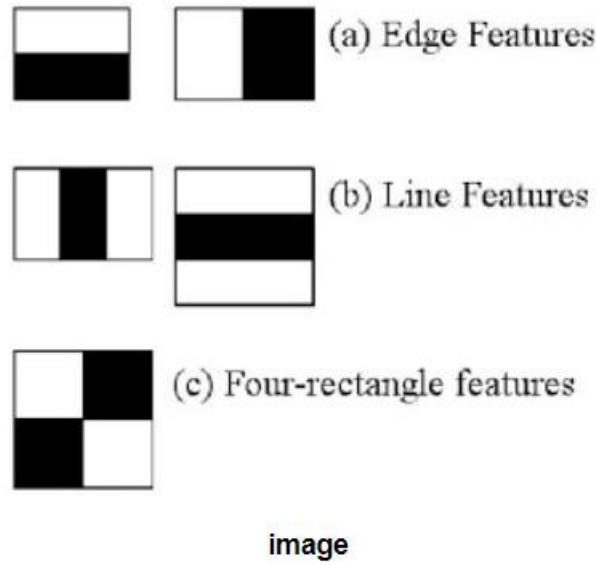


Figure 4-7 Haar Features [19]

HaarCascades are xml files in OpenCV used as trainers to detect objects, our system detects faces so it uses already trained classifiers provided by OpenCV such as classifiers for eyes, smiles, nose etc. the xml files can be found in the HaarCascades folder. In this project eyes and faces classifiers are to be used, applying an approach to get face first the eyes second, and sort a way to join both for a real-time-face detection. [19]

4.2.7 Blink Detection

There are many methods and approaches proposed to implement blink detection. However, an efficient method based on detecting eyes landmarks is available and implemented in this project. This method is based on real-time facial landmark detectors which captures face features, including eye corners and eyelids. Those landmarks are trained using datasets to different illuminations and facial expressions, and therefore they provide sufficient locations. On the other hand, Eye Aspect Ratio (EAR) is depicted using eye

landmarks by the following equation:
$$EAR = \frac{||p2-p6||+||p3-p5||}{2||p1-p4||}$$

Where p_1, \dots, p_6 are the eye landmarks.

The EAR value is a constant value most of the time when the eye is open. However, it drops down close to zero when the eye is closed. Therefore, each time the EAR's value drops close to zero, it is said that the eye has closed. Similarly, when it returns to its constant range after dropping down, it is said the eyes have reopened and as a result a blink is detected.

Blink Detection has been used in our project as a second algorithm for Liveness Detection. Since the first algorithm is not sufficient when it comes to printed faces, there should be a solution for this problem. Therefore, blink detection provides efficiency in this situation as it can detect both, printed images and phone images.

4.2.8 Facial Recognition:

Facial Recognition refers to the activity done by an application to recognize people through analyzing their faces. This is achieved by receiving a person's image as input and comparing it with a set of pre-processed and recognized images. When a matching occurs, then the input image is said to be recognized and identity is determined by looking at the name assigned to the matched image.

Facial Recognition is implemented through using Machine Learning which extracts and compares feature vectors of faces. A machine learning algorithm receives an image (with human face) as input. After that it runs a neural network on the input image to analyze different patterns of the face (e.g., nose, eyes, ...etc.) and measure their features (e.g., heights, widths, colors, ...etc.). Then, the algorithm maps the face into a feature vector by arranging the measures done during the previous phase in a certain order. Once the feature vector is obtained, the algorithm matches/compares the feature vector obtained with a set of stored feature vectors, stored in local memory or database, and extracts the distance between the obtained vector and each individual vector. A matching occurs when the distance between two images is very small. [17]

In this project, we have utilized “face-recognition” library which creates vectors consisted of 128 features. Our system implements facial recognition in order to recognize visitors and take automatic actions accordingly. If a visitor matches a feature vector stored at the local dataset, then the door is opened automatically.

4.3 Implementation Summary

This chapter presents the implementation and demonstration of the system including both parts, the hardware (Raspberry Pi) and the software (Android app). It also provides an introduction and usage of each tool utilized in this project. It also shows the communication method between the Pi and the app using database.

4.3.1 Main System

4.3.1.1 Preparing Environment:

The main board utilized in this system is the Raspberry Pi 4 Model B which is running Debian-based operating system Raspbian, Figure 4-8.

Depending on the system being used to install the Raspbian OS, navigate to the following link <https://www.raspberrypi.org/software/>, and choose a compatible Raspberry Pi OS Imager executable file. After that, open the software and choose Raspberry Pi OS (32-bit) option, insert an SD card to the computer using an SD card reader and then press on Download to download the system on the SD card.

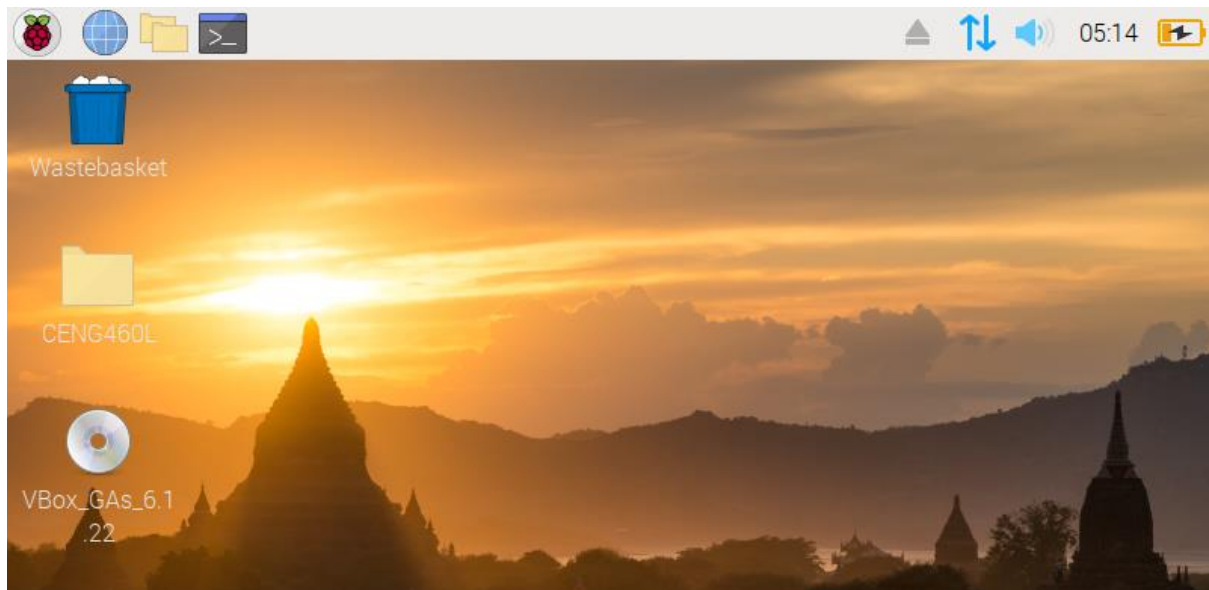


Figure 4-8 Raspbian OS

Also, there must be a web server on the Raspberry Pi that will enable the system to handle the user's web requests (e.g., http requests). For this purpose, a bundle of software and modules (Apache + PHP + MySQL) should be installed on the Raspbian to have a functioning web server.

All the software required can be installed using the system's terminal. Refer to Appendix C.

4.3.1.2 Creating the Database:

The database used in this system is very simple. It is composed of two tables dedicated for two-way communication between the main system unit and the Android App.

The E/R Diagram of the database is shown in Figure 4-9.

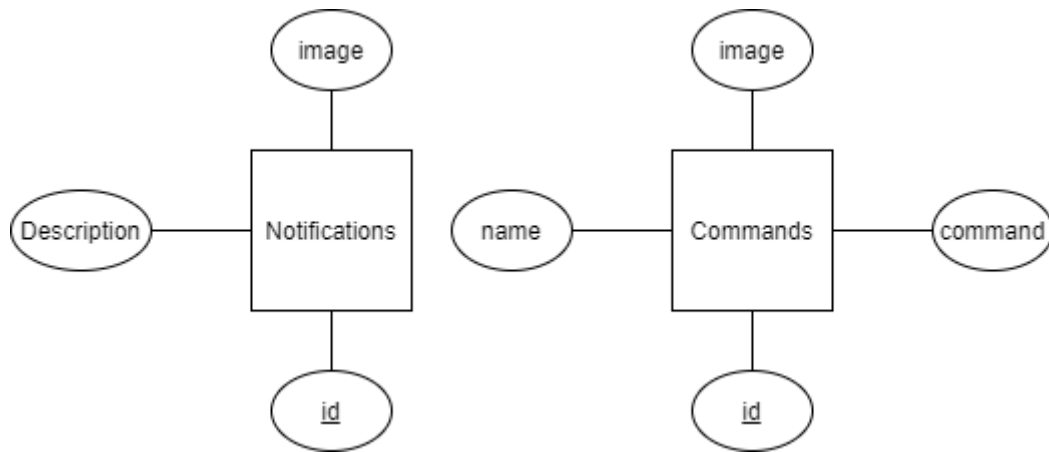


Figure 4-9 Database E/R Diagram

Table “Notifications” is used by the Pi to send notifications to the Android app while table “Commands” is used by the user to send his/her commands to the Pi. Both tables have no relationship to each other.

4.3.1.3 Implementing System’s Functionality:

The system uses Computer Vision and Machine Learning in order to perform an accurate and real recognition of human faces. To accomplish this final task, there are different steps that should be followed to make the implementation easier.

Many separate modules which are integrated together have been created using different tools and methods. These modules include Face Detection, Liveness Detection, Blink Detection, and Facial Recognition.

This project is designed to recognize humans by employing an algorithm to detect their faces. Therefore, Face Detection has been used frequently throughout this system. First, when the system first operates, it executes the Face Detection algorithm to check for any humans’ faces around. In case it detects a human, the system can proceed with executing the following modules or repeats itself otherwise. Also, for the purpose of implementing a Liveness Detector, there should be a large dataset which contains samples of fake and real

faces of humans. To guarantee a decent accuracy, the number of sample images must be thousands. However, snipping these faces manually is impossible and time-consuming, instead we have used a Face Detector that could operate on a video of humans and store all the faces that appear in each frame, and this makes the process of building a dataset much faster and easier.

The second step is to implement the Liveness Detection. This detector is executed after the face detection, it runs an algorithm based on Deep Learning on the camera frames and checks whether the person is real or fake (e.g., a picture on a phone screen being shown to the camera). The first step to implement Liveness Detection is to train a CNN on the dataset prepared in the first module and then to classify faces based on the trained model.

After executing the Liveness Detection algorithm, a second Liveness Detection algorithm is executed to double-check the results from the first one. Blink Detection is an algorithm based on Deep Learning dedicated to detecting the movement of the person's eyes. When s/he closes and then reopens her/his eyes, the algorithm detects this movement and classifies the person as real, and fake in case their eyes are still.

Once a person is classified to be real, his/her face is processed by the Facial Recognition algorithm to determine his/her identity if already recognized. This algorithm extracts the face's features in the first place, and then compares these features with a set of stored and recognized faces. In case a matching occurs, the algorithm returns the name assigned to the matched picture and sends it to the system in order to determine the operation to be executed.

To implement this system, the steps mentioned above should be followed to prepare the final program to be executed by the Raspberry Pi.

To begin with, a bunch of python libraries have been utilized throughout the project. Some are used for face detection, others for liveness detection, and a few for facial recognition. To install these libraries, refer to appendix C.

Then, refer to the attached CD and install the directory “Liveness Detection” on your system, Figure 4-10.

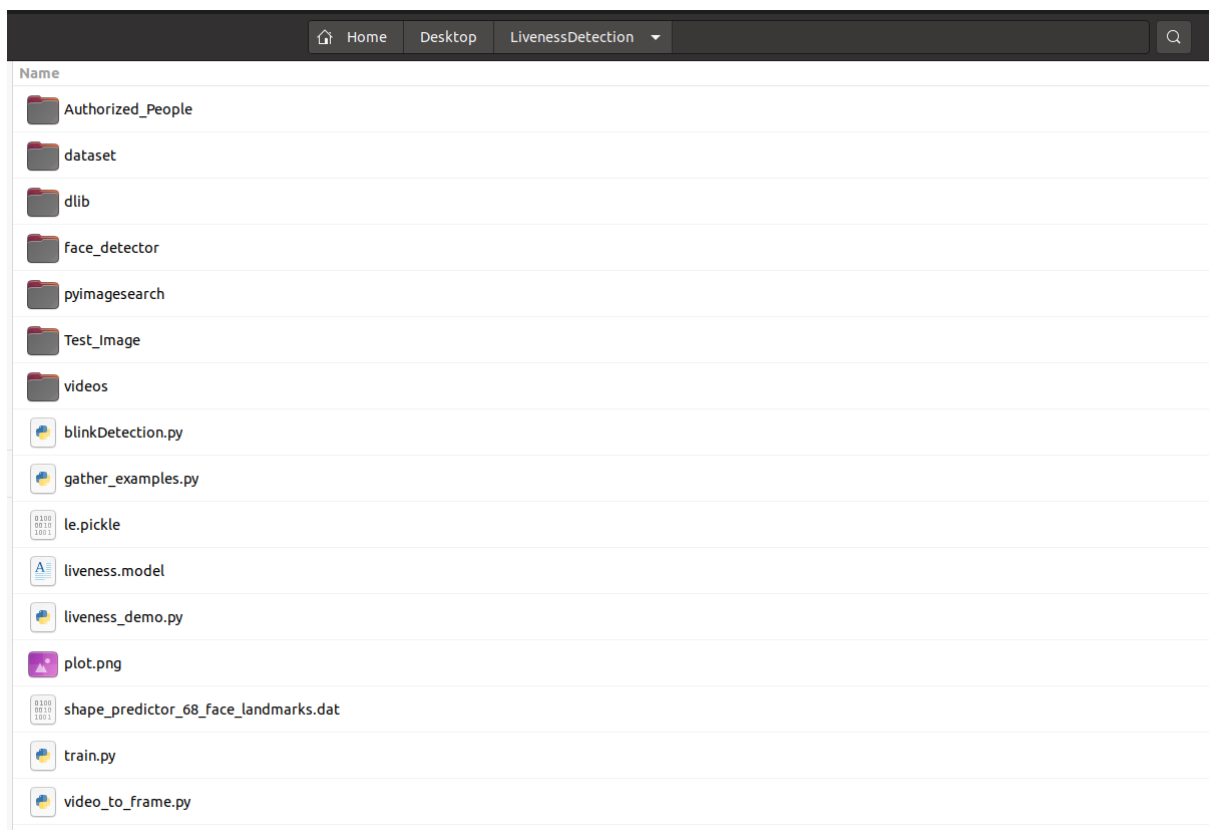


Figure 4-10 Liveness Detection Directory

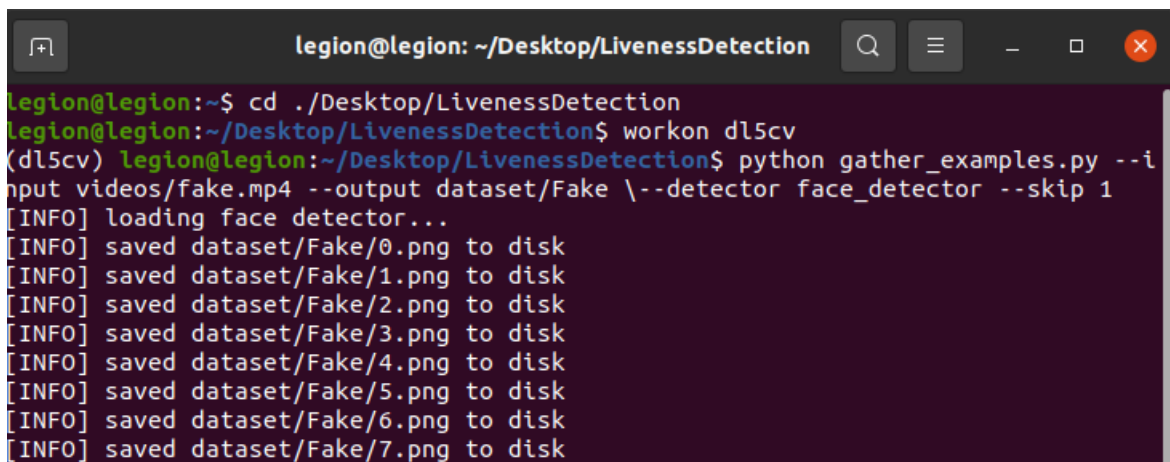
For the Face Detector, the following directories and files use face detection algorithm to work: face_detector, gather_examples.py, and le.pickle. However, the module used to detect faces in images is based on Caffe model using OpenCV’s deep neural networks. The caffe model requires two files to work, **.prototxt** file (to define the model architecture, i.e., layers), and **.caffemodel** (which contains the weights of the layers). These two files are found

in the “face_detector” directory, and therefore this is the actual folder that performs face detection. This directory is given as a parameter for other modules to use face detection.

For the Liveness Detection, we should prepare the dataset for the fake and real faces, and then train a Convolutional Neural Network on this dataset to distinguish between real and fake faces.

First, to prepare the “fake” dataset, you can either use the provided “fake.mov” video in the “videos” directory, Figure 4-10, or create your own custom video by playing a video of people from different origins on a device and filming the device’s screen from another device/camera. Also, you should prepare the “real” dataset through installing another video which contain people from different origins or use the provided “real.mov” video in the “videos” directory.

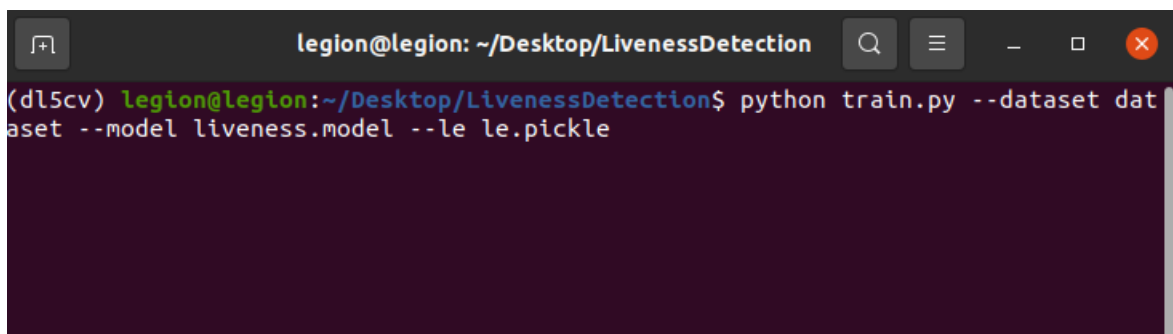
After that, to save the face ROIs (Region of Interests) from all the frames in each video into the dataset, you should input both videos to the face detector and provide the destination folder, Figure 4-11, by typing the following commands in the terminal: “python gather_examples.py --input videos/video_name --output dataset/fake \--detector face_detector --skip 1” (for fake), and “python gather_examples.py --input videos/video_name --output dataset/real \--detector face_detector --skip 1” (for real).

A terminal window titled "legion@legion: ~/Desktop/LivenessDetection" with standard window controls. The terminal shows the following commands and output:

```
legion@legion:~$ cd ./Desktop/LivenessDetection
legion@legion:~/Desktop/LivenessDetection$ workon dl5cv
(dl5cv) legion@legion:~/Desktop/LivenessDetection$ python gather_examples.py --i
input videos/fake.mp4 --output dataset/Fake \--detector face_detector --skip 1
[INFO] loading face detector...
[INFO] saved dataset/Fake/0.png to disk
[INFO] saved dataset/Fake/1.png to disk
[INFO] saved dataset/Fake/2.png to disk
[INFO] saved dataset/Fake/3.png to disk
[INFO] saved dataset/Fake/4.png to disk
[INFO] saved dataset/Fake/5.png to disk
[INFO] saved dataset/Fake/6.png to disk
[INFO] saved dataset/Fake/7.png to disk
```

Figure 4-11 Building Dataset Command

After executing both commands, the dataset for both, real and fake faces, will be full and ready to be used. Second step is to train a CNN on the images inside the dataset in order to distinguish between fake and real images. This is done through typing the following command in the terminal: “python train.py --dataset dataset --model liveness.model --le le.pickle”, Figure 4-12, where the parameter “dataset” is the path to the dataset directory, “model” is the path to the output model file, and “le” is the path to the serialized label encoder file.

A terminal window with a dark background. The title bar shows 'legion@legion: ~/Desktop/LivenessDetection'. The command prompt is '(dl5cv) legion@legion:~/Desktop/LivenessDetection\$'. The command entered is 'python train.py --dataset dataset --model liveness.model --le le.pickle'.

```
(dl5cv) legion@legion:~/Desktop/LivenessDetection$ python train.py --dataset dataset --model liveness.model --le le.pickle
```

Figure 4-12 Model Training Command

Note that the “train.py” file will apply the “livenessnet.py” script in “pyimageserach” folder on the dataset to create the model file. This script is the actual CNN that applies the several layers mentioned earlier on the input images. Figure 4-13 shows the python code to implement the CNN.

```

11 class LivenessNet:
12     @staticmethod
13     def build(width, height, depth, classes):
14         # initialize the model along with the input shape to be
15         # "channels last" and the channels dimension itself
16         model = Sequential()
17         inputShape = (height, width, depth)
18         chanDim = -1
19         # if we are using "channels first", update the input shape
20         # and channels dimension
21         if K.image_data_format() == "channels_first":
22             inputShape = (depth, height, width)
23             chanDim = 1
24
25         # first CONV => RELU => CONV => RELU => POOL layer set
26         model.add(Conv2D(16, (3, 3), padding="same",
27             input_shape=inputShape))
28         model.add(Activation("relu"))
29         model.add(BatchNormalization(axis=chanDim))
30         model.add(Conv2D(16, (3, 3), padding="same"))
31         model.add(Activation("relu"))
32         model.add(BatchNormalization(axis=chanDim))
33         model.add(MaxPooling2D(pool_size=(2, 2)))
34         model.add(Dropout(0.25))
35         # second CONV => RELU => CONV => RELU => POOL layer set
36         model.add(Conv2D(32, (3, 3), padding="same"))
37         model.add(Activation("relu"))
38         model.add(BatchNormalization(axis=chanDim))
39         model.add(Conv2D(32, (3, 3), padding="same"))
40         model.add(Activation("relu"))
41         model.add(BatchNormalization(axis=chanDim))
42         model.add(MaxPooling2D(pool_size=(2, 2)))
43         model.add(Dropout(0.25))
44
45         # first (and only) set of FC => RELU layers
46         model.add(Flatten())
47         model.add(Dense(64))
48         model.add(Activation("relu"))
49         model.add(BatchNormalization())
50         model.add(Dropout(0.5))
51         # softmax classifier
52         model.add(Dense(classes))
53         model.add(Activation("softmax"))
54         # return the constructed network architecture
55         return model

```

Figure 4-13 Convolutional Neural Network Implementation

To double-check the output of the first liveness detection algorithm, another algorithm is executed once the first is finished. Blink Detection detects eyes landmarks and reads EAR value constantly. When it drops down close to zero, a blink is countered.

Files used for blink detection are “blinkDetection.py script” to performs blink detection on video stream, and “shape_predictor_68_face_landmarks.dat” which is the dlib’s pre-trained facial landmark detector used to place landmarks on video frames.

After confirming that the person is real, the program executes the Facial Recognition module which is designed to identify visitors. This module receives the visitor’s image as

input and compares its features to all pre-stored images' features and determines whether it is recognized or not.

To implement this module, first store all the initial recognized people's faces in the directory "Authorized_People" and name the files with their names, then adjust the "liveness_demo.py" script, Figure 4-14, to add the facial recognition method in case the visitor is found to be real.

```

154         face = np.expand_dims(face, axis=0)
155         # pass the face ROI through the trained liveness detector
156         # model to determine if the face is "real" or "fake"
157         preds = model.predict(face)[0]
158         j = np.argmax(preds)
159         label = le.classes_[j]
160
161         if label == "fake":
162             #print("This face is fake")
163             while True: #replace this code with a notification to the user and then reading any potential commands
164                 print("This face is fake")
165                 key = cv2.waitKey(1) & 0xFF
166                 # if the 'q' key was pressed, break from the loop
167                 if key == ord("q"):
168                     break
169             else:
170                 run_face_recognition(frame)
171
172         # draw the label and bounding box on the frame
173         label = "{}: {:.4f}".format(label, preds[j])
174         cv2.putText(frame, label, (startX, startY - 10),
175                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

```

Figure 4-14 Facial Recognition Method Call

Implementing run_face_recognition() method is done using OpenCV and face_recognition libraries. First, return all the images stored in "Authorized_People" directory, then create a method to extract each single image's features and put them inside an array. After that, captures an image, extract its features, and compare these features with the features inside the array. In case a matching occurs, return the matched image's name, and open the door lock. If no matching occurs, send a notification to the user. Refer to the attached CD for the code.

For the communication with the android application, it is achieved through uploading images/messages to the database (sending notifications) or reading images/messages from the database (reading user's commands). For contrast, when someone tries to spoof the doorbell

system (through showing a fake face), the system malfunctions immediately and sends a notification to the owner telling him/her that the system is being spoofed by someone. It then waits for a command from the user to know whether to resume functioning or keep malfunctioning.

To implement this, when a face is determined to be fake, first a message is uploaded to the database (telling the user a spoofing trial is detected), and then the program reads the last command sent by the user to the database. If it is already executed (its ID is already stored), the system keeps reading the last command until its ID is higher than the stored ID (ID of last executed command) by 1. At this point, the system returns the command and checks its value. In case the value is “reset”, the system can resume functioning, and keep malfunctioning otherwise, Figure 4-15.

```

if label == "fake":
    sqls = "INSERT INTO Notifications (image, description) VALUES (%s,%s)"
    im = "A"
    desc = "Someone is trying to spoof your doorbell system"
    myCursor.execute(sqls,(im,desc))
    myDB.commit()
    while True:
        #return the maximum command id from the database
        myDB = mysql.connector.connect(host="localhost", user="root", password="cenG@495", database="CENG495")
        myCursor = myDB.cursor()
        getIdStatement = "SELECT MAX(id) from commands"
        myCursor.execute(getIdStatement)
        ids = myCursor.fetchall()[0][0]
        #return the command associated with the maximum id.
        command = "SELECT command from commands where id = %s"
        myCursor.execute(command, (ids,))
        cmd = myCursor.fetchall()[0][0]

        if ids == MAXID: # No new command has been received yet
            print("No command has been received yet")
        elif ids > MAXID: # A new command has been received
            MAXID = ids
            if cmd == "reset": # Resume functioning (break from malfunction loop)
                break
            else: # Stay in this loop (keep malfunctioning)
                print("Stay here")
    else:
        run_face_recognition(frame)

```

Figure 4-15 Notification-Command Code

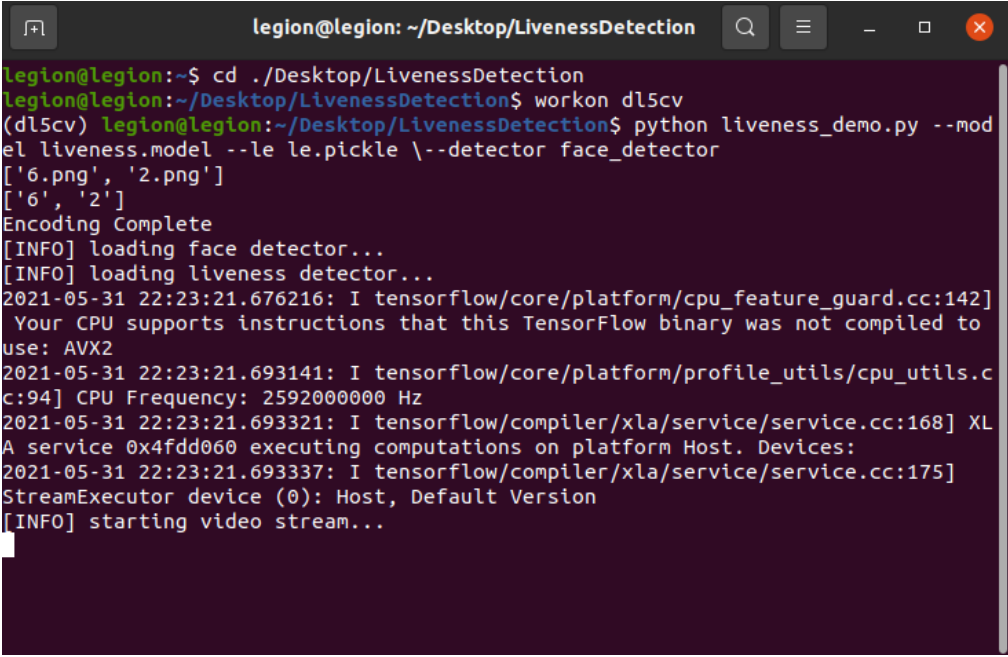
Also, when a user sends a command which is expected to turn a hardware device on, this command is handled by a function which utilizes the GPIO of the Raspberry Pi. Similar to receiving a “reset” request as depicted in Figure 4-15, inside the while loop (Refer to attached CD for code) a function is included to listen to the latest commands sent by the user,

if the command's value is "Open Door", then the program executes a function in which a GPIO pin connected to the Lock (PIN in our case) is setup to output and high state is sent to it. This unlocks the door lock. Turning the alarming system on works in a similar manner.

Images are uploaded to and returned from the database in the form of a text encodings using base64 library. Images are encoded to text before they are uploaded to the database and decoded after they are returned. This is achieved using base64 encoding and decoding methods. Refer to attached CD for codes.

Finally, to run the whole program, type the following command into the terminal:
"python liveness_demo.py --model liveness.model --le le.pickle \--detector face_detector",

Figure 4-16.

A terminal window titled 'legion@legion: ~/Desktop/LivenessDetection' with standard window controls. The terminal shows the following commands and output:

```
legion@legion:~$ cd ./Desktop/LivenessDetection
legion@legion:~/Desktop/LivenessDetection$ workon dl5cv
(dl5cv) legion@legion:~/Desktop/LivenessDetection$ python liveness_demo.py --model liveness.model --le le.pickle \--detector face_detector
['6.png', '2.png']
['6', '2']
Encoding Complete
[INFO] loading face detector...
[INFO] loading liveness detector...
2021-05-31 22:23:21.676216: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
2021-05-31 22:23:21.693141: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2592000000 Hz
2021-05-31 22:23:21.693321: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x4fdd060 executing computations on platform Host. Devices:
2021-05-31 22:23:21.693337: I tensorflow/compiler/xla/service/service.cc:175] StreamExecutor device (0): Host, Default Version
[INFO] starting video stream...
```

Figure 4-16 Program Running.

4.3.2 Android Application

4.3.2.1 Home Activity

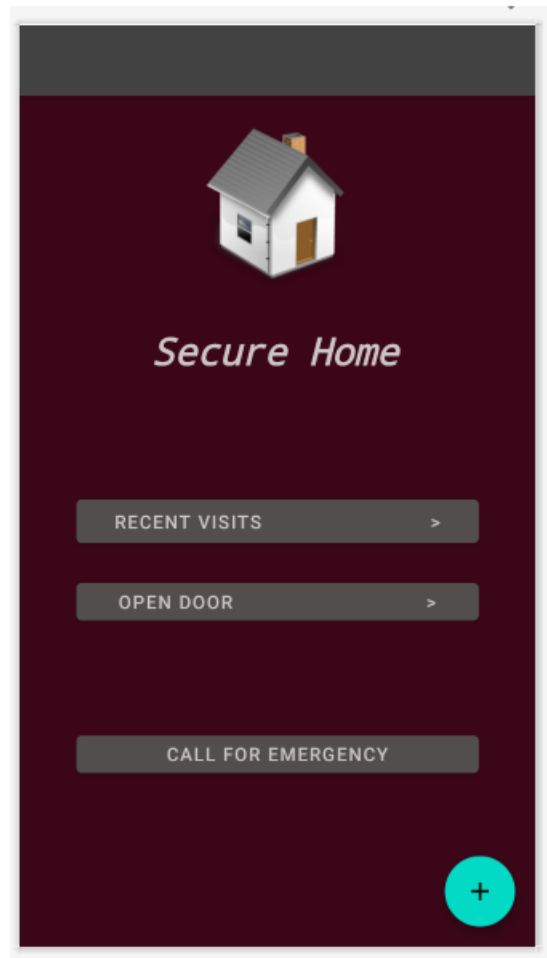


Figure 4-17 Main Activity

Home Activity, Figure 4-17, which is the main activity represents the control panel of the application. It represents the activity which the user navigates to when s/he first opens the application. Controls in this activity allow user to open the door, check recent visits, add a new guest, and call an emergency (turn alarming system on) when the user doubts in a visitor.

4.3.2.2 Recent Visits Activity

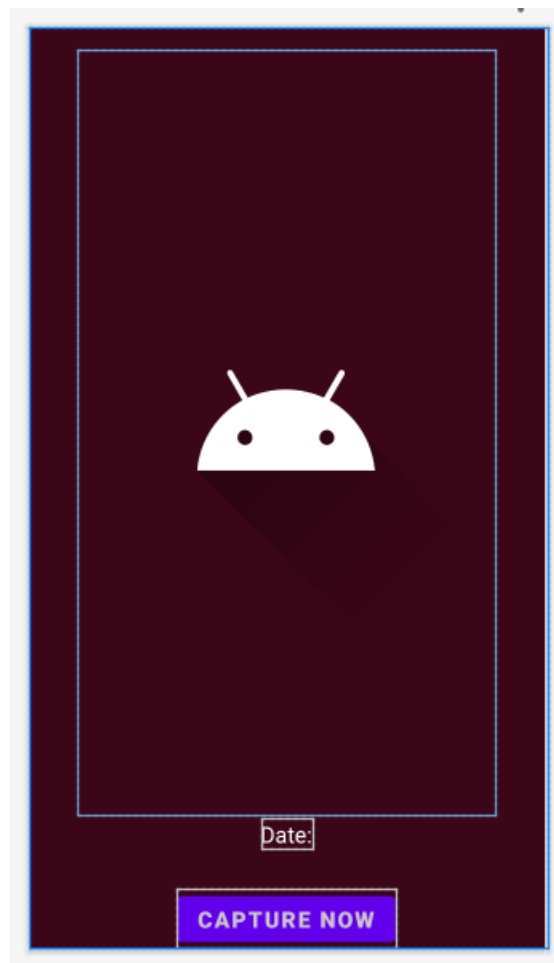


Figure 4-18 Recent Visits Activity

Figure 4-18 represents where images of visitors are presented. It is linked with a php file that gets all the images from the database and stores them in this activity as a list view. Each image is listed with the date of its capture, so the user can check the time.

Images brought from database are string formatted images, in this activity each image is redisplayed using the base64 class to reconvert it to a PNG format.

The capture now button allows the user to take a picture instantly by the pi, this is used to see the view at the current time.

4.3.2.3 Add Guest Activity

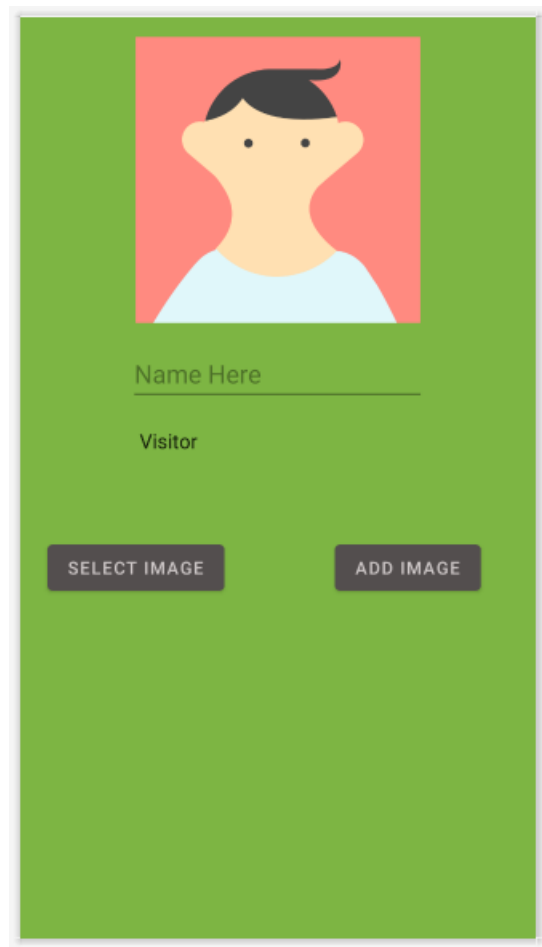


Figure 4-19 Add Guest Activity

After the (+) button is pressed in the main activity, the user is navigated to the Add Guest Activity, Figure 4-19. This activity is composed of a form (ImageView and DataField) for the user to fill in the image and name of the new authorized visitor. The add image button executes a PHP script allowing to add an image to the database. The image after being selected is converted to a byte Array type and then encoded to string using java class base64. After that, it is uploaded to the database in the format of a string.

4.3.2.4 Notifications and Services

```
while (ServiceIsRun){

    String url = "http://10.0.2.2:80/images/getItem.php";
    RequestQueue queue = Volley.newRequestQueue(getApplicationContext());
    StringRequest request= new StringRequest(Request.Method.GET, url, new Response.Listener<String>() {

        @RequiresApi(api = Build.VERSION_CODES.O)
        @Override
        public void onResponse(String response) {

            try {
                JSONObject jsonObject = new JSONObject(response);
                Response = jsonObject.getString( name: "response");
                image =jsonObject.getString( name: "image");
                decodedString = Base64.decode(image,Base64.DEFAULT);
                //Toast.makeText(getApplicationContext(),decodedString.toString(),Toast.LENGTH_LONG).show();
                y =Integer.parseInt(Response);

            }

            catch (JSONException e){
                Toast.makeText(getApplicationContext(),e.toString(),Toast.LENGTH_LONG).show();
            }

        }

    });
```

Figure 4-20 the Service Class

The section of code shown in Figure 4-20 represents the service running in the background. An infinite loop starts every time the application runs; this loop executes a PHP file which accesses the database and returns the last image uploaded using the last ID technique. After that, the ID of the image is returned and compared with the ID of the image which was returned before it as shown in Figure 4-21. If the ID just returned is greater than the ID which was returned just before it, the new ID is an ID of a new image sent by the system. Then a notification is generated for the user to see. In case the user presses on the notification, s;/he is navigated to the image of the new visitor.

```

if (y > x) {

    Intent i = new Intent(getApplicationContext(), RecentVisits.class);
    i.putExtra( name: "image", decodedString);
    i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    startActivity(i);

    i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
    PendingIntent pendingIntent = PendingIntent.getActivity(getApplicationContext(), requestCode: 0, i, flags: 0);
    NotificationCompat.Builder NA = new NotificationCompat.Builder(getApplicationContext(), channelId: "sam");
    NA.setSmallIcon(R.drawable.ic_launcher_foreground);
    NA.setContentTitle("new notification");
    NA.setContentText("New Activity Nearby" +String.valueOf(y));
    NA.setPriority(NotificationCompat.PRIORITY_DEFAULT);
    NA.setContentIntent(pendingIntent);
    NA.setAutoCancel(true);
    NotificationManager notificationManager = null;
    NotificationChannel channel = new NotificationChannel( id: "sam", name: "myChannel", NotificationManager.IMPORTANCE_LOW);
    notificationManager = (NotificationManager) getApplicationContext().getSystemService(NotificationManager.class);
    notificationManager.createNotificationChannel(channel);
    notificationManager.notify( id: 0, NA.build());
    editor.putString("key",String.valueOf(y));
    x= y;
}

```

Figure 4-21 Notification

```

2  <?php
3  require_once 'init.php';
4  //$query = "select id from admin where id>0";
5  $max_query = "SELECT MAX(id) FROM admin" ;
6
7
8
9  if($return=mysqli_query($con,$max_query)){
10     $result=mysqli_fetch_row($return);
11
12     $key = $result[0];
13     $query="select image FROM admin where id = '$key'";
14
15
16
17  if($return1=mysqli_query($con,$query)){
18     $result1 = mysqli_fetch_row($return1);
19     $re=$result1[0];
20     $encoded_data = base64_encode(file_get_contents($re));
21     //$im=file_get_contents($result1);
22     //$image= base64_encode($im);
23
24     echo json_encode(array('response'=>$key, 'image'=>$encoded_data));
25  }
26  else
27     echo 'connection error';
28
29  }
30  else
31     echo 'connection error1';
32  ?>

```

Figure 4-22 PHP Getting the Images

As seen in Figure 4-22, the script gets the maximum ID, which is the last one, and the image. It then sends it to Android as a JSON array containing two objects which are the ID and the encoded data (image).

Refer to Appendix B for app and system's instruction manual.

4.4 Test Cases and Acceptance Criteria

The system was successfully tested in a Local Area Network (LAN) with simple electronic components representing door lock and alarming system. The communication point between the Raspberry Pi and the Android app was the Apache server and phpMyAdmin database.

Figure 4-23 shows the main and final system unit which has been tested in this project. There three main electronic components and one module connected to the Raspberry Pi's GPIOs. The buzzer which represents alarming system, is connected to a digital GPIO. PIR sensor which detects any motion inside its operating range, is also connected to a digital GPIO. A red LED which represents the door lock, is connected digitally to the Raspberry Pi. Last module is the Raspberry Pi Camera Module which is connected to the Raspberry Pi's CSI connector.

During testing, the system operated with no errors and was able to detect human presence sufficiently through processing camera footage for human faces. Moreover, the system operated at a considerable amount of speed when a spoofing attempt had been applied. An instant notification was received at the android application and then it was obvious that the system stopped working as it longer could recognize authorized visitors, this was as expected. Also, the system was able to resume functioning after a user sent a "reset" command to the Pi.

For the autonomous part of the system, it worked perfectly as it was able to automatically open the door when an authorized visitor approached the camera.

The system was not tested at night, and it is expected to operate with less efficiency as the camera is not provided with a lighting for night. Therefore, recognizing faces and detecting spoofing attempts will be more challenging and will cause the system to not behave as expected.

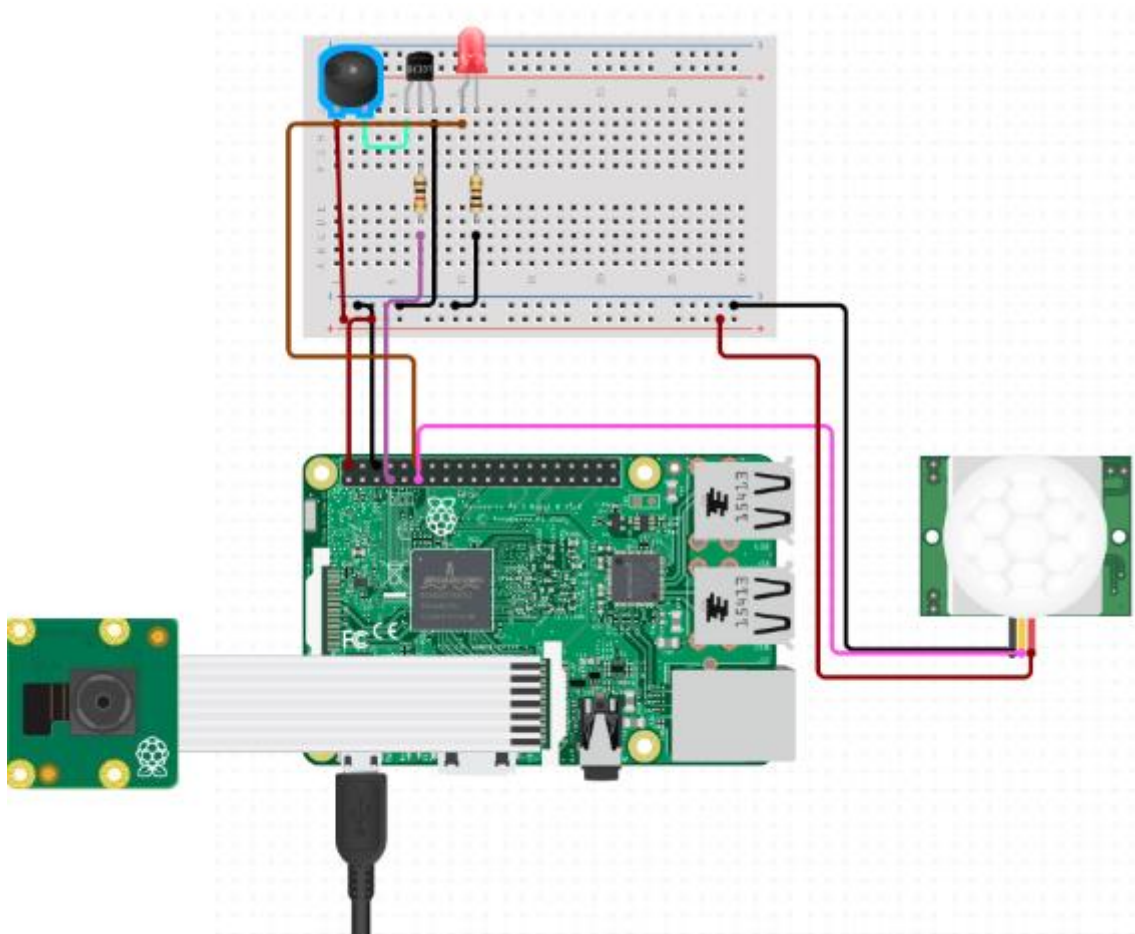


Figure 4-23 System Circuit

System Requirements:

1. The Raspberry Pi and Android application must be connected to the same network.
2. The Raspberry Pi should operate constantly without performing any operation as long as there is no human presence.

3. The Pi should receive input from the motion sensor readings and send notification to the android app.
4. The Raspberry Pi should detect human presence through receiving camera input.
5. The Pi should be able to distinguish real humans and catch spoofing attempts.
6. The Pi should alert the user through app notification when spoofing is attempted.
7. The Pi should alert user when a real visitor is not recognized through pushing a notification with the visitor's image.
8. The Pi should not alert the user when a visitor is real and recognized.
9. The Pi should automatically open the door lock when a visitor is recognized.
10. The Android application should receive notifications sent by the Pi for: spoofing attempts and unrecognized visitors.
11. The user should be able to send commands to the Pi by the Android application to: open the door, turn the alarming system on, and add new recognized visitor.

Input:

1. The camera footage.
2. The PIR Motion sensor readings.
3. The user's commands sent by the Android app.

Output:

1. The Pi send outputs to control door lock and alarming system.
2. The camera footage of unrecognized humans is inserted to the database.
3. Description on spoofing attempts is inserted to the database.

4.5 Conclusion

The system is consisted of two parts. Hardware part and Software part. Each of them is implemented using different set of tools. The android application was implemented mainly

using Java for functionality and XML for design, while Python has been used extensively in the Raspberry Pi. It was found during this project that Python is a powerful language as many modules have been implemented by Python, ranging from sending outputs and reading input from the Raspberry Pi GPIOs to implementing complex Machine Learning algorithm for Liveness Detection. Moreover, the ability of Python to run on different environments is capable to classify it as a universal language.

The system was tested successfully under certain conditions. However, there must an update to the system to work under more severe weather conditions and lighting.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The rapid development in industry and wide scalability of technology have changed every single aspect in the way we live and accomplish our duties. This includes the practices taken to maintain security and have safety. This field has improved dramatically as a consequence of technology development and tools improvement. The Internet of Things (IoT) and Computer Vision are the primary tools which have made a difference and assisted the field of security to scale very rapidly. Unlike traditional methods taken to provide security, which include placing IP cameras at the corners of house courtyard and beside windows and doors and having them recording every activity at these spots 24/7. This method of security can provide a way of documenting incidents later on. However, it is not able to, at least, contribute to prevent them, and that is due the fact that they see but do not envision. In this report, IoT and Computer Vision have been integrated together to propose a fundamental way of enhancing security by contributing to increase safety. This is done through capturing images for visitors/people around the spotted area and identifying them. People who are found to be unrecognized are reported in real-time to an administrator, who's in turn can act reversely to prevent any unwanted/suspicious presence of some people around his/her property. Also, administrators' actions are executed in real-time through sending them across the internet to the system which in turn performs a deterring action by turning the alarming system on. This system is not limited to acting as a doorbell system, but rather it can be placed anywhere to report any presence of humans. However, being a doorbell system enables it to act autonomously when recognized people approach it by opening the door to them automatically.

This system can be used by a wide segment of people, specifically everybody who is interested to secure a property and always have real-time information on human presence.

Throughout this report, all the tools required were presented. Also, some topics were shortly introduced to provide an insight on their mechanism, and then a detailed approach was provided on how to implement the system. Finally, a simple demonstration was presented to test the behavior of the system.

5.2 Future Work

Since this system has been developed and implemented with a tight amount of time, there are still many aspects which require further improvement as follows:

1. The system should support main setup and control unit to be placed inside the property which provides control over other users of the system.
2. It should provide a plug-and-play (PnP) port which provides automatic configuration and identification of other systems (e.g., alarming system).
3. The system can be improved to support two-way audio communication and live streams over the internet.
4. Video recording would be an advantage for such system.
5. Also, footage performance and quality can be enhanced by using high-quality infrared camera.

APPENDIX A: IMPLEMENTATION DETAILS

All codes and pre-trained models utilized in this system are burned on the attached CD.

APPENDIXB: USER MANUAL

First, make sure to connect the Raspberry Pi to the same LAN as the mobile phone and local server. Then, start by opening the Raspbian terminal and installing all the necessary libraries for the project. Refer to Appendix C. After that, install Liveness Detection directory from the attached CD and start implementation.

Using the Android application is simple, since it is a GUI with buttons each one having the name of its function.

The home activity buttons:

- Recent Visits: shows the user recent visits to his/her house.
- Plus (+) floating button: takes the user to a new activity for adding new authorized visitors.
- Open door: calls the command for opening the main door.

The Add Guest Activity buttons:

- Select image: takes user to the library where an image of the person is to be selected from.
- Add image: Adds the image to the database.

Note: you cannot add an image unless you provide a name for it.

Recent Visits Activity:

In this activity pictures of people approached to your camera are displayed.

The capture now allows user to see what is going on around his/her front door.

APPENDIX C: DEPLOYMENT AND CONFIGURATION MANUAL

1. To install web server on the Raspberry Pi, you can follow the steps on the following link: <https://howtoraspberrypi.com/how-to-install-web-server-raspberry-pi-lamp/> .
2. First step is to install python libraries “OpenCV”. You can either follow the steps in the following link: <https://linuxize.com/post/how-to-install-opencv-on-ubuntu-20-04/> or type these two commands in the terminal (make sure to be in the LivenessDetection directory):
 - `sudo apt update`
 - `sudo apt install libopencv-dev python3-opencv`
3. To install numpy library, use one of the following commands (depending on python version):
 - `sudo apt install python-numpy` (python 2), or
 - `sudo apt install python3-numpy` (python 3).
4. To install the TensorFlow library, refer to the following link and follow the steps: <https://www.pyimagesearch.com/2019/12/09/how-to-install-tensorflow-2-0-on-ubuntu/>
5. To install dlib and face-recognition library. Follow the steps in this link: <https://ourcodeworld.com/articles/read/841/how-to-install-and-use-the-python-face-recognition-and-detection-library-in-ubuntu-16-04>
6. Type the following commands in terminal (LivenessDetection Directory) to install python “base64” library:
 - `sudo apt-get update -y`
 - `sudo apt-get install -y cl-base64`

REFERENCES

- [1] Google Support, “Learn About the Nest Hello video doorbell before you buy”, [Online]. Available: <https://support.google.com/googlenest/answer/9243617?hl=en#zippy=> , [Accessed 15 March 2021].
- [2] Arlo, “Arlo Essential Wireless Video Doorbell”, [online]. Available: <https://www.arlo.com/en-us/doorbell/video/arlo-essential-wireless-video-doorbell.html>, [Accessed 15 March 2021].
- [3] Wikipedia, “Home Security”, 17 March 2021. [Online]. Available: https://en.wikipedia.org/wiki/Home_security , [Accessed 18 March 2021]
- [4] Wikipedia, “Internet of Things”, 12 March 2021. [Online]. Available: https://en.wikipedia.org/wiki/Internet_of_things , [Accessed 18 March 2021]
- [5] CCTV Camera Pros, “AI Security Cameras”. [Online]. Available: <https://www.cctvcamerapros.com/AI-security-cameras-s/1512.htm>, [Accessed 18 March 2021]
- [6] Raspberry Pi, “Products”. [Online]. Available: <https://www.raspberrypi.org/products/>, [Accessed 1 May 2021].
- [7] Divine Okoi, “20 Best Operating Systems You Can Run On Raspberry Pi in 2021” 25 March 2021. [Online]. Available: <https://www.fossmint.com/operating-systems-for-raspberry-pi/>. [Accessed 19 May 2021].
- [8] [Online] Available: <https://www.electronicwings.com/raspberry-pi/raspberry-pi-introduction>. [Accessed 19 March 2021].
- [9] Lady Ada, “PIR Motion Sensor” 28 January 2014. [Online]. Available: <https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor> . [Accessed 19 May 2021].

- [10] “Python (Programming Language)” 19 May 2021. [Online]. Available: [https://en.wikipedia.org/wiki/Python_\(programming_language\)#Indentation](https://en.wikipedia.org/wiki/Python_(programming_language)#Indentation). [Accessed 21 May 2021].
- [11] “Advantages of Using Python For Computer Vision” 02 May 2019. [Online]. Available: <https://fullscale.io/blog/advantages-using-python-computer-vision/> [Accessed 21 May 2021].
- [12] OpenCV, “About OpenCV”. [Online]. Available: [https://opencv.org/about/#:~:text=OpenCV%20\(Open%20Source%20Computer%20Vision,p erception%20in%20the%20commercial%20products](https://opencv.org/about/#:~:text=OpenCV%20(Open%20Source%20Computer%20Vision,p erception%20in%20the%20commercial%20products) [Accessed 21 May 2021].
- [13] Serdar Yegulalp, “What is TensorFlow? The machine learning library explained ” 18 June 2019. [Online]. Available: <https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html> . [Accessed 22 May 2021].
- [14] Daphne Cornelisse, “An intuitive guide to Convolutional Neural Networks ” 24 April 2018. [Online]. Available: <https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/> . [Accessed 22 May 2021].
- [15] Jason Brownlee, “How Do Convolutional Layers Work in Deep Learning Neural Networks?” 17 April 2019. [Online]. Available: <https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/> . [Accessed 22 May 2021].
- [16] Stanislav Kovář, Jan Valouch, Hana Urbančoková, Milan Adámek, “Impact of Security Cameras on Electromagnetic Environment in Far and Near-field”. 5-7 July 2016. [Online]. Available: <https://ieeexplore.ieee.org/document/7557166/keywords#keywords> [Accessed 31 May 2021].
- [17] Aman Goel, “A Simple Introduction to Facial Recognition (with Python codes) ”. 30 August 2018. [Online]. Available: <https://www.analyticsvidhya.com/blog/2018/08/a-simple-introduction-to-facial-recognition-with-python-codes/> [Accessed 31 May 2021].

- [18] raspberrypi.org, “Camera Module V2”, [Online]. Available: <https://www.raspberrypi.org/products/camera-module-v2/>, [Accessed 1 June 2021].
- [19] opencv.org, “Face Detection using Haar Cascades”. [Online]. Available: https://docs.opencv.org/master/d2/d99/tutorial_js_face_detection.html, [Accessed 1 June 2021].
- [20] Adrian Rosebrock, “Liveness Detection with OpenCV”, 11 March 2019. [Online]. Available: <https://www.pyimagesearch.com/2019/03/11/liveness-detection-with-opencv/>, [Accessed 1 May 2021].
- [21] Adrian Rosebrock, “Eye Blink Detection with OpenCV, Python, and dlib”, 27 April 2017. [Online]. Available: <https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/>, [Accessed 28 May 2021].