

**EVALUATION OF SECURITY METRIC FOR
ARTIFICIAL INTERNET OF THINGS SMART
LOCKING SYSTEM**

SITI NUR HANANI BINTI MOHD RIZAL

UNIVERSITI TUNKU ABDUL RAHMAN

**EVALUATION OF SECURITY METRIC FOR ARTIFICIAL INTERNET OF
THINGS SMART LOCKING SYSTEM**

SITI NUR HANANI BINTI MOHD RIZAL

**A project report submitted in partial fulfilment of the requirements for the
award of**

Master of Engineering in Electronic System


Faculty of Engineering and Green Technology

Universiti Tunku Abdul Rahman

August 2023

DECLARATION


I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 
Name : Siti Nur Hanani Binti Mohd Rizal
ID No : 21AGM06711
Date : 18/8/2023

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**EVALUATION OF SECURITY METRIC FOR ARTIFICIAL INTERNET OF THINGS SMART LOCKING SYSTEM**” was prepared by Siti Nur Hanani Binti Mohd Rizal has met the required standard for submission in partial fulfilment of the requirements for the award of Master of Engineering in Electronic System at Universiti Tunku Abdul Rahman.

Approved by,

Signature :  _____
Supervisor : Dr. Lee Han Kee
Date : 18/8/2023

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2023, Siti Nur Hanani Binti Mohd Rizal. All right reserved.

EVALUATION OF SECURITY METRIC FOR ARTIFICIAL INTERNET OF THINGS SMART LOCKING SYSTEM

ABSTRACT

The increasing significance of delivering a smart lock system continues to grow. Modern technologies are being explored and utilised to accomplish this objective. Nonetheless, recent years have witnessed a noticeable surge in security breaches targeting the IoT systems, raising concerns for consumers. Drawing attention to this issue, the need to prioritise security should be considered during the development of the system, rather than treating it as an afterthought.

The primary goal of this dissertation is to evaluate the security metric for IoT based locking system and design an enhanced smart locking system that leverages the capabilities of Artificial Intelligence for facial recognition from the ground up. This system employs the Raspberry Pi controller to grant access to a private premises without relying on physical keys like access cards. The designed smart lock system incorporates smartphones, enabling the sending of emails to the owner upon detecting a potential intruder. The SMTP protocol library is evaluated to enable the email transactions features from the microcontroller. In terms of hardware implementation, a microcontroller is chosen, and a testing environment is established to explore and enhance security measures.

This thesis outlines the architectural model of the smart locking system, the design of the face recognition algorithm, and the development of a face recognition reader using the Raspberry Pi 3 model B microcontroller. Throughout this process, all developed sub-hardware components are designed to interact seamlessly between one another, ensuring a functional and secure flow within the AIoT framework.

TABLE OF CONTENTS

DECLARATION	iii
APPROVAL FOR SUBMISSION	iv
ABSTRACT	vi
TABLE OF CONTENTS	vii
LISTS OF TABLES	x
LISTS OF FIGURES	x
LIST OF ABBREVIATIONS	xii
CHAPTER	
1 INTRODUCTION	1
1.1 Background	1
1.2 Objective	2
1.3 Problem Statement	3
1.4 Scope and Limitations of The Project	3
1.5 Organization of Report	3
2 LITERATURE REVIEW	4
2.1 Related Work	4
2.2 Security Concerns in Existing Work	7
2.3 Hardware Review	10
2.3.1 Smart Mobile Device	10
2.3.2 Microcontroller	10
3 METHODOLOGY	12
3.1 Proposed Smart Locking System	13
3.1.1 Face Recognition using CNN	14
3.2 Software Specification	16
3.2.1 Raspbian OS	16

3.2.2	Python IDE	17
3.2.3	Visual Studio Code IDE	17
3.3	Hardware Specification	18
3.3.1	Webcam	19
3.3.2	Raspberry Pi 3 Model B	19
3.3.3	Relay	19
3.3.4	Electromagnet Lock	20
4	SOFTWARE DEVELOPMENT AND IMPLEMENTATION	21
4.1	Software Setup	21
4.1.1	Raspbian OS	21
4.1.2	Python IDE	22
4.1.3	Gmail – Python Account	22
4.2	Software Development	22
4.2.1	Datasets Preparation Process	24
4.2.2	Model training and Face Recognition Process	25
4.2.3	Email Sending Process	26
4.3	Microcontroller Code Implementation	27
5	HARDWARE DEVELOPMENT AND IMPLEMENTATION	28
5.1	Bipolar Junction Transistor (BJT) Amplifier	28
5.1.1	Saturation Mode	29
5.1.2	Cut-off Mode	30
5.2	Hardware Configuration	31
6	RESULT AND DISCUSSION	32
6.1	Datasets Collection	32
6.2	System Testing	32
6.3	Alerting System	34
6.4	Challenges	34
6.5	Low Power Consumption Effort	35
6.5.1	Ultrasonic Sensor	35
6.5.2	Heatsink	36
6.6	Locking Mechanism Substitute	38
6.7	Final Product	39

7	CONCLUSION	40
7.1	Future Work	40
7.2	Project Reflection	41
	REFERENCE	42
	APPENDICES	46
	APPENDIX A – Source Code – Main.py	46
	APPENDIX B – Source Code – face_detection.py	47
	APPENDIX C – Source Code – real_time_face_recognition.py	51
	APPENDIX D – Source Code – gpio_utils.py	55

LISTS OF TABLES

TABLE	TITLE	PAGE
Table 2.1	Advantages, Disadvantages and Security Concerns in Existing Research Works	7
Table 2.2	Comparison between Raspberry Pi 3 Model B and Arduino UNO On Different Selection Criterias	10
Table 5.1	BJT Amplifier Design Criteria	28
Table 6.1	CNN Model Confidence Level Under Different Brightness Condition	35
Table 6.2	Microcontroller Drawn Current and Temperature Based on Different Scenarios	38

LISTS OF FIGURES

FIGURE	TITLE	PAGE
Figure 3.1	Project Methodology	12
Figure 3.2	High-Level Architecture of the Proposed System	13
Figure 3.3	Facial Recognition Mechanism of the Proposed System	15
Figure 3.4	Typical Desktop View of Raspbian OS	16
Figure 3.5	Python Software Foundation Logo	17
Figure 3.6	Visual Studio Code IDE Logo	17
Figure 3.7	High-Level Hardware Diagram	18

Figure 3.8	Webcam Module	19
Figure 3.9	Raspberry Pi Model B Microcontroller	19
Figure 3.10	Relay Module	19
Figure 3.11	Electromagnetic Lock	20
Figure 4.1	Proposed System Software Flowchart	23
Figure 4.2	Transfer Learning Method for Datasets Preparation	24
Figure 4.3	Code Snippet for Frame Resizing and Face Detection	24
Figure 4.4	Code Snippet of Saving Image Captured	25
Figure 4.5	Face Recognition Development Method	25
Figure 4.6	Sending Email with Python Method	26
Figure 4.7	Code Snippet of Initializing Email Content	26
Figure 4.8	Code Snippet of Email Sending Process	26
Figure 4.9	Detailed Hardware-Software Communication Diagram	27
Figure 5.1	BJT Amplifier Circuit using NPN transistor.	29
Figure 5.2	Theoretical Graph Result of the Design Circuit	30
Figure 5.3	Hardware Pin Configuration of the Proposed Smart Lock System	31
Figure 6.1	Datasets Collection	32
Figure 6.2	Screenshot of Authorized Personnel with Mask	33
Figure 6.3	Screenshot of Authorized Personnel Without Mask	33
Figure 6.4	Email Sent to Owner	34
Figure 6.5	HC-SR04 Ultrasonic Sensor	35
Figure 6.6	Ultrasonic Sensor Pin Configuration	36
Figure 6.7	Heatsink Placement on Microcontroller	37
Figure 6.8	Microcontroller Power Consumption Based on Different Scenarios	37
Figure 6.9	Locking Mechanism Indicator Pin Configuration	38
Figure 6.10	Final Product Prototype	39

LIST OF ABBREVIATIONS

IoT	Internet of Things
AI	Artificial Intelligence
AIoT	Artificial Intelligence of Things
SMTP	Simple Mail Transfer Protocol
OS	Operating System
MitM	Man-in-the-Middle
PIN	Personal Identification Number
Wi-Fi	Wireless Fidelity
NFC	Near Field Communication
RFID	Radio Frequency Identification
OTP	One Time Password
IDE	Integrated Development Environment
GCP	Google Cloud Platform
CNN	Convolutional Neural Network
API	Application Programming Interface
GPIO	General-purpose Input/Output
DC	Direct Current
SD	Secure Digital
IP	Internet Protocol
SSL	Secure Sockets Layer
BJT	Bipolar Junction Transistor
LED	Light Emitting Diode

CHAPTER 1

INTRODUCTION

1.1 Background

The convergence of Artificial Intelligence (AI) and the Internet of Things (IoT) in smart lock system has significantly increasing in terms of the adoption, and while the development is exciting, it has raised security concerns among the users. Provided that, the security metrics of AIoT smart lock systems are vital to avoid the authorised access against the unauthorised access and protect user privacy. The security metrics can be evaluated based on several factors (Zhang *et al*, 2023):

1. Authentication and authorization: User authentication and access grant being done based on their authorization level.
2. Confidentiality: User's data and credentials should be encrypted to avoid hacking attacks.
3. Integrity: System must ensure data is not modified or lost during transmission or storage.
4. Availability: Responsive to user requests
5. Non-repudiation: System should provide evidence of user actions and transactions to prevent unauthorised or false failure.

As technology progresses, people are demanding for new technologies that can serve both purposes: conveniency and reliability. Technology like face recognition has a strong presence in the market and highly compatible with various smartphone operating systems (OS). As a result, a smartphone being used to do various tasks such as phone screen lock, validation of user locations and identity, as well as approving bank transactions.

For example, there are many bank applications in Malaysia such as HSBC, CIMBClicks or Maybank2U app turn to biometrics like facial recognition as a form of payment authentication, not just to better address the security issues, but also to make people life easier by shortening the authentication process (Normalini *et al*, 2015).

The idea of this research is to understand and implement the complex biometrics recognition to the existing door lock system that is available in the market. A camera will be implemented to do the facial recognition. If facial features detected is from registered personal based on the database, access will be granted, and the controller triggers the electromagnet lock to be released. Otherwise, door remain locked. It must also be noted that the facial recognition algorithm must be able to detect faces either with or without the face mask on and map back to the facial features from the database. To grant access, the controller request for authentication from the database for the facial features being scanned. A smartphone will be integrated to the lock system to allow owner to be notified when there are unauthorized faces at the door.

As can be conclude, digital locks offer enhanced security and convenience, especially with the growing reliability and accessibility of mobile technologies. This research study suggests the development of a digital locking mechanism using AI-driven facial recognition. The lock method chosen due to all facial features are unique to one's features, hence makes it complex for hackers. Other than that, the fact that smartphone is already inserted in human's daily life, adopting the smartphone for the super user which is the owner makes system more user-friendly.

1.2 Objective

1. To design a smart locking system based on IoT
2. To design a face detection system which can detect both face that is covered by mask and without the mask.
3. To generate an internet enabled locking system which can notify the owner when there are unfamiliar faces.

1.3 Problem Statement

The security of the IoT has become an increasing pressing concern. Although significant research has been dedicated to this subject, there is a noticeable absence of effort in terms of practical implementation or standardise solutions to tackle this issue (Zhang *et al.*, 2023). There are several issues that were identified in the existing smart lock system which include privacy problem, accuracy issues, vulnerability to hacking and weak authentication mechanism. Apart from that, manufacturing companies are treating security as an afterthought, as they focuses more on faster product release and profit. In turn, it has caused the systems to be more vulnerable to attacks. As a result, people stick to the normal mechanical-lock system using physical keys. Having said that, the design on an IoT based door lock system should offer better-secured access, more time-saving and accessible to everyone, aligned to the main purpose of technology advancement, which is to make human lives easier.

1.4 Scope and Limitations of The Project

The scope of this project is to develop an AIoT framework encompassing a locking system that is characterized by functionality, affordability, convenience, and above all, robust security. The boundary of the project is limited to Python programming language and the Raspbian Operating System (OS), along with the ability to utilize cloud storage for the database.

1.5 Organization of Report

The introduction to project which includes background, scope, objective and problem statement is presented in Chapter 1. On the other hand, Chapter 2 discuss and reviews of the existing research cited in the project and thesis. The software and hardware is also evaluated. Chapter 3 gives introduction to the product's system architecture, including descriptions of its features. This section also provides a list of hardware and programming software utilised in the project. Chapter 4 and Chapter 5 presents the software of the proposed framework and physical implementation of the proposed system presented in Chapter 3. Chapter 6 discuss about the results of the completed AIoT framework. Chapter 7 concludes the project and provide insights for future work and project reflection.

CHAPTER 2

LITERATURE REVIEW

2.1 Related Work

The advancement of technology has allowed the development of smart door lock system which replaces traditional keys with digital keys. This has been made possible with the adoption of the IoT framework to the lock mechanism. Smart door lock system has been widely adopted in the residency areas, business places as well as university's lecture rooms. The wide implementation is also because user data can be collected and monitored, which is also known as the monitoring system controlled by the administrative of the place. The smart door lock system must be reliable or "smart" enough to safeguard restricted premises by being able to differentiate authorized personals against the unauthorized ones.

Though it is no doubt that smart door lock system has made people's lives easier, this system can also be exposed to security vulnerability. Looking back at the normal physical keys, the security concern is that the lock can be circumvented, or the keys can be easily duplicated and use for mass usage. Hence, the smart door lock system is innovated to overcome these problems. However, the security vulnerability in the IoT can be hugely damaging. A breach in door lock system can cause security threats such as replay attacks; identity and data theft; Man-in-the-Middle (MitM) attacks and many more. Various research has been carried out to enhance the security and reliability of smart door lock system.

Password based door lock system has garnered people's attention ever since it is first introduced. G. Sowmya *et al.* (2018, p.223) studied the usage of matrix keypad to replace physical keys by having a predefined Personal Identification Number (PIN) code for user to gain access. The proposed system is programmed in such way that the password can be set or reset without having to use an external device. In research conducted by Akshay K. *et al.* (2018) the authors have developed a password- based door lock system with an anti-guess mechanism where the system is blocked after four failed attempts. The system will have to be reset in order to re-enable the system, Arduino microcontroller board is being used to integrate the system with an additional Wi-Fi module to store data to cloud for monitoring the attempts made.

Y. Kim *et al.* (2021) presented an AIoT-based smart lock system that integrates smart tags for access control. The lock is smart enough to detect smart tags that are attached to objects such as keys or cards. It uses AI algorithms to recognize them and grant access to users. Although the author claimed to offer advantages over traditional locks, the smart tags could still be lost or stolen, leading to unauthorized access. . On the other hand, research conducted by P. N. Narkar *et al.* (2017) uses the Near Field Communication (NFC)-enabled tag to act as a key to perform action. This technology needs Wi-Fi, Bluetooth and NFC enabled devices. NFC technology is based on the Radio Frequency Identification (RFID) protocols where it works on short range communication. NFC tags uses card emulation method where an NFC enabled devices can be used to act as keys to perform transactions.

The One Time Password (OTP) technology is more commonly known in the banking payment system. Seung-Soo Shin *et al.* (2013, p.436) have proposed this method for university lab and classroom usage management. This requires a server to store user information and generate the OTP seed which uses a cryptographic technique to generate the number. This OTP Code is then sent to user's phone and the server will compare with the user input. Aggarwal (2017) on the other hand, studied an OTP Based authentication method using Bluetooth transmission. The OTP Code is generated at random, and it is sent via Bluetooth to user's device with limited number of attempts allowed.

In the recent years, door lock system uses biometric technology where an individual's physical characteristics are being used as the key for access. Akanbi *et al.*

(2020) and Rao *et al.* (2019) proposed a door lock system using fingerprints technology for hotel room management and unlocking car door respectively. Both proposed scenarios used a fingerprint sensor module to capture fingerprints patterns to save and compare. In research conducted by S. Yedulapuram *et al.* (2020), they have proposed facial recognition method to identify authorised personals using image processing method in real time. In this proposed work, phone is integrated to notify owner when doors opened or when unrecognised face is captured from the door.

In a previous study conducted by Waseem *et al.* (2020), face recognition is used as a form of authentication for unlocking system. While the suggested system successfully meets the need to remove the reliance on physical keys, it falls short in terms of adaptability to identify individuals who are wearing mask during the authentication process. As a result, authorized user is not able to be matched correctly to the database causing no access. This demonstrates the system's inefficiency and a similar evident of this inefficiency is visible in research conducted by Khalimov *et al.* (2020, pp 244). Moreover, the design described in this research lack the ability to promptly notify the owner about granted or denied access to owner in real-time. This causes security risk in which, attackers or unauthorized persons can enter the restricted places without anyone noticing. This proves that the systems mentioned defeat the security purposes as anyone that can bypass the systems is allowed to enter without being flagged appropriately.

Furthermore, a token-based authentication has also been used in the smart lock system where it relies on the use of tokens instead of passwords to make request authorisation. The token is typically a unique string of characters that is generated by the server and stored on the client-side. Siddique *et al.* (2019) proposed a token-based home access control that implement mobile app usage to generate and manage the tokens. For this proposed system, the lock device in this system communicates with the mobile app over Bluetooth transmission. On the other hand, the research proposed Gunasinghe *et al.* (2018), the authors implemented the usage of central server in order to generate and manage the tokens. The system integrated the usage of IoT devices that communicate with the server via standard IoT communication protocols.

While there are various kind of technologies used for door lock system, some authors have also studied and proposed a combination of technologies to enhance the security features. A. Hemalatha and G. Gandhimathi (2019) proposed three-factor authentication by combining the RFID, password, and OTP technologies to unlock doors. In this proposed system, user needs to present their RFID cards as the first level, followed by password entry on keypad and OTP code send to the registered mobile devices to be input to the keypad on the door. Each level of authentication must be validated by the system before user can move to the next level. D. B. Mei Yin *et al.* (2016, pp 1-8) in their research work have proposed a door lock system which uses NFC and facial recognition technologies. This proposed system worked by capturing user NFC sticker as their first level of authentication which then followed by a face detection on the camera. In research conducted by Motwani *et al.* (2021, pp. 7973), they have proposed a door lock system that uses the password and facial recognition technologies for the door lock system. This system integrates keypad to input password as their first level and facial recognition method as their second level of authentication.

2.2 Security Concerns in Existing Work

Table 2.1 summaries the advantages and the disadvantages in the existing research done by other scholars, as well as the security concerns identified based on the authentication method used.

Table 2.1 Advantages, Disadvantages and Security Concerns in Existing Research Works

Authenticati on Method	Advantage	Disadvantage	Security Concerns
Passwords	Key-less solution	Anyone with password pin can enter	<ul style="list-style-type: none"> • Vulnerable to brute-force attacks, • Password reuse, • Dictionary attacks.
OTP Codes	<ul style="list-style-type: none"> • Key-less solution • Integrates mobile devices 	Limited time usage	<ul style="list-style-type: none"> • Replay attacks • Man-in-the-Middle (MitMs)

			<p>Attacks</p> <ul style="list-style-type: none"> • Weak PIN Selections
Smart tags	Easier to use than the physical keys	<ul style="list-style-type: none"> • Complex system • Tags/cards can be lost, stolen or replicated for mass usage 	<ul style="list-style-type: none"> • Identity theft • Data Theft
Biometric	<ul style="list-style-type: none"> • Key-less solution • Convenient 	<ul style="list-style-type: none"> • Big Storage • Costly • Need high resolution camera to get accurate result • Complex system 	<ul style="list-style-type: none"> • Privacy risks • Accuracy issues • Increased surveillance • False positive/negative
Token-based	<ul style="list-style-type: none"> • Improvement in user experience • Scalable system 	<ul style="list-style-type: none"> • Complex • Maintenance is costly and time consuming 	<ul style="list-style-type: none"> • Vulnerability to spoofing, • False positives/negatives. • Man-in-the-Middle (MitMs) Attacks
Multi-factor	<ul style="list-style-type: none"> • Multiple authentication method, • Adding layers of protection; making it harder for attackers to decipher layers 	<ul style="list-style-type: none"> • Complex, • Lengthy process, • Costly 	<ul style="list-style-type: none"> • Device vulnerabilities • Insider threats

Several papers have been reviewed and they are connected to the research purpose in terms of using the IoT framework in the proposed system. However, none of the works had established a system that can tackle most of the security concerns highlighted. The existing works in [9-13] have recognised the potential of biometric technology as the authentication method for smart lock system. With regards to facial recognition technology, several security vulnerabilities have been identified in the existing works which include privacy risk, accuracy issues, vulnerability to hacking, and the increased surveillance. Despite these concerns that facial recognition is a relatively new method, it has the potential to enhance security and convenience in not just smart lock context, but also others such as mobile devices and public spaces. To achieve the objective, this project explores the development of a more accurate and reliable facial recognition algorithms that are robust and more secured.

2.3 Hardware Review

The proposed system centres around three primary hardware modules: a mobile device., microcontroller board, and face recognition reader.

2.3.1 Smart Mobile Device

A smartphone device is necessary to test out the email sending features when developing the project. In general, any smartphone with stable Internet connection can be used for this project. The Apple iPhone 11 is being utilised.

2.3.2 Microcontroller

A microcontroller module is an embedded system that is equipped with various input and output pins, enabling its connection to components or modules required to perform the required task. It enhances the door locking and unlocking process by interpreting incoming signals to execute specific actions. Moreover, it's crucial to highlight that the microcontroller device must facilitate internet connectivity for seamless communication with cloud storage. In the low-cost microcontroller category, both Arduino and Raspberry Pi are being considered for this purpose. The comparison between these two microcontrollers are tabulated in Table 2.2.

Table 2.2 Comparison between Raspberry Pi 3 Model B and Arduino UNO On Different Selection Criterias

Criteria	Raspberry Pi 3 Model B (<i>Raspberry Pi Foundation, no date</i>)	Arduino UNO (<i>Arduino, 2019</i>)
Processor	Quad-core ARM Cortex A53	ATmega328P 8-bit
Memory	1GB RAM	2KB SRAM, 32KB Flash
Internet Connectivity	Wi-Fi, Ethernet	None
Power Consumption	~2.5 W	~0.5 W
Working Temperature	0-50 °C	-40-85 °C
Camera Module	Compatible with webcam or Pi camera	Not natively supported
Programming Language	Python, C/C++ and more	C/C++ (Arduino IDE)
Cost	Around RM150	Around RM35

In conclusion, the selection of the suitable microcontroller holds immense significance for the project. The Raspberry Pi emerges as a remarkable selection for this endeavour although the price is slightly on a higher range, hence it will serve as the primary controller for the smart locking system. This decision is also influenced by past familiarity and expertise in microcontroller operations and programming language offered. For this project, the Python programming language has been chosen for the program development.

CHAPTER 3

METHODOLOGY

This chapter delves into the methodologies and the foundation to carry out the project execution. The primary stages for this project comprise research on existing literature, hardware and software specifications selection, design development and implementation, as well as experimental results as shown is Figure 3.1.

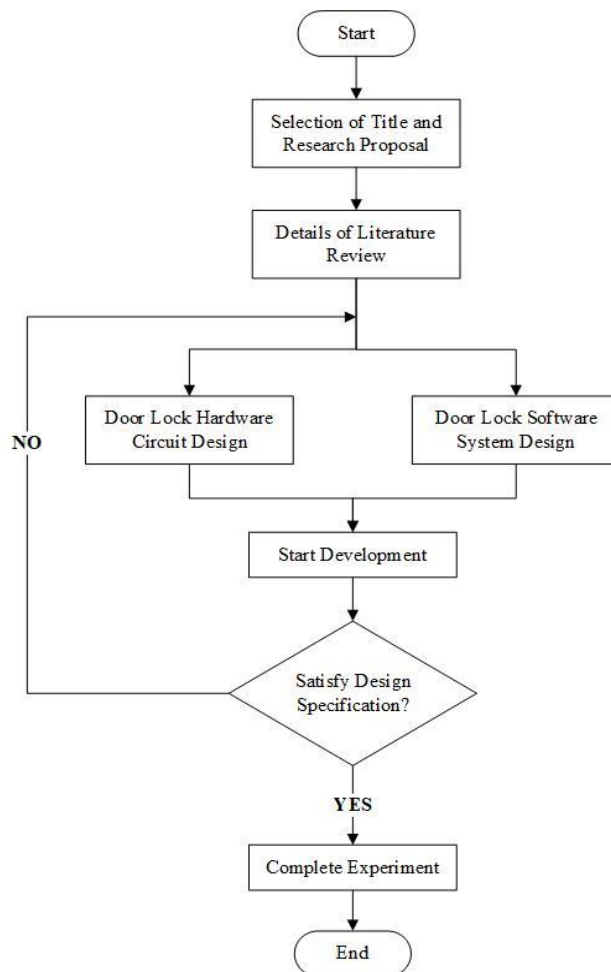


Figure 3.1 Project Methodology

In Chapter 2, the research reviews has been undertaken prior to designing the project specification and executing the project. An adequate level of literature on the primary topic and concerns of these works has been collected. Initially, the AIoT within the smart locking system was conducted, accompanied by a discussion on various locking mechanisms. This evaluation leads to the evaluation of facial biometrics, which has been selected to symbolize the digital or biometric key for this project. Given the extensive communication between devices exposing the system to possibility of malicious attacks, it is necessary to focus on the face recognition algorithm. Thus, the concerns on the existing work is tabulated to provide as a pre-requisite that must be satisfied to develop an AIoT system.

The design specification incorporate insights from relevant research papers as well as the highlighted security concerns. This includes the hardware finalization such as choosing a well-suited microcontroller and face recognition reader that can adapt at real-time video recording. Additionally, a cloud storage system selection is included to enhance the security for managing the smart locking system.

3.1 Proposed Smart Locking System

A secured system architecture of the proposed project is as shown in Figure 3.2. There are two main components in the software implementation: Microcontroller and Firebase storage. In this proposed system, microcontroller act as the image collector, and database storage using the Firebase storage. Firebase storage are one of many services offered by the Google Cloud Platform (GCP) (*Google*, no date).

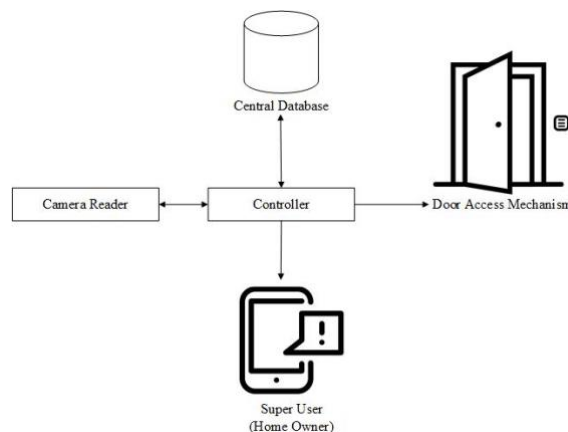


Figure 3.2 High-Level Architecture of the Proposed System

The software architecture of the proposed system ensures a secure structure for user authentication and validation, ensuring that only authorized users gain access after receiving permission from the controller. While the project emphasis rests on security, it's crucial to consider the convenience and reliability for the users. Thus, in conjunction with the security as the focus, the proposed design aims to mitigate finicky design and difficult authorization process. This is to ensure that the potential of users to deter the system can be avoided.

The Firebase Storage, will act as a database to store all the image collections and dataset received. The cloud functions, which is a user defined function server have the access to all the main entities of the system. The microcontroller unit is integrated to the IoT system and will have two main functionalities – to receive the datasets from the database and perform authorization which resulting to door being unlock in case if user is authorised, and vice versa. Other than that, the microcontroller will also interact with other sub -hardware components incorporated to the input and output ports that forms the locking system. The system will adhere to a secure communication protocols for acquiring the datasets from the database, and it will also dispatch essential messages to the owner via email in the event of intruders being detected.

In terms of security, the microcontroller does not have direct access or communication with the database. This is to ensure that any possible attacks can be prevented or at least, delayed.

3.1.1 Face Recognition using Convolutional Neural Networks (CNNs)

The face recognition mechanism in this proposed system is as shown in Figure 3.3. The project aims to create a robust framework to allow scalability and re-training of machine learning model. Thus, the algorithm will be divided into three main parts – dataset generation, training model, facial recognition.

The dataset generation involves the image collection process where a set of images containing faces of an individual person is pre-collected. These images are later pre-process to enhance feature extraction experience in the later stage. The images are later saved in a folder labelled by individual's unique identifier. To attain the objective of this stage, the proposed system will incorporate the OpenCV library, enabling the microcontroller to process images taken by the face recognition reader.

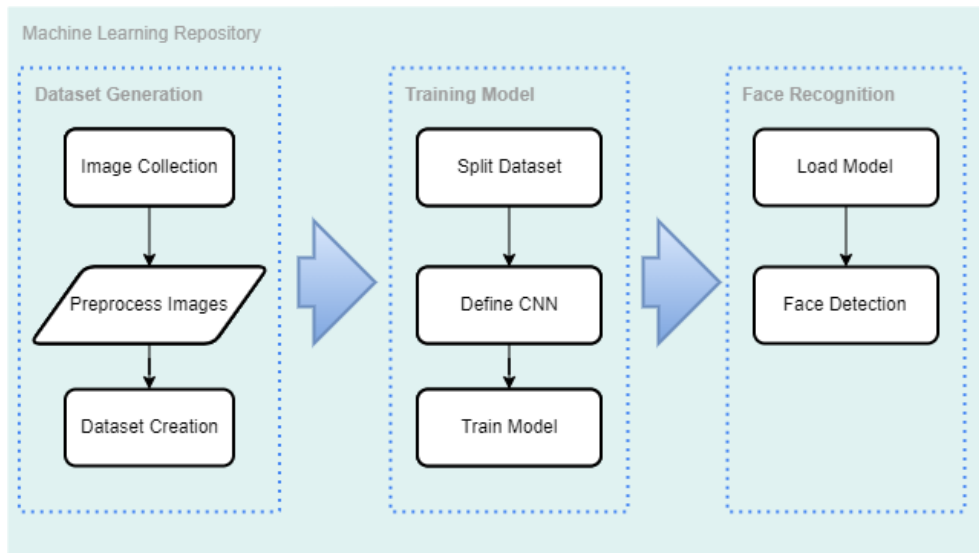


Figure 3.3 Facial Recognition Mechanism of the Proposed System

At the training model stage, the dataset is loaded into the algorithm in order to perform the face training. The project aims to adopt the Convolutional Neural Networks (CNNs) model which is a deep learning model used for image classification tasks (F. Li *et al.* 2015). To achieve this, additional python library which is TensorFlow Keras package will be implemented. The TensorFlow is an open-source machine learning framework that provides various Application Programming Interfaces (APIs). Keras which is one of its API will be used in implementing the model training algorithm as it helps to simplify the process of building, training, and deploying the CNN model (Nath, J. 2021). The face recognition stage involves the process of loading the trained model and perform a simple algorithm to send signal to other hardware component on the locking mechanism.

3.2 Software Specification

3.2.1 Raspbian OS

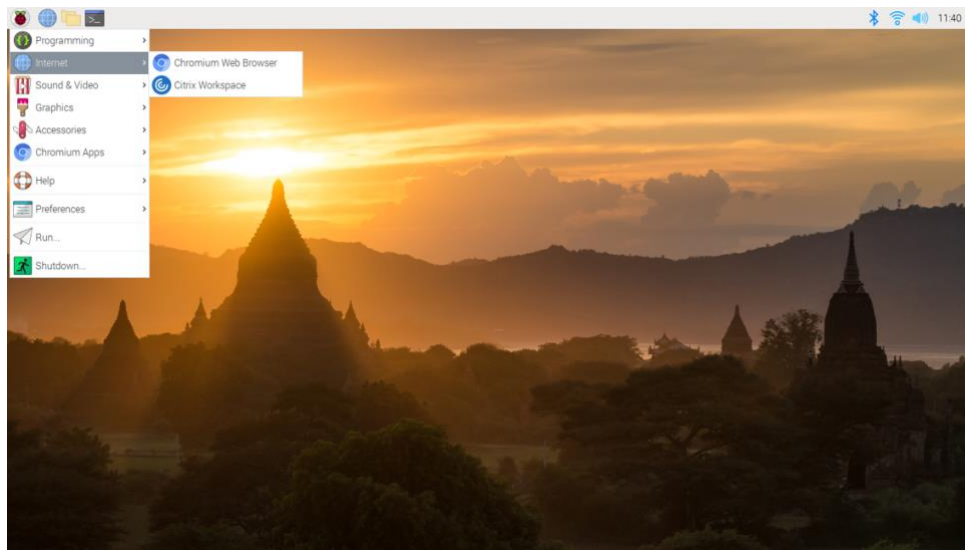


Figure 3.4 Typical Desktop View of Raspbian OS

Figure 3.4 shows a typical desktop view of the Raspbian OS. Raspbian is the free and foundation's official supported operating system based on Debian optimized for the raspberry pi hardware. Raspbian provide more than pure OS if compare to the other operating system. It comes with over 35000 packages, pre- compiled software bundled in a nice format for easy installation on Raspberry Pi. Software like Python IDE, Scratch and more are included in this OS (*Raspberry Pi Foundation*, no date).

3.2.2 Python IDE

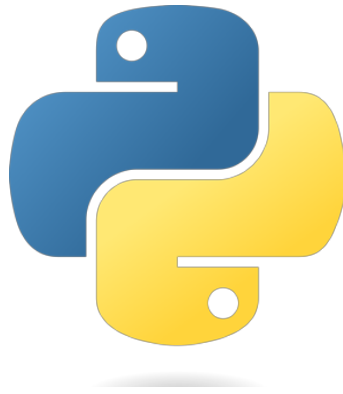


Figure 3.5 Python Software Foundation Logo

Python IDE is a free and open source programming software and also is an IDE stand for integrated development environment for Python. In Raspbian OS, python IDE is a built-in software and installed with python2 and pyhton3 (*Raspberry Pi Foundation*, no date). Apart from that, Python supports a virtual environment which is a tool that can isolate dependencies required for specific project usage (*Python*, no date). This is to make sure no redundant libraries being run with the code that might affect the code functionalities. In this project python IDE will be used to code most of the program including the face recognition and face detection with the use of OpenCV and TensorFlow library.

3.2.3 Visual Studio Code IDE



Figure 3.6 Visual Studio Code IDE Logo

Visual Studio Code, created by Microsoft, serve as a versatile coding tool for scripting, debugging and refactoring. This helps the developers to develop code efficiently and in a timely-manner. It also have multi-language support which includes but not limited to Python, C#, C++, HTML and JavaScript (*Visualstudio.com*, 2016). In this project Visual Studio Code will aid the software development process ensuring a successful outcome.

3.3 Hardware Specification

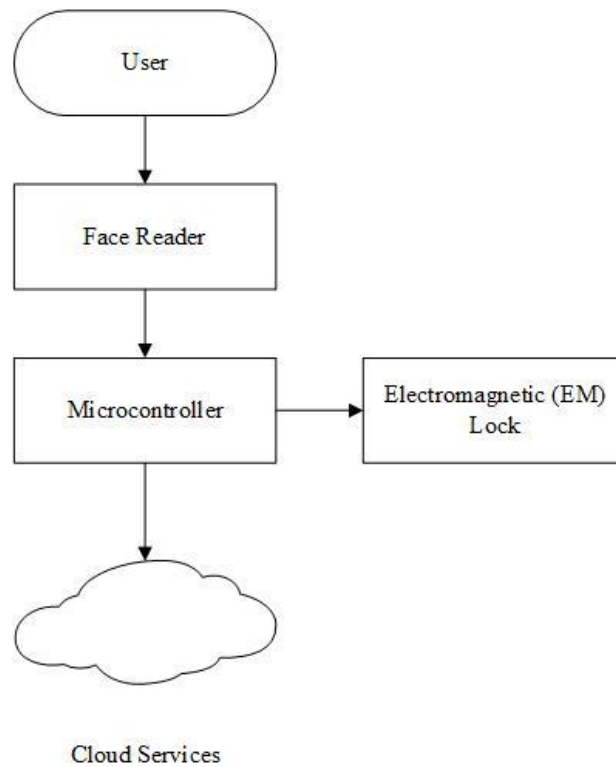


Figure 3.7 High-Level Hardware Diagram

Figure 3.7 illustrates the hardware flowchart for the proposed design. The smart locking system encompassed an electromagnet lock affixed to a door, and a camera for gathering people images. These components are interconnected through a Raspberry Pi 3 model B microcontroller. In project scope, a user can enter locked premises by showing their face on the camera. The face recognition reader detects and captures image collections and sends them to the microcontroller. The microcontroller subsequently undertakes the face verification to determine the authentication status.

3.3.1 Webcam



Figure 3.8 Webcam Module

The webcam functions as a video camera, that captures real-time video for facial recognition task. It scans and gathers data which is the image whenever an individual approaches the camera. The acquired data is subsequently transmitted to the controller for subsequent task execution.

3.3.2 Raspberry Pi 3 Model B



Figure 3.9 Raspberry Pi Model B Microcontroller

The Raspberry Pi microcontroller act as a main “runner” that retrieve datasets from the database, make decisions, and transmitting the executing command to the relay that governs an electromagnet lock.

3.3.3 Relay



Figure 3.10 Relay Module

A relay functions as a mechanical switch that facilitate the transition from a low input signal to a significantly higher output signal. For the proposed system, direct connection between electromagnet lock and microcontroller could lead to damage on the controller. Hence a relay will be used to bridge the GPIO voltage output to the electromagnet Lock.

3.3.4 Electromagnet Lock



Figure 3.11 Electromagnetic Lock

An electromagnet lock operates on DC12V and comprises a magnet, steel plate, and accompanying mounting hardware. In this project, the electromagnet lock waits a command from the microcontroller. Upon receiving a “success” signal, the power is disengaged, demagnetizing the electromagnet lock and vice versa.

CHAPTER 4

SOFTWARE DEVELOPMENT AND IMPLEMENTATION

4.1 Software Setup

4.1.1 Raspbian OS

Installing the Raspbian Operating System is required in order to use the Raspberry Pi microcontroller. The Raspberry Pi has its own official OS imager tool where developer can choose the OS type and version that is suited for the project. Upon choosing, the imager can also perform write to the microSD. At the end, the microSD can be slotted to the microcontroller and boot the system.

The initial set up includes setting up user name and password, followed by connecting the Raspberry Pi to the Internet wirelessly. With the Internet access, the operating system is updated to the latest version. In order to avoid the changing of Raspberry Pi's IP address while rebooting Router or Raspberry Pi, it's a necessity that we use a fixed IP address. The IP address has been set to a static address within the network range. With this being fixed, the IP address can be used for the VNC viewer from personal computer. Apart from that, as the project is using webcam instead of the Pi-camera itself, there are additional library needed to be installed, which is *fswebcam* (*Debian Webmaster*, 2012). This is required in order to properly operate the webcam. The required library will be installed and stored in the virtual environment created for the project.

4.1.2 Python IDE

Although Raspbian OS comes with a pre-installed python2 and python3, it must be noted that the version can satisfy the project requirement such as the required libraries and dependencies. For this project, the main additional libraries required are TensorFlow and OpenCV. For ease of usage, the required dependencies will be installed on a virtual environment configured solely for this project.

4.1.3 Gmail – Python Account

In order to enable email sending feature in the system, a Gmail account is needed. A two-step verification must first be activated in order to generate a Python password under application password for project usage. Once password is created, it can then be copied and use in the project.

4.2 Software Development

A complete software code flowchart is as shown in Figure 4.1. The sensor distance threshold is set at 20cm. This means that if an object detected is within 0 to 20 cm proximity from the sensor, the microcontroller will trigger the webcam to start collecting the data, or images for facial recognition. The webcam will remain turn on until the person's face is evaluated or if the time is up. The time is adopted for cases to ensure that the camera does not turn on forever that could cause a surge in current drawn by the microcontroller. For this project, the camera timer is set to 20 seconds.

Within the timer range, The face recognition algorithm is being run to analyse any face detected. The following sub-topics will explain in details on the face recognition process.

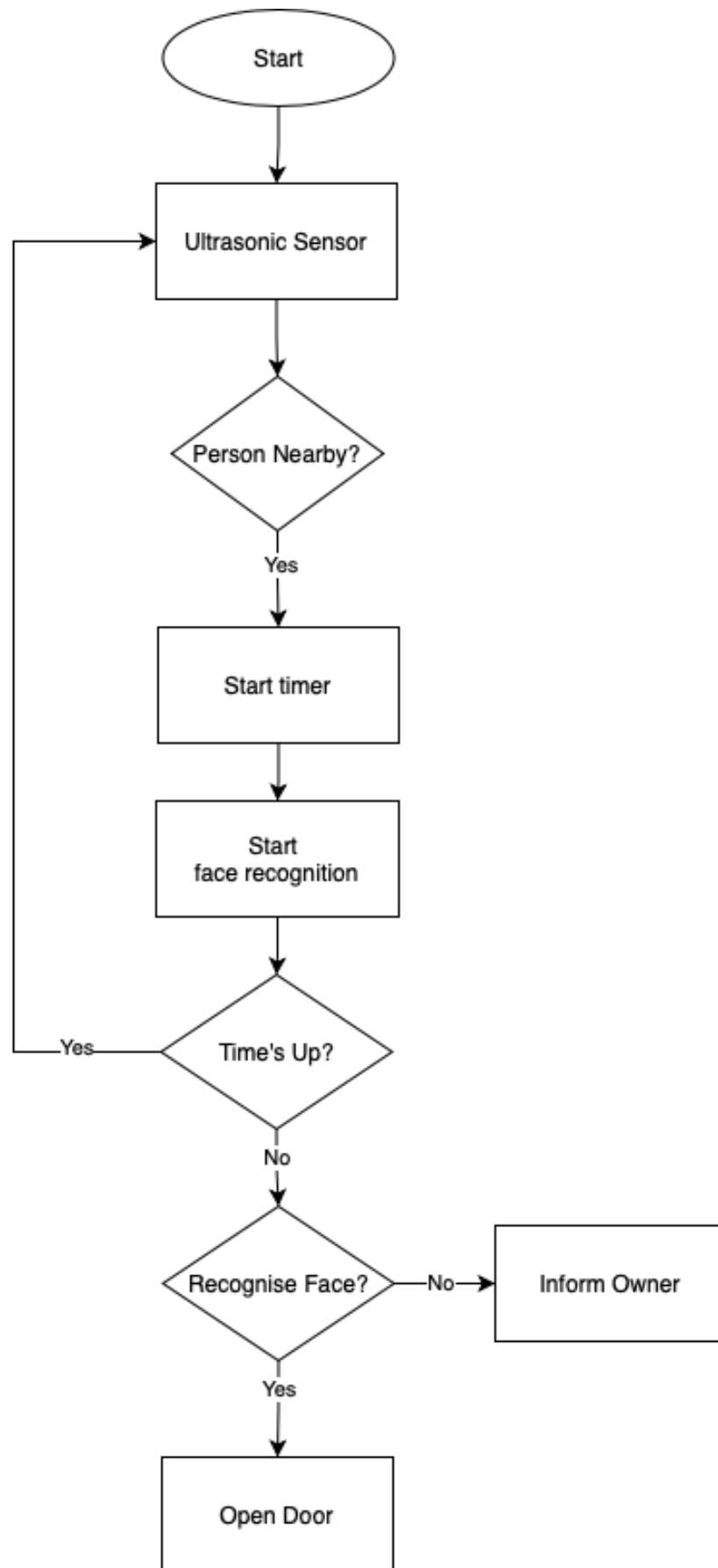


Figure 4.1 Proposed System Software Flowchart

4.2.1 Datasets Preparation Process

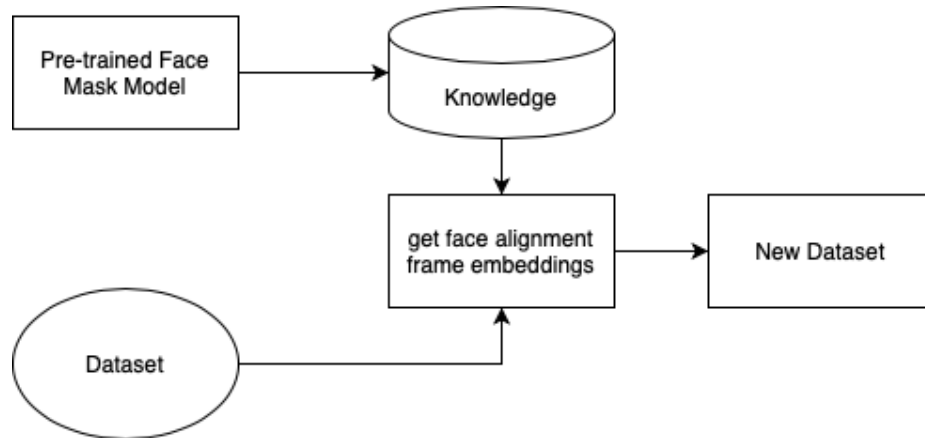


Figure 4.2 Transfer Learning Method for Datasets Preparation

There are many face recognition standards and methods being used to develop a face recognition system. Figure 4.2 shows the flow diagram of the datasets preparation for this project, which is using the transfer learning method for face detection. A pre-trained CNN model, which is a model pre-trained for face detection is loaded to the system. From this model, the frame size required to properly detect faces is obtained by using the TensorFlow library. Once frame sized is obtained, the face detected on the image is performed and bounding boxes are placed on the dataset. The dataset is then saved into a new form of datasets which only have the faces detected.

```
#---image processing
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_rgb = img_rgb.astype(np.float32)
img_rgb /= 255

#---frame resize and face detection
img_fd = cv2.resize(img_rgb, fmd.img_size)
img_fd = np.expand_dims(img_fd, axis=0)

bboxes, re_confidence, re_classes, re_mask_id = fmd.inference(img_fd, height, width)
if len(bboxes) > 0:
    for num, bbox in enumerate(bboxes):
        cv2.rectangle(img, (bbox[0], bbox[1]), (bbox[0] + bbox[2], bbox[1] + bbox[3]), (0, 255, 0), 2)
```

Figure 4.3 Code Snippet for Frame Resizing and Face Detection

Figure 4.3 shows the code snippets of creating the new datasets. The dataset frame is resize using the following the pre-trained image size requirement (fmd.image). the face detection is performed with the new resize frame by passing the image to inference model. The bounding boxes (bboxes) then return the rectangles form of the face detected. Using the cv2 library the image is saved (cv2.imwrite) to the specified path and naming (save_path), as shown in Figure 4.4.

```

#----keys handle
k = cv2.waitKey(1)
if k%256 == 27: #press ESC
    break
elif k%256 == 32: #press space
    if len(bboxes) > 0:
        img_temp = img[bbox[1]:bbox[1] + bbox[3], bbox[0]:bbox[0] + bbox[2], :]
        save_path = f"{user}_{img_count}.jpg"
        save_path = os.path.join(ref_dir,save_path)
        cv2.imwrite(save_path,img_temp)
        img_count += 1
        print("An image is saved to ",save_path)

```

Figure 4.4 Code Snippet of Saving Image Captured

4.2.2 Model training and Face Recognition Process

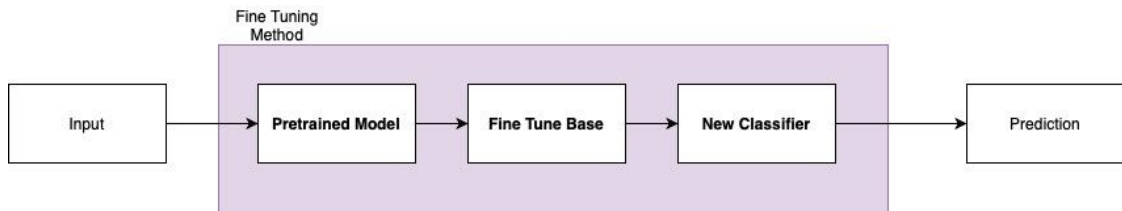


Figure 4.5 Face Recognition Development Method

The model training includes using a pre-trained face recognition model to reduce the heavy-duty of training simple features and the base weight of this model is used to train new model. Figure 4.5 illustrates the training method being used for this project. The pre-trained model is in Protobuf format (.pb) which contains the graph definition as well as the weights of the model. When webcam module is running and image is capture, the trained model is loaded and get the prediction value using Tensorflow's tf.session.run() function. The lower the prediction value, the higher the rate of accuracy. This means lower error rate on recognition. For this project, the prediction value threshold is set to 0.8.

4.2.3 Email Sending Process



Figure 4.6 Sending Email with Python Method

Function `send_message_to_owner()` is created and the process is as show in Figure 4.6. Python IDE have built-in email sending library. The required packages include OS, SSL, email and SMTP which comes pre-built with the installed python IDE. The setup process includes initializing elements which is the sender email and password, receiver email, and email content. The elements is then mapped and defined for the `EmailMessage()` class. In order to add extra layer of security, the SSL library is implemented to define the SSL certificate. Secure Sockets Layer (SSL) is a standard internet protocols that helps to ensure that internet connection is kept secured and safeguarding any sensitive data that is being sent between two system. In order to send the email, the SMTP library is imported to login and define the email server. Then the email can be sent through this SMTP server when an intruder is detected as shown in the system design flowchart.

```
#-----Initialize info with EmailMessage() -----
em = EmailMessage()
em["From"] = email_sender
em["To"] = email_reciever
em["Subject"] = subject
em.set_content(body)
em.add_attachment(open("image.jpg", "rb").read(), maintype='application', subtype='png', filename='image.png')
```

Figure 4.7 Code Snippet of Initializing Email Content

Figure 4.7 shows the code snippet of the setting content to send the email. The `set_content` and `add_attachment` helps define the body content of the email and the captured image respectively. Apart from that, `ssl.create_default_context` helps to set the SSL certificate as shown in Figure 4.8. Once everything is convert, login is done using `smtp.login` and email is send using `smtp.send_mail()`.

```
#-----set SSL Certificate-----
context = ssl.create_default_context()
with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as smtp:
    smtp.login(email_sender, email_password)
#----- send email -----
smtp.sendmail(email_sender, email_reciever, em.as_string())
```

Figure 4.8 Code Snippet of Email Sending Process

4.3 Microcontroller Code Implementation

Figure 4.9 represents the detailed sequence of the communication between the microcontroller, database, SMTP mail provider and other hardware components.

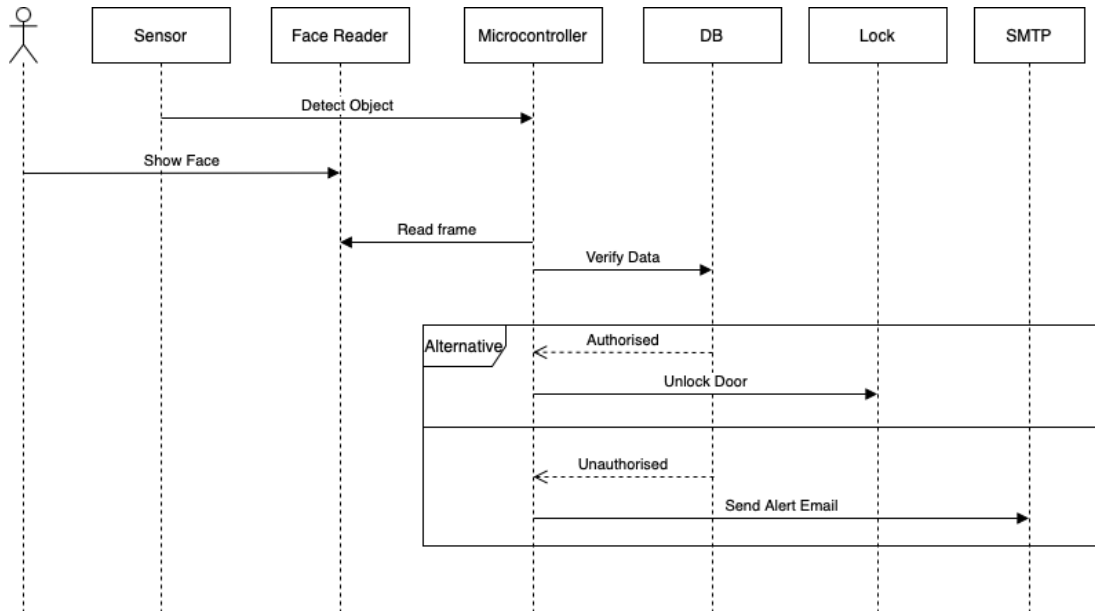


Figure 4.9 Detailed Hardware-Software Communication Diagram

CHAPTER 5

HARDWARE DEVELOPMENT AND IMPLEMENTATION

5.1 Bipolar Junction Transistor (BJT) Amplifier

An external controlling circuit is necessary to transmit the signal from microcontroller GPIO pin to the relay's IN pin. According to the datasheet, the relay operates on an active-low logic, where it interprets a logic low signal as $0V$, and a logic high as $5V$. The relay will then deliver the received logic values from the BJT amplifier circuit to the load, which is the electromagnet lock.

Referring to the microcontroller datasheet (*Raspberry Pi Foundation*, no date), the GPIO pins ranges from $1.6-3.3V$ for a logic high output, and less than $1V$ for a logic low mode. However, the pull-in (trigger) voltage is above $3.75V$. Aligning the GPIO pin's output voltage range with the trigger voltage of the relay, the concept of an inverter logic is used. In this project, an NPN transistor is reviewed for this purpose. The design specification for the BJT amplifier is as tabulated in Table 5.1 below.

Table 5.1 BJT Amplifier Design Criteria

Transistor Operation	Input Voltage Range (V)	Output Voltage Range (V)	Transistor State
Cut-off Mode	0 – 1	3 – 5	Off (Open)
Saturated Mode	1.6 – 3.3	0 – 1	On (Close)

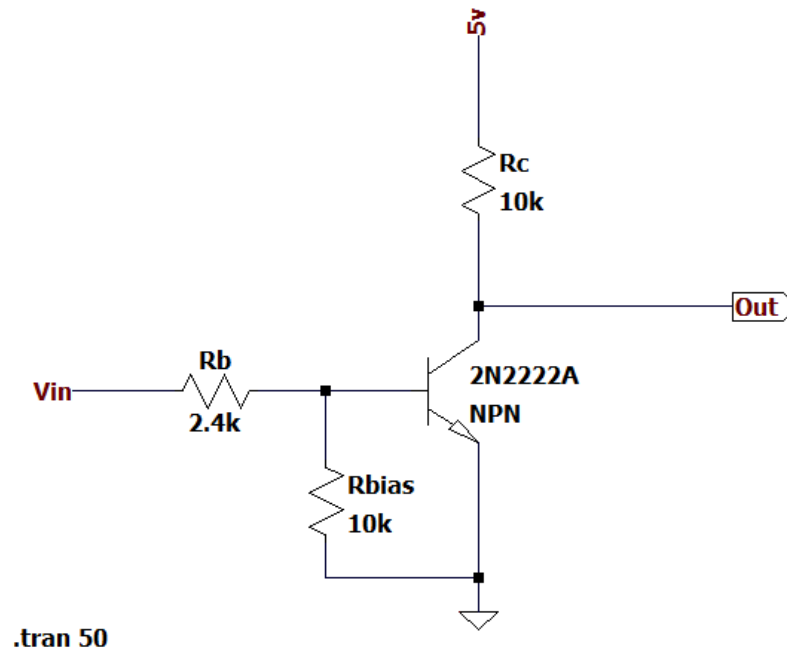


Figure 5.1 BJT Amplifier Circuit using NPN transistor.

5.1.1 Saturation Mode

Based on the NPN 2N2222A transistor datasheet (*First Components International*, no date), the maximum voltage value for collector-emitter voltage V_{CE} and the based-emitter voltage V_{BE} are $0.3V$ and $1.2V$ respectively. By applying the Kirchoff's voltage law, 5.1 is deduced. To maintain control over the transistor, the collector current must be sufficient while keeping power consumption low. It is also crucial to note on the floating state, where the voltage acquired is unstable and causes the relay to continuously switch between $5V$ and $0V$. This floating stage should be minimised or eliminated.

Assuming a current of $0.5mA$ at R_c , the value calculated for R_c is $9.4k\Omega$. considering the standard resistor values, $10k\Omega$ is selected for R_c . This resistor functions as a pull-up resistor, helping to guide the input to a high state when the voltage is floating.

$$5V - V_{CE} = I_C R_C \quad 5.1$$

In order to counter the floating state and set it to a low state, a pull-down resistor R_{bias} is utilised. Consequently, $10k\Omega$ for R_{bias} is placed. With NPN transistor's minimum beta value H_{Fe} being 100, the base current of $5\mu A$ required to saturate the NPN transistor is verified using equation 5.2.

$$I_C = H_{Fe} I_B \quad 5.2$$

5.1.2 Cut-off Mode

The transistor cut-off mode denotes current not flowing thus, zero at base current. Having this in mind and V_{BE} as $1.2V$ in saturated mode, the cut-off mode rules are defined by the following bullets:

- Current draws at R_b and R_{bias} is equivalent
- Voltage should be less than $1.2V$ across R_{bias}

Equation 5.3 is used to define the resistors value. Adhering to the design criteria, the calculated value for R_b is $2.5k\Omega$ when the input voltage is $1.5V$.

$$\frac{V_i - V_{BE}}{R_b} = \frac{V_{BE}}{R_{bias}} \quad 5.3$$

Prior to transferring the theoretical amplifier circuit design to a soldering board for practical use, a simulation is performed using Ltspice. This simulation is crucial to validate the design's performance before the actual implementation to ensure that the design criteria align with the desired outcome. This helps in minimizing the risk of damage to both of the hardware component, Raspberry Pi microcontroller and the relay module.

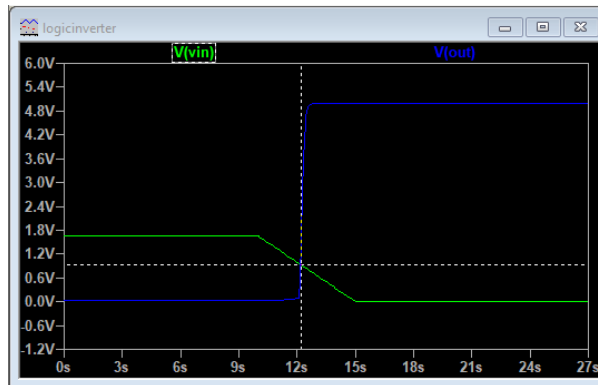


Figure 5.2 Theoretical Graph Result of the Design Circuit

The design of the amplifier circuit is simulated using Ltspice to observe its output behaviour. As can be seen in Figure 5.2, when the input is at 1.6V, the NPN transistor becomes fully saturated, pulling the 5V to ground. This experimental result demonstrates the successfully attained the logic high state for the design circuit. Conversely, the supply voltage should be below 1V in order to yield 5V of output voltage for the logic low state. As shown in the graph result using Ltspice simulator, when the input voltage is approximately 1V, the output voltage registers at 5V. This observation signifies the NPN transistor being powered off, which is entering the cut-off mode. As a result, the design circuit has achieved the intended objective in theory.

5.2 Hardware Configuration

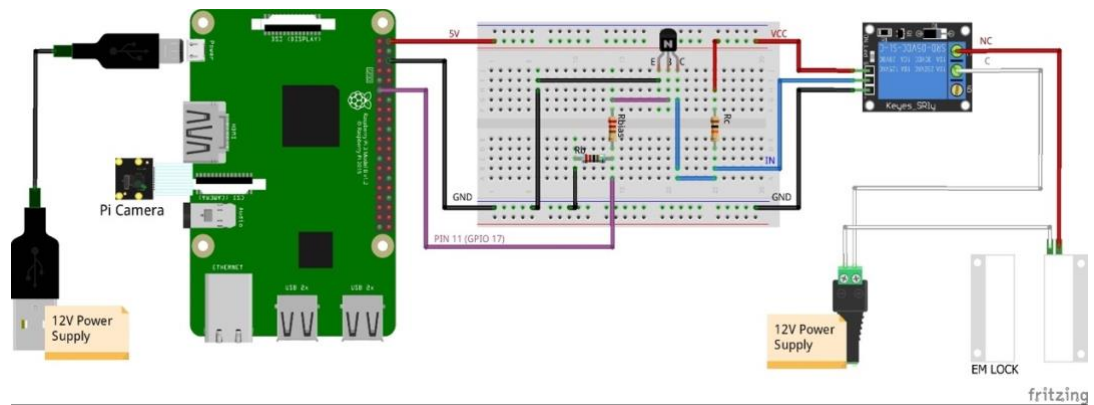


Figure 5.3 Hardware Pin Configuration of the Proposed Smart Lock System

CHAPTER 6

RESULT AND DISCUSSION

6.1 Datasets Collection

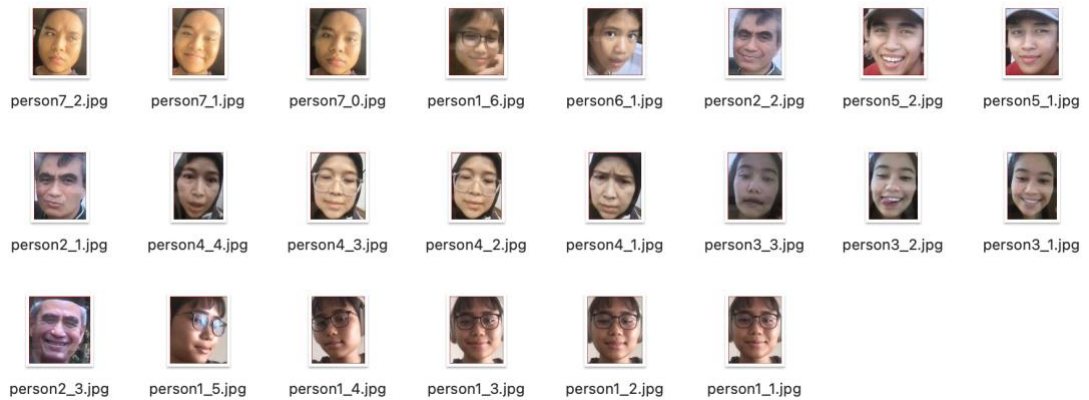


Figure 6.1 Datasets Collection

6.2 System Testing

The performance of the facial recognition execution is recorded using *save_video_recording()* function created in the microcontroller to save the capturing process to the computer screen. This leverages the use of cv2 python library as it has already been integrated into the project. The system able to detect the person face with and without mask within 2 seconds timeframe as shown in Figure 6.2 and Figure 6.3 respectively.

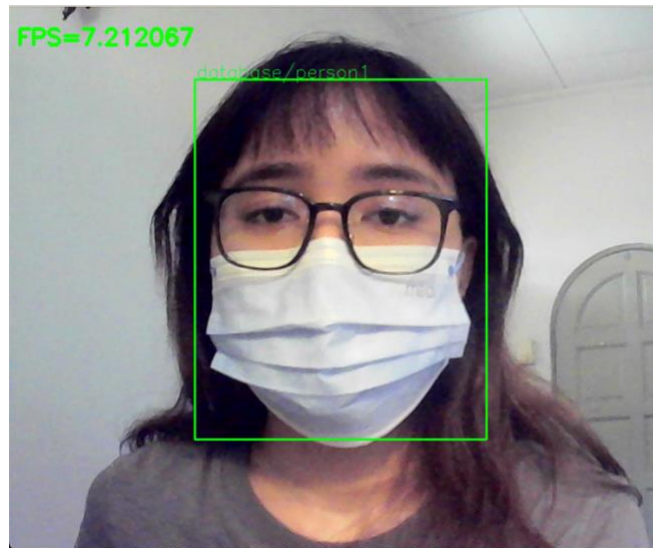


Figure 6.2 Screenshot of Authorized Personnel with Mask

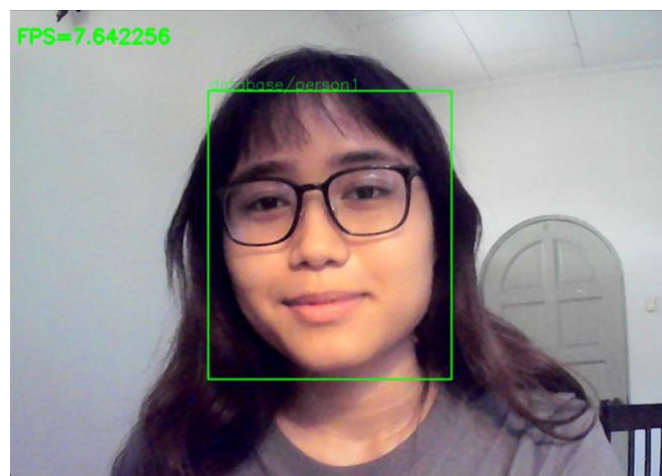


Figure 6.3 Screenshot of Authorized Personnel Without Mask

6.3 Alerting System

The alerting system is tested to send email to owner when there's unfamiliar face being detected. As shown in Figure 6.4, image is captured and email is sent to the owner's registered email address.

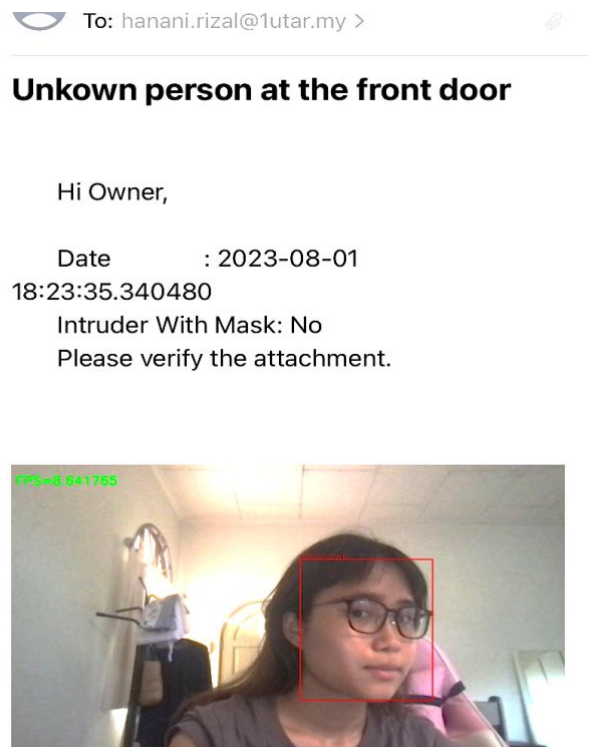


Figure 6.4 Email Sent to Owner

6.4 Challenges

The face recognition algorithm is tested on a personal computer with different camera module before transferring to the Raspbian OS SD Card. Having said this, the built-in laptop camera have different brightness compared to the webcam module attach to the microcontroller under the same light condition. Due to this, the confidence level are re-adjusted to accommodate to new default lighting condition. Table 6.1 summarised the working confidence level for the microcontroller. As can be since, the confidence level need to be increased in order to correctly detects the person.

Table 6.1 CNN Model Confidence Level Under Different Brightness Condition

	Confidence Level	Brightness
Webcam Module	0.8	High
Built-in laptop Module	0.5	Medium

Apart from that, the system also faces issue on dealing with small diversity of faces and datasets to train the model. This trade-off the accuracy level of the algorithm to implement the tasks. That being said, a public contributed pre-trained model had been used to further enhances the accuracy, robustness and efficiency of the algorithm (Schroff *et al.*, 2015)

6.5 Low Power Consumption Effort

The Raspberry Pi 3 Model B controller is not built-in with a sleep or hibernate mode, hence the controller always remain active when powered on (*Raspberry Pi Foundation*, no date). Moreover, the camera module also remained activated throughout the process causing the controller to draws a lot of current at approximately 600mA and heat up fast. In order to fix this issue, the system had accommodate additional installation to minimise the power consumption. This includes the ultrasonic sensor and the heatsink.

6.5.1 Ultrasonic Sensor



Figure 6.5 HC-SR04 Ultrasonic Sensor

The HC-SR04 Ultrasonic sensor outputs 5V (*Spark Fun Electronic*, no date) thus the voltage divider rule formula will be used to shift 5V to 3.3V. This is because, the raspberry pi GPIO pins only accept 3.3V. By having this level shifter, it can prevent the sensor from frying the microcontroller. According to the 6.1 where input voltage V_{in} and output voltage V_{out} are 5V and 3.3V respectively, the resistor R1 will be 1k Ω and resistor R2 will be approximately 2k Ω .

$$V_{out} = V_{in} \frac{R_2}{R_1 + R_2} \quad 6.1$$

The pin configuration for sensor connection and the microcontroller is as shown in Figure 6.6.

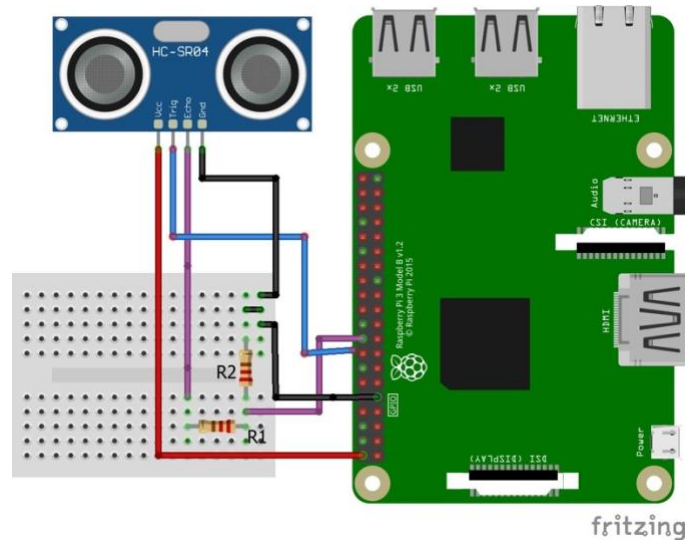


Figure 6.6 Ultrasonic Sensor Pin Configuration

The software code of the system is modified to accommodate these changes. Having this installed, the current draws is relatively smaller at about 590mA.

6.5.2 Heatsink

Figure 6.7 shows the heatsink being attached over the controller's processor. Having this installed, the microcontroller power consumption is relatively smaller at approximately 550mA current drawn within a 10 minutes timeframe.

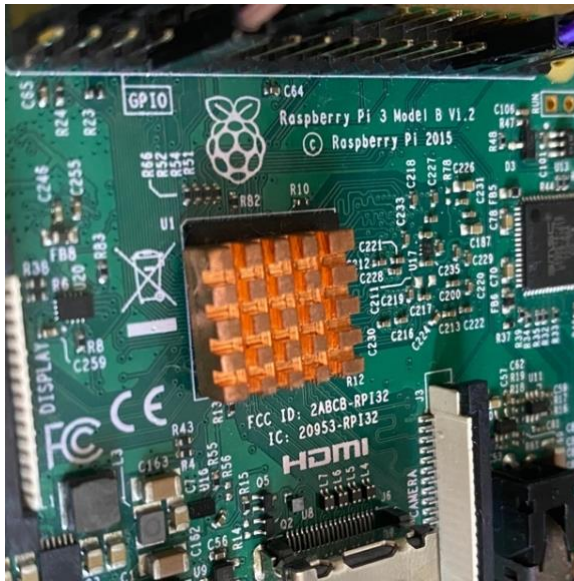


Figure 6.7 Heatsink Placement on Microcontroller

Graph shown in Figure 6.8 summarized the controller power consumption in four difference scenarios within 10 minutes time frame. The scenarios include the microcontroller without additional installation, microcontroller with ultrasonic sensor, microcontroller with heatsink, and microcontroller installed with both heatsink and ultrasonic sensor.

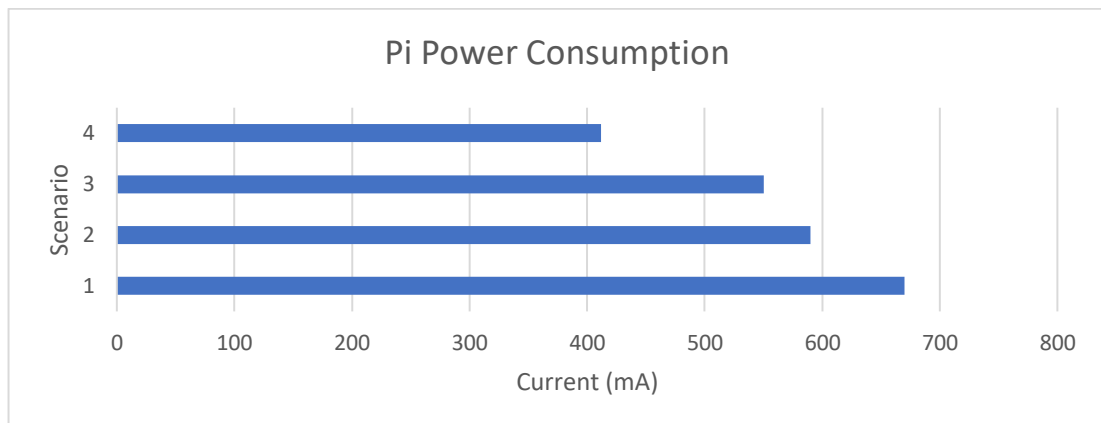


Figure 6.8 Microcontroller Power Consumption Based on Different Scenarios

As can be seen in Table 6.2, the last scenarios draw up the least current at about 412mA and temperature of 56°C. Thus, the system remained with the microcontroller with both components installed.

Table 6.2 Microcontroller Drawn Current and Temperature Based on Different Scenarios

Scenario	Description (10 minutes run)	Current(mA)	Temperature (°C)
1	Microcontroller	670	78
2	Microcontroller + Ultrasonic Sensor	590	64
3	Microcontroller + Heatsink	550	60
4	Microcontroller + Heatsink + Ultrasonic Sensor	412	56

6.6 Locking Mechanism Substitute

From the hardware configuration, it is noted that the relay and the electromagnet lock will represent the locking mechanism. However for the implementation testing, two LEDs are being used to reduce cost. When the door is unlocked, the green LED lights up for 5 seconds, else red LED remains on to indicate that the door is locked. Figure 6.9 shows the pin connection to the Raspberry Pi GPIO. Red LED is connected to GPIO 6 whereas Green LED connected to GPIO 18.

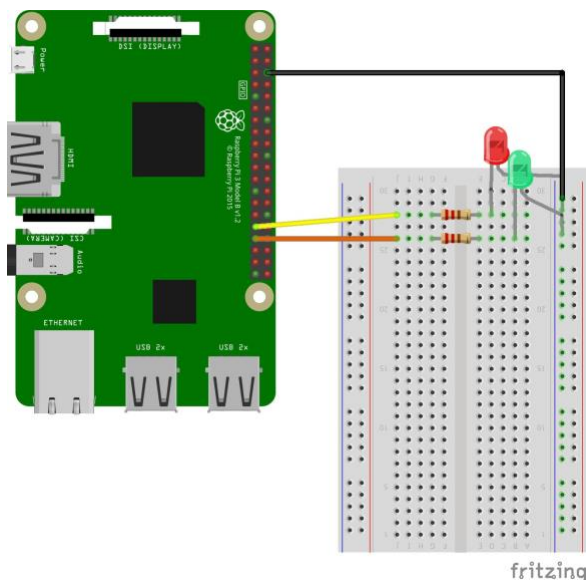


Figure 6.9 Locking Mechanism Indicator Pin Configuration

6.7 Final Product

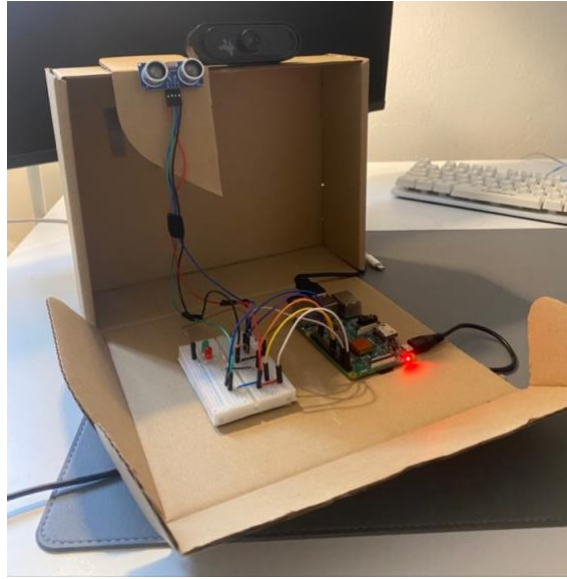


Figure 6.10 Final Product Prototype

CHAPTER 7

CONCLUSION

The project effectively attains its initial aims and objective. A smart locking system using an AIoT framework had been developed, leveraging the cost-effective Raspberry Pi microcontroller and the python programming language. The face recognition algorithm also proven to be able to detect a person with and without mask, as well as notifying the owner via email when an intruder is detected. Notable enhancements include a study on a low consumption effort to optimize the power consumption of the system for long-term usage. In conclusion, the system's architecture is clearly presented, seamlessly integrating all platforms. Towards the end, a complete system with a hardware prototype had been developed, all parts were integrated, tested and validated.

7.1 Future Work

The subsequent phase of the development for this technology will involve its expansion for use across various industries that necessitate locking system such as libraries, condominium complexes and private parking areas. In cases where no further feature expansion is needed but the controller is desired in multiple locations, duplicating the storage media which is the MicroSD card, from the Raspberry Pi to a new device can be achieved without concerns about the kernel or driver compatibility.

In terms of security, there are several improvements that could be implemented to enhance the system by introducing data encryption for microcontroller and the database. This modification aims to restrict the decoding of the information to only designated controller. Apart from that, the face recognition reader could be upgraded to an infrared camera, offering improved quality and immunity to varying light conditions. Additionally, as the proposed system only logged or notify owner for intruder detected, the system can also be upgraded to logged authorised entries for owner to trace.

7.2 Project Reflection

As is common in most projects, there were instances of unexpected events influencing milestones, either advancing or delaying them, and this project is no exception. the project schedule has encountered a minor conflict on the hardware-software development. However in the first half of the project schedule, the project managed to complete 50% of the hardware design development by running a simulation on the expected result and it is expected to be tested on physical circuit, moving forward. To compensate on the delay, the project schedule has been adjusted where the software code development is expected to run in parallel with the remaining hardware testing. Ultimately, the project outcome have both hardware – software design integrated and run seamlessly to achieve the main project objective. Overall, the project has progressed well, and finished on time and within the scheduled timeline.

REFERENCE

Zhang, F., Pan, Z. and Lu, Y., 2023. AIoT-enabled smart surveillance for personal data digitalization: Contextual personalization-privacy paradox in smart home. *Information & Management*, 60(2), p.103736.

Normalini, M.K. and Ramayah, T., 2015. Understanding security in consumer adoption of internet banking: Biometrics technology implementation in the Malaysian banking context. In *Banking, Finance, and Accounting: Concepts, Methodologies, Tools, and Applications* (pp. 685-698). IGI Global.

G. Sowmya, G. Divya Jyothi, N. Shirisha, K. Navya, and B. Padmaja, 2018 "IOT based Smart Door Lock System," *International Journal of Engineering & Technology*, vol. 7, no. 3.6, p. 223.

Akshay Krishnadas Bhat, Siddhesh Praveen Kini 2018, "Password Enabled Door Locking System using Arduino and IoT," *International Journal of Engineering Research & Technology (IJERT)* , Vol. 7 Issue 5

Y. Kim, J. Choi, and J. Park, 2021 "An AIoT-based Smart Lock System with Smart Tags for Secure and Convenient Access Control," *IEEE Access*, vol. 9, pp. 101501-101511.

P. N. Narkar, R. Fernandes, R. Dmello, and J. Louis, 2017 "Study on Access Control and Management Using NFC," *International Journal of Latest Engineering Research and Applications (IJLERA)*, vol. 02, pp. 126–129.

Shin, S.S., Han, K.H. and Jin, K.Y., 2013. Digital door lock on the access control system using otp-based user authentication. *International Journal of Digital Content Technology and its Applications*, 7(11), p.436.

Aggarwal, M., 2017. Secure Electronic Lock Based on Bluetooth Based OTP System. In *Elins International Journal of Science Engineering and Management* (Vol. 2, No. 1). Department of Electronics and Communication Engineering

Akanbi, C.O., Ogundoyin, I.K., Akintola, J.O. and Ameenah, K., 2020. A prototype model of an iot-based door system using double-access fingerprint technique. *Nigerian Journal of Technological Development*, 17(2).

Rao, S., Panguluri, M.A.H. and Sriharika, C., 2019. Automatic door unlock system using IOT and RFID. *International Journal of Innovative Technology and Exploring Engineering*, 8(5), pp.619-23.

Yedulapuram, S., Arabelli, R., Mahender, K. and Sidhardha, C., 2020, December. Automatic door lock system by face recognition. In *IOP Conference Series: Materials Science and Engineering* (Vol. 981, No. 3, p. 032036). IOP Publishing.

Waseem, M., Khowaja, S.A., Ayyasamy, R.K. and Bashir, F., 2020, October. Face recognition for smart door lock system using hierarchical network. In *2020 International Conference on Computational Intelligence (ICCI)* (pp. 51-56). IEEE.

Khalimov, R., Rakhimbayeva, Z., Shokayev, A., Kamalov, B. and Ali, M.H., 2020, July. Development of intelligent door locking system based on face recognition technology. In *2020 11th International Conference on Mechanical and Aerospace Engineering (ICMAE)* (pp. 244-248). IEEE.

M. Siddique, H. Sohail, and M. Sarwar, 2019 "A smart lock system based on token authentication for home security," 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), pp. 1-6, doi: 10.1109/CAIS48617.2019.9031984.

C. Gunasinghe, S. K. Halgamuge, and P. M. A. Siriwardana , 2018 "Token-based security for IoT home automation systems," 2018 IEEE International Conference on Industrial Technology (ICIT), pp. 1225-1230.

Hemalatha, A. and Gandhimathi, G., 2019. RFID, Password and OTP based Door Lock System using 8051 Microcontroller. *International Journal of Engineering Research & Technology (IJERT) CONFCALL*, 7(11).

Mei Yin, D.B., Kamal, M.I., Azmanuddin, N.S., Ali, S.H.S., Othman, A.T. and Chik, R.Z.W., 2016, January. Electronic door access control using MyAccess two-factor authentication scheme featuring near-field communication and eigenface-based face recognition using principal component analysis. In *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication* (pp. 1-8).

Motwani, Y., Seth, S., Dixit, D., Bagubali, A. and Rajesh, R., 2021. Multifactor door locking systems: A review. *Materials Today: Proceedings*, 46, pp.7973-7979.

Arduino (2019). *Arduino Uno Rev3*. Available at Arduino.cc. Available at: <https://store.arduino.cc/usa/arduino-uno-rev3> (Accessed: 16 April 2023).

Raspberry Pi Foundation (no date). *Raspberry Pi hardware - Raspberry Pi Documentation*. Available at: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md> (Accessed: 16 April 2023).

Google (no date). *Google Cloud overview*. Available at: <https://cloud.google.com/docs/overview>. (Accessed: 16 April 2023).

F. Li, A. Karpathy, and J. Johnson (2015) "Convolutional Neural Networks for Visual Recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1-9.

Nath, J. (2021). *Introduction to TensorFlow Keras*. Medium. Available at: <https://towardsdatascience.com/introduction-to-tensorflow-keras-53aa859fcc93> (Accessed: 16 April 2023).

Raspberry Pi Foundation (no date) *Raspberry pi documentation, Raspberry Pi OS*. Available at: <https://www.raspberrypi.com/documentation/computers/os.html> (Accessed: 16 April 2023).

Python (2023) *Venv - creation of Virtual Environments, Python documentation*. Available at: <https://docs.python.org/3/library/venv.html> (Accessed: 17 August 2023).

Visualstudio.com, (2016). *Documentation for Visual Studio Code*. Available at <https://code.visualstudio.com/docs> (Accessed: 16 April 2023).

Debian Webmaster, D.W. (2012) *Package: Fswebcam, Debian*. Available at: <https://packages.debian.org/sid/graphics/fswebcam> (Accessed: 17 August 2023).

Ningbo Songle Relay Co. Ltd (2012) "Relay" SRD-05VDC-SL-C datasheet.

First Components International, no date "NPN General Purpose Transistor," 2N2222A datasheet.

Schroff, F., Kalenichenko, D. and Philbin, J., 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815-823).

Spark Fun Electronic (no date) *Ultrasonic ranging module HC - SR04 - SparkFun Electronics, sparkfun*. Available at: <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf> (Accessed: 16 April 2023).

APPENDICES

APPENDIX A – Source Code – Main.py

```
from face_recognition.real_time_face_recognition import *

import time

def main():
    start_system()

def start_system():
    cam = 0

    while True:
        try:
            # Using ultra sonic distance sensor to measure the distance of object
            setup_gpio_pins()
            dist = measure_average()
            time.sleep(1)
            # Whenever you press ctrl+c script stop running
        except KeyboardInterrupt:
            GPIO.cleanup()
            break

    if object_detected(dist):
        print("HERE : ", dist)
        print("Starting Face Recognition")
        face_recognition(camera_source=cam, resolution="720", to_write=True,
```

APPENDIX B – Source Code – face_detection.py

```
import numpy as np

from face_recognition.parameters import *

import tensorflow as tf
tf.compat.v1.disable_v2_behavior()

class FaceMaskDetection():
    def __init__(self,pb_path,margin=44,GPU_ratio=0.1):

        # ===generate anchors
        anchors = self.generate_anchors(feature_map_sizes, anchor_sizes, anchor_ratios)
        anchors_exp = np.expand_dims(anchors, axis=0)

        sess, tf_dict = get_sess_from_pre_trained_pb(pb_path, node_dict,GPU_ratio = GPU_ratio)
        tf_input = tf_dict['input']
        model_shape = tf_input.shape # [N,H,W,C]
        print("model_shape = ", model_shape)
        img_size = (int(tf_input.shape[2]),int(tf_input.shape[1]))
        print("img_size = ", tf_input.shape[2], tf_input.shape[1])
        detection_bboxes = tf_dict['detection_bboxes']
        detection_scores = tf_dict['detection_scores']

        # ----local var to global
        self.model_shape = model_shape
        self.img_size = img_size
        self.sess = sess
        self.tf_input = tf_input
        self.detection_bboxes = detection_bboxes
```

```

def decode_bbox(self, anchors, raw_outputs, variances=[0.1, 0.1, 0.2, 0.2]):

    anchor_centers_x = (anchors[:, :, 0:1] + anchors[:, :, 2:3]) / 2
    anchor_centers_y = (anchors[:, :, 1:2] + anchors[:, :, 3:]) / 2
    anchors_w = anchors[:, :, 2:3] - anchors[:, :, 0:1]
    anchors_h = anchors[:, :, 3:] - anchors[:, :, 1:2]
    raw_outputs_rescale = raw_outputs * np.array(variances)
    predict_center_x = raw_outputs_rescale[:, :, 0:1] * anchors_w + anchor_centers_x
    predict_center_y = raw_outputs_rescale[:, :, 1:2] * anchors_h + anchor_centers_y
    predict_w = np.exp(raw_outputs_rescale[:, :, 2:3]) * anchors_w
    predict_h = np.exp(raw_outputs_rescale[:, :, 3:]) * anchors_h
    predict_xmin = predict_center_x - predict_w / 2
    predict_ymin = predict_center_y - predict_h / 2
    predict_xmax = predict_center_x + predict_w / 2
    predict_ymax = predict_center_y + predict_h / 2
    predict_bbox = np.concatenate([predict_xmin, predict_ymin, predict_xmax, predict_ymax],

```

axis=1)

```

def single_class_non_max_suppression(self, bboxes, confidences, conf_thresh=0.2, iou_thresh=0.5,
keep_top_k=-1):

    if len(bboxes) == 0: return []

    conf_keep_idx = np.where(confidences > conf_thresh)[0]

    bboxes = bboxes[conf_keep_idx]
    confidences = confidences[conf_keep_idx]

    pick = []
    xmin = bboxes[:, 0]
    ymin = bboxes[:, 1]
    xmax = bboxes[:, 2]
    ymax = bboxes[:, 3]

    area = (xmax - xmin + 1e-3) * (ymax - ymin + 1e-3)
    idxs = np.argsort(confidences)

    while len(idxs) > 0:
        last = len(idxs) - 1
        i = idxs[last]
        pick.append(i)

        # keep top k
        if keep_top_k != -1:
            if len(pick) >= keep_top_k:
                break

        overlap_xmin = np.maximum(xmin[i], xmin[idxs[:last]])
        overlap_ymin = np.maximum(ymin[i], ymin[idxs[:last]])
        overlap_xmax = np.minimum(xmax[i], xmax[idxs[:last]])
        overlap_ymax = np.minimum(ymax[i], ymax[idxs[:last]])
        overlap_w = np.maximum(0, overlap_xmax - overlap_xmin)
        overlap_h = np.maximum(0, overlap_ymax - overlap_ymin)
        overlap_area = overlap_w * overlap_h
        overlap_ratio = overlap_area / (area[idxs[:last]] + area[i] - overlap_area)

        need_to_be_deleted_idx = np.concatenate(([last], np.where(overlap_ratio > iou_thresh)[0]))

```

```

def inference(self, img_4d, ori_height, ori_width):
    # ---var
    re_boxes = list()
    re_confidence = list()
    re_classes = list()
    re_mask_id = list()

    y_bboxes_output, y_cls_output = self.sess.run([self.detection_bboxes, self.detection_scores],
                                                  feed_dict={self.tf_input: img_4d})

    # remove the batch dimension, for batch is always 1 for inference.
    y_bboxes = self.decode_bbox(self.anchors_exp, y_bboxes_output)[0]
    y_cls = y_cls_output[0]

    # To speed up, do single class NMS, not multiple classes NMS.
    bbox_max_scores = np.max(y_cls, axis=1)
    bbox_max_score_classes = np.argmax(y_cls, axis=1)

    # keep_idx is the alive bounding box after nms.
    keep_idxs = self.single_class_non_max_suppression(y_bboxes, bbox_max_scores,
conf_thresh=self.conf_thresh,
                                                    iou_thresh=self.iou_thresh )

    # ====draw bounding box
    for idx in keep_idxs:
        conf = float(bbox_max_scores[idx])
        #print("conf = ", conf)
        class_id = bbox_max_score_classes[idx]
        bbox = y_bboxes[idx]
        #print(bbox)

        xmin = np.maximum(0, int(bbox[0] * ori_width - self.margin / 2))
        ymin = np.maximum(0, int(bbox[1] * ori_height - self.margin / 2))
        xmax = np.minimum(int(bbox[2] * ori_width + self.margin / 2), ori_width)
        ymax = np.minimum(int(bbox[3] * ori_height + self.margin / 2), ori_height)

        re_boxes.append([xmin, ymin, xmax - xmin, ymax - ymin])
        re_confidence.append(conf)
        re_classes.append('face')
        re_mask_id.append(class_id)
    return re_boxes, re_confidence, re_classes, re_mask_id

```

APPENDIX C – Source Code – real_time_face_recognition.py

```
import cv2, os, time, math, csv
import numpy as np

from face_recognition.face_detection import FaceMaskDetection, get_sess_from_pre_trained_pb
from face_recognition.parameters import *
from face_recognition.message_sending_utils import *
from face_recognition.utils import *
from gpio_utils import *

import tensorflow as tf
tf.disable_v2_behavior()
import tensorflow.compat.v1.gfile as gfile

isWriteToCSV = 0

def get_cam_resolution_decision(resolution=DEFAULT_RESOLUTION):
    if resolution in resolution_dict.keys():
        width = resolution_dict[resolution][1]
        height = resolution_dict[resolution][0]
    else:
        width = DEFAULT_FRAME_WIDTH
        height = DEFAULT_FRAME_HEIGHT

    return width, height

def save_video_recording(to_write=False, resolution=DEFAULT_RESOLUTION,
save_dir=RECORDING_DIRECTORY):
    writer = None
    if to_write is True:
        fourcc = cv2.VideoWriter_fourcc(*'MP4V')

        date, time = get_date_time_now()
        save_path = f'video_capture_{date}_{time}.mp4'
```

```

def video_init(camera_source=0,resolution="480",to_write=False,save_dir=None):
    cap = cv2.VideoCapture(camera_source)

    width, height = get_cam_resolution_decision(resolution)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)

    writer = save_video_recording(to_write, width)

    return cap,height,width,writer

def get_image_from_database(tf_sess, tf_embed, model_shape, feed_dict, tf_input):
    d_t = time.time()
    paths = [file.path for file in os.scandir(REFERENCE_DIRECTORY) if file.name[-3:] in img_format]
    len_ref_path = len(paths)
    if len_ref_path == 0:
        print("No images in ", REFERENCE_DIRECTORY)
    else:
        ites = math.ceil(len_ref_path / BATCH_SIZE)
        embeddings_ref = np.zeros([len_ref_path, tf_embed.shape[-1]], dtype=np.float32)

        for i in range(ites):
            num_start = i * BATCH_SIZE
            num_end = np.minimum(num_start + BATCH_SIZE, len_ref_path)

            batch_data_dim =[num_end - num_start]
            batch_data_dim.extend(model_shape[1:])
            batch_data = np.zeros(batch_data_dim,dtype=np.float32)

            for idx,path in enumerate(paths[num_start:num_end]):
                img = cv2.imread(path)
                if img is None:
                    print("read failed:",path)
                else:
                    print(img.shape,(type(model_shape[2]),model_shape[1]))
                    img = cv2.resize(img,(int(model_shape[2]),int(model_shape[1])))
                    img = img[:,:,:-1]#change the color format
                    batch_data[idx] = img
            batch_data /= 255
            feed_dict[tf_input] = batch_data

            embeddings_ref[num_start:num_end] = tf_sess.run(tf_embed,feed_dict=feed_dict)

    d_t = time.time() - d_t

```



```

def get_sess_from_pre_trained_pb(pb_path, node_dict, GPU_ratio=None, is_face_reg_pb = 0):
    tf_dict = dict()
    with tf.Graph().as_default():
        config = tf.ConfigProto(log_device_placement=True, #print out GPU or CPU is adopted
                                allow_soft_placement=True, #allow tf to use alternative devices
                                )
        if GPU_ratio is None:
            config.gpu_options.allow_growth = True # The program can access as much resource as
possible
        else:
            config.gpu_options.per_process_gpu_memory_fraction = GPU_ratio # limit the GPU
resource
        sess = tf.Session(config=config)
        with gfile.FastGFile(pb_path, 'rb') as f:
            graph_def = tf.GraphDef()
            graph_def.ParseFromString(f.read())
            sess.graph.as_default()

            if is_face_reg_pb == 1:
                for node in graph_def.node:
                    if node.op == 'RefSwitch':
                        node.op = 'Switch'
                        for index in range(len(node.input)):
                            if 'moving_' in node.input[index]:
                                node.input[index] = node.input[index] + '/read'
                    elif node.op == 'AssignSub':
                        node.op = 'Sub'
                        if 'use_locking' in node.attr: del node.attr['use_locking']

            tf.import_graph_def(graph_def, name='') # import the calculation graph

        sess.run(tf.global_variables_initializer())

        for key, value in node_dict.items():

```

```

def face_recognition(camera_source=0,resolution="480",to_write=False,save_dir=None):

    #---Video streaming initialization
    cap,height,width,writer = video_init(camera_source=camera_source, resolution=resolution, to_write=to_write, save_dir=save_dir)

    # ---face detection init
    fmd = FaceMaskDetection(FACE_MASK_MODEL_PATH, MARGIN, GPU_ratio=None)

    # ----face recognition init
    tf_sess, tf_dict = get_sess_from_pre_trained_pb(FACE_RECOGNITION_MODEL_PATH, tf_node_dict, GPU_ratio=None,
is_face_reg_pb=1)
    print(tf_dict)
    tf_input = tf_dict['input']
    tf_phase_train = tf_dict['phase_train']
    tf_embed = tf_dict['embeddings']
    model_shape = tf_input.shape.as_list()
    print("The mode shape of face recognition:",model_shape)
    feed_dict = {tf_phase_train: False}
    if 'keep_prob' in tf_dict.keys():
        tf_keep_prob = tf_dict['keep_prob']
        feed_dict[tf_keep_prob] = 1.0

    len_ref_path, embeddings_ref, paths = get_image_from_database(tf_sess, tf_embed, model_shape, feed_dict, tf_input)

    # ----tf setting for calculating distance
    if len_ref_path > 0:
        with tf.Graph().as_default():
            tf_tar = tf.placeholder(dtype=tf.float32, shape=tf_embed.shape[-1])
            tf_ref = tf.placeholder(dtype=tf.float32, shape=tf_embed.shape)
            tf_dis = tf.sqrt(tf.reduce_sum(tf.square(tf.subtract(tf_ref, tf_tar)), axis=1))
            # ----GPU setting
            config = tf.ConfigProto(log_device_placement=True,
                allow_soft_placement=True,
                )
            config.gpu_options.allow_growth = True

```

APPENDIX D – Source Code – gpio_utils.py

```
import time
import RPi.GPIO as GPIO
from face_recognition.parameters import GPIO_TRIGGER, GPIO_ECHO, GPIO_LED_GREEN,
GPIO_LED_RED, OBJECT_NEARBY_THRES

def measure():
    # This function measures a distance
    GPIO.output(GPIO_TRIGGER, True)
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)
    start = time.time()

    while GPIO.input(GPIO_ECHO)==0:
        start = time.time()

    while GPIO.input(GPIO_ECHO)==1:
        stop = time.time()

    elapsed = stop-start
    distance = (elapsed * 34300)/2

    return distance

def measure_average():
    # This function takes 3 measurements and
    # returns the average.
    distance1=measure()
    time.sleep(0.1)
    distance2=measure()
    time.sleep(0.1)
    distance3=measure()
    distance = distance1 + distance2 + distance3
    distance = distance / 3
    return distance

def unlock_door():
```

```
print("starting to unlock_door ")
GPIO.output(GPIO_LED_RED, False)
GPIO.output(GPIO_LED_GREEN, True)
time.sleep(3)
GPIO.output(GPIO_LED_GREEN, False)
GPIO.output(GPIO_LED_RED, True)
```

```
def setup_sensor_gpio_pins():
    # Use BCM GPIO references
    # instead of physical pin numbers
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)

    # Set pins as output and input
    GPIO.setup(GPIO_TRIGGER,GPIO.OUT) # Trigger
    GPIO.setup(GPIO_ECHO,GPIO.IN)    # Echo

def setup_LED_gpio_pins():
    GPIO.setwarnings(False)
    GPIO.setup(GPIO_LED_RED, GPIO.OUT)
    GPIO.setup(GPIO_LED_GREEN, GPIO.OUT)

    # Set trigger to False (Low)
    GPIO.output(GPIO_TRIGGER, False)

def setup_gpio_pins():
    setup_sensor_gpio_pins()
    setup_LED_gpio_pins()
```