

**HANDGESTURESENSE: HAND GESTURE RECOGNITION-CONTROLLED  
SMART HOME AUTOMATION**

**BY**

**TAN TECK SHENG**

**A REPORT**

**SUBMITTED TO**

**Universiti Tunku Abdul Rahman**

**in partial fulfillment of the requirements**

**for the degree of**

**BACHELOR OF COMPUTER SCIENCE (HONOURS)**

**Faculty of Information and Communication Technology**

**(Kampar Campus)**

**January 2024**

## REPORT STATUS DECLARATION FORM

**Title:** HANDGESTURESENSE: HAND GESTURE RECOGNITION-CONTROLLED  
SMART HOME AUTOMATION

**Academic Session:** January 2024

I TAN TECK SHENG

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in  
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



\_\_\_\_\_  
(Author's signature)



\_\_\_\_\_  
(Supervisor's signature)

**Address:**

50, Jalan Padi 2,

Taman Baru Uda,

81200 Johor Bahru, Johor

Tseu Kwan Lee \_\_\_\_\_

Supervisor's name

**Date:** 20 April 2024

**Date:** 2024.04.26

<b>Universiti Tunku Abdul Rahman</b>			
Form Title : <b>Sample of Submission Sheet for FYP/Dissertation/Thesis</b>			
Form Number: <b>FM-IAD-004</b>	Rev No.: <b>0</b>	Effective Date: <b>21 JUNE 2011</b>	Page No.: <b>1 of 1</b>

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**  
**UNIVERSITI TUNKU ABDUL RAHMAN**

Date: 20 April 2024

**SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS**

It is hereby certified that Tan Teck Sheng (ID No: 20ACB03875 ) has completed this final year project entitled “ HANDGESTURESENSE: HOME GESTURE RECOGNITION-CONTROLLED SMART HOME AUTOMATION ” under the supervision of Tseu Kwan Lee (Supervisor) from the Department of Computer Science , Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,



\_\_\_\_\_  
(TAN TECK SHENG)

\*Delete whichever not applicable

## DECLARATION OF ORIGINALITY

I declare that this report entitled “**HANDGESTURESENSE: HAND GESTURE RECOGNITION-CONTROLLED SMART HOME AUTOMATION**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : \_\_\_\_\_  \_\_\_\_\_

Name : TAN TECK SHENG

Date : 24/4/2024

## **ACKNOWLEDGEMENTS**

I would like to express thanks and appreciation to my supervisor, Ms. Tseu Kwan Lee and my moderator, Dr. Tong Dong Ling who have given me a golden opportunity to participate on the Internet of Things field study. Besides that, they have given me a lot of guidance to complete this project. When I was facing problems in this project, their advice from them always assists me in overcoming the problems. Again, a million thanks to my supervisor and moderator.

## **ABSTRACT**

This project focuses on smart home automation controlled by gesture recognition, integrating computer vision, machine learning and home automation technologies. The main goal is to develop an accurate system that can effectively detect and interpret gestures while enhancing the system's adaptability and performance. The system enables users to seamlessly control every aspect of their smart home without the need for physical contact. The proposed project utilizes computer vision algorithms to capture and analyze live image or gesture video streams. Through extensive training and fine-tuning using machine learning techniques, the system learns to recognize a series of gestures, each corresponding to a different command or function in a smart home environment. These gestures may include actions such as opening, closing, pointer or other hand movements, with a particular focus on controlling and monitoring fan and light appliances. The proposal has important implications for people with limited mobility or disabilities, allowing them to easily navigate and manage their living spaces. Additionally, gesture recognition systems have the potential to improve energy efficiency and safety by responding to gestures indicating a user's presence or absence, thereby adjusting lighting and surveillance systems accordingly.

Keywords: gesture recognition, machine learning, home automation, lighting

# TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>REPORT STATUS DECLARATION FORM</b>	<b>ii</b>
<b>FYP THESIS SUBMISSION FORM</b>	<b>iii</b>
<b>DECLARATION OF ORIGINALITY</b>	<b>iv</b>
<b>ACKNOWLEDGEMENTS</b>	<b>v</b>
<b>ABSTRACT</b>	<b>vi</b>
<b>TABLE OF CONTENTS</b>	<b>vii</b>
<b>LIST OF FIGURES</b>	<b>x</b>
<b>LIST OF TABLES</b>	<b>xi</b>
<b>LIST OF SYMBOLS</b>	<b>xii</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xiii</b>
<b>CHAPTER 1 INTRODUCTION</b>	
1.1 Problem statement	3
1.2 Motivation	5
1.3 Research Objectives	7
1.4 Project Scope and Direction	8
1.5 Contributions	10
1.6 Report Organization	11
<b>CHAPTER 2 LITERATURE REVIEW</b>	
2.1 Previous Works on Deep Learning	12
2.1.1 Using Gestures to Interact with Home Automation Systems: A Socio-Technical Study on Motion Capture Technologies for Smart Homes	12
2.1.2 Hand Gesture Recognition System for Controlling Home Appliances	14
2.1.3 Smart Home Automation-Based Hand Gesture Recognition Using Feature Fusion and Recurrent Neural Network	16
2.1.4 Hand Gesture Recognition for Home Automation	18

2.1.5 Smart and Energy Efficient Gesture Controlled Home Automation	20
2.1.6 Techniques and Challenges of Face Recognition: A Critical Review	24
2.2 Limitation of Previous Works	25
2.3 Proposed Solutions	26

### **CHAPTER 3 SYSTEM METHODOLOGY/APPROACH**

3.1 System Design Diagram/Equation	31
3.1.1 System Architecture Diagram	31
3.1.2 Use Case Diagram and Description	31
3.1.3 Activity Diagram	34
3.2 Hand Gesture Recognition Method	35
3.3 Hand Gesture Recognition Model	35
3.3.1 CNN Hand-Gesture Recognition Model	35
3.3.2 Random Forest Model	37
3.3.3 NN Model (Neuro Network)	39
3.4 Flowchart	40
3.5 Function to Extract RGB Value of The Environment	41
3.6 Gantt Chart	43

### **CHAPTER 4 SYSTEM DESIGN**

4.1 System Block Diagram	45
4.1.1 Hand Gesture Recognition	45
4.1.2 Face Recognition	48
4.2 System Components Specifications	49
4.2.1 Hardware	49
4.2.2 Software	51
4.3 Circuit and Components Specifications	52
4.3.1 System Circuit Diagram	52



## **CHAPTER 5 SYSTEM IMPLEMENTATION**

5.1	Setting and Configuration	55
5.1.1	Hardware Setup	55
5.1.2	Software Setup	55
5.2	Python Code Explanation	62
5.2.1	Hand Gesture Recognition Code Explanation	62
5.2.2	Face Recognition Code Explanation	68
5.2.3	Code to Extract RGB Value and Manual and Auto Increase/Decrease Brightness	72
5.3	System Operation	74
5.4	Implementation Issues and Challenges	75

## **CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION**

6.1	System Testing and Performance Metrics	76
6.1.1	Performance Metrics for Hand Gestures Recognition Model	76
6.1.2	Testing Hand Gestures Control Smart Home	77
6.1.3	Testing Face Recognition	78
6.1.4	Testing the convertScaleAbs Functions	78
6.2	Testing Result	79
6.2.1	Confusion Metrics for Recognition Model	79
6.2.2	Result of Testing Hand Gestures control Smart Home	85
6.2.3	Result of Face Recognition	86
6.2.4	Result of Using cv2.convertScaleAbs() Functions	87
6.3	Project Challenges	90
6.4	Objective Evaluation	91

## **CHAPTER 7 CONCLUSION AND RECOMMENDATION**

7.1	Conclusion	91
7.2	Recommendation	93

## **REFERENCES 93**

## **APPENDIX 95**

## **WEEKLY LOG 115**

<b>POSTER</b>	<b>121</b>
<b>PLAGIARISM CHECK RESULT</b>	<b>122</b>
<b>FYP2 CHECKLIST</b>	<b>131</b>

## LIST OF FIGURES

<b>Figure Number</b>	<b>Title</b>	<b>Page</b>
Figure 1	Architecture diagram of the hand gesture interaction for HAS	12
Figure 2	Hand-gesture dictionary visualised with Kinect	14
Figure 3	Architecture diagram of Hand gesture based remote control.	15
Figure 4	Image segmentation result	15
Figure 5	Architecture diagram of the proposed system for hand gesture recognition	17
Figure 6	Demonstrates the comparison of the accuracy of RNN with other-state-of the-art algorithms	18
Figure 7	Segmentation results of the hand gesture	19
Figure 8	Image processing algorithm	20
Figure 9	Flowchart of the recognition process and home automation system	22
Figure 10	(a) Original image (b) Image obtained after converting the image into HSV color model (c) Image after converting the (b)image into Grayscale image.	23
Figure 11	Face detection process	24
Figure 12	Smart home architecture diagram	30
Figure 13	Use case diagram for Face Recognition System	31
Figure 14	Use case diagram for Face Recognition System	32
Figure 15	Activity diagram of whole system	34
Figure 16	Hand Gesture Recognition Method	35
Figure 17	Machine learning model	37
Figure 18	RGB Matrix for CNN	37
Figure 19	Random Forest model architecture	39
Figure 20	Neural networks model architecture	40
Figure 21	Flowchart of Hand-gestures recognition system	41
Figure 22	Colored Scatter Plot in Dark Environment	41

Figure 23	Colored Scatter Plot in Normal Environment	41
Figure 24	HSV Scatter Plot in Normal Environment	42
Figure 25	Gantt Chart for FYP1	43
Figure 26	Gantt Chart for FYP2	44
Figure 27	Block Diagram of hand gestures recognition system	45
Figure 28	Block Diagram of face recognition system	47
Figure 29	Circuit diagram of the project	51
Figure 30	System circuit for operation.	73
Figure 31	Issues when install opencv in Raspberry Pi	74
Figure 32	Confusion matrix of CNN model	78
Figure 33	Confusion matrix of Random Forest Model	79
Figure 34	Confusion matrix of Random Forest Model	80
Figure 35	Result of face recognition system	85
Figure 36	Normal environment and before using this function	86
Figure 37	Result of using this function to tune the frame	86
Figure 38	Result of using function cv2.convertScaleAbs	88

## LIST OF TABLES

<b>Table Number</b>	<b>Title</b>	<b>Page</b>
Table 1	Comparison of 5 literature	29
Table 3.1.2.1	Use case description for face recognition	32
Table 3.1.3.2	Use case description for face recognition	33
Table 4.2.1	Specifications of laptop	48
Table 4.2.2	Specifications of hardware	49
Table 4.2.3	Specifications of software	50
Table 4.3.1	Mapping relationship between components	52
Table 5.1.1	Details of hardware setting up	54
Table 6.1.1.1	Hand gestures class.	77
Table 6.1.2.1	Gestures used to control smart home when testing the system	77
Table 6.1.2.2	Face used when testing the system	77
Table 6.1.4	Testing tuning function	77
Table 6.2.1.1	Classification report of CNN model	81
Table 6.2.1.2	Classification report of Random Forest Model	82
Table 6.2.1.3	Classification report of Neural Networks Model	83
Table 6.2.2.1	Result of Testing Hand Gestures Control Smart Home	85
Table 6.2.4.1	Result of image after tuning	89

## LIST OF SYMBOLS

$\beta$	beta
$\Omega$	Ohm (resistance)

## LIST OF ABBREVIATIONS

<i>CNN</i>	Convolutional Neural Network
<i>GND</i>	Ground
<i>HAS</i>	Home Automation System
<i>HCI</i>	Human Computer Interaction
<i>IDE</i>	Integrated Development Environment
<i>I/O</i>	Input / Output
<i>IoT</i>	Internet of Things
<i>kWh</i>	Kilowatt hour
<i>LED</i>	Light-Emitting Diode
<i>ML</i>	Machine Learning
<i>TAM</i>	Technology Acceptance Model
<i>NN</i>	Neuro Network

### **Chapter 1: Introduction**

In recent years, home automation has become a popular research field. In the rapidly evolving world of technology, the integration of smart devices and systems into our living spaces has reached unprecedented levels. As a result, the concept of smart home automation emerged, promising greater convenience, efficiency, and comfort for residents. [16] When comes to intelligence, however, traditional smart device management methods often rely on physical interfaces such as switches, remote controls, or mobile apps. Traditional methods require the use of multiple devices, including mice, keyboards, touch screens, remote controls, etc., to meet the needs of users to operate various home appliances with only their hands. Although these interfaces are fully functional and have high accuracy, they fail to provide truly seamless human-computer interaction. Due to this limitation, researchers have begun to focus on the fields of gesture recognition and facial recognition. These fields use computer vision and machine learning to interpret and respond to human gestures and facial shapes. Then, after gesture recognition, the system interprets the recognized gesture and maps it to a predefined action or command. For example, a gesture might mean turning on a light or turning a fan on or off. This mapping allows users to meaningfully interact with the smart home ecosystem and execute smart home commands.

By mapping the interpreted gestures to corresponding actions, the smart home system executes the required command. For example, if a user makes a specific gesture to indicate that the room temperature is rising, the thermostat will adjust accordingly. Likewise, gestures can control lighting, sound systems, and other smart devices in the home. This provides research in the field of smart home automation with a basis for exploring the intersection of human actions and smart environments. These studies demonstrate results in terms of feasibility, user experience, and potential accessibility benefits. Researchers have developed algorithms and models that can accurately detect and interpret a range of gestures, paving the way for gesture-driven smart home ecosystem interactions. Additionally, these studies explore the challenges and opportunities for real-time processing of video, training of machine learning models,



## CHAPTER 1

and seamless integration of gesture-driven control with existing smart home architectures.

Additionally, integrating gesture recognition into smart home automation not only improves user convenience but also helps improve the overall energy efficiency and sustainability of residential spaces. Since smart homes are designed to optimize resource usage, gesture-driven controls provide a way to operate devices hands-free and reduce contact with traditional interfaces. This is not only in line with today's push for environmentally friendly technologies, but also addresses issues related to energy conservation and environmental impact. Easily controlling every aspect of your home environment through intuitive gestures aligns with the wider goal of creating smart, responsive living spaces that adapt to user preferences while promoting sustainable practices. As researchers improve gesture recognition algorithms and solve real-world challenges, the potential for widespread adoption of these technologies will open up more possibilities for a more harmonious convergence between human behavior and the intelligent systems that govern our lives.

### 1.1 Problem Statement

This project requires the creation and integration of a seamless system, which involves a certain level of technical complexity. Software needs to adapt to the hardware and deal with software and hardware compatibility issues. The integrity of the project relied on selecting the best software and hardware components, which required trial and error with different hardware and software configurations to ensure they connected and adapted seamlessly. This iterative process is critical to creating a complete IoT system that seamlessly integrates gestures to control hardware components such as light bulbs and fans.

Although significant advances have been made in gesture recognition technology, these advances have also created challenges that require further research. The existing potential for gesture-controlled smart home automation shows promise, but the inherent complexities of the technology require careful examination of the barriers that prevent its seamless integration and widespread adoption. One of the key challenges is the reliability of gesture detection systems under various lighting conditions [1]. Changes in ambient lighting conditions can significantly affect image processing algorithms, especially tasks such as human skin detection [1]. These systems may perform very well in controlled environments, but accuracy may degrade when exposed to changing lighting conditions, such as low light or strong backlighting. With constantly changing ambient lighting, it becomes imperative to ensure better feature tuning for higher performance and accuracy.

Another challenge is the lack of comprehensive gesture vocabulary datasets, which poses a significant obstacle to the development of machine learning (ML) methods for gesture recognition and pose estimation. Annotated datasets are crucial to the standard learning process [4][17]. However, existing datasets mainly focus on gesture recognition or gesture estimation, so it is difficult to find a single dataset containing annotations for both tasks [4]. The lack of comprehensive datasets poses a major obstacle to effectively bridging the gap between gesture recognition and gesture estimation during model training.

## CHAPTER 1

To address these challenges, this proposal introduces a novel semi-supervised training scheme that allows the extraction of shared features from hand images with different annotation levels, thereby transferring hand pose estimation knowledge to the gesture recognition task. Establishing a set of universally recognized gestures that correspond to specific actions or functions is critical to creating a consistent and intuitive user interface. Without a standardized gesture vocabulary, users may encounter differences in gesture interpretation across devices or platforms, impacting usability and learnability.

### 1.2 Motivation

The project blends innovation and technology to explore and demonstrate the capabilities of emerging technologies such as computer vision, machine learning and the Internet of Things. It aims to show how these technologies can be integrated to create cutting-edge smart home systems.

The first motivation is the integration of gesture-controlled smart home automation with an increasing emphasis on user experience.[8] Lighting, temperature, entertainment and security systems can be adjusted with simple gestures or hand movements. This technology simplifies daily life and makes life more efficient. [7] For example, when a gesture is recognized and executed, clear feedback is provided, including visual (lightbulb switch), auditory (fan spinning) or tactile feedback, informing the user that their command has been received and executed. This feedback enhances the user's sense of control and understanding. Home automation encourages self-actualization by freeing people from repetitive and mundane tasks, allowing them to focus on important aspects such as self-actualization, personal growth and richer experiences.

Another motivation is to enhance the adaptability of computer vision to various environments, ensuring that it can detect user gestures for control regardless of environmental conditions. This prevents the system from becoming unintelligent due to harsh environments and hindering user control. Computer adaptability is achieved, and the system can handle gesture control autonomously without the need for users to constantly adjust computer vision functions, which can be achieved using artificial intelligence, making the system more convenient and efficient.

The next motivation is to learn adaptability between different hardware and software. Not all hardware and software versions are suitable for optimal use. This project analyzes and identifies the hardware and software versions best suited for smart home gesture control. The project aims to achieve hardware compatibility and optimal hardware performance levels, as well as software adaptation to that hardware version. Certain software versions may become overly complex and frequently updated,

## CHAPTER 1

requiring more value from the hardware and causing adaptation issues. On the other hand, outdated versions may lack the features needed to fully operate the hardware. Therefore, this project required in-depth research and tuning of hardware and software.

Another important motivation is to optimize energy usage. [18] AI can assist in analyzing users' energy consumption behavior to support home automation and reduce energy consumption. Gesture recognition can dynamically control lighting and energy-consuming devices based on the user's presence or absence. For example, users can easily turn off lights, entertainment systems and other energy-consuming devices, optimizing power usage, increasing sustainability, reducing environmental impact, and saving on energy costs. Additionally, smart home systems can use sensors to recognize gestures to detect user activity and presence, such as turning off lights when a room is unoccupied.

In conclusion of this part, the motivation behind the project extends into the areas of accessibility and inclusion. Gesture-controlled smart home automation can greatly benefit individuals with physical disabilities or limited mobility, providing them with a more convenient way to interact with their living environment. By relying on gestures rather than physical devices or manual controls, individuals facing physical limitations can enjoy greater independence and control over their surroundings. This is consistent with the broader goal of using technology to improve the quality of life for everyone, regardless of physical ability. Integrating gesture recognition technology into smart homes can not only enhance users' sense of empowerment, but also help create a more inclusive living space for different individuals. As technology continues to advance, this project aims to contribute to the ongoing conversation about how smart home automation innovations can positively impact and improve the lives of people with diverse needs and abilities

### 1.3 Research Objective

In order to gain a better understanding of this area, research is ongoing on existing hand-gesture control smart home systems. Analyzes and investigations were conducted. By understanding the strengths and weaknesses of previous jobs. This project is integrated into a seamless project by implementing ML, computer vision and hardware, from extracting gestures to operating smart home.

Sub-goals of the proposed project:

- To study the existing hand-gesture control smart home system, learn the basic IoT system so that hardware and software can be seamlessly connected and adapted, and understand how to extract gestures step by step, train model, use computer vision to extract video, and finally use ML to get the gesture given by the user, and finally perform the corresponding operation
- To propose a series of lighting problems, which can be solved from software or hardware. (From replacing hardware or using better software tuning method to improve the entire detection capability)
- To develop hand-gesture control smart home system to perform different smart home operations based on user's gestures in real time
- To evaluate the proposed system by using performance metrics and model accuracy

### 1.4 Project Scope and Direction

In this proposed project, a smart home automation system will be developed. The main scope of the project is to control the home through gestures. For this smart home system, the camera captures the user's gestures and then adjusts them according to the brightness of the room, so that the gestures can be captured in any situation to control the smart home. The point of this project is the need for a computer core, the Raspberry Pi, to have a complete control over all functions and any settings. This core connects all hardware so that their information can be transferred to any device.

The direction of this project is to focus on creating more gesture vocabulary, adding more devices and solving the problem of lighting environment. Thus, in the future project, I will include more devices, specifically fans and lighting. Then I'll expand the gesture vocabulary such that all devices have their unique opening gestures. The next stage is to handle the issue of ambient lighting, as users may be in a variety of locations, some of which are too dark or harsh for gesture control.

The direction of the next project is to build a face recognition system. The system will be able to detect the user-face, in order to prevent unauthorized user in accessing the system. Upon the authentication of the user, the system will recognize the user gesture in allowing them to control the smart home appliances.

The project scope also includes trying out various ML/DL to identify which model is more suitable to be used in the project. The project scope also includes crafting manual and automatic adjustments to adapt computer vision to the user's different environments in order to detect and act on the user's gestures at any time.

Project scope not included is:

Although implementing security measures to protect user data and privacy is crucial, especially in smart home automation systems, security and privacy are not yet included in our scope. Cameras or sensors used for gesture recognition may raise privacy concerns in home environments. Balancing the needs of gesture recognition with the privacy rights of occupants is also a complex challenge [5].

## CHAPTER 1

Wireless connectivity between the Raspberry Pi and other devices, although implementing this will make smart home systems more efficient and convenient. But all wirelessly connected wireless devices must include power and internet connectivity chips, which would make the project very expensive.

Because the hardware capabilities of Raspberry Pi 4b are limited, this will limit my scope, that show face recognition, 3D scatters plots, and manual/automatic tuning on the Raspberry Pi 4b, but it can be displayed on the computer. The assumption is that if you have better equipment, everything will be better, get faster responses and feedback, and be more efficient.

Anything outside the project scope would have to exclude a facial recognition system, which requires a large amount of data to learn due to the differences in faces across nationalities. This simple face recognition system only extracts a small amount of data, so it is not that accurate and it is not merged into the hand gesture control inside the smart home.



### 1.5 Contribution

The proposal aims to make a significant contribution in several key areas:

This research strives to contribute to the advancement of gesture recognition technology by developing precise algorithms to detect and interpret gestures. The role of ML algorithms in technological progress is mainly reflected in the development and implementation of powerful ML algorithms. Integrating these algorithms into gesture recognition systems can significantly improve the success rate of gesture detection and interpretation.

This research aims to improve the reliability and usability of gesture-controlled smart home automation systems by dealing with lighting changes. Utilizing convolutional neural networks (CNN) is the first choice for gesture recognition projects in smart home automation systems. CNNs are well suited for image-based tasks such as gesture recognition in video data because of their ability to automatically extract and learn complex spatial features from images. CNNs can be effectively applied to your project through a series of key steps, including data preprocessing for gesture isolation, appropriate data representation, and designing a CNN architecture suitable for gesture recognition. By training a model using annotated gesture data, and possibly leveraging transfer learning from a pre-trained model, you can significantly improve its performance. Real-time reasoning and continuous learning mechanisms further ensure adaptability to different lighting conditions and different hand sizes.

By studying the automatic video tuning system, the use of smart environments has become easier. The system will tune captured videos according to different brightnesses. This change has significantly improved the user experience for gesture-controlled smart home automation. One of its main contributions is enhanced convenience. [12] People only need to use body movements and gestures to communicate with remote displays and gadgets. This streamlined interaction brings a seamless feel in controlling the home environment and makes everyday chores easier and more fun. [14] In terms of accessibility, gesture control extends the benefits of smart home technology to individuals with physical disabilities or limitations. Doing so lowers the barrier to entry

and ensures more users can take advantage of home automation. Additionally, gesture-controlled smart home systems can improve energy efficiency. Users can easily adjust lights and other devices with precise gestures to reduce unnecessary energy consumption [17]. AI activity recognition can also help users connect their behavior to their current home devices and then provide recommendations when energy waste is detected. This not only helps save costs but is in line with an eco-conscious lifestyle and increases customer satisfaction by promoting sustainability.

We studied the face recognition system so that in a multi-person environment, the system is able to detect who is the owner of the home and allow the owner to use gestures to control the smart home. This study also compares and compares several models to determine which models are more suitable for this project.

### **1.6 Report Organization**

In the standard structure of this final annual project report, Chapter 1 identifies the problem statement, motivation, project objectives, project scope and direction, and contributions. The problem statement provides a clear explanation of the problem, while the motivation Explains why this problem is worth solving. The project objectives are outlined. what is the project objective that to be achieved. For the project scope and direction describe the boundaries and objectives research project, what included in project scope, and describe scope that didn't covered. This contribution explains the expected results of the study. Chapter 2 introduces previous research on hand gestures and smart home, to know their advantages, disadvantages and limitations, compare them and get the results. And then proposed solutions to the limitations under discussion. explained in Chapter 3. The structure of the entire project and the functions used by the project. Chapter 4 will be Hardware and software specifications and project structure flow. In Chapter 5 explain all the function that used in this system and system operations of coding and system operations, as well as hand gesture recognition methods, and the setting up of the entire project. Chapter 6 will evaluate whether the tuning method used can improve the detection capabilities and the effect capabilities of each model, and Chapter 7 will be the conclusion of the system.

## Chapter 2: Literature Review

### 2.1.1 Using Gestures to Interact with Home Automation Systems: A Socio-Technical Study on Motion Capture Technologies for Smart Homes

Three significant areas are included in the review of related literature in this study: home automation systems, motion capture technologies, gesture interaction, and socio-technical aspects (shown in Figure 1) [11]. In combination, these sections provide an in-depth understanding of the background and context of the research on gesture-based control for home automation systems.

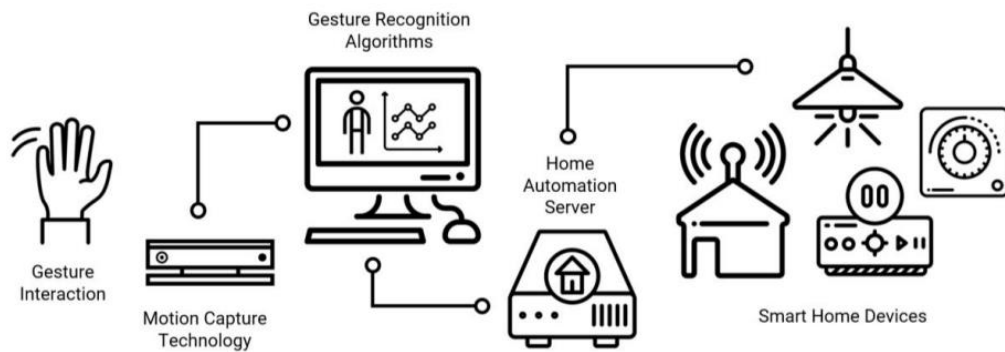


Figure 1: Architecture diagram of the hand gesture interaction for HAS [11]

Building a fundamental understanding of ideas linked to home automation and smart homes is the main goal of the section on home automation systems. It is vital to emphasize that home automation entails integrating a variety of technologies and services into the living area with the goal of automating tasks, improving security, increasing comfort, enabling communication, and managing technical aspects. The idea of a "home automation system (HAS)" is introduced in this section to refer to a wider range of ideas, such as smart houses and home automation. The phrase "smart home technology" is also defined to refer to a variety of devices, such as sensors and actuators, that make it easier to develop and utilize a HAS.

The section on motion capture technology goes into the tools that make it possible to manage home automation systems with gestures. It goes on to how gadgets like the

Microsoft Kinect and LEAP Motion controller have advanced gesture technology. The advancement of microcontrollers, the availability of 3D cameras and depth sensors, machine vision software, and other variables are significant factors in the development of gesture recognition. The two primary categories of gesture control technologies are perceptual and non-perceptual in this section. While non-perceptual technologies require physical touch with the input device or object, perceptual technologies enable gesture recognition without it. However, the study mainly concentrated on perception technologies, with Microsoft Kinect receiving special attention because of its capacity to enable gesture detection without requiring physical touch. Non-aware technology is recognized for its relevance and use with Kinect.

The socio-cultural aspect of gesture-based interaction is presented in the section on Gesture Interaction and Socio-Technical Aspects. It emphasises how vital it is to take social and cultural factors into account when determining meaningful interaction gestures. This section emphasises the necessity for gesture-based interfaces to be acceptable, usable, and culturally relevant while recognizing the difficulty of incorporating technology into daily life. The Technology Acceptance Model (TAM), a framework for measuring users' acceptance of technology, was developed to overcome these problems by emphasising perceived usefulness and usability.

The method is systematic and consists of problem identification, the definition of the requirements, developing the artefacts, observation and feedback, and evaluation. Using Microsoft Kinect as the principal technology for gesture capture, data acquisition, preprocessing, feature extraction, training and test sets, and post-processing using algorithms like AdaBoost Trigger and RFRP Progress, algorithmic and technical implementation details are also provided. In order to apply gesture-based control in home automation systems, a classification model is finally constructed to identify and determine user motions. Hand-gesture dictionary visualised with Kinect shows in Figure 2.



Figure 2: Hand-gesture dictionary visualized with Kinect [11]

### **2.1.2 Hand Gesture Recognition System for Controlling Home Appliances**

This paper proposes an innovative approach to solve a gesture recognition system connected with a webcam. [13] By utilizing camera-based gesture recognition, individuals can use detected and recognized gestures as command signals to control various devices and user interfaces. The technology, designed to support natural hand control, has the potential to change the way people with physical disabilities interact with their surroundings.

The concept of gesture-based remote controls takes center stage in this research, aiming to combine the functionality of numerous home remote controls into a single device. Remote controls have traditionally been used to manage appliances such as TVs, CD players, air conditioners, DVD players and music systems, as well as perform functions such as controlling lights and door openers. The device aims to replace the need for multiple remote controls with a unified control mechanism. Thus, Figure 3 shows the architecture diagram of Hand gesture based remote control.

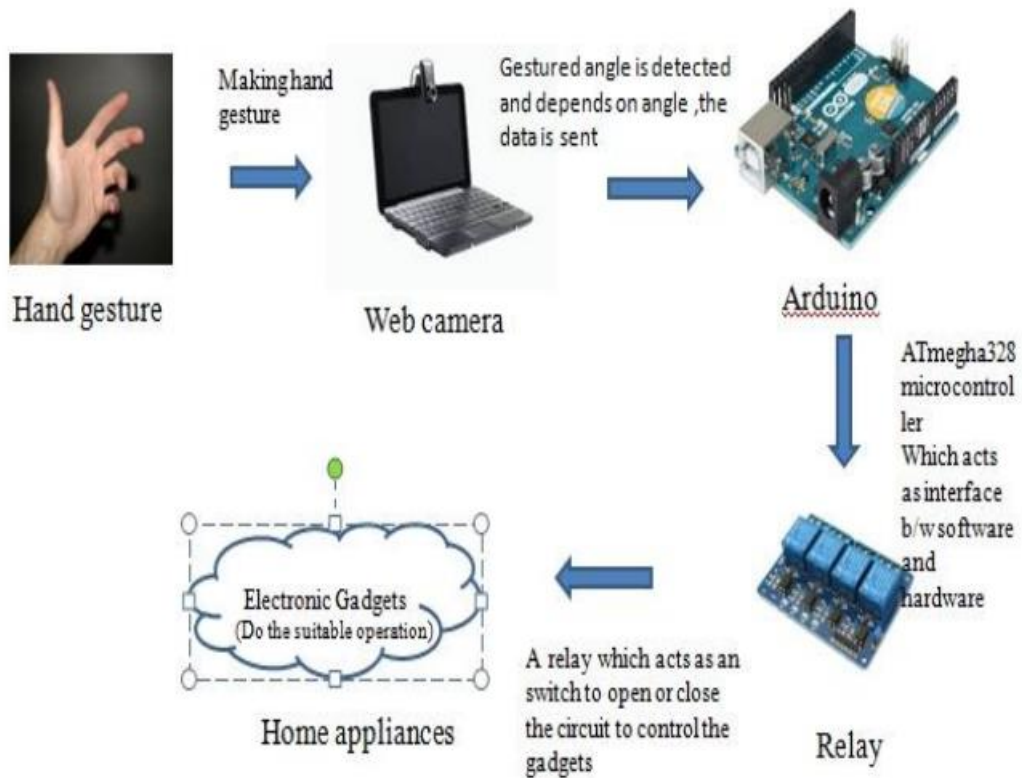


Figure 3: Architecture diagram of Hand gesture based remote control.

The classification and segmentation methods involve RGB color detection, which requires a pixel-by-pixel comparison of color values in the RGB color space. The comparison is made against a threshold used as a filtering parameter. Also, background subtraction emerged as a technique in image processing to distinguish moving objects from stationary ones. The algorithm captures two images at different times and highlights objects that change or move. As a result, Figure 4 shows the result of background subtraction.

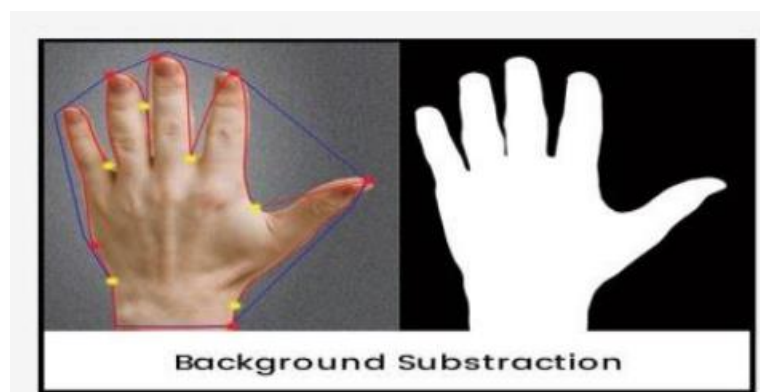


Figure 4: Image segmentation result

The proposed method revolves around an experimental setup aimed at controlling household appliances and electronics through gestures. The core components of this setup include a camera, Arduino Uno with ATmega328, relays, electronics, MATLAB software and enough RAM. The architecture of the system is shown in the figure, illustrating the interaction between the different modules. Four modules were identified as components of the system: gestures, image segmentation, software and hardware interfaces, and electronic gadgets. These modules cooperate to detect gestures, process images, interface with hardware, and control electronics based on detected gestures.

They implement this system with a series of steps. These steps include starting the process, capturing the gesture, calculating the angle of the gesture, relaying that angle to the Arduino, controlling the electronics through a relay, and finally stopping the process.

### **2.1.3 Smart Home Automation-Based Hand Gesture Recognition Using Feature Fusion and Recurrent Neural Network**

This research paper has delved into hand gestures as a more interactive and intuitive approach. [2] This paper explores different methods employed for hand gesture recognition, with a particular focus on dynamic gestures. It also introduces a proposed model for dynamic gesture recognition and discusses the challenges and technologies involved.

In the realm of HCI, conventional approaches utilized inertial sensor-based and vision-based methods for gesture recognition. Inertial sensor-based methods employ arrays of sensors to track hand position, velocity, and acceleration. These motion features are then used for controlling home appliances like TVs, radios, and lights. However, this approach's sensitivity can be a drawback, requiring high-quality sensors for better results, which can increase costs and durability concerns. On the other hand, vision-based approaches rely on cameras to capture RGB and depth images for gesture recognition.

The proposed model in this research paper focuses on dynamic hand gestures, introducing a novel approach to overcome limitations. Videos are initially converted into frames for processing. Image pre-processing steps involve noise reduction using an adaptive median filter and gamma correction for light intensity adjustment. Hand detection is carried out in two phases: skin tone pixel extraction followed by saliency map application. This two-way model enhances precision in hand localization. After hand detection, the system proceeds to identify the hand skeleton, distinguishing between the palm and fingers. A fusion of full-hand features (joint color cloud and neural gas) and point-based features (directionally active model) is extracted for gesture recognition. These features are optimized using an active bee colony algorithm. Finally, a deep learning algorithm, Recurrent Neural Network (RNN), is implemented for gesture classification. The architecture diagram of the proposed system is shown in Figure 5.

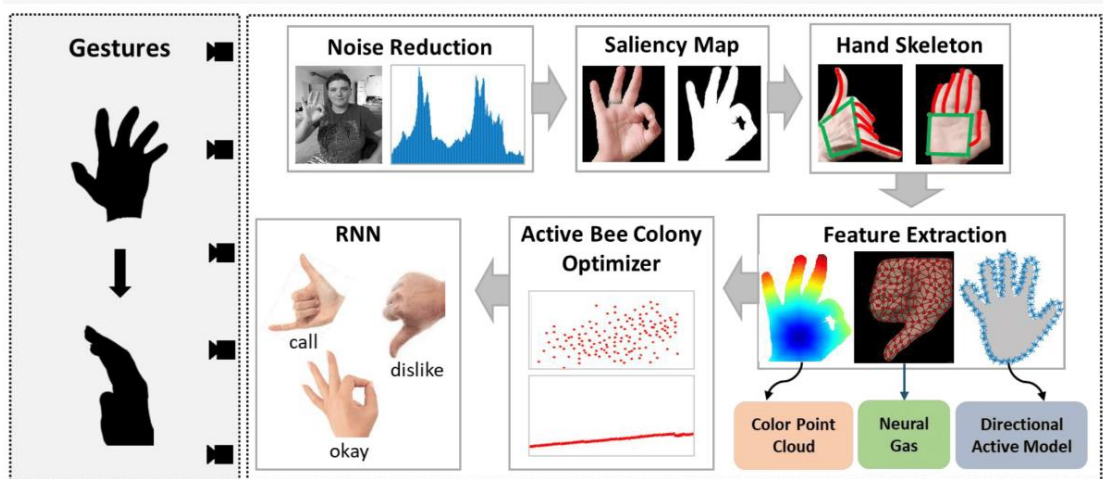


Figure 5: Architecture diagram of the proposed system for hand gesture recognition

[14]





Figure 6 demonstrates the comparison of the accuracy of RNN with other-state-of-the-art algorithms.[2]

This Figure 6 shows that the Hagrid got the highest accuracy compared to others by using (RNN, Random Forest, Multi-layer Perceptron and WLASL ML) method to evaluate the accuracy.

#### **2.1.4 Hand Gesture Recognition for Home Automation**

This research paper presented here delves into the field of gesture recognition, specifically in the context of home automation.[6] It begins by highlighting the critical role of image processing, treating images as two-dimensional signals, and how MATLAB is a versatile tool in this field. Gesture recognition is considered an important means of non-verbal communication, and gestures, especially gestures, can convey a wide range of emotions and information. The paper highlights the importance of this technology in controlling a variety of home appliances and systems, especially for individuals with physical disabilities. MATLAB is positioned as a key tool that facilitates tasks such as matrix manipulation, mathematical calculations, data analysis, and graphical representation.

The paper also discusses the key process of feature extraction, focusing on the use of discrete wavelet transform (DWT) and F-ratio-based feature selection to simplify gesture recognition. Furthermore, it provides insight into existing systems and their high-precision gesture recognition, which ultimately paves the way for the proposed gesture-controlled home automation system using MATLAB. From the respondents, the average rate was 98.50%. In essence, this research paper highlights the transformative potential of gesture recognition, and MATLAB is an important tool in realizing this vision. As a result, Figure 7 shows the result of segmentation of the hand gesture. Besides, Figure 8 is the image processing algorithm.

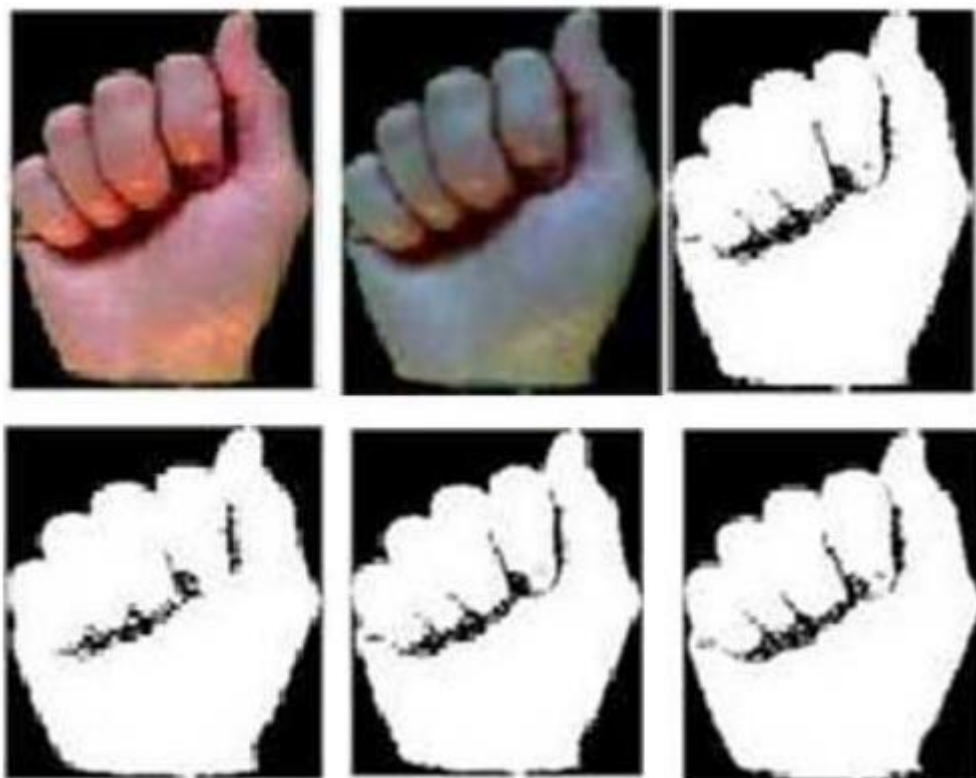


Figure 7: Segmentation result of the hand gesture

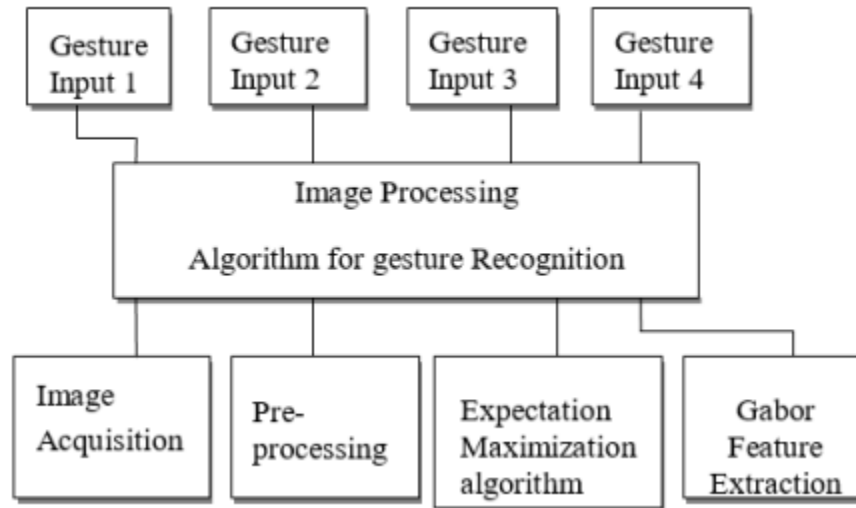


Figure 8: Image processing algorithm [6]

### **2.1.5 Smart and Energy Efficient Gesture Controlled Home Automation**

This research paper discusses a system for intelligent human-computer interaction using gestures in the context of home automation. [15] The literature survey reveals various techniques and methods employed in this field. Several noteworthy methods and components of the proposed system are highlighted, including a gesture recognition system and a home automation component.

**Glove-based American Sign Language Recognition System:** This research paper mentions the development of a glove-based portable communication system for American Sign Language recognition. The system uses multiple sensors and ZigBee communication to transmit gestures to driver circuits to control appliances. It uses feature extraction and artificial neural network (ANN) for gesture recognition combined with a microcontroller (8051) for device control.

**Skin detection and segmentation:** The system utilizes skin-based thresholding in YCbCr color space to identify the presence of hands. The technology marks skin-like pixels in the image, helping to distinguish hands from the background. Additionally, range calculations based on the HSV color model are used to create the boundaries of

skin pixels. Edge detection and morphological operations are employed to facilitate static gesture recognition.

**Color-based preprocessing:** This research paper highlights the importance of color information for gesture recognition. It describes how to use the HSV (hue, saturation, value) color space for preprocessing. The system uses predefined color component ranges for skin detection in HSV space. Grayscale conversion is also part of the preprocessing pipeline.

Is

**Home automation system:** This paper outlines the architecture of a home automation system, emphasizing that its goal is to operate electronic devices without additional input hardware such as gloves or sensors. After successfully recognizing the gesture, the system transmits the code to the microcontroller, which in turn controls the connected home appliance through a relay circuit.

This research paper also mentions the workflow of gesture recognition: A gesture recognition system involves several key steps, including image capture, preprocessing, and skin detection. It relies on a gesture reference database for training. During real-time operation, input gestures are compared to a database to find the closest match, allowing the system to recognize the intended gesture. For reference, Figure 9 is the flowchart of the recognition process and home automation system as well as Figure 10 shows (a) Original image (b) Image obtained after converting the image into HSV colour model (c) Image after converting the (b)image into Grayscale image.

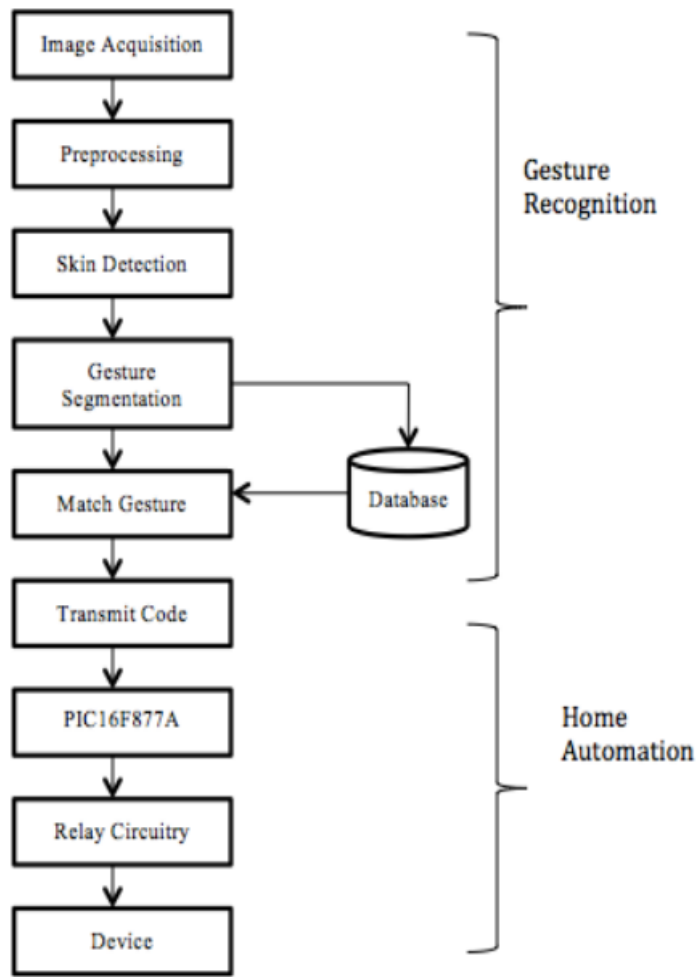


Figure 9: Flowchart of the recognition process and home automation system

[15]



(a)

(b)



(c)

Figure 10: (a) Original image (b) Image obtained after converting the image into HSV color model (c) Image after converting the (b)image into Grayscale image.

[15]

### 2.1.6 Techniques and Challenges of Face Recognition: A Critical Review

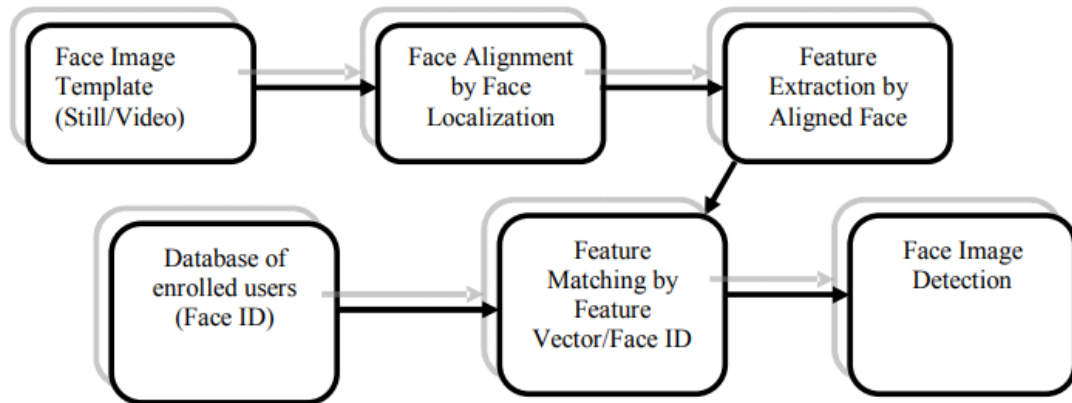


Figure 11: Face detection process [10]

This paper explains that human identification can be achieved through different facial features, such as fingerprints, eyes/iris, body structure, spot markings, etc. [10] However, faces play a crucial role in identifying individuals. Resolution is particularly important in facial recognition systems, especially in surveillance or CCTV applications. The process typically begins with face detection in images, where faces are broken down into key features such as eyes, lips, nose, and mouth for identification. Initially, 2D face recognition involves four steps: face detection, alignment, feature extraction, and matching from a database of registered users. However, this method has limitations such as dependence on image capture conditions, lighting, head orientation, partial occlusion, and facial expressions. To address these limitations, this paper introduces 2D-3D face recognition technology, which combines 2D visual images and 3D models to improve accuracy. These techniques include eigenfaces, stereo vision, and principal component analysis (PCA). Additionally, 3D face recognition minimizes issues related to pose changes, occlusions, and lighting conditions, providing more accurate recognition. Various methods such as local feature descriptors and bidirectional re-illumination are employed to enhance feature extraction and improve recognition rates. The motivation for using 3D face recognition is to overcome the shortcomings of 2D systems and achieve higher accuracy, especially in challenging scenarios.

## **2.2 Limitation of Previous Works**

### **2.2.1 Limitation of Previous Studies**

This paper does not delve into the range and complexity of gestures that can be recognized. Having a rich gesture vocabulary that is intuitive and covers a wide range of commands is essential for a practical home automation system. Limiting gestures may limit user interaction.

### **2.2.2 Limitation of Previous Studies**

Previous studies did not cover the recognition of complex hand-gestures. The system's ability to recognize complex and detailed gestures may be limited. While the system may be good at detecting basic gestures, it may struggle to accurately recognize more complex gestures. [9] The model parameters specified by skeleton-based recognition can enhance the detection of complex features. In order to concentrate on geometric and statistical features, it describes geometric attributes and constraints and easily translates features and correlations of data where the various representations of skeleton data for the hand model can be used for classification.

The second limitation is the changing lighting conditions can affect the accuracy of the system. [10] The effectiveness of hand motion recognition sensors can be considerably impacted by lighting. For instance, the sensor might not be able to catch the hand gesture precisely in low light. Changes in ambient lighting can cause changes in color and contrast, which can affect the performance of RGB color-based gesture recognition.

### **2.2.3 Limitation of Previous Studies**

The challenges in this research paper are light intensity, clutter sensitivity, and skin color. Hand localization is a critical step.

### **2.2.4 Limitation of Previous Studies**

Gesture variability: People can gesture in a variety of ways, causing the same gesture to appear differently in different situations. This variability can make it difficult for systems to accurately generalise and recognize gestures.

### **2.2.5 Limitation of Previous Studies**

One of the grand challenges of this research paper is to achieve robust and accurate gesture recognition under different lighting conditions. Different lighting conditions



can affect the appearance and skin tone of the hand, making it difficult to consistently detect and recognize gestures.

### **2.2.6 Limitation of Previous Studies**

The limitation faced by this article is that when performing face recognition, faces will change over time due to the influence of age, making it difficult to accurately identify individuals for a long time. The second is that thermal images have different characteristics and require suitable technology to accurately recognize faces. What follows is that accurate extraction and recognition of IRIS features is crucial for facial biometric analysis. Finally, there are facial expressions, which can vary greatly and affect facial recognition accuracy.

## **2.3 Proposed Solutions**

### **2.3.1 Proposed Solutions**

Allows users to customize or define their own gestures for specific commands. This is made possible through a user-friendly interface where users can record and assign gestures to desired actions. This way, users can create gestures that are most suitable to them.

### **2.3.2 Proposed Solutions**

The way to address this limitation is to incorporate with others ML techniques for example (CNN). By training the system with different gesture datasets, it can get better at recognizing a variety of gestures, including complex ones.

To overcome the second limitation, adaptive color thresholding techniques can be explored. These technologies dynamically adjust color thresholds based on current lighting conditions, ensuring consistent recognition in different environments.

### **2.3.3 Proposed Solutions**

To address the challenge, the researcher let the conventional systems be divided into different steps to achieve better accuracy while taking the challenges into account. First, data acquisition is performed, followed by hand detection. For hand detection, a variety of methods are used, including segmentation, tracking, and color-based extraction. Features are extracted using different algorithms. After that, the gesture is

recognized. For a given method, images and videos are collected. Static images provide static gestures, while videos provide dynamic gestures, as changes in gestures from one frame to the next are noticed. Static gestures are still images that require less computational cost.

### **2.3.4 Proposed Solutions**

Employing ML techniques, such as deep learning with recurrent neural networks (RNN), can enhance the system's ability to recognize gestures by learning from different training examples. RNNs are able to capture temporal dependencies in gesture sequences, which helps to handle variability.

### **2.3.5 Proposed Solutions**

To address this challenge, research papers can implement adaptive lighting correction techniques. These technologies can automatically adjust the brightness and contrast of the image to ensure consistent lighting conditions for gesture recognition. Additionally, the use of multispectral or depth-sensing cameras such as infrared or time-of-flight cameras can provide more reliable data for gesture recognition, as they are less affected by changes in visible light. Furthermore, a comprehensive dataset containing various lighting conditions for training the gesture recognition model will help improve its robustness in different scenarios.

### **2.3.6 Proposed Solutions**

The proposed solution to the aging problem is to use techniques such as Active Appearance Model (AAM), Coupled Autoencoder Network (CAN), nonlinear analysis, and sparse constraint methods for face recognition with aging factors. These methods focus on reducing image dimensionality, analyzing aging effects, and improving feature recognition performance.

To solve the problem of thermal image recognition, we use multi-feature extraction technology, Gabor jet descriptor, fusion algorithm and entropy analysis for thermal image recognition. These methods aim to improve the accuracy, recognition rate, and robustness of thermal face recognition systems. To solve the iris recognition problem, this article discusses the use of bihashing technology, mobile engagement systems that combine face and iris recognition, and multi-biometric systems to achieve accurate iris recognition. Iris recognition. To address the facial expression problem, expression-

## CHAPTER 2

invariant 3D face recognition, active shape models, sparse representation classification (SRC), collaborative representation classification (CRC), and facial expression analysis techniques are discussed. These methods focus on accurately recognizing faces despite differences in facial expressions.

**Table 1: Comparison of 5 literature**

	[11]	[13]	[2]	[6]	[15]
Accuracy	No mention	No mention	92.57%	98.50%	No mention
Deep learning model	Deep learning models in neural networks	No mention	RNN	HMM (Hidden Markov Model)/Random Forest	ANN
Capture Gesture	Microsoft Kinect	OpenCV	Computer vision	NI Vision	Computer Vision
Software	Microsoft Kinect	MATLAB	Computer vision	MATLAB	MATLAB
Hardware	3D cameras and XBOX Kinect sensors	Web camera, Arduino, Relay	RGB Sensor	Remote Control, web camera and microcontroller	Digital camera, microcontroller, relay driver ULN2003A, sensors

## Chapter 3: System Methodology/Approach

### 3.1 System Design Diagram/Equation

#### 3.1.1 System Architecture Diagram

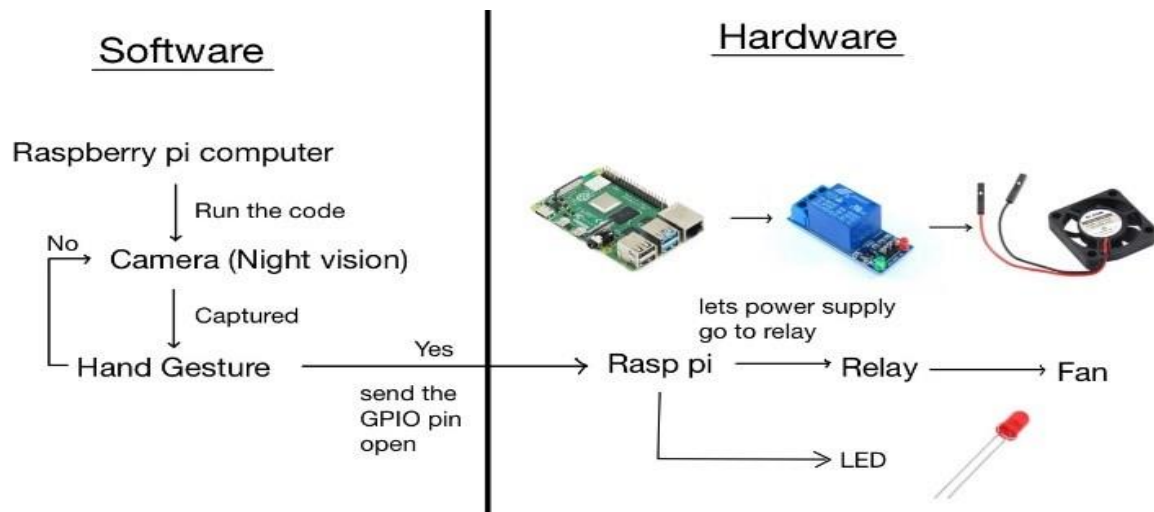


Figure 12: Smart home architecture diagram

In this project Raspberry pi multiprocessor and multicontroller are used to control the entire system. First, use the Raspberry Pi Thonny programming application to capture the gestures using mediapipe and opencv. When the gesture is detected, the system will automatically distinguish the gesture, when the open gesture is displayed, the relay will let the power pass to the fan. If display the close gestures, the relay will block the power. If display the pointer gestures, the LED light will turn on, if display OK gestures, the LED light will turn off.

3.1.2 Use Case Diagram and Description

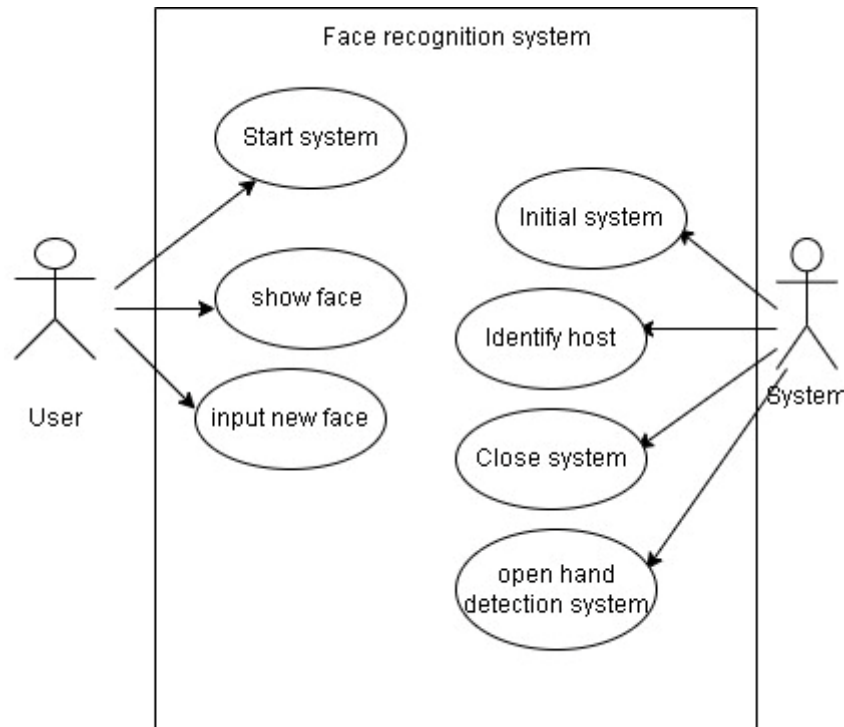


Figure 13: Use case diagram for Face Recognition System

<b>Use Case Name:</b> Face Recognition	<b>ID:</b> 1	<b>Importance Level:</b> High
<b>Primary Actor:</b> User	<b>Use Case Type:</b> Detail, Essential	
<b>Stakeholders and Interests:</b> User: Scan or record the face recognition input		
<b>Brief Description:</b> This use case outlines all procedures of user doing the face recognition system		
<b>Trigger:</b> User want to show the face to CCTV to show the user is a host in the house.		
<b>Type:</b> External		
<b>Relationships</b>		
<b>Association:</b> User		
<b>Include:</b> Null		
<b>Extend:</b> Null		
<b>Generalization:</b> Null		
<b>Normal Flow of Events:</b>		
<ol style="list-style-type: none"> <li><b>The user start the face recognition system.</b></li> </ol>		

<ol style="list-style-type: none"> <li>2. The user gives face input.</li> <li>3. The system Initial system, open the frame to capture the face.</li> <li>4. The system can Identify Host or unknown.</li> <li>5. The system can open hand detection system automatically.</li> </ol>
<b>SubFlows:</b>
<b>Alternate/ Exceptional Flows: Input the unknown face record.</b>
<b>Not Applicable.</b>

Table 3.1.2.1 Use case description for face recognition

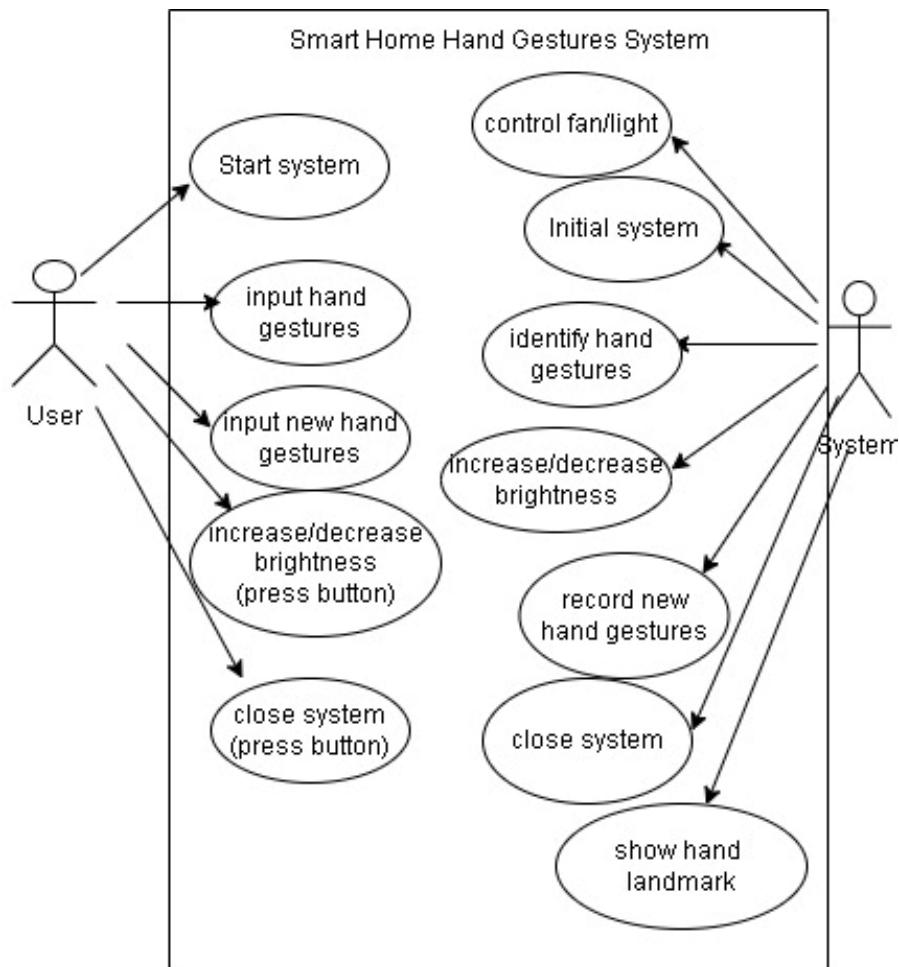


Figure 14: Use case diagram for Face Recognition System

<b>Use Case Name:</b> Hand Gestures Recognition	<b>ID:</b> 2	<b>Importance Level:</b> High
<b>Primary Actor:</b> User	<b>Use Case Type:</b> Detail, Essential	

<b>Stakeholders and Interests:</b> User: Control smart home using hand gestures recognition
<b>Brief Description:</b> This use case outlines all procedures of user control the smart home using the hand-gestures
<b>Trigger:</b> Users want to control the fan or light using the hand gestures recognition <b>Type:</b> External
<b>Relationships</b> <b>Association:</b> User <b>Include:</b> Null <b>Extend:</b> Null <b>Generalization:</b> Null
<b>Normal Flow of Events:</b> <ol style="list-style-type: none"> <li>1. The user starts the hand gestures recognition system.</li> <li>2. The user gives hand gestures input.</li> <li>3. The user can control the brightness of the real time video.</li> <li>4. The user can input the new hand gestures record.</li> <li>5. The user can close the program.</li> <li>6. The system Initial system, open the frame to capture the hand gestures.</li> <li>7. The system can Identify Hand gestures.</li> <li>8. The system can control the fan or light.</li> <li>9. The system can increase/ decrease the brightness.</li> <li>10. The system can record the new hand gestures.</li> <li>11. The system can close the frame.</li> <li>12. The system can show the hand-gestures landmark.</li> </ol>
<b>SubFlows:</b>
<b>Alternate/ Exceptional Flows:</b> Input the unknown hand gestures record. <b>Not Applicable.</b>

Table 3.1.2.2 Use case description for face recognition



3.1.3 Activity Diagram

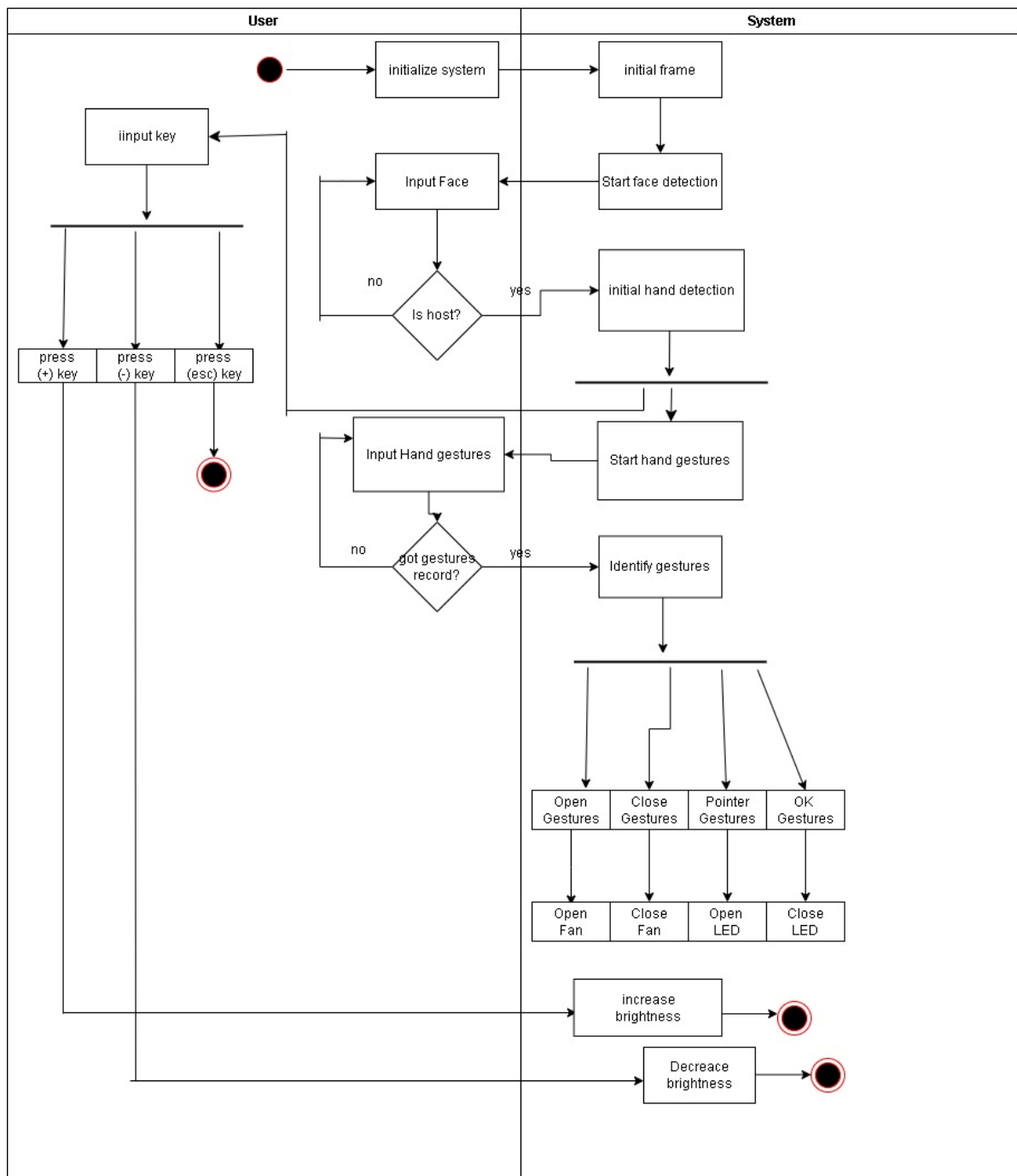


Figure 15: Activity diagram of whole system

This activity diagram shows the entire system. First, the user needs to start the system. Then the system will initial frame, which is, turn on the camera and then start face recognition. If the system detects that it is the Host, the system will directly start hand gesture detection. Else, the

system will continue to detect until the host is detected. When the hand gesture system starts, the user can give the gesture. If the system detects that the gesture is in the record, the system will open fan based on the gesture you had provided like close fan and more functions. If no gesture is detected, the system will always wait for the host to give instructions. When the hand gestures recognition system is turned on, the user can give instructions according to the keyboard to reduce or increase the brightness of the video, and or roll out this system.

### 3.2 Hand Gesture Recognition Method

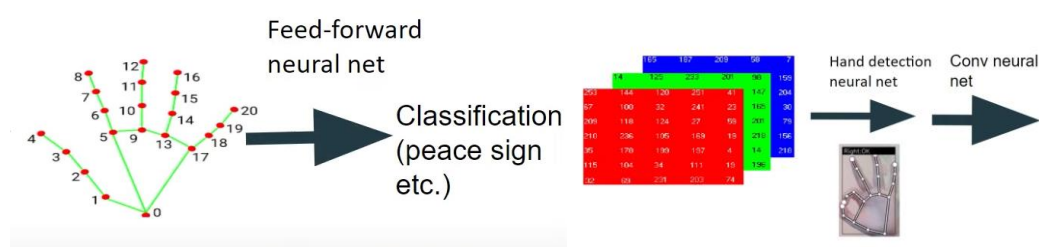


Figure 16: Hand Gesture Recognition Method

OpenCV was then added to help capture clearer gestures. First, we get the frames from the camera for grabbing or get the RGB matrix, which contains a lot of color information of the image we captured, next, we pass these frames from the camera to the hand detection model, and then the gesture is passed to the Conv neural network to extract the hand landmarks and finally classified only through the feedforward neural network. During all of this, the Mediapipe framework will be the tool that helps capture gestures.

### 3.3 Hand Gestures Recognition Model

#### 3.3.1 CNN Recognition Model

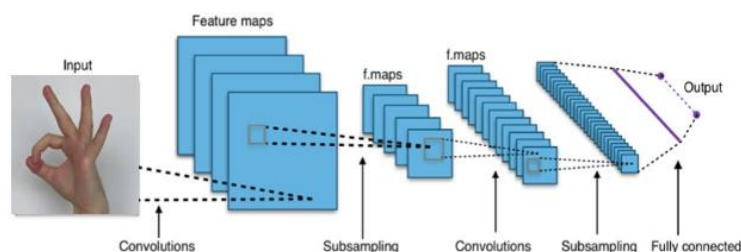


Figure 17: Machine learning model

## CHAPTER 3

Convolutional neural networks (CNN) are a specialized deep learning architecture that is particularly good at image-related tasks and is well suited for gesture detection in smart home automation systems. Here's an overview of how to use a CNN to achieve this:

**Input data preparation:** CNN takes images or frames from a camera as input. These images can be preprocessed to extract regions of interest (hands) or resize them into a specific format required by the network.

**Architecture design:** CNN architecture usually contains convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply filters on the input image to extract features, gradually learning more complex patterns as information passes through the network. Pooling layers reduce spatial dimensions, improve computational efficiency, and create more abstract image representations.

**Training using labeled data:** To train a CNN for gesture recognition, a labeled dataset containing various gestures is required. Each image or frame is labeled with a corresponding gesture. CNN learns to distinguish and classify these gestures through a supervised learning process.

**Feature Learning:** When a CNN is trained, it automatically learns the features necessary to distinguish between different gestures. These features may include contours, shapes, locations, or relationships between key points on the hand.

**Optimization and Validation:** The parameters of the CNN are optimized via backpropagation and gradient descent methods to minimize the difference between predicted and actual gestures. The model's performance is evaluated on a separate validation dataset to ensure that it generalizes well to unseen data.

**Deployment and real-time inference:** Once trained, CNN models can be deployed on a Raspberry Pi or other computing device in a smart home system. It analyzes incoming video frames in real time, recognizes gestures based on learned patterns and triggers corresponding actions in the home automation system.

This CNN-based approach enables the system to accurately recognize and interpret various gestures, allowing users to interact seamlessly with smart home devices. Adjusting the network architecture, expanding the data set, or fine-tuning the parameters can enhance the accuracy and robustness of the CNN in recognizing gestures under different environmental conditions.

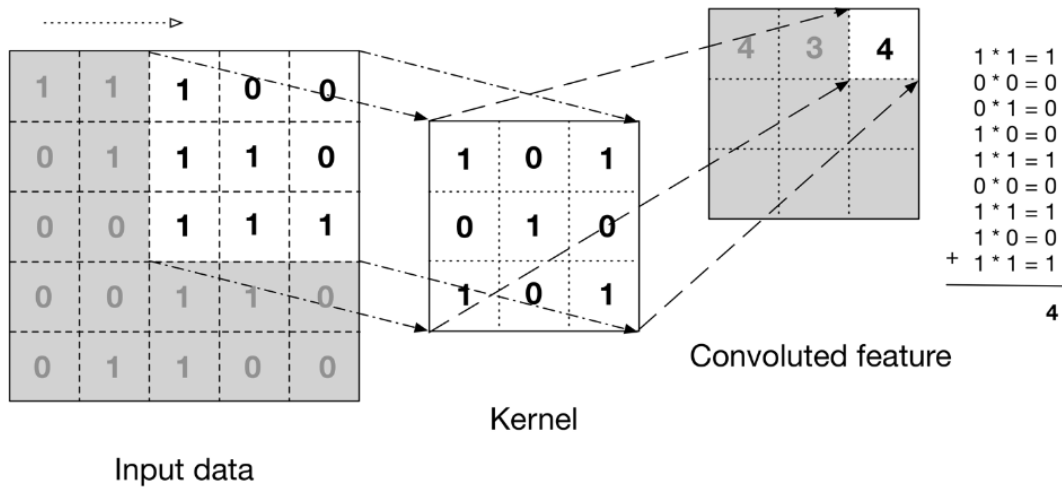


Figure 18: RGB Matrix for CNN

This image shows what a convolution is. Take a filter/kernel (3x3 matrix) and apply it to the input image to get the convolved feature. This convolved feature is passed on to the next layer.

### 3.3.2 Random Forest Model

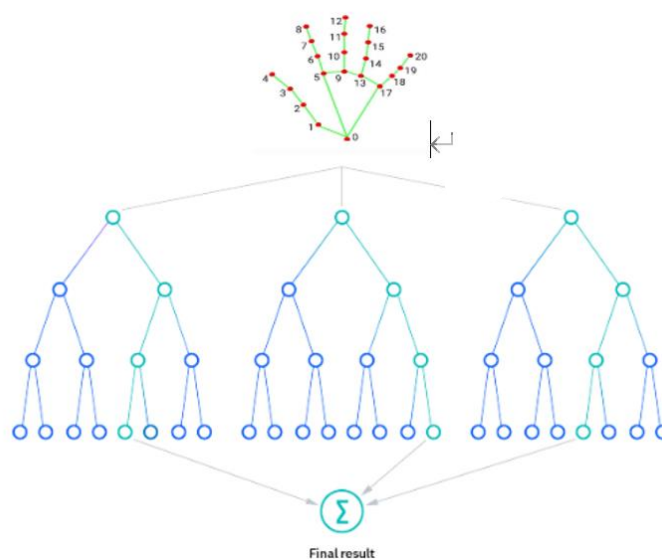


Figure 19: Random Forest model architecture

Hand gesture recognition using random forests involves several steps. First, a dataset of gesture images is collected and labeled with corresponding gestures (e.g. fist, open palm, etc.). These images are then preprocessed, which typically involves resizing and converting them into a consistent format suitable for analysis.

Next, features are extracted from these preprocessed images. These features can include various aspects such as the shape of the hand, the distribution of pixels within the hand area, the location of key points such as fingertips, and other relevant features. The goal is to capture meaningful information that distinguishes different gestures.

The random forest model itself is an ensemble learning technique that combines multiple decision trees. Each tree in a random forest is trained on a random subset of features in the dataset. During training, the tree learns to make predictions based on these subsets of features. The final prediction of a random forest model is usually determined by aggregating the predictions of individual trees, often using techniques such as averaging or voting.

After training the random forest model on the prepared dataset, its performance is evaluated on a separate dataset to assess how well it generalizes to unseen gestures. This evaluation step helps ensure that the model can accurately classify gestures outside of the training set, demonstrating its robustness and effectiveness.

Once the random forest model is trained and validated, it can be deployed on devices such as the Raspberry Pi for real-time gesture recognition. During this deployment phase, the model analyzes video frames captured in real time, extracting features from these frames using the same techniques applied during training. The extracted features are then compared with the learned patterns stored in the model. Based on this comparison, the model can recognize specific gestures and trigger corresponding actions in the smart home system, such as adjusting lighting, controlling devices, or activating specific functions.

Overall, the random forest approach provides a simpler and potentially less computationally intensive approach to gesture recognition than some other deep learning techniques. It leverages ensemble learning and feature-based analysis to achieve accurate gesture classification and enable seamless interaction with smart home systems.

### 3.3.3 NN model (Neurol Network)

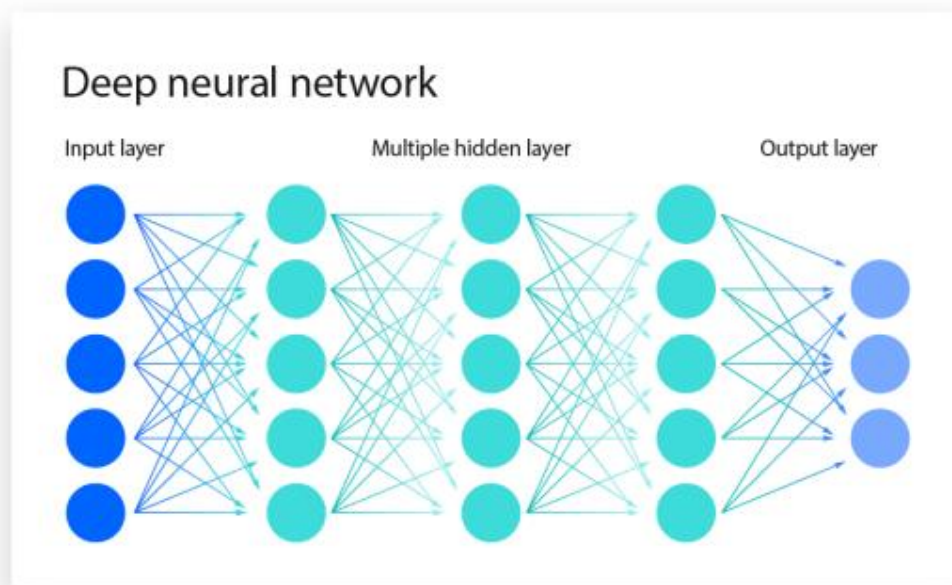


Figure 20: Neural networks model architecture

Neural networks (NN) are suitable for gesture recognition because of their ability to learn complex patterns and make accurate predictions. A typical neural network consists of interconnected nodes, or neurons, organized into layers: input layer, hidden layer, and output layer. The input layer receives initial data or features representing the gesture. Each node in this layer corresponds to a specific feature extracted from the gesture image, such as the shape of the hand or the pixel intensity distribution. Hidden layers are intermediate layers that perform computations on these input features through weighted connections and activation functions such as ReLU or Sigmoid. These calculations enable neural networks to learn complex relationships and representations of gestures. Deep neural networks with multiple hidden layers can capture the hierarchical structure of features, making them good at recognizing complex gestures. Finally, the output layer generates predictions based on the learned patterns, classifying gestures into predefined categories such as fist, open palm, or peace gesture. The neural network is trained using labeled gesture images, adjusting its internal parameters (weights and biases) to minimize prediction errors during training. After training, the neural network can accurately recognize gestures in real-time applications, allowing intuitive interaction with smart devices or systems based on the detected gestures.

## 3.4 Flowchart

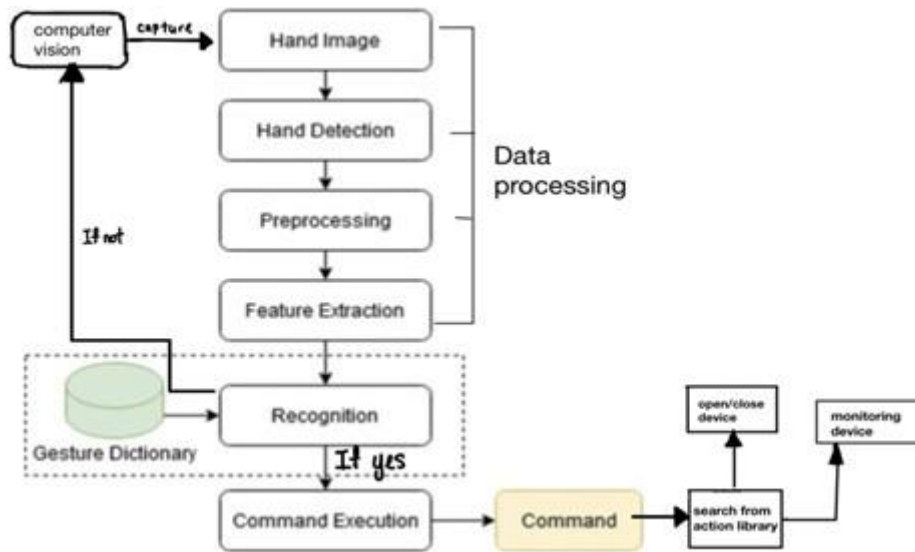


Figure 21: Flowchart of Hand-gestures recognition system

First, use thonny programming in raspberry pi to use computer vision to capture hand gestures. Then the system will first detect the hand and then perform preprocessing. This is for Improvement of image data that suppresses undesirable distortion or enhances certain image features important for further processing, and then perform feature extraction. Next, the system will check from gesture dictionary to see what gesture you are doing, and then give instructions to the raspberry pi based on your gesture, whether it should be a switch or a control.

**3.5 Function to Extract RGB Value of The Environment.**

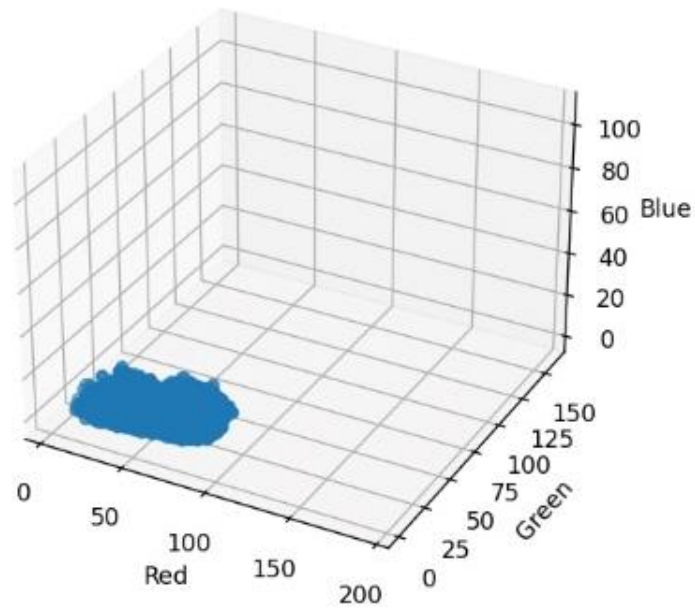


Figure 22: Colored Scatter Plot in Dark Environment

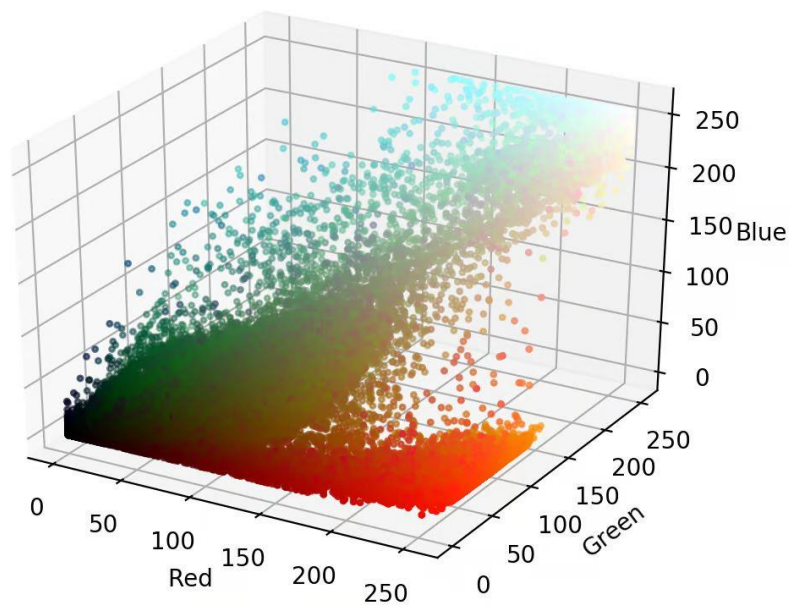


Figure 23: Colored Scatter Plot in Normal Environment [20]



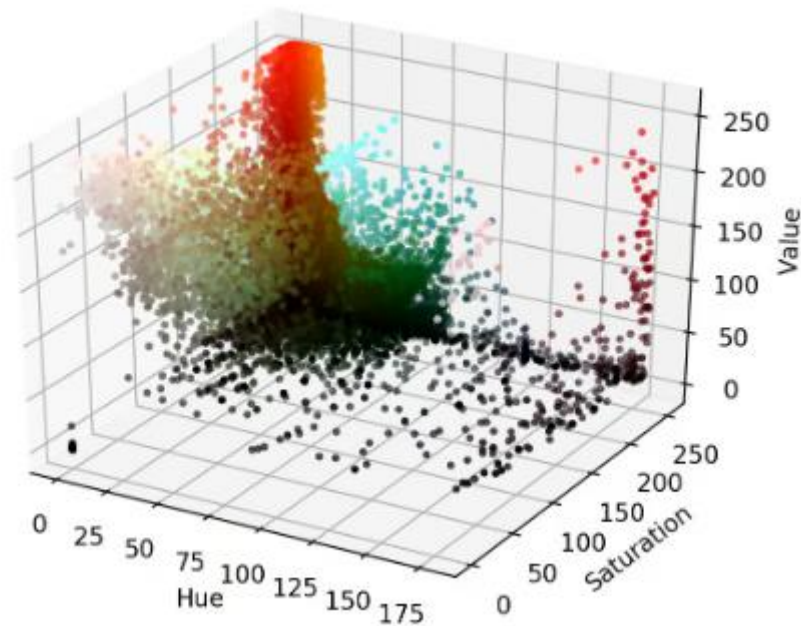


Figure 24: HSV Scatter Plot in Normal Environment [20]

These three images above show how to extract RGB and HSV from a frame. These RGB colors are used for adjustments in subsequent parts of the project. RGB represents color using red, green, and blue components, each ranging from 0 to 255. Meanwhile, HSV stands for hue, saturation, and value/lightness. Hue determines how light or dark a color is, saturation refers to its intensity and value/brightness indicates the color's brightness level.

In the RGB method, the colors of the video are converted from BGR to RGB. The RGB values are rescaled into a list, and a 3D scatter plot is created to visualize the RGB components. However, during testing we found that separate RGB extraction was better suited for this project due to its ability to adjust brightness based on color changes.

HSV conversion, on the other hand, involves converting raw BGR frames into HSV format. Extract hue, saturation and value/luminance channels for analysis. While HSV is beneficial for colorful scenes where brightness is affected by color, we determined that using a specific method for grayscale conversion and average brightness calculation was more effective for solving the lighting issues in this project. Grayscale conversion more closely approximates darker colors, allowing precise brightness calculations and subsequent adjustments to captured video frames.

## CHAPTER 3

By implementing these color space technologies and adjusting brightness accordingly, the system ensures accurate gesture recognition regardless of lighting conditions.

### 3.6 Gantt Chart

Project Task	Project Week				
	Week 2	Week 3	Week 4	Week 5	Week 6
<b>Chapter 1: Introduction</b>					
Problem Statement and Motivation		17/11/2023			
Research Objectives		18/11/2023			
Project Scopes and Direction			21/11/2023		
Contributions			22/11/2023		
Report Organization			24/11/2023		
<b>Chapter 2: Literature Review</b>					
Finding Related and Relevant etc, Papers, blogs, news, books	6/11/2023				
Study resources	7/11/2023				
Limitations of Studies	8/11/2023				
Proposed Solutions	8/11/2023				
LR table comparison	10/11/2023				
<b>Chapter 3: Proposed Method / Approach</b>					
System Requirement			25/11/2023		
Application Diagram			26/11/2023		
System Architecture Design				27/11/2023	
Hand Gesture Recognition Method/Machine Learning Model				28/11/2023	
Challenge				28/11/2023	
Flowchart				29/11/2023	
Timeline (Gantt Chart)				30/11/2023	
<b>Chapter 4: Preliminary Work</b>					
Proposed method - system design, architecture, framework				30/11/2023	
Hardware/Software involved				1/12/2023	
Code Explanation				2/12/2023	
Others: UML diagram				3/12/2023	
<b>Report Checking</b>					
Draft Checking					7/12/2023
<b>Chapter 5: Conclusion</b>					
Conclusion					6/12/2023
Final Report Submission					8/12/2023

Figure 25: Gantt Chart for FYP1

CHAPTER 3

Project Task	Project Week					Project Week						
	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
<b>Development Process</b>												
Add vocabulary of gestures	█											
Hand gestures model		█										
3D scatter plot to retrieve brightness value			█									
Auto/manual tuning				█								
Face detection model					█							
<b>Chapter 5: Experiment / Simulation</b>												
Setup report work						█						
Coding report work							█					
<b>Chapter 6: System Testing and Discussion</b>												
System Testing and performance metrics								█				
Testing result									█			
Objectives Evaluation										█		
Challenge Faced											█	
<b>Chapter 7: Conclusion</b>												
Conclusion											█	
Summarize overall report											█	
<b>Report Checking</b>												
Draft Checking												█
Final Report Submission												█

Figure 26: Gantt Chart for FYP2

## Chapter 4: System Design

### 4.1 System Block Diagram

#### 4.1.1 Hand Gesture Recognition

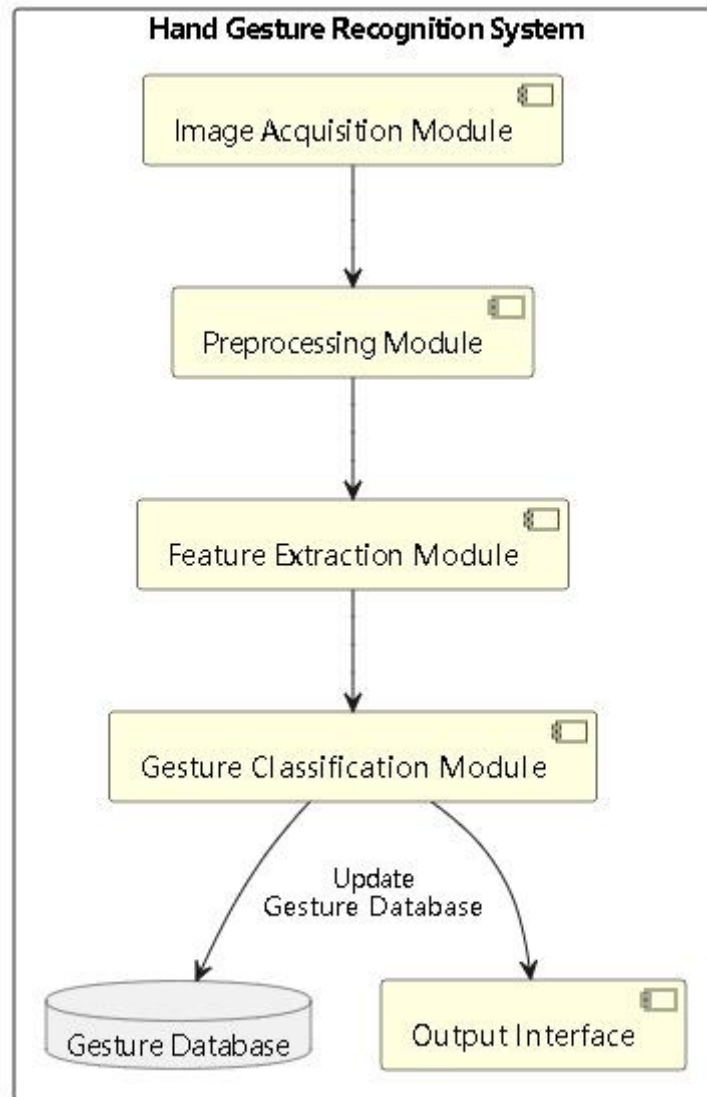


Figure 27: Block Diagram of hand gestures recognition system

The gesture recognition system block diagram illustrates the various interconnected components and their role in recognizing gestures. At the heart of the system is a gesture database, which stores a set of predefined gesture patterns for comparison during the recognition process. The system starts with the Image Acquisition Module (IAM), which is

responsible for capturing images or video frames containing gestures. These raw input data are then passed to the preprocessing module (PM), which performs basic tasks such as noise reduction, image enhancement, normalization and segmentation. The preprocessed data is then fed into the Feature Extraction Module (FEM), which extracts relevant features from the input data. These features can include shape, texture, color, motion or spatial information characteristics of different gestures.

The extracted features are then analyzed by the Gesture Classification Module (GCM) using machine learning algorithms, pattern recognition techniques or rule-based systems. GCM compares the extracted features with the patterns stored in the gesture database to determine the corresponding gesture. If there is necessary, GCM able to update the gesture database with the new gesture information. Once a gesture is recognized, an Output Interface (OI) displays the results to the user or other system. This output can be in the form of a graphical representation which recognized the gesture on the GUI, provide feedback or trigger an action based on the recognized gesture.

The connections between components illustrate the flow of data and control within the system. IAM provides raw data to PM for preprocessing and then PM passes the processed data to FEM for feature extraction. FEM provides the extracted features to GCM for gesture classification and GCM updates the gesture database and sends the recognized gesture information to OI for display. Overall, this block diagram outlines a systematic approach to capturing, processing, extracting features, classifying and displaying gestures in a recognition system.

### 4.1.2 Face Recognition

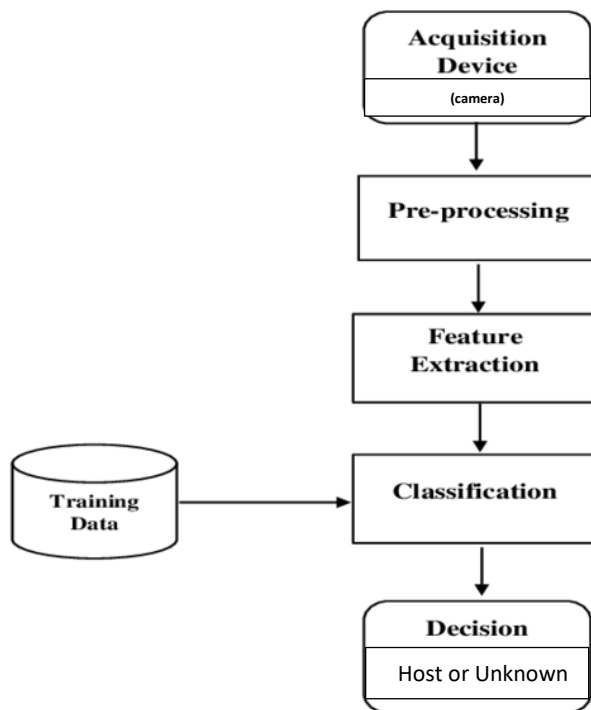


Figure 28: Block Diagram of face recognition system

The face recognition process begins with a capture device (usually a camera) that captures an image or video frame containing a human face. The equipment must be in a high quality to ensure facial images are clear and detailed without distortion or occlusion. Once the image is obtained, it is preprocessed, which involves various techniques such as noise reduction, image enhancement and normalization. Noise reduction removes unwanted artifacts, while image enhancement improves clarity and quality and standardization ensures consistency in attributes such as size and orientation for accurate identification.

After preprocessing, the next step is featuring extraction, which extracts relevant facial features from the image, such as facial shape, key landmarks (eyes, nose, mouth), texture patterns and other unique features. Feature extraction algorithms analyze image data to create compact representations of these features suitable for comparison and identification.

The extracted features are then fed into the classification algorithm. The algorithm compares features of the input face to features stored in training data or a database. This database contains

pre-learned patterns or templates of known faces and their corresponding identities. Classification algorithms assign a likelihood score, or similarity measure, between the input face and each face in the database to determine the person's identity.

Finally, based on the output of the classification algorithm, the system decides to select the user as a host or an unknown user. If the similarity score exceeds a certain threshold and matches a known face in the database, the system identifies that person as a moderator or registered user. This allows the system to trigger specific actions or provide access based on the user's identity. If the similarity score is below the threshold or does not match any known face, the system identifies the person as unknown or unauthorized and takes appropriate action based on system settings or security protocols.

## 4.2 System Components Specifications

### 4.2.1 Hardware

The hardware involved in this project is computer, raspberry pi 4 model b, relay module, raspberry pi camera module 2, 5v fan. First, use a computer to remotely control the raspberry pi by using a VNC viewer to run the entire system. The camera is used to capture gestures, and the relay module is used to control or switch the output power to ensure that the fan will not break down due to excessive power. The fan is one of the devices of a smart home.

Table 4.2.1 and 4.2.2 illustrated specification of laptop and hardware used.

Description	Specifications
Model	Asus TUF-gaming A15
Processor	Intel Core i5-12100U
Operating System	Windows 11
Graphic	NVIDIA GeForce RTX 1080
Memory	8GB DDR4 RAM

Storage	1TB SATA HDD
---------	--------------

Table 4.2.1 Specifications of laptop






Hardware Components	Descriptions
	Raspberry Pi 4model b acts as slave microcontroller and multiprocessor in this project.It controls all programs for the project By using hardware communication methods.
	Raspberry Pi camera module 2, a camera to capture all the hand gestures.
	Relay modules control the supply of electricity
	5V fan
	Led light

Table 4.2.2 Specifications of hardware



## 4.2.2 Software


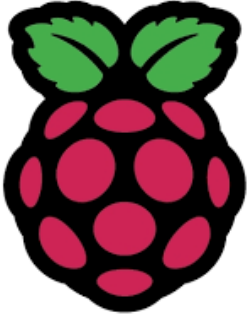
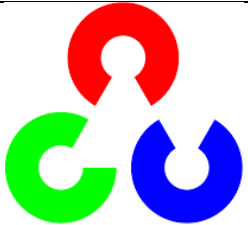

Software	Description
 <p>Python</p>	<p>Version: 3.12</p> <p>Purpose : For test the code</p>
 <p>Raspberry pi</p>	<p>Version: 6.1</p> <p>Purpose : To run the system in raspberry pi 4b.</p>
 <p>OpenCV</p>	<p>Version: 4.9.0</p> <p>Purpose: To capture and tune the quality of real time hand gestures video.</p>
 <p>Teachable Machine</p>	<p>Version:2.0</p> <p>Purpose: To make the face recognition model</p>

Table 4.2.3: Specifications of software

### 4.3 Circuit and Component Design

#### 4.3.1 System Circuit Diagram

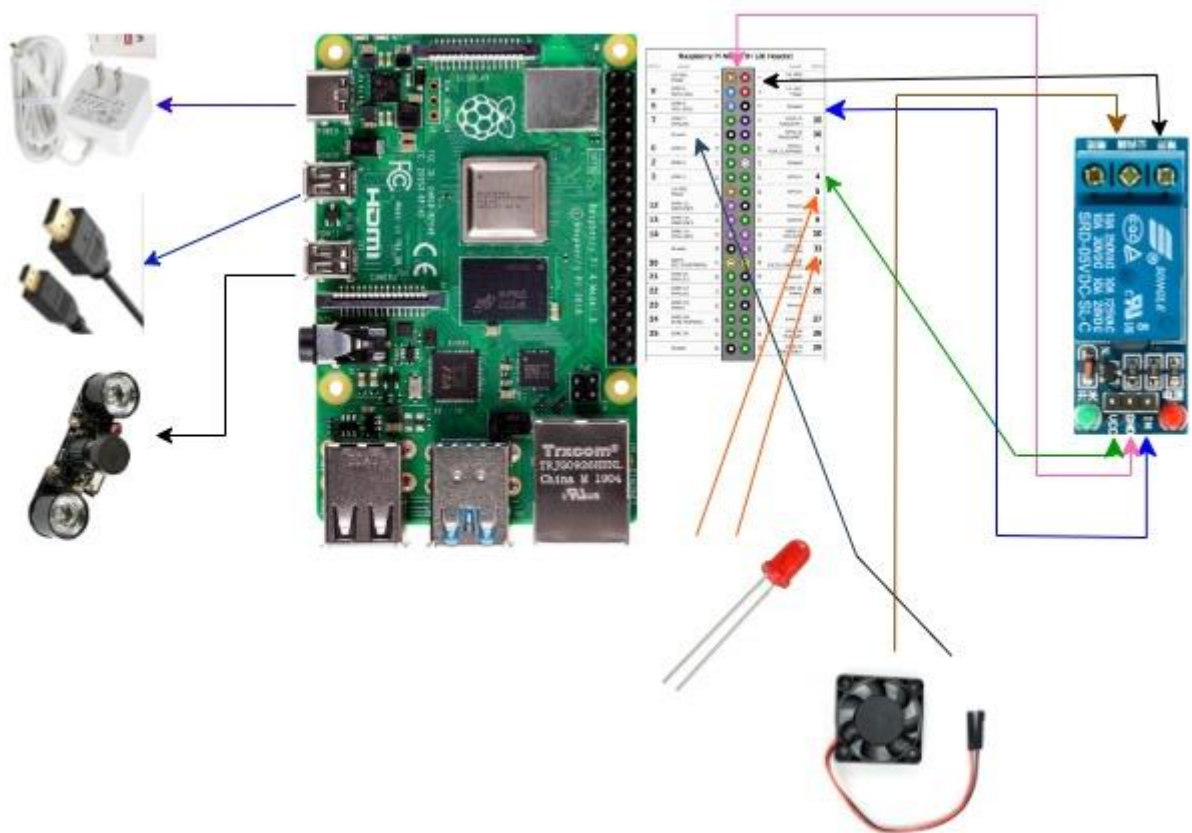


Figure 29: Circuit diagram of the project

Raspberry Pi model b	Pin	Component	Pin
	3.3V DC Power	relay module -> <-Raspberry pi (+)	5V power
	GPIO 2		5V power
	GPIO 3	raspberry pi (-) ->	Ground
	GPIO 4		GPIO 14
	Ground	<- fan (-)	GPIO 15
	GPIO 17		GPIO 18
	GPIO 27		Ground
	GPIO 22	Relay signal ->	GPIO 23
	3v3 power		GPIO 24
	GPIO 10		Ground

GPIO 9	Led signal à	GPIO 25
GPIO 11		GPIO 8
Ground		GPIO 7
GPIO 0		GPIO 1
GPIO 5		Ground
GPIO 6		GPIO 12
GPIO 13		Ground
GPIO19		GPIO 16
GPIO26		GPIO 20
Ground	βled (-)	GPIO 21

Table 4.3.1: Mapping relationship between components.

In each system. It lists components such as Raspberry Pi 4 model b, relay model, Raspberry pi camera module 2 and 5V fan. And then map all the pins connected.

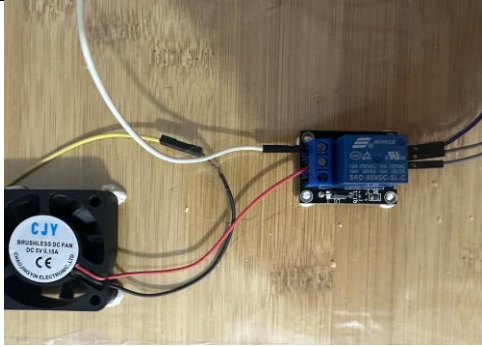
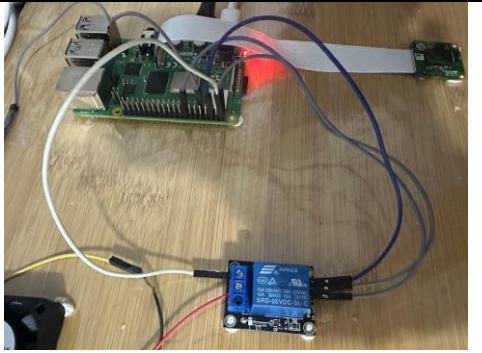
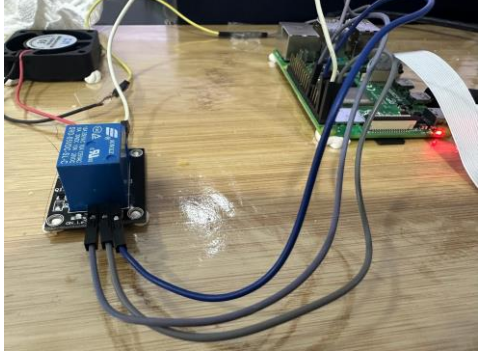
The Raspberry Pi is connected to the fan and relay module via digital pin 1, pin 2, pin 6, pin 9 and Pin 16. Raspberry Pi Camera Module 2 connected to Raspberry Pi Model 4b Camera Plug.

The first 3V pin of the Raspberry Pi goes to the + power supply of the relay module, then the second 5V pin goes to the relay power supply, then the 6th ground pin goes to the ground hole of the relay module, and then the 9th hole It is the (-) line connected to the fan, and pin 16 is connected to the signal plug of the relay module. The relay module has another line that goes to the red line of the fan. Finally, the led (-) connect to the Ground, and led (+) connected to GPIO2.

## Chapter 5: System Implementation

### 5.1 Setting and Configuration

#### 5.1.1 Hardware Setup

	<p>Connect the relay module with the fan, to control the power supply to the fan. The red jumper cable connects to the relay + plug and then the fan (-) cable connects to the raspberry pi ground.</p>
	<ul style="list-style-type: none"> <li>-Plug type-c adapter to give raspberry pi power supply.</li> <li>-Connect the raspberry pi camera module 2 with the raspberry pi 4 model b.</li> <li>- Input mouse and keyboard (optional)</li> </ul>
	<ul style="list-style-type: none"> <li>-The (+) of the relay module connects with the raspberry pi 5V pin.</li> <li>-The (-) of the relay module connects with the raspberry pi Ground pin.</li> <li>-The (Sig) of the relay module connects with the raspberry pi GPIO 23 pin.</li> </ul>

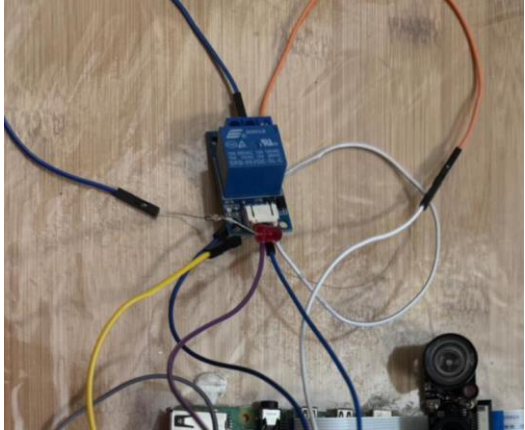
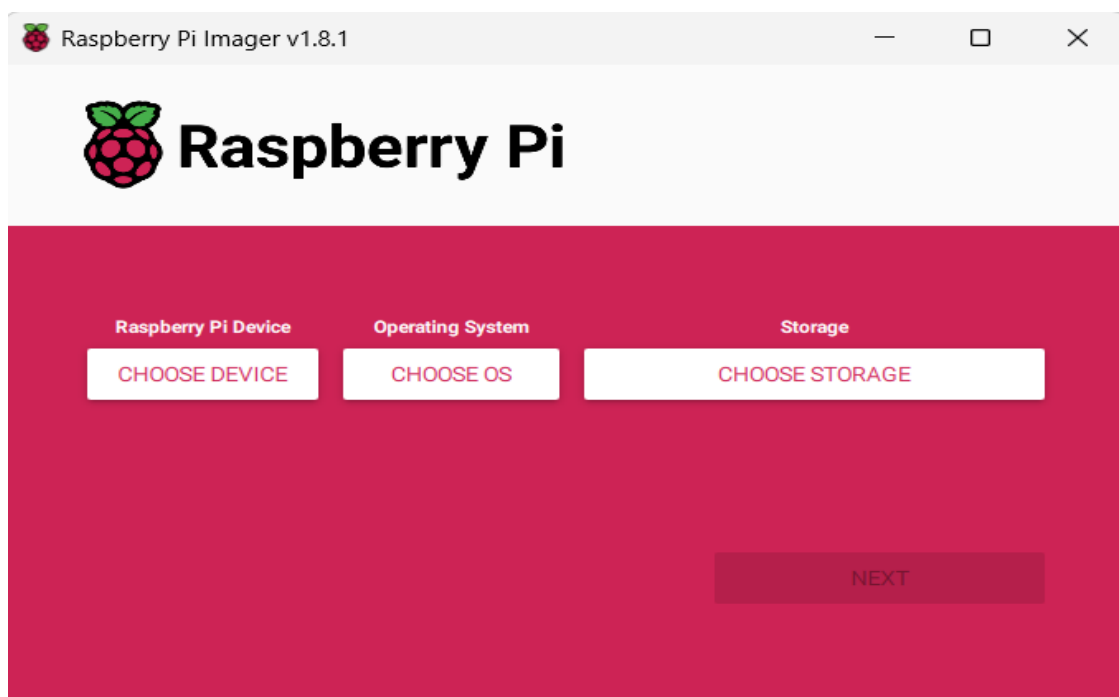
	<p>-The (+) of the LED light connects with the raspberry pi 3V pin.</p> <p>-The (-) of the LED light connects with the raspberry pi Ground pin.</p>
---	---

Table 5.1.1: Details of hardware setting up

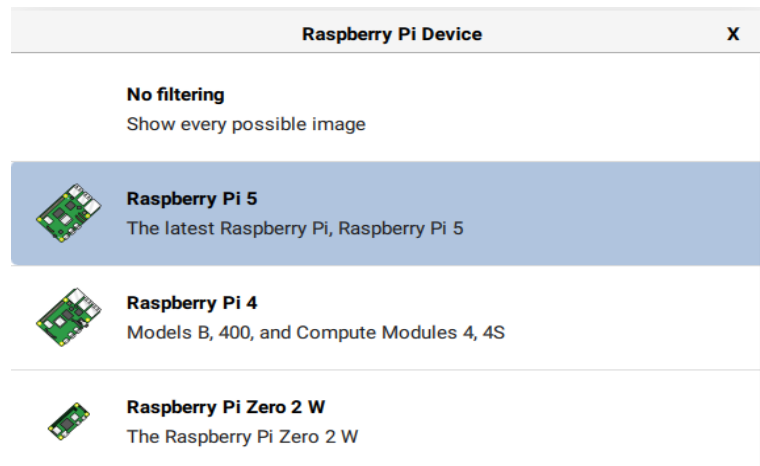
### 5.1.2 Software Setup

Before starting to develop the project, there are six software needed to be installed and downloaded in the laptop and raspberry pi 4 model b:

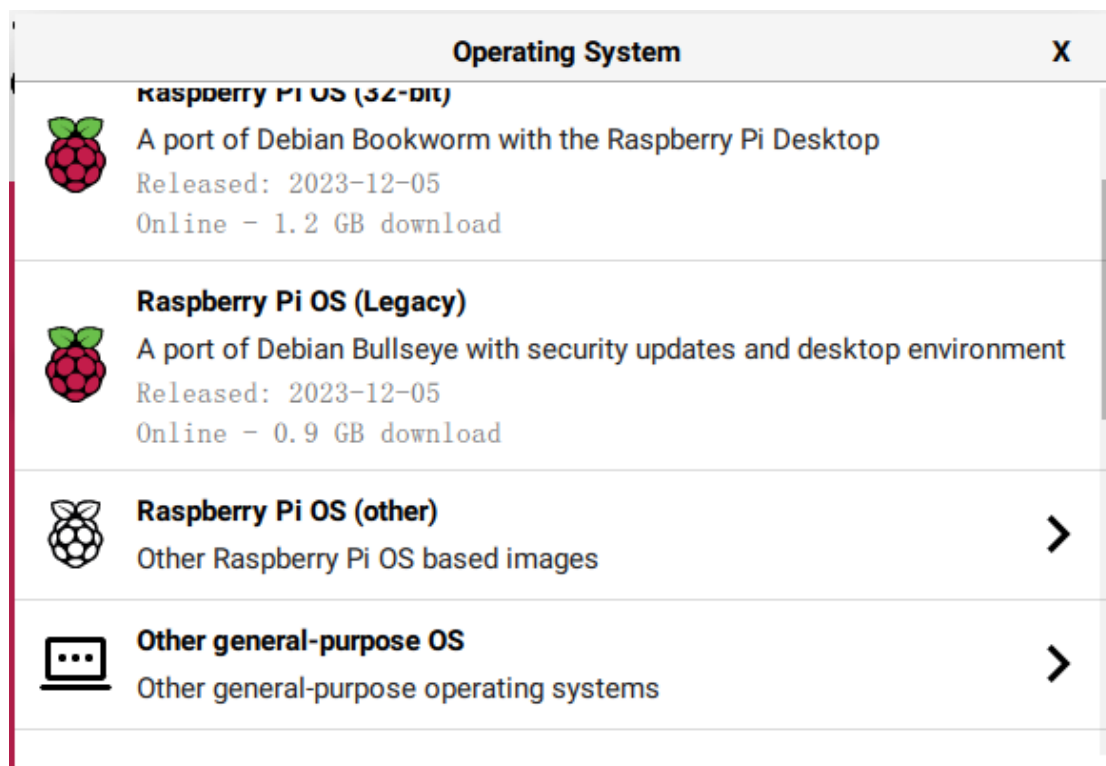
1. Raspberry pi Imager (to install Legacy version of raspberry pi OS into raspberry pi)



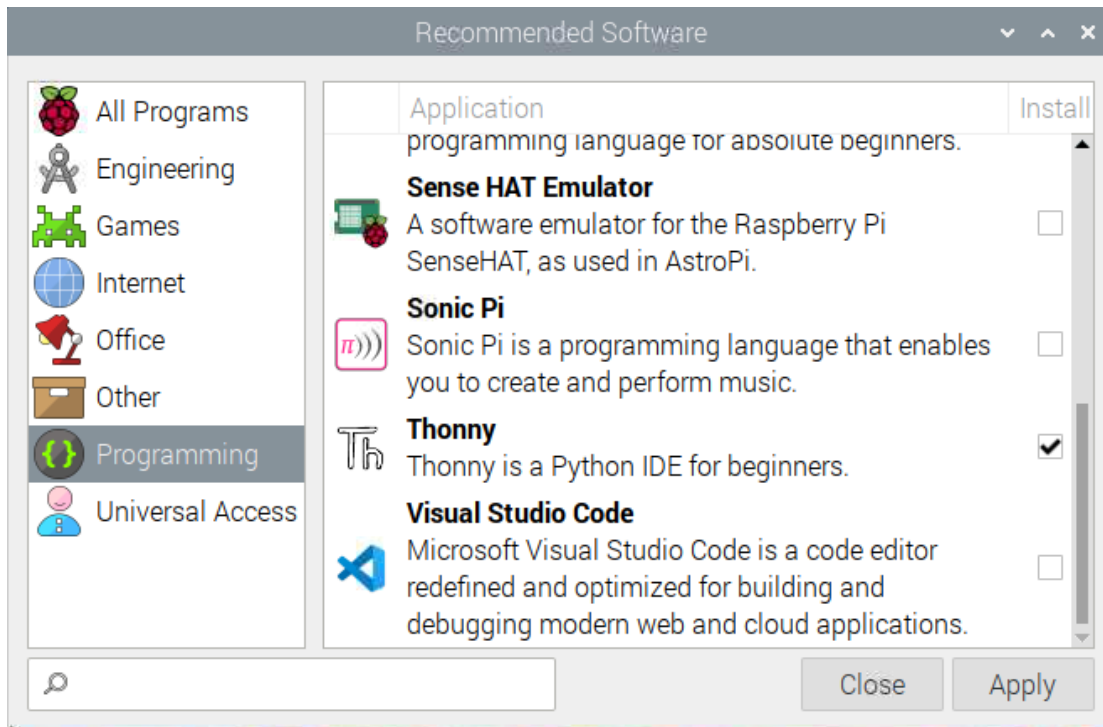
- i. Install raspberry pi imager from website
- ii. Then choose a device.



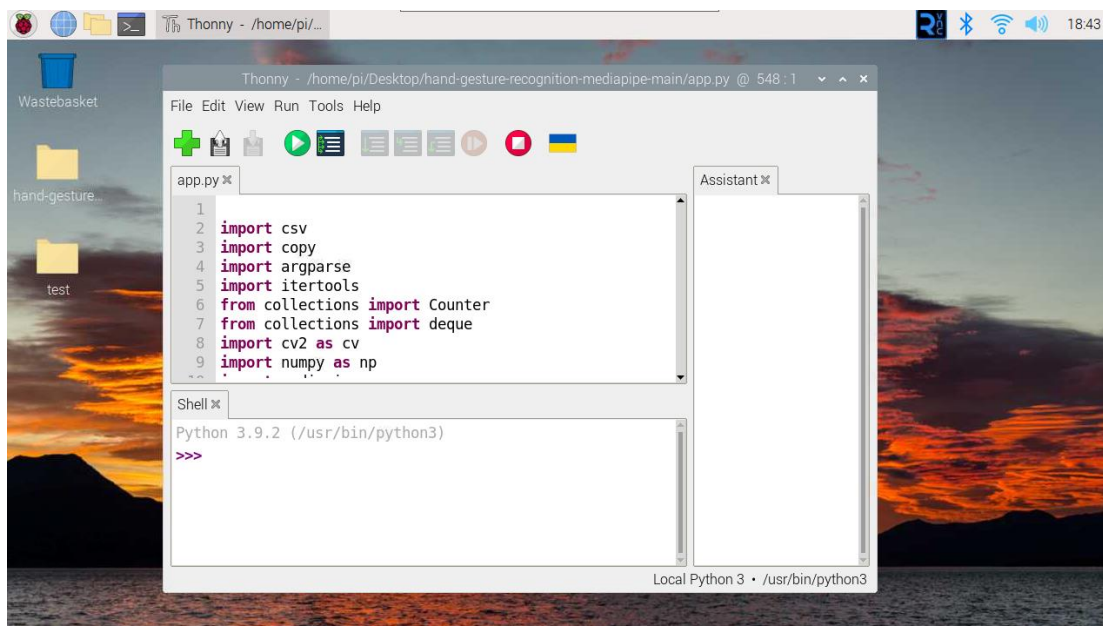
- iii. Choose the Legacy version Raspberry pi OS



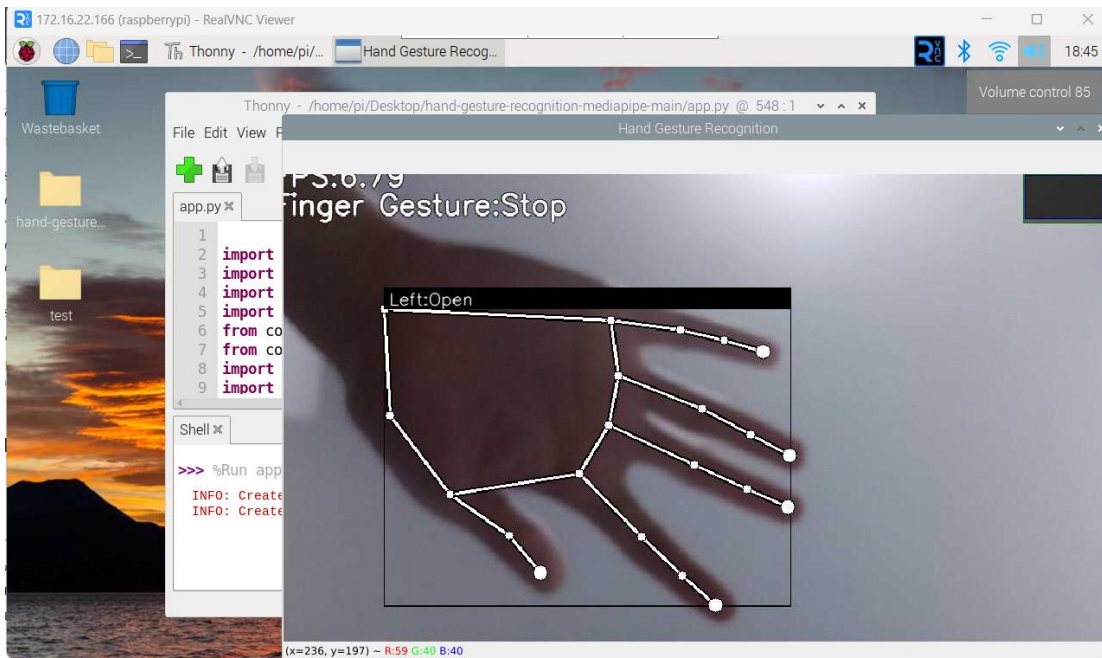
- iv. Then choose the SD card you put in , then start to write the OS inside the SD card.
- v. Thonny programming (for raspberry pi programming)



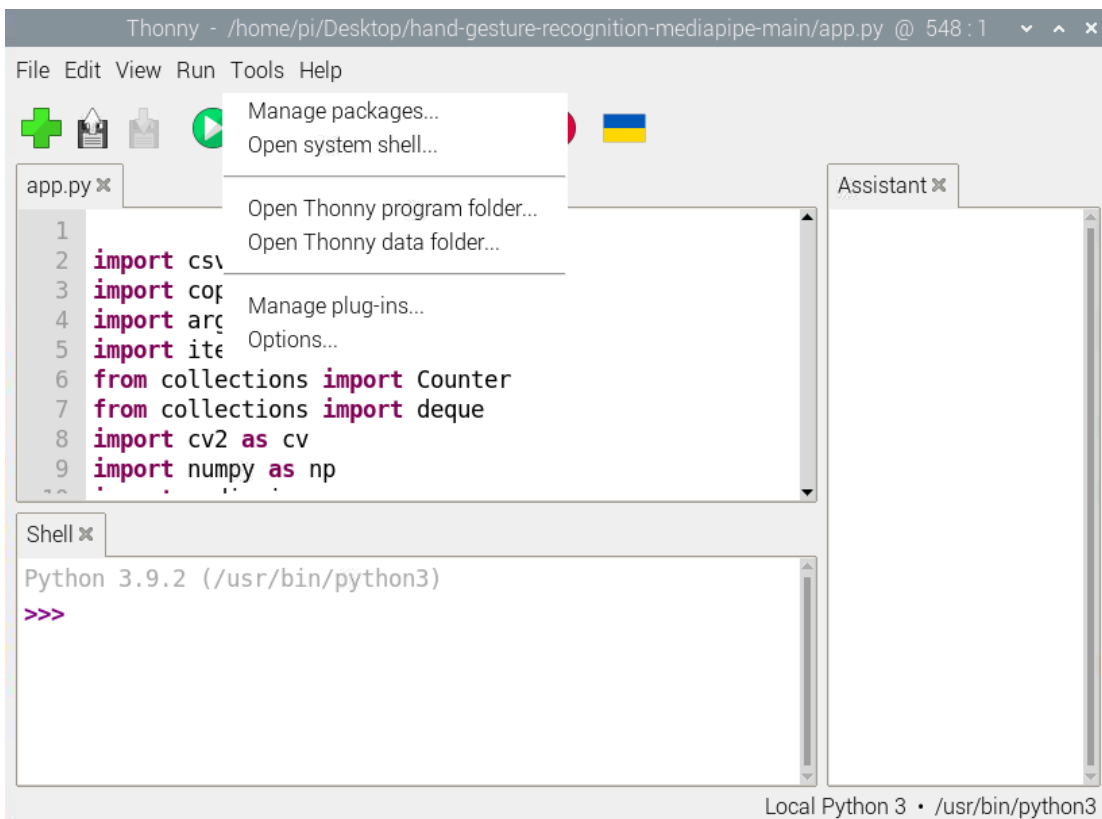
- vi. Search the Thonny in recommended software and apply it.



- vii. Put the code inside the Thonny programming and then run.

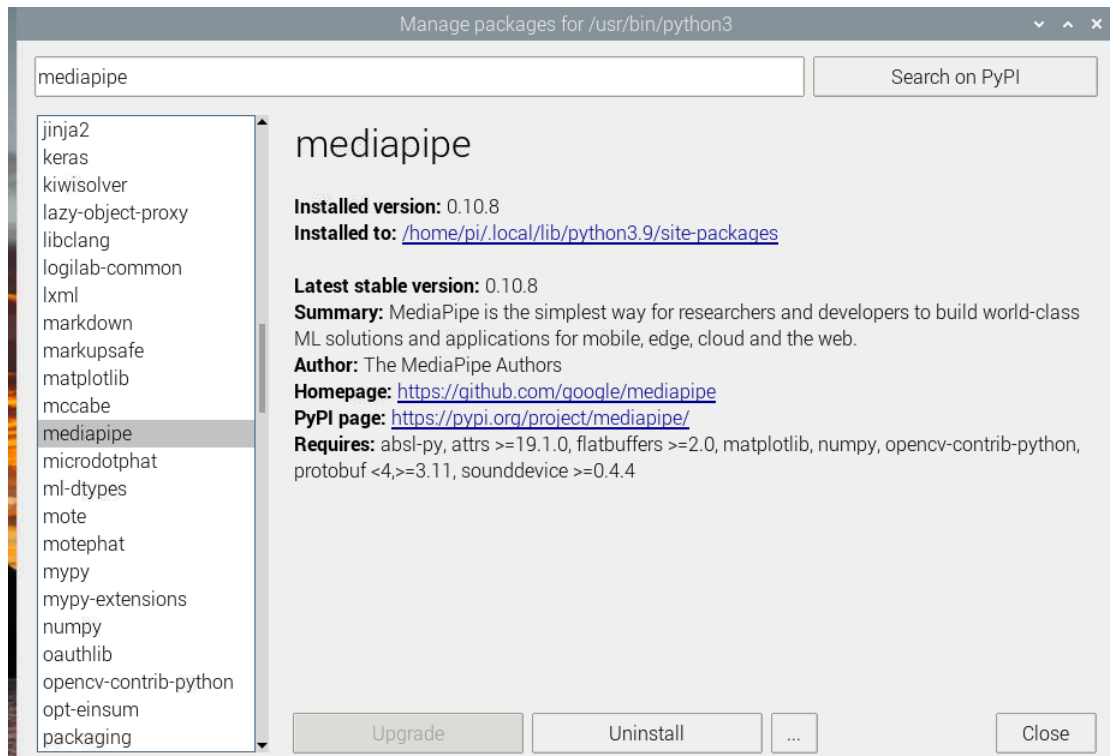


3. Mediapipe (for hand gestures recognition use) and OpenCv (to capture handgesture using raspberry pi)



- i. Inside the Thonny programming Tools part you can press the manage packages

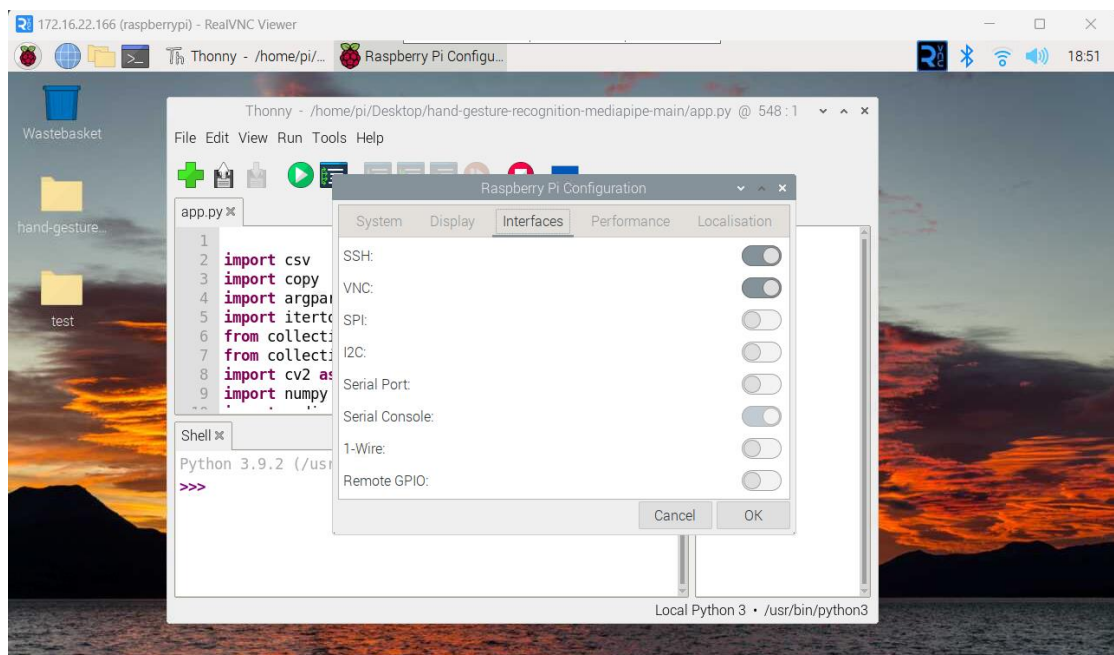




- ii. And then install the mediapipe and opencv on that page.
- iii. If can't successfully install the mediapipe and opencv you can go terminal and write "pip install mediapipe and pip install opencv.

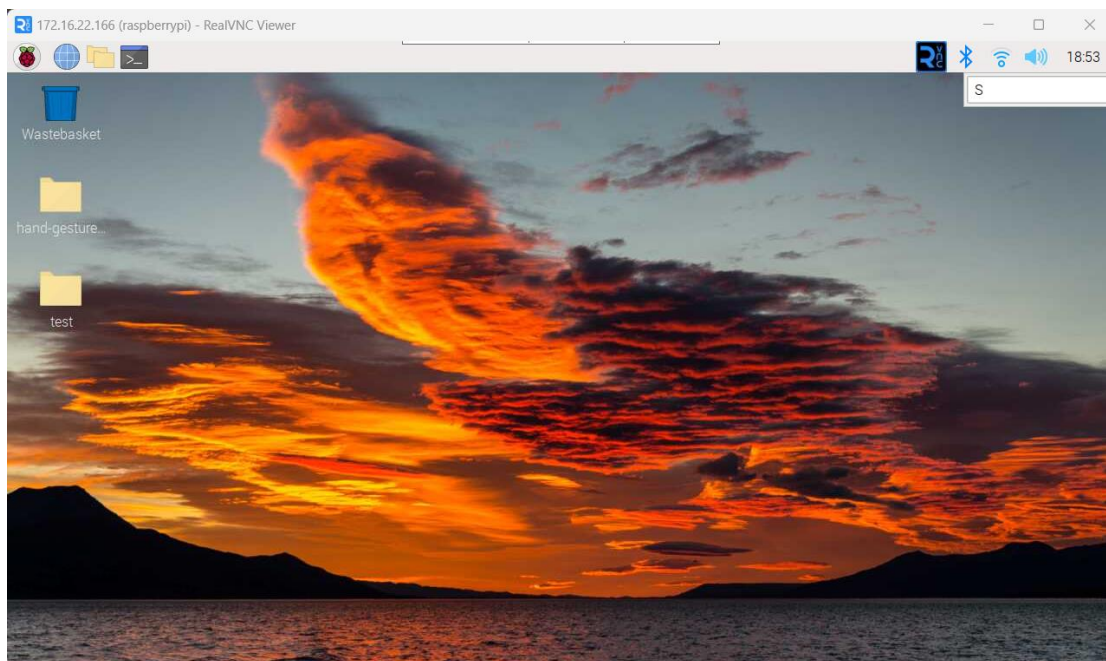
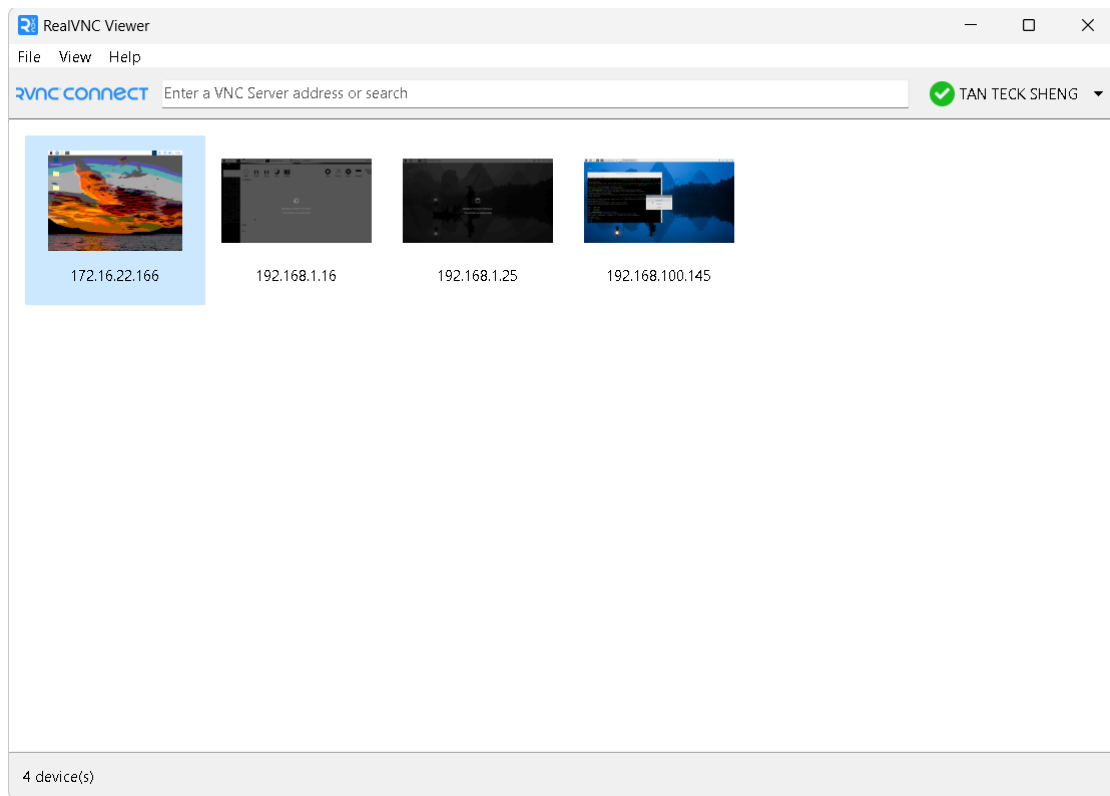
## 5. VNC viewer (to control raspberry pi)

- i. Remember to enable the VNC access first on the raspberry pi configuration page.



- ii. And then on your PC , you can install the VNC viewer from the website.

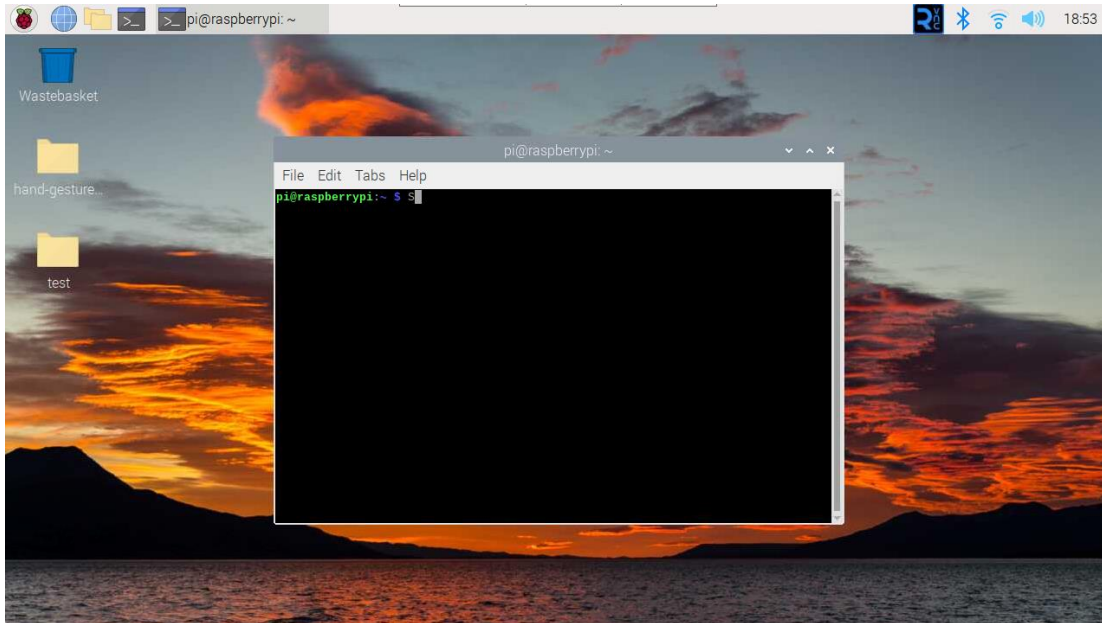
## CHAPTER 5



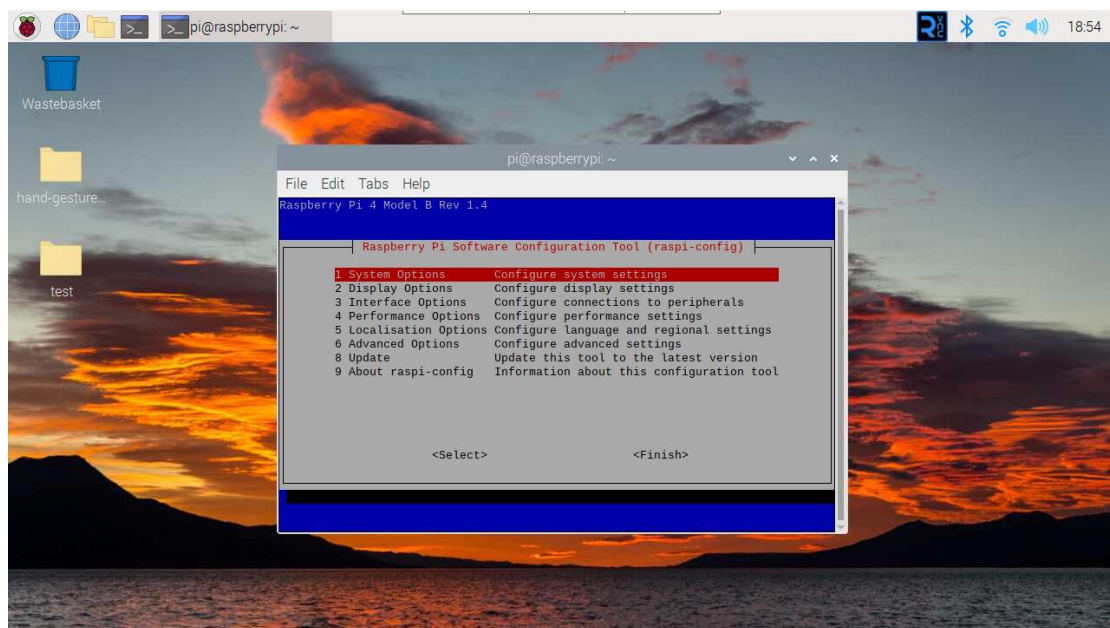
iii. Lastly, you can successfully control the raspberry pi by using the VNC viewer.

6. Install the camera module.

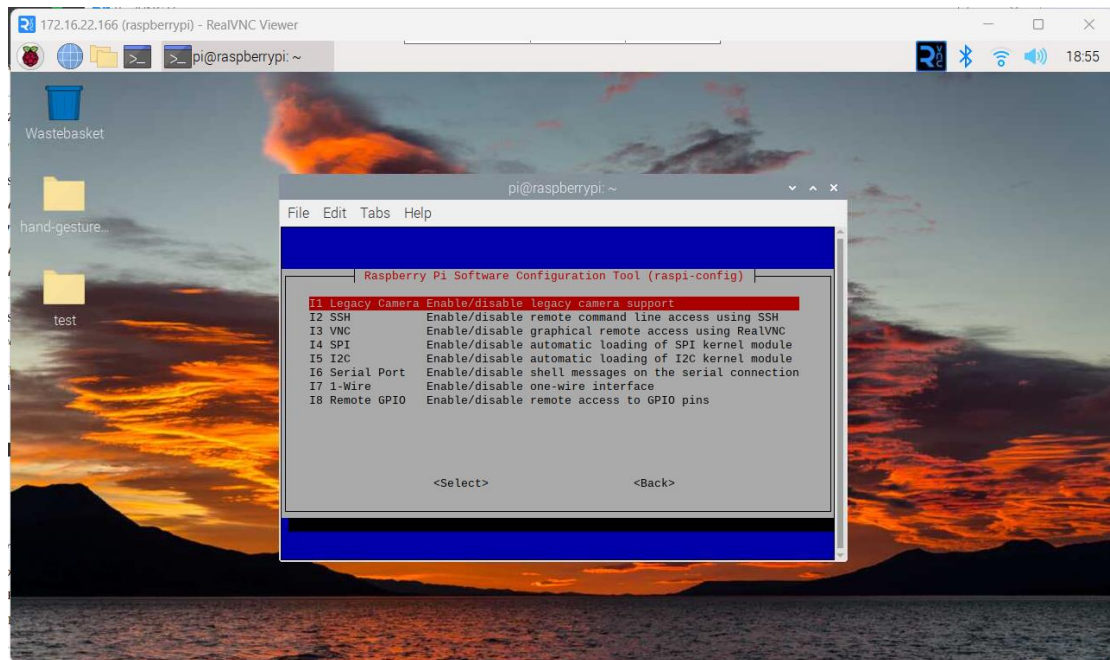
## CHAPTER 5



- i. Go to Terminal
- ii. And then press “sudo raspi-config”



- iii. And then choose the interface option.



iv. And then the last step is enabling the Legacy Camera.

## 5.2 Python Code Explanation

### 5.2.1 Hand Gestures Recognition Code Explanation

```
import csv
import copy
import argparse
import itertools
from collections import Counter
from collections import deque
import cv2 as cv
import numpy as np
import mediapipe as mp
from utils import CvFpsCalc
from model import KeypointClassifier
from model import PointHistoryClassifier
from time import sleep
from gpiozero import LED
```

Code Explanation:

The code is leveraging a robust set of libraries and modules to develop a hand gesture recognition system. It uses CSV for efficient handling of CSV files, aiding in data storage and retrieval. The copy module enables safe duplication of objects and data structures, ensuring data integrity in the system. Argparse allows seamless integration of command-line arguments, enhancing program flexibility during execution. Itertools provides versatile tools for data manipulation and iteration, while Counter simplifies counting occurrences of elements within iterables. The deque structure is employed for efficient double-ended queue operations. cv2 (OpenCV) forms the backbone for computer vision and image processing tasks, while NumPy supports numerical operations crucial for computations. Mediapipe focuses on specialized computer vision tasks like hand tracking, and the project's utils and model modules contain custom utility and model classes tailored for this application. Time handles time-related functionalities like creating delays, and gpiozero enables access to GPIO pins, particularly facilitating LED control, likely for visual feedback within the hand gesture recognition system.

```
led = LED(23) # Initialize the GPIO pin for the LED

def get_input_args():
    argument_parser = argparse.ArgumentParser()
    argument_parser.add_argument("--device-id", type=int, default=0)
    argument_parser.add_argument("--frame-width", help='set frame width', type=int,
default=960)
    argument_parser.add_argument("--frame-height", help='set frame height',
type=int, default=540)
    argument_parser.add_argument('--static-image-mode', action='store_true')
    argument_parser.add_argument("--detection-confidence",
        help='set min detection confidence',
        type=float,
        default=0.7)
    argument_parser.add_argument("--tracking-confidence",
        help='set min tracking confidence',
        type=int,
```

```

        default=0.5)

    parsed_args = argument_parser.parse_args()

    return parsed_args

```

Code Explanation:

This code snippet initializes an LED object led, which is used to control the GPIO pin number 23 using the gpiozero library. It also defines a function, get\_args(), that uses argparse to handle command-line parameters related to device configuration, image size, and confidence thresholds. The function returns the parsed parameters.

```

def start():
    parsed_arguments = retrieve_input_args()

    camera_device = parsed_arguments.device_id
    camera_width = parsed_arguments.frame_width
    Camera_height = parsed_arguments.frame_height

    is_static_image_mode = parsed_arguments.static_image_mode
    Minimum_detection_confidence = parsed_arguments.Detection_confidence
    Minimum_tracking_confidence = parsed_arguments.tracking_confidence

    use_rectangle = True

```

Code Explanation:

This main() function uses argparse to perform parameter parsing to collect and process user-defined settings for devices (such as cameras), image sizes, and confidence thresholds for detection and tracking. Then, assign the parsed arguments to the corresponding variables for use in the function. Also, use\_brect is set to True in the function.

```

def prepare_camera():

```

```

Camera = cv.VideoCapture(camera_device)
camera.settings(cv.CAP_PROP_FRAME_WIDTH, camera width)
camera.settings(cv.CAP_PROP_FRAME_HEIGHT, camera height)

# Load model##### #
#####

mp_hands = mp.solutions.hands
hands = mp_hands.Hands(
    static_image_mode=static_image_mode,
    Maximum lot size=1,
    Minimum detection confidence = detection confidence,
    min_tracking_confidence=tracking_confidence,
)

keypoint_classifier = KeyPointClassifier()

point_history_classifier = PointHistoryClassifier()

```

Code Explanation:

Camera preparation: It initializes the capture object using OpenCV (`cv.VideoCapture`) to access the video stream from the specified device. It sets the width and height of the captured frame based on the provided dimensions.

Model loading:

It loads the hand tracking model from MediaPipe. Initialize the hand tracking model (`hand`) with various parameters, such as whether to use static image mode, the maximum number of hands to detect, and the minimum confidence threshold for detection and tracking. Initialize instances of the `KeyPointClassifier` and `PointHistoryClassifier` classes for gesture classification.

```

# Detection implementation
image = cv.cvtColor(image, cv.COLOR_BGR2RGB)

```

```

image.flags.writeable = False
results = hands.process(image)
image.flags.writeable = True

if results.multi_hand_landmarks is not None:
    for hand_landmarks, handedness in zip(results.multi_hand_landmarks,
                                         results.multi_handedness):
        # Bounding box calculation
        brect = calc_bounding_rect(debug_image, hand_landmarks)
        # Landmark calculation
        landmark_list = calc_landmark_list(debug_image, hand_landmarks)

        # Conversion to relative coordinates / normalized coordinates
        pre_processed_landmark_list = pre_process_landmark(
            landmark_list)
        pre_processed_point_history_list = pre_process_point_history(
            debug_image, point_history)
        # Write to the dataset file
        logging_csv(number, mode, pre_processed_landmark_list,
                   pre_processed_point_history_list)

        # Hand sign classification
        hand_sign_id = keypoint_classifier(pre_processed_landmark_list)
        if hand_sign_id == 0:
            led.on()
        elif hand_sign_id == 1:
            led.off()

```

Code Explanation:

The script initializes label lists by reading CSV files, establishing FPS calculation, and setting up data structures to maintain coordinate and finger gesture histories. It enters a continuous loop, managing FPS and user input, capturing frames, and processing images by detecting hand



landmarks. For each detected hand, it computes the bounding box and extracts landmarks, preparing this data for processing. The pre-processed landmarks and history points are logged, and the pre-processed landmarks are classified using a keypoint classifier. Based on the identified hand sign, it triggers LED actions—illuminating or turning off—depending on the specific hand sign identification. This forms a closed-loop system where hand gestures prompt corresponding LED responses, a core functionality within the hand gesture recognition framework.

```
# Gesture classification
gesture_id = 0
history_length = len(pre_processed_point_history_list)
if history_length == (max_history * 2):
    gesture_id = point_history_classifier(pre_processed_point_history_list)

# Track and visualize gestures
gesture_history.append(gesture_id)
most_common_gesture_id = Counter(gesture_history).most_common()

# Visualization components
debug_image = draw_bounding_box(use_brect, debug_image, brect)
debug_image = draw_detected_landmarks(debug_image, landmark_list)
debug_image = draw_information(
    debug_image,
    brect,
    handedness,
    keypoint_classifier_labels[hand_sign_id],
    point_history_classifier_labels[most_common_gesture_id[0][0]],
)

# Handling undetected landmarks
else:
    point_history.append([0, 0])
```

```
# Visualize history and system info
debug_image = draw_point_history_info(debug_image, point_history, fps, mode,
number)
```

#### Code Explanation:

The script begins by converting the camera-captured image to RGB and utilizes MediaPipe to detect hand landmarks. It processes each detected hand by calculating its bounding box and retrieving a set of landmarks representing key hand points. These landmarks are then normalized for further analysis, and their historical data is stored in a dataset file. Subsequently, a keypoint classifier is used to identify gestures from the preprocessed landmarks. When a gesture is recognized as 0, indicating a specific symbol, the code triggers the LED to turn on. Conversely, if the recognized gesture is 1, it prompts the LED to turn off—a clear action based on the identified hand gesture, demonstrating the interactive behavior of the system.

### **5.2.2 Face Recognition Code Explanation**

```
#To capture New Face Record
video = cv2.VideoCapture(0)

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

count = 0
max_count = 300 # Maximum number of images to capture

nameID = str(input("Enter your name: ")).lower()

path = 'images/' + nameID

isExist = os.path.exists(path)

if isExist:
```

```

print("Name already taken")
nameID = str(input("Enter your name again: "))
else:
    os.makedirs(path)

while True:
    ret, frame = video.read()
    faces = facedetect.detectMultiScale(frame, 1.3, 5)
    for x, y, w, h in faces:
        count = count + 1
        name = './images/' + nameID + '/' + str(count) + '.jpg'
        print("Creating Images....." + name)
        cv2.imwrite(name, frame[y:y + h, x:x + w])
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)
        cv2.imshow("WindowFrame", frame)
        cv2.waitKey(1)
        if count >= max_count: # Break the loop when count reaches max_count
            break

    if count >= max_count: # Break the main loop when count reaches max_count
        break

# Release the video capture and close all OpenCV windows
video.release()
cv2.destroyAllWindows()

```

Code Explanation:

Use the OpenCV library to capture video frames from a webcam and detect faces in real time using the pre-trained Haar Cascade classifier. The system prompts the user for a name, creates a directory based on the user's name to store the captured face images, and then enters an infinite loop to capture and process the frames. For each frame, it detects faces using the Haar Cascade classifier, crops the face region, saves it as an image file in a specified directory, and visually highlights the detected face with a rectangle. This process continues until the

maximum number of images specified by maximum of count is reached. Finally, it releases the video capture object and closes all OpenCV windows.

```
#Start the Face Recognition system
facedetect = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
font = cv2.FONT_HERSHEY_COMPLEX

model = load_model('keras_model.h5')

def get_className(classNo):
    if classNo == 0:
        return "Host"
    elif classNo == 1:
        return "Unknown"

def detect_host(imgOriginal):
    faces = facedetect.detectMultiScale(imgOriginal, 1.3, 5)
    for x, y, w, h in faces:
        crop_img = imgOriginal[y:y + h, x:x + w]
        img = cv2.resize(crop_img, (224, 224))
        img = img.reshape(1, 224, 224, 3)
        prediction = model.predict(img)
        classIndex = np.argmax(prediction, axis=-1) # Using argmax to get the class
index
        probabilityValue = np.amax(prediction)
        if classIndex == 0:
            cap.release()
            cv2.destroyAllWindows() # Close the current window
            subprocess.run(["python", "app.py"]) # Run app.py using subprocess
```

```

        exit() # Exit the script after running app.py
    return probabilityValue # Return probabilityValue from the function

while True:
    success, imgOriginal = cap.read()
    probabilityValue = detect_host(imgOriginal) # Get probabilityValue from
detect_host function
    if probabilityValue is not None:
        cv2.putText(imgOriginal, str(round(probabilityValue * 100, 2)) + "%", (180,
75), font, 0.75, (255, 0, 0), 2, cv2.LINE_AA)
        cv2.imshow("Result", imgOriginal)
        k = cv2.waitKey(1)
        if k == ord('q'):
            break

cap.release()
cv2.destroyAllWindows()

```

Code Explanation:

This face recognition starts by importing the necessary libraries such as TensorFlow, Keras, NumPy, cv2 (OpenCV) and subprocess. This script uses OpenCV's CascadeClassifier to load a pre-trained face detection classifier named "haarcascade\_frontalface\_default.xml". It also loads a pre-trained deep learning model named "keras\_model.h5" using Keras's load\_model function.

The main part of the code is the detector\_host function, which takes the raw image captured from the webcam, detects faces using a CascadeClassifier, crops the detected face area, and resizes it to (224, 224) pixels. The resized images are then input into a pre-trained deep learning model for inference using the prediction function. The model predicts the class index and probability of a detected face as "Host" (class index 0) or "Unknown" (class index 1). If the predicted class index is 0 (host), it releases the video capture, closes the OpenCV window, runs "app.py" using a subprocess, and exits the script.

The while loop uses `cap.read()` to continuously capture frames from the webcam, calls the `detector_host` function to perform face recognition on each frame, and uses `cv2.imshow()` to display the results on the screen. The script also checks if the 'q' key is pressed to exit the loop and releases the video capture resources and closes the OpenCV window before terminating.

### 5.2.3 Code to Extract RGB Value and Manual and Auto Increase/Decrease Brightness

```
#To Extract RGB Value
def increase_brightness(frame, brightness_increase):
    new_brightness = brightness_increase
    if brightness_increase > 0:
        new_brightness = min(brightness_increase, 255)
    elif brightness_increase < 0:
        new_brightness = max(brightness_increase, -255)
    return cv.convertScaleAbs(frame, alpha=1, beta=new_brightness)

fps = cvFpsCalc.get()

#           Process           Key           (ESC:           end)
#####

key = cv.waitKey(10)
if key == 27: # ESC
    break
elif key == 107 and brightness_increase <= -255: # '+' key to increase
brightness
    brightness_increase += 10
elif key == 109 and brightness_increase >= -255: # '-' key to decrease
brightness
    brightness_increase -= 10
number, mode = select_mode(key, mode)
```

```

# Camera capture
#####
ret, image = cap.read()
if not ret:
    break
gray_frame = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
brightness = int(gray_frame.mean())
image = cv.flip(image, 1) # Mirror display
brightened_frame = increase_brightness(image, brightness_increase)
debug_image = copy.deepcopy(brightened_frame)

if brightness <= 60:
    image = increase_brightness(image, -160) # Decrease brightness by 100
if brightness >= 130:
    image = increase_brightness(image, -50)

debug_image = copy.deepcopy(image)

```

#### Code Explanation:

This part of the code defines a function called `increase_brightness`, which is used to adjust the brightness of the input frame. This function takes two parameters: `frame` (representing the input image frame) and `brightness_increase` (specifying the amount by which the brightness should be increased or decreased).

Inside the `increase_brightness` function, it checks whether the `brightness_increase` value is positive or negative. If positive, ensures that the new brightness value does not exceed 255, if negative, ensures that the new brightness value does not go below -255. This is achieved using the `min` and `max` functions.

After calculating the new brightness value, the function uses OpenCV's `convertScaleAbs` function to adjust the brightness of the input frame. The `ConvertScaleAbs` function applies a linear transformation to an input array (frames), scaling pixel values by a specified factor (alpha) and adding a specified value (beta) to each pixel. In this case, the alpha value is set to 1 (scaling

unchanged) and the beta value is set to the calculated new\_brightness, which determines the brightness level of the output frame.

The code snippet also includes other functions such as capturing frames from the camera (cap.read()), converting the captured frames to grayscale (cv.cvtColor(image, cv.COLOR\_BGR2GRAY)), flipping the frames horizontally (cv.flip( Image, 1)), adjust the brightness according to the average brightness value (brightness) of the grayscale frame. Additionally, there are conditions for further adjusting the brightness based on certain brightness thresholds ( $\leq 60$  or  $\geq 130$ ).

### 5.3 System Operation

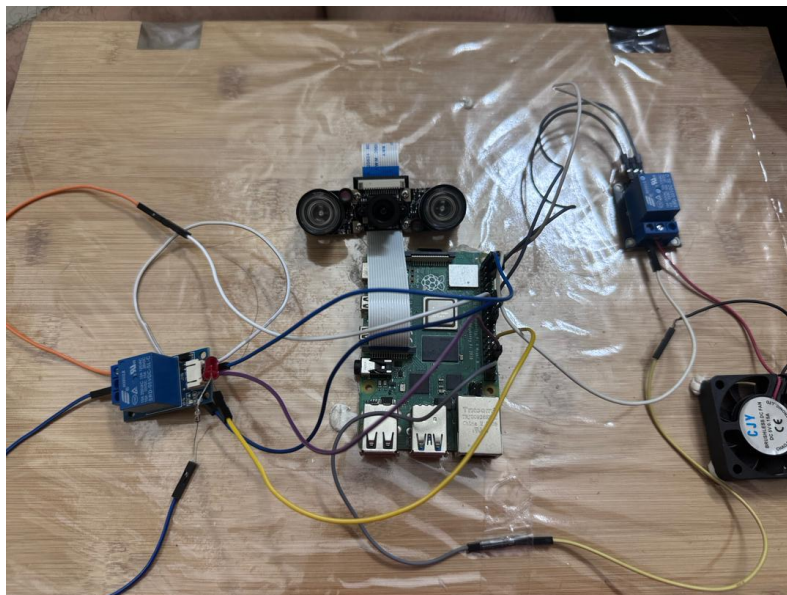


Figure 30 shows the System operation.

This subsection presents the results of the project development and main target. This project is to monitor various gestures. These gestures can operate light bulbs and fans at any time. Figure 5.3 The overall circuit of the system is running smoothly.



## 5.4 Implementation Issues and Challenges

```
chmod +x *.sh
g++ -std=c++11 -std=c++11 -Iinclude/ -I3rdparty/stb/include -DOPENCV `pkg-config
--cflags opencv4 2> /dev/null || pkg-config --cflags opencv` -Wall -Wfatal-erro
rs -Wno-unused-result -Wno-unknown-pragmas -fPIC -Ofast -DOPENCV -c ./src/image_
opencv.cpp -o obj/image_opencv.o
Package opencv was not found in the pkg-config search path.
Perhaps you should add the directory containing `opencv.pc'
to the PKG_CONFIG_PATH environment variable
No package 'opencv' found
./src/image_opencv.cpp:16:10: fatal error: opencv2/core/version.hpp: No such fil
e or directory
#include <opencv2/core/verston.hpp>
                    ^~~~~~
compilation terminated.
Makefile:182: recipe for target 'obj/image_opencv.o' failed
make: *** [obj/image_opencv.o] Error 1
```

Figure 31 shows Issues when install opencv in Raspberry Pi.

When implementing these software and hardware, we encountered a lot of hardware compatibility and hardware damage problems. The software version is also very important, because some software versions will not match the hardware, so the correct software must be selected to adapt to the hardware. Hardware must also be selected to be more It is suitable for projects to prevent the system from lagging and crashing. This can also indirectly improve the practicality and efficiency of the system.

## **Chapter 6: System Evaluation and Discussion**

### **6.1 System Testing and Performance Metrics**

#### **6.1.1 Performance Metrics for Hand Gestures Recognition Model**

System testing and performance metrics are an important part of evaluating a program's system performance. In this project, CNN, NN and random forest models are used to evaluate the accuracy of gesture detection. First, use the confusion matrix to determine the gesture recognition accuracy of the three models of true positive, false positive, true negative and false negative, and finally determine the performance of these three models. In this detection, we classify four gestures, from 0 to 3, representing open, close, pointer and OK Gestures. After that, a classification report is generated which displays precision, recall, f1 score, support, accuracy, macro average, weighted average, and loss. Precision is the proportion of true predictions out of all the positive predictions made by the model. Recall, also known as sensitivity, measures the proportion of true positive predictions out of all actual positive instances in the data set. The F1 score is the harmonic mean of precision and recall, providing a balanced measure of model performance. Support refers to the number of instances of each class in the dataset. Accuracy is the overall correct prediction rate of the model. Macro-avg calculates the unweighted average of precision, recall, and F1 scores across all classes. Weighted average calculates the average of precision, recall, and F1 score, weighted by the support of each class. Loss represents the difference between the model's predicted output and the actual target output during training, quantifying how well the model performs relative to its training goals.

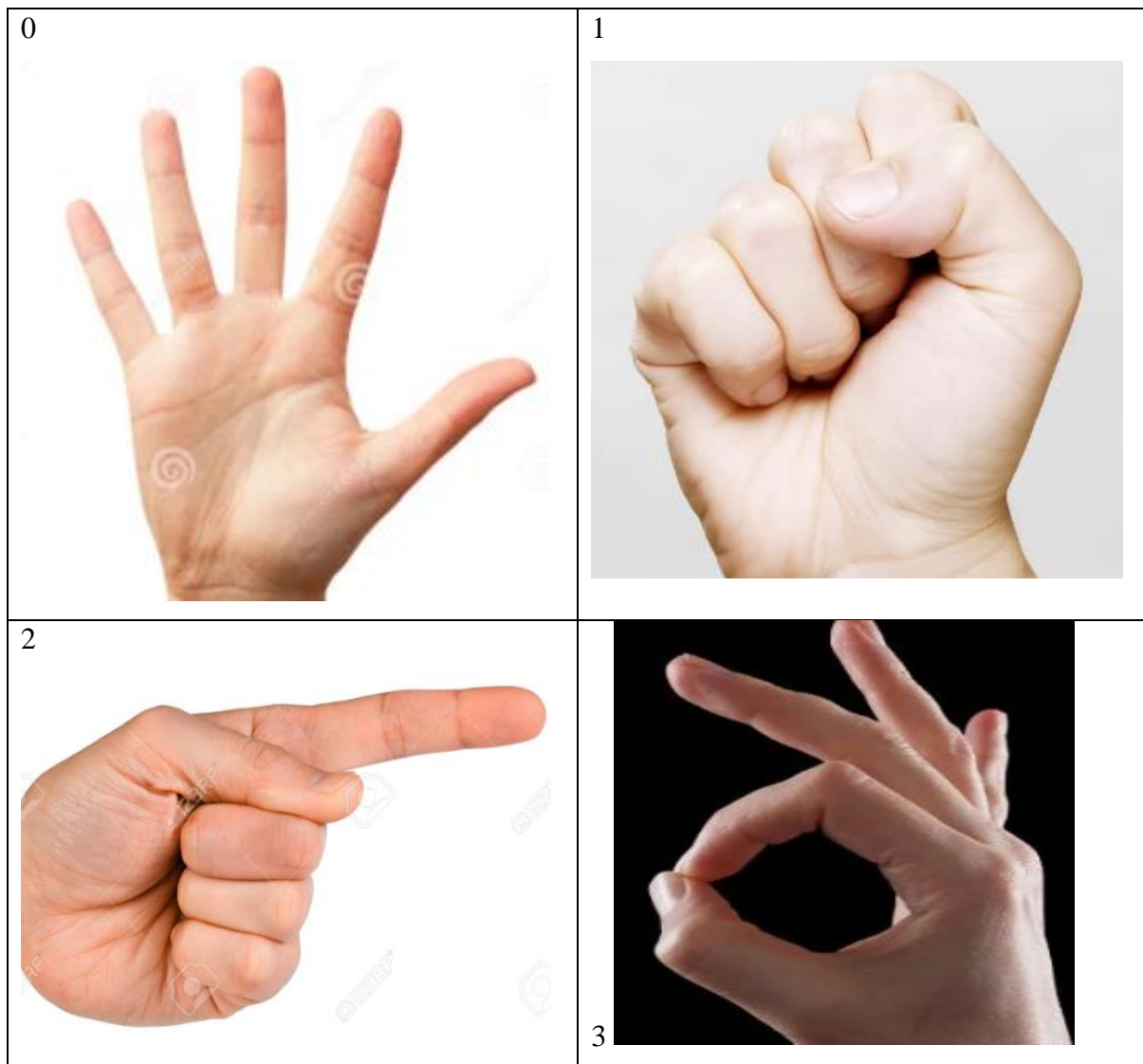


Table 6.1.1.1 Show the hand gestures class.

### 6.1.2 Testing Hand Gestures Control Smart Home

Input Hand Gestures	Action of smart home
Open	Open the Fan
Close	Close the Fan
Pointer	Open the LED
OK	Close the LED

Table 6.1.2.1 show the gestures used to control smart home when testing the system.

When user input the specific hand gestures, what is the system action.

### 6.1.3 Testing Face Recognition

Input Face	Action of system
Owner	Start the hand gestures recognition system
Unknown	No Action

Table 6.1.2.2 show the face used when testing the system.

When user show is the Owner face, the system will automatically close, and then open the hand gestures recognition system. If show the Unknown, the system will not take any action.

### 6.1.4 Testing the convertScaleAbs Functions

In the above explanation, we get the brightness value, now test the `cv2.convertscaleabs` function will really improve the probability of the system detecting gestures in dark places. 6.2 Testing Result. First, use brightness to give real-time feedback on the brightness of the frame at that time, and testing these gestures can be detected after using this function in black environment.


Input Image	Can detect hand after this function?
	<ul style="list-style-type: none"> <li>● Yes, what is the minimum brightness?</li> <li>● No, manual adjust</li> </ul>

Table 6.1.4 show the Testing tuning function

## 6.2 Testing Result

### 6.2.1 Confusion Metrics for Recognition Model

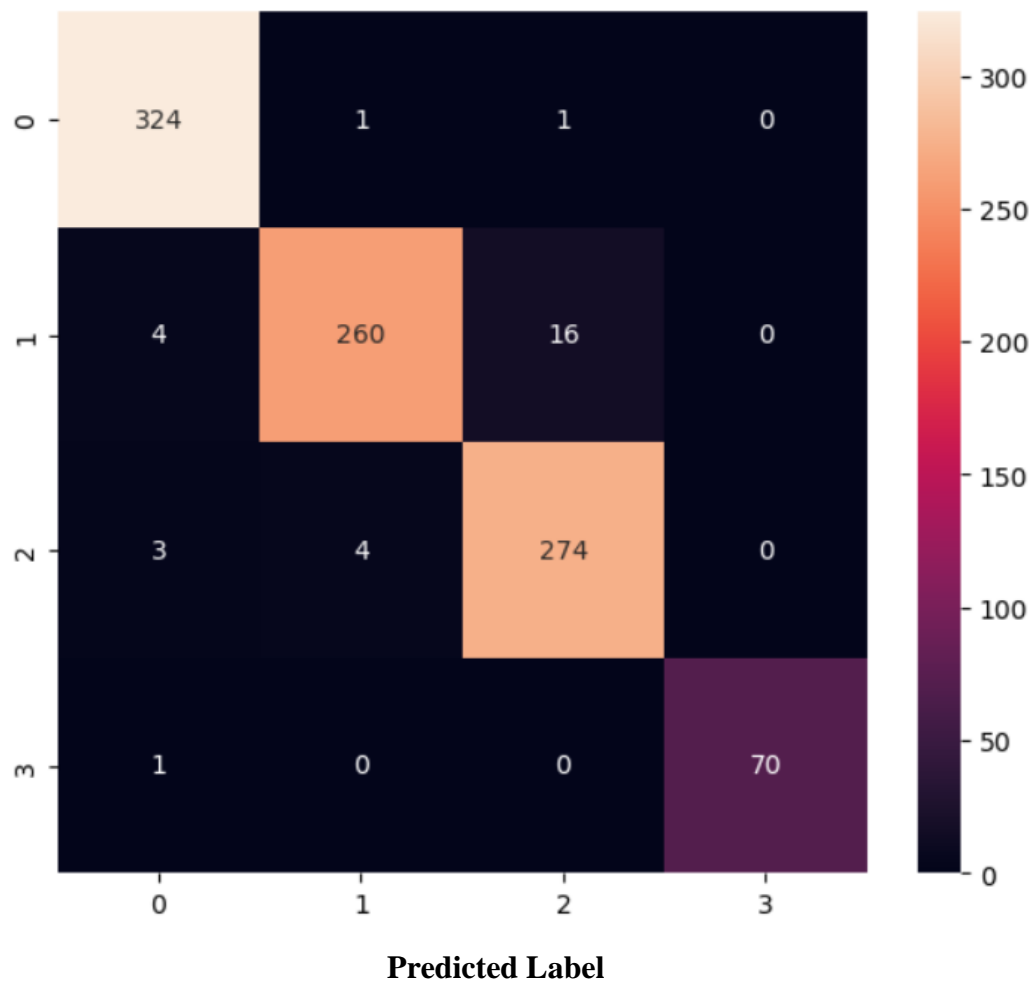


Figure 32 shows confusion matrix of CNN model

This confusion matrix indicates the model's performance in the gesture recognition task. This model demonstrates a strong ability to correctly recognize most gestures, as evidenced by high true positive (TP) values for different categories (0, 1, 2, 3). For example, Category 3 has a TP of 70, indicating that the model accurately recognizes that specific gesture in most cases. However, the false positive (FP) and false negative (FN) values highlight some areas for improvement. The model sometimes misclassified instances, as shown by the FP values, especially for classes 0, 1, and 2, where it incorrectly predicted some instances as belonging to these classes when in fact they belonged to other classes. Additionally, there are some FN instances, showing cases where the model is unable to recognize certain gestures, such as 20 instances of class 1 being incorrectly predicted for other classes. Overall, while the model

shows strong performance in terms of TP and TN values, solving for FP and FN instances can further improve its gesture recognition accuracy.

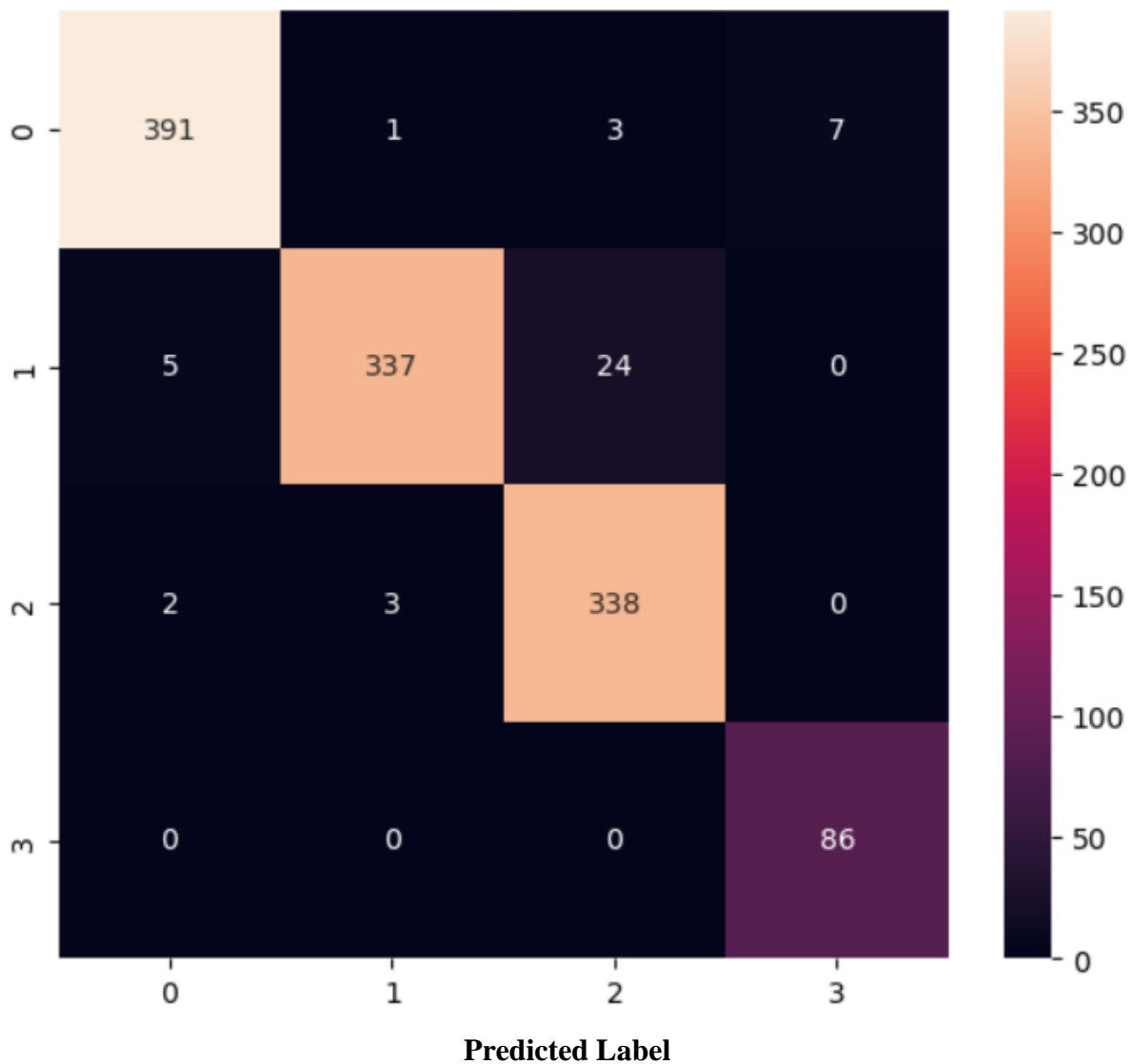


Figure 33 shows confusion matrix of Random Forest Model

This confusion matrix reflects the performance of the random forest model in the gesture recognition task. The model shows high accuracy and effectiveness in recognizing most gestures, as shown by the large number of true positive (TP) values for different categories (0, 1, 2, 3). For example, category 0 shows a TP of 391, indicating that the model correctly recognized the gesture in most instances. However, the false positive (FP) and false negative (FN) values highlight some areas for improvement. The model occasionally misclassified instances, as shown by the FP values, especially for classes 0, 1, and 2, predicting that some

instances belonged to these classes when in fact they belonged to other classes. Additionally, there are some FN instances, showing cases where the model is unable to recognize certain gestures, such as 5 instances of class 0 being incorrectly predicted as other classes. Overall, while the random forest model shows strong performance at high TP values.

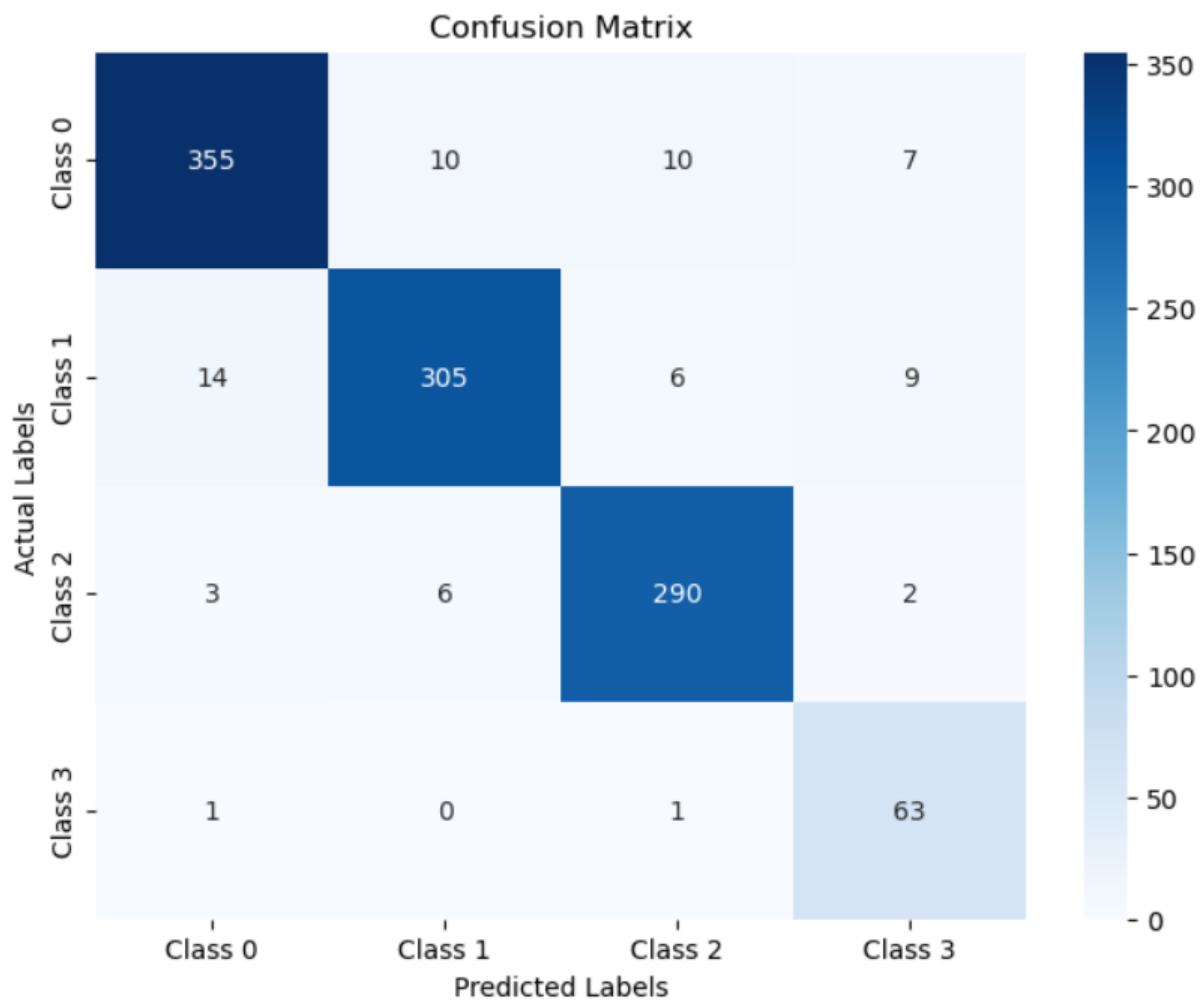


Figure 34 shows confusion matrix of Random Forest Model

This confusion matrix reveals the performance of neural network (NN) models in different categories. The diagonal elements of the matrix represent instances that were correctly classified by the model, while the off-diagonal elements represent misclassifications. Overall, the neural network model shows strong performance, as evidenced by the high values of the diagonal.

Specifically, looking at accuracy (a measure of the proportion of true positives out of all predicted positives), the model showed high accuracy values for most classes. For example, the accuracy values for categories 0, 1, and 2 are approximately 0.95 to 0.96, indicating a low false positive rate. However, for class 3, the accuracy drops to around 0.79, indicating a higher false positive rate for this class.

Likewise, recall (sensitivity) measures the proportion of actual positives correctly identified by the model, and recall is high for most classes, with class 0, 1, and 2 having recall around 0.95 to 0.96. However, the recall for class 3 is lower, around 0.79, which indicates that the model misses some instances of this class.

The F1 score combines precision and recall into one metric and is typically higher for classes 0, 1, and 2, reflecting a good balance between precision and recall. However, the F1 score for Class 3 is relatively low, indicating a trade-off between precision and recall when classifying lass.

	Precision	Recall	F1-score	Support
0	0.98	0.99	0.98	326
1	0.98	0.93	0.95	280
2	0.94	0.98	0.96	281
3	1.00	0.99	0.9	71
Accuracy			0.97	958
Macro-avg	0.97	0.97	0.97	958
Weighted avg	0.97	0.97	0.97	958
Loss = 0.2064				

Table 6.2.1.1: Classification report of CNN model.

According to the classification report, the gesture recognition model showed strong performance on various metrics. Precision, recall, and F1 scores for each category indicate high accuracy in distinguishing different gestures. The overall accuracy of 97% reflects the model's ability to correctly classify gestures in the dataset. Precision measures how accurate the model is for each class. The values shown are all very high, which is around 0.93 to 1.00, indicating the model rarely makes mistakes in assigning a class label. Recall measures how well the model



finds all the relevant data points for each class. Again, the values obtained are high, which is around 0.91 to 1.00, signifying the model captures most of the relevant data points for each class. F1-score is a harmonic mean between precision and recall, offering a balanced view of both metrics. A high F1-score indicates the model is both precise and good at capturing relevant data points. The F1-score values are all close to the corresponding precision and recall values, further emphasizing the model's strong performance across all classes. Support metrics show the number of data points present in each class. The distribution value seems balanced with class sizes around 300 each. Accuracy is the overall percentage of predictions the model got right. With an accuracy of 0.97, the model performs very well, correctly classifying nearly 97% of the data points. Macro-avg and weighted avg are used to calculate the average precision, recall, and F1-score, with minimal bias toward any class. Both methods give very similar values, suggesting a consistent performance across all classes. Lastly the loss value, which is 0.2064 is specific to the training algorithm used and represents how well model performs on unseen data. A lower loss generally indicates a better performance.

	Precision	Recall	F1-score	Support
0	0.98	0.97	0.98	402
1	0.99	0.92	0.95	366
2	0.93	0.99	0.95	343
3	0.92	1.00	0.96	86
Accuracy			0.96	1197
Macro-avg	0.96	0.97	0.96	1197
Weighted avg	0.96	0.96	0.96	1197
Loss = 0.2106				

Table 6.2.1.2: Classification report of Random Forest Model

Classification report based on random forest model. Accuracy values ranging from 0.92 to 0.99 indicate that the model performs well in assigning class labels. However, there is a slight decrease compared to the previous model results, especially for categories 2 and 3. Recall values range from 0.92 (category 1) to 1.00 (category 3). Level 1 is a much steeper drop than before. This suggests that the model may be missing some relevant data points for Class 1. The F1 scores are both close to the corresponding precision and recall values, indicating a balance

between precision and recall, but are slightly lower than the previous model results. Category 3 has a significantly smaller number of data points (86) compared to the other supports (approximately 350-400). This imbalance affects the overall interpretation of the indicator. The model still performs well with an accuracy of 0.96 but is slightly down from before (0.97)

	precision	recall	f1-score	support
0	<b>0.96</b>	0.93	0.94	382
1	0.97	0.91	0.95	334
2	0.98	0.97	0.99	298
3	0.99	0.98	0.99	64
Accuracy			0.96	1078
Macro-avg	0.97	0.96	0.97	1078
Weighted avg	0.96	0.96	0.96	1078
Loss = 0.3127				

Table 6.2.1.3: Classification report of Neural Networks Model

The classification report reveals the performance of a neural network (NN) model designed for a hand gesture recognition task. With an overall accuracy of 0.96, the model showcases a strong ability to correctly classify gestures across various classes. Precision scores ranging from 0.96 to 0.99 indicate the model's precision in predicting each class, with class 2 and 3 achieving particularly high precision values. However, the recall values for class 1 show a slight drop to 0.91, suggesting that the model may struggle to identify all instances of class 1 accurately. The F1-scores, which balance precision and recall, remain relatively high across classes, especially for class 2 and 3. Despite the slightly reduced accuracy compared to previous models, the NN model maintains robust performance, as evidenced by its balanced F1-scores and overall accuracy. The loss value of 0.2413 further reflects the model's optimization during training, with lower values indicating better alignment between predicted and actual outputs.

**6.2.2 Result of Testing Hand Gestures control Smart Home**

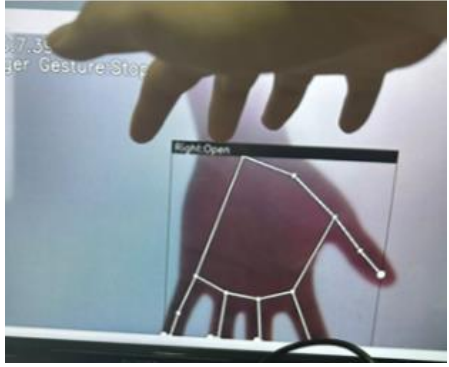

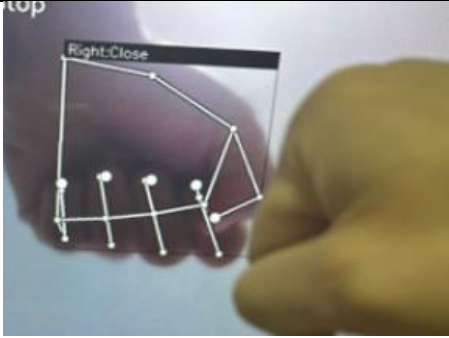

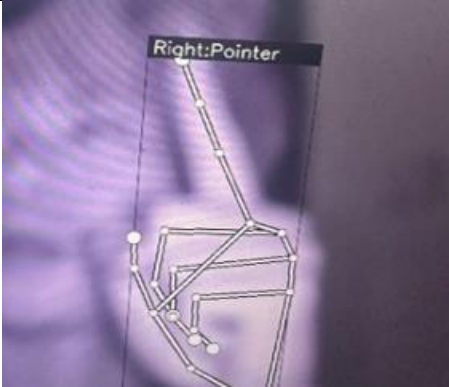

Input Hand Gestures	Action of smart home
	
	
	



Table 6.2.2.1: Show Result of Testing Hand Gestures Control Smart Home

The hand gestures recognition can successfully be detected, and the smart home can be controlled by using specify hand gestures.

### 6.2.3 Result of Face Recognition

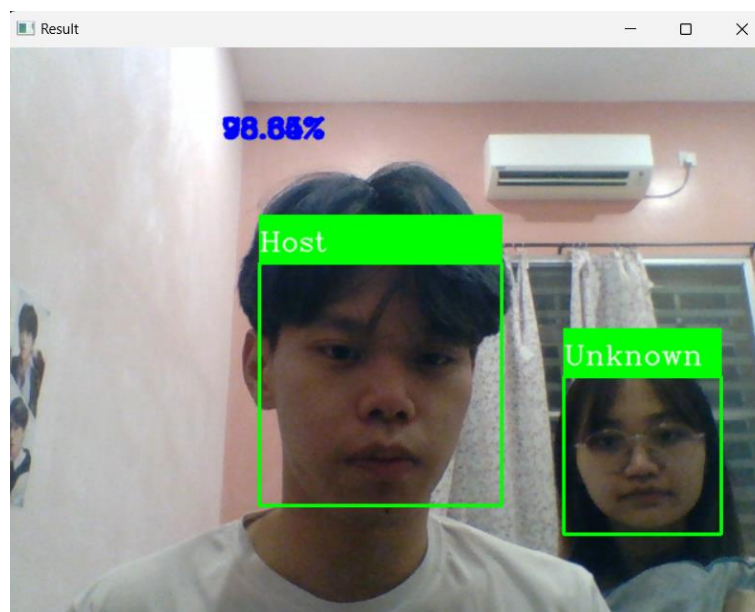


Figure 35: Result of face recognition system

Figure 18 show the result of face recognition system, because face recognition requires a lot of data to learn, and because everyone has a different face, this was outside the scope of the project. This simple facial recognition system only extracts a small amount of data, so it is not all that accurate. This face recognition is incorporated into gesture detection in my smart home. This face detection is to prevent there being no host in a room or that everyone in the room can control it, so only when the host is detected and can control it.

### 6.2.4 Result of Using cv2.convertScaleAbs() Functions

This function is used to change the contrast and brightness of an image. [19]

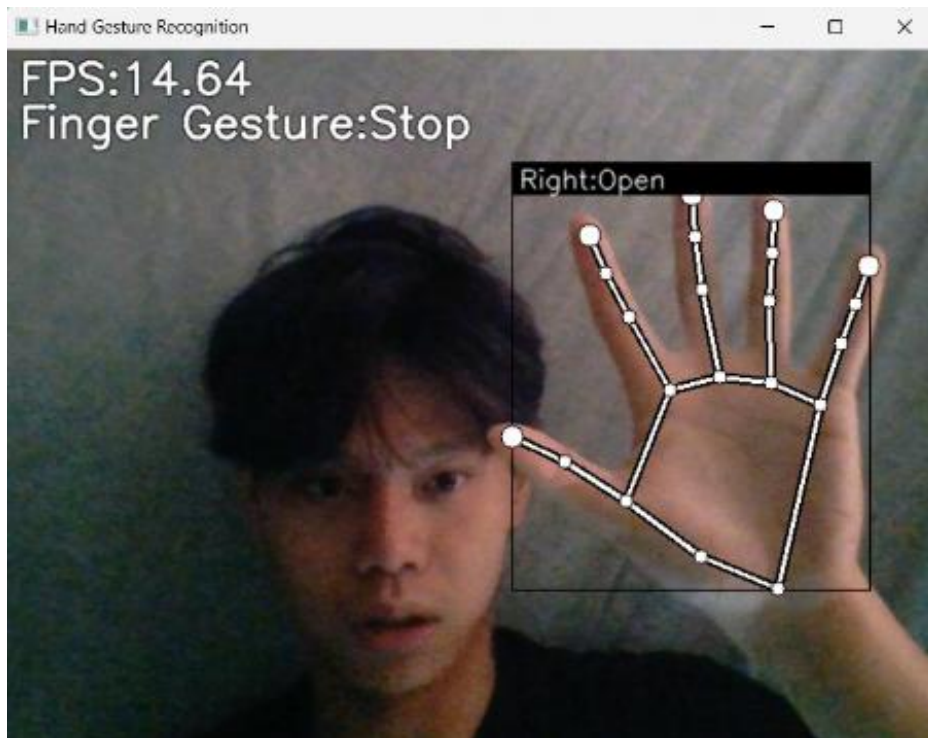


Figure 36: Normal environment and before using this function.

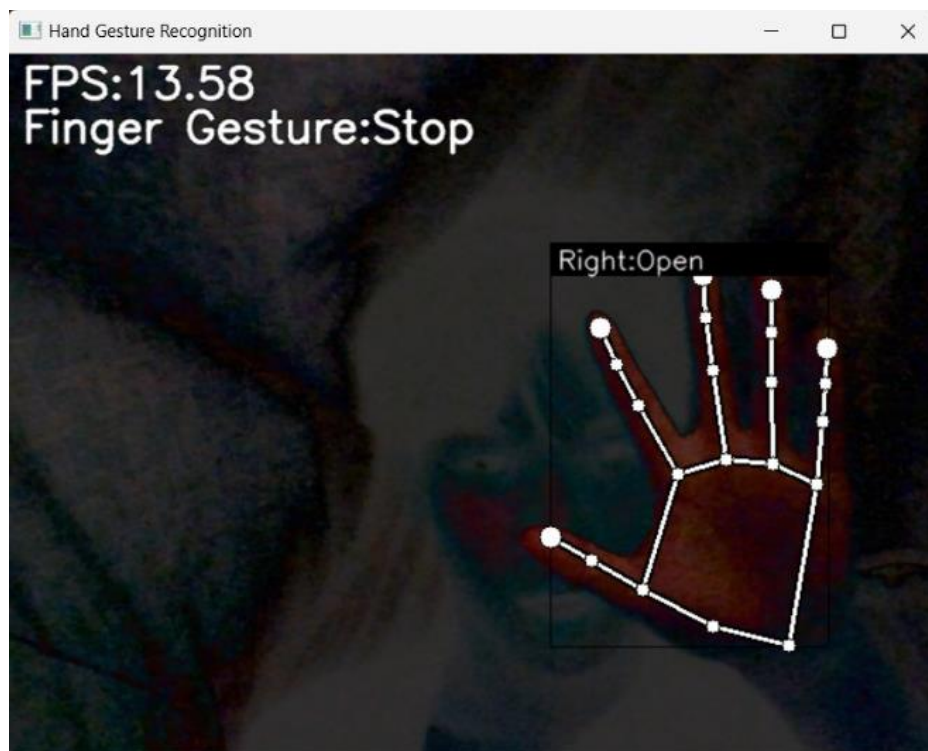


Figure 37: Result of using this function to tune the frame.

Syntax: `cv2.convertScaleAbs(image, alpha, beta)`

- Parameters:
- i. image – original input image
  - ii. alpha – optional scale factor that controls contrast.
  - iii. beta – optional delta added to the scaled values.

To adjust the contrast and brightness of an image using OpenCV, first, ensure you've imported the necessary library. Then, read the input image using the `cv2.imread()` method, specifying the full path of the image. Next, define the alpha parameter to control the contrast and the beta parameter to control the brightness. Use either the `cv2.convertScaleAbs()` function method to apply the desired adjustments to the image. `cv2.convertScaleAbs()` directly applies linear transformation to the image based on the specified alpha and beta, returning the adjusted image.

The `cv2.convertScaleAbs` function can adjust the brightness of the captured video to make it darker or brighter. From the figure 16 (left) is showing the normal brightness around 70-100, which is 8am to 6 pm. From the figure 16 (right), it simulates the brightness in the room when the lights are turned off between 7:30 and 7:45 pm. Even when the brightness is below 40 (where it is difficult to see the surrounding environment clearly), gestures can still be captured in low-light conditions using this method. It also depends on training the dataset more on nighttime, which could improve the speed and accuracy of gesture capture.

When using this function, set manual and automatic modes. In manual mode, users can adjust the brightness using the (+) and (-) buttons. Typically, in darker conditions, users will use (-) to lower the brightness, as using this method in the dark will further darken the picture until only black and white can be discerned. The effect is like using a night vision camera, effectively indirectly illuminating the video. In automatic mode, set the automatic method. This method automatically reduces the brightness of the video when it falls below a certain threshold, ensuring that gestures can be detected in any room lighting condition. Of course, this can be adjusted depending on the environment; for example, in bright sunlight we increase the brightness, like adjusting the brightness on a smartphone.



Figure 38: Result of using function cv2.convertScaleAbs

Hand Gestures	Brightness	Before	After
Open	Minimum (6)		
Close	Minimum (9)		
Pointer	Minimum (12)		

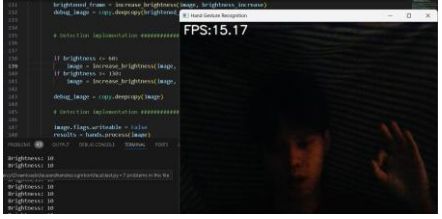

OK	Minimum (10)		
----	-----------------	---	--

Table 6.2.4.1 show the result after tuning.

### 6.3 Project Challenges

In this project, I faced a lot of challenges during this project which were hardware physical issues like broken jumpers, broken SD card readers. The camera version is not suitable with the Raspberry Pi, the power supply problem. I couldn't choose the best power supply for this project because some devices are AC, and some are DC devices. Finally, there are many versions of the Raspberry Pi operating system. For what kind of project, you must choose the correct version. When I came into the OpenCV part, I faced a frame error, this is because the camera cannot be satisfied with the frame size. Next, because the old version of the camera, the capture speed and FPS are very low, so sometimes, the opencv will display the hand-gestures that are detected error.

There are still many challenges encountered in this project, it needs to relate to more devices, such as: lights, speakers, etc., to make this project more intelligent and also create more control functions for the whole device, for example, not only Not only can it be turned on and off, but it can also control the brightness of the light. The next challenge is to add more gesture vocabulary so that users can use more gestures to control different devices, but this will also increase the difficulty of programming. Then, the next challenge is the environment, because users may use gestures to operate in various environments, so in order to adapt to customer operations in various environments, I will choose to use infrared cameras, because infrared cameras can use infrared lighting to capture images in low-light scenes or even complete darkness. These cameras detect heat emitted by objects and are very effective in challenging lighting conditions. However, this also requires reconfiguring the camera and then finding suitable infrared images from the frames detected by OpenCV.



The adaptability between software and hardware limits the scope of what I can do differently. For example, during model training, due to software version restrictions, the software version cannot be installed, resulting in failure, or the hardware is too old and cannot keep up with the new one. software.

### **6.4 Objective Evaluation**

The first objective of this project was to study the existing hand-gesture control smart home systems. To understanding the weaknesses and strengths of earlier systems, this involves conducting comprehensive research and analysis. By referring to the previous work, a new enhancement project was proposed a series of lighting problems, which can be solved from software or hardware. This objective was being achieve by replacing hardware or using better software tuning method to improve the entire detection capability.

Next, on the hardware side, the project uses a Raspberry Pi Night Vision to solve the lighting problem, allowing it to detect gestures even in low-light conditions. On the other hand (Software Side), it utilizes a specific adjustment method (`cv.convertScaleAbs`) to adjust frame brightness. This has effectively resolved the series of lighting issues, which can be solve from software and hardware.

Furthermore, hand-gestures control smart home system have also been successfully developed as the system allows users to control various smart home devices using specific gestures. For instance, performing the "On" gesture will turn on the fan, while performing the "Off" gesture will turn off the fan. Likewise, using the Pointer gesture will turn the LED light on and the OK gesture will turn it off.

Last but not least, to determine the best model for hand-gesture recognition, confusion metrics which indicated the performance metrics and model accuracy were evaluated by training 3 different types of models, which are NN (Neural Network), CNN (Convolutional Neural Network) and Random Forest in order to compare their effectiveness in different task.

## **Chapter 7: Conclusion**

### **7.1 Conclusion**

The Internet of Things is a rapidly growing technology that connects a variety of devices, equipment, and devices systems that enable them to communicate and exchange data over the Internet. IoT can also increase efficiency and reduce operating costs. Looking to the future, as the number of connected IoT devices continues to grow exponentially, IoT technology is expected to play an even more important role. With the emergence of technologies such as 5G and edge computing and the advancement of artificial intelligence, IoT systems will become smarter, and detection and perception speeds will become faster.

In this project direction, more devices that can be controlled and monitored will be added, not just switching devices, but also more in-depth operations, such as adjusting the brightness of light bulbs according to the size of gestures, making the entire system more intelligent. Then in the next plan, the application will be used in this project, allowing users to use mobile phones and computers to observe the status of smart home devices.

In summary, this project reviews existing smart home systems and gesture methods to explore basic IoT concepts and understand the entire traffic and communication protocols between IoT devices. Through this project, a new smart home system will be launched, using the monitoring system to capture gestures throughout the process, aiming to reduce energy consumption and promote intelligence. Smart homes reduce the need for direct physical control by humans using controllers or plugs. The development of this energy-efficient smart home solution has the potential to revolutionize smart home practices and contribute to a more sustainable future. This project has proven its effectiveness in reducing direct communication in smart home environments. Integrating IoT technology into smart homes can increase efficiency and reduce operating costs. This research therefore holds significant promise for transforming smart homes and ensuring their long-term survival. Going forward, mobile must continue to refine and expand these innovative solutions, foster collaboration among stakeholders, and facilitate widespread adoption of IoT-based smart home systems. By doing this, we can work together to build stronger, more efficient smart homes for people with disabilities.

Next, I will focus on how to make this smart home system more perfect, which is to connect more devices so that the entire smart home has more space for manipulation, and then add more gesture vocabulary, such as OK Gestures, pointer gestures, to make the whole project more practical, and finally solve the problem of lighting environment, because users may perform gestures in different environments, so in order to prevent users from using this smoothly in dark environments system, the camera will be changed, that is, the infrared camera, and then the camera can detect the user's gestures under any circumstances, making the entire project perfect.

### **7.2 Recommendation**

After completing this project, several comprehensive recommendations can be implemented to optimize and enhance gesture recognition for smart home systems. First, prioritize algorithm enhancements and continually refine machine learning algorithms to increase accuracy and eliminate false positives. Second, data augmentation methods are employed to improve model generalization capabilities and diversify training data, thereby improving accuracy and flexibility.

In addition, combined with the user feedback mechanism, the algorithm is adjusted to adapt to different environmental conditions to further improve accuracy and flexibility. Enhance user experience and system stability by implementing ongoing monitoring and maintenance procedures, user training programs, and customization options. Exploring the integration of gesture detection with voice control can provide consumers with a seamless and versatile interactive experience.

Overall, these recommendations aim to create a more precise, flexible, and intuitive smart home system. Investing in better hardware can enhance the system's fluidity and operability, while deeper learning and application in facial recognition can improve its accuracy and responsiveness. Additionally, expanding the gesture or facial dataset used for training will enhance the detection capabilities of the system.

Consider additional tuning methods to improve your system's detection and responsiveness, such as optimizing frame clarity. Furthermore, incorporating adjusted datasets into the training process can help the system better adapt to real-world scenarios. The extraction and training of relevant data can further improve the detection capabilities of the system.

## REFERENCES

- [1] Ankit Chaudhary et al., “Light invariant real-time robust hand gesture recognition,” *Optik*, <https://www.sciencedirect.com/science/article/abs/pii/S0030402617315784>
- [2] B. I. Alabdullah et al., “Smart home automation-based hand gesture recognition using feature fusion and recurrent neural network,” *MDPI*, <https://www.mdpi.com/1424-8220/23/17/7523>
- [3] B. Kurian, J. Regi, D. John, H. P and T. Y. Mahesh, "Visual Gesture-Based Home Automation," 2023 3rd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS), Kalady, Ernakulam, India, 2023, pp. 286-290, doi: 10.1109/ACCESS57397.2023.10200895.
- [4] C. Xu, Y. Jiang, J. Zhou, and Y. Liu, “Semi-supervised Joint Learning for hand gesture recognition from a single color image,” *MDPI*, <https://www.mdpi.com/1424-8220/21/3/1007#B7-sensors-21-01007>
- [5] G. Diraco, G. Rescio, A. Caroppo, A. Manni, and A. Leone, “Human action recognition in smart living services and applications: Context awareness, data availability, personalization, and privacy,” *MDPI*, <https://www.mdpi.com/1424-8220/23/13/6040>
- [6] J. Katti, A. Jadhav, A. Kulkarni, P. Nikam, and A. Pachange, “review paper on: Home Automation techniques based on hand gesture recognition,” *IJSR*, <https://issuu.com/irjet/docs/irjet-v8i8126>
- [7] L. Y. Rock, F. P. Tajudeen, and Y. W. Chung, “Usage and impact of the internet-of-things-based smart home technology: A quality-of-life perspective - universal access in the information society,” *SpringerLink*, <https://link.springer.com/article/10.1007/s10209-022-00937-0>
- [8] M. R. Khandakar et al., “Wireless Electric Appliance Control for smart buildings using indoor location tracking and BIM-based Virtual Environments,” *Automation in Construction*, <https://www.sciencedirect.com/science/article/abs/pii/S092658051830997X>

- [9] M. Oudah, A. Al-Naji, and J. Chahl, "Hand Gesture Recognition Based on Computer Vision: A Review of Techniques," <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8321080/>
- [10] M. Secure&trade;, "Hand gesture recognition sensor- A comprehensive guide," LinkedIn, <https://www.linkedin.com/pulse/hand-gesture-recognition-sensor-comprehensive-guide-medtechsecure>
- [11] M. Villanueva and O. Drögehorn, "Using gestures to interact with home automation systems: A socio-technical study on Motion Capture Technologies for Smart Homes," Accueil - Archive ouverte HAL, <https://hal.science/hal-02179898>
- [12] P. Vogiatzidakis and P. Koutsabasis, "Mid-air gesture control of multiple home devices in spatial augmented reality prototype," MDPI, <https://www.mdpi.com/2414-4088/4/3/61>
- [13] S. Imran, "Hand gesture recognition system for controlling home appliances," IJSR, <https://www.ijsr.net/archive/v8i8/ART20199651.pdf>
- [14] S. Kshirsagar, S. Sachdev, N. Singh, A. Tiwari and S. Sahu, "IoT Enabled Gesture-Controlled Home Automation for Disabled and Elderly," 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2020, pp. 821-826, doi: 10.1109/ICCMC48092.2020.ICCMC-000152.
- [15] S. Sarkar and A. Gade, "Smart and Energy Efficient Gesture Controlled Home Automation," IJSET, [https://ijset.com/vol3/v3s4/IJSET\\_V3\\_I4\\_09.pdf](https://ijset.com/vol3/v3s4/IJSET_V3_I4_09.pdf)
- [16] V. Govindraj, M. Sathiyarayanan and B. Abubakar, "Customary homes to smart homes using Internet of Things (IoT) and mobile application," 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon), Bengaluru, India, 2017, pp. 1059-1063, doi: 10.1109/SmartTechCon.2017.8358532.
- [17] Wojciech-Marusarz, "The challenges and opportunities of gesture recognition," nexocode, <https://nexocode.com/blog/posts/gestures-recognition-challenges-and-opportunities/#the-challenges-and-opportunities-of-gesture-recognition>
- [18] X. Guo, Z. Shen, Y. Zhang, and T. Wu, "Review on the application of Artificial Intelligence in smart homes," MDPI, <https://www.mdpi.com/2624-6511/2/3/25>

- [19] Shahid Akhtar Khan, “How to change the contrast and brightness of an image using opencv in python?,” Tutorialspoint, <https://www.tutorialspoint.com/how-to-change-the-contrast-and-brightness-of-an-image-using-opencv-in-python>
- [20] Real Python, “Image segmentation using color spaces in opencv + python,” Real Python, <https://realpython.com/python-opencv-color-spaces/>
- [21] S. Singh and S. V. A. V. Prasad, “Techniques and challenges of face recognition: A critical review,” Procedia Computer Science, vol. 143, pp. 536–543, 2018. doi:10.1016/j.procs.2018.10.427

## APPENDIX

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import csv
import copy
import argparse
import itertools
from collections import Counter
from collections import deque

import cv2 as cv
import numpy as np
import mediapipe as mp
```

```
from utils import CvFpsCalc
from model import KeyPointClassifier
from model import PointHistoryClassifier

def increase_brightness(frame, brightness_increase):
    new_brightness = brightness_increase
    if brightness_increase > 0:
        new_brightness = min(brightness_increase, 255)
    elif brightness_increase < 0:
        new_brightness = max(brightness_increase, -255)
    return cv.convertScaleAbs(frame, alpha=1, beta=new_brightness)

def get_args():
    parser = argparse.ArgumentParser()

    parser.add_argument("--device", type=int, default=0)
    parser.add_argument("--width", help='cap width', type=int, default=960)
    parser.add_argument("--height", help='cap height', type=int, default=540)

    parser.add_argument('--use_static_image_mode', action='store_true')
    parser.add_argument("--min_detection_confidence",
                        help='min_detection_confidence',
                        type=float,
                        default=0.7)
    parser.add_argument("--min_tracking_confidence",
                        help='min_tracking_confidence',
                        type=int,
                        default=0.5)

    args = parser.parse_args()

    return args
```

```

def main():
    #                                Argument                                parsing
    #####

    brightness_increase = 20 # Adjust this value as needed

    args = get_args()

    cap_device = args.device
    cap_width = args.width
    cap_height = args.height

    use_static_image_mode = args.use_static_image_mode
    min_detection_confidence = args.min_detection_confidence
    min_tracking_confidence = args.min_tracking_confidence

    use_brect = True

    #                                Camera                                preparation
    #####

    cap = cv.VideoCapture(cap_device)
    cap.set(cv.CAP_PROP_FRAME_WIDTH, cap_width)
    cap.set(cv.CAP_PROP_FRAME_HEIGHT, cap_height)

    #                                Model                                load
    #####

    mp_hands = mp.solutions.hands
    hands = mp_hands.Hands(
        static_image_mode=use_static_image_mode,
        max_num_hands=1,
        min_detection_confidence=min_detection_confidence,
        min_tracking_confidence=min_tracking_confidence,

```



```

)

keypoint_classifier = KeypointClassifier()

point_history_classifier = PointHistoryClassifier()

# Read labels #####
with open('model/keypoint_classifier/keypoint_classifier_label.csv',
          encoding='utf-8-sig') as f:
    keypoint_classifier_labels = csv.reader(f)
    keypoint_classifier_labels = [
        row[0] for row in keypoint_classifier_labels
    ]
with open(
    'model/point_history_classifier/point_history_classifier_label.csv',
    encoding='utf-8-sig') as f:
    point_history_classifier_labels = csv.reader(f)
    point_history_classifier_labels = [
        row[0] for row in point_history_classifier_labels
    ]

#                               FPS                               Measurement
#####

cvFpsCalc = CvFpsCalc(buffer_len=10)

#                               Coordinate                               history
#####

history_length = 16
point_history = deque(maxlen=history_length)

# Finger gesture history #####
finger_gesture_history = deque(maxlen=history_length)

```

```

#
#####

mode = 0

while True:
    fps = cvFpsCalc.get()

    #          Process          Key          (ESC:          end)
    #####

    key = cv.waitKey(10)
    if key == 27: # ESC
        break
    elif key == 107 and brightness_increase <= -255: # '+' key to increase brightness
        brightness_increase += 10
    elif key == 109 and brightness_increase >= -255: # '-' key to decrease brightness
        brightness_increase -= 10
    number, mode = select_mode(key, mode)

# Camera capture #####
ret, image = cap.read()
if not ret:
    break
gray_frame = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
brightness = int(gray_frame.mean())
image = cv.flip(image, 1) # Mirror display
brightened_frame = increase_brightness(image, brightness_increase)
debug_image = copy.deepcopy(brightened_frame)

#          Detection          implementation
#####

```

```

if brightness <= 60:
    image = increase_brightness(image, -0) # Decrease brightness by 100
if brightness >= 130:
    image = increase_brightness(image, -50)

debug_image = copy.deepcopy(image)

#                               Detection                               implementation
#####

image.flags.writeable = False
results = hands.process(image)
image.flags.writeable = True

#
#####

if results.multi_hand_landmarks is not None:
    for hand_landmarks, handedness in zip(results.multi_hand_landmarks,
        results.multi_handedness):
        # Bounding box calculation
        brect = calc_bounding_rect(debug_image, hand_landmarks)
        # Landmark calculation
        landmark_list = calc_landmark_list(debug_image, hand_landmarks)

        # Conversion to relative coordinates / normalized coordinates
        pre_processed_landmark_list = pre_process_landmark(
            landmark_list)
        pre_processed_point_history_list = pre_process_point_history(
            debug_image, point_history)
        # Write to the dataset file
        logging_csv(number, mode, pre_processed_landmark_list,
            pre_processed_point_history_list)

```

```

# Hand sign classification
hand_sign_id = keypoint_classifier(pre_processed_landmark_list)
if hand_sign_id == 2: # Point gesture
    point_history.append(landmark_list[8])
else:
    point_history.append([0, 0])

# Finger gesture classification
finger_gesture_id = 0
point_history_len = len(pre_processed_point_history_list)
if point_history_len == (history_length * 2):
    finger_gesture_id = point_history_classifier(
        pre_processed_point_history_list)

# Calculates the gesture IDs in the latest detection
finger_gesture_history.append(finger_gesture_id)
most_common_fg_id = Counter(
    finger_gesture_history).most_common()

# Drawing part
debug_image = draw_bounding_rect(use_brect, debug_image, brect)
debug_image = draw_landmarks(debug_image, landmark_list)
debug_image = draw_info_text(
    debug_image,
    brect,
    handedness,
    keypoint_classifier_labels[hand_sign_id],
    point_history_classifier_labels[most_common_fg_id[0][0]],
)
else:
    point_history.append([0, 0])

debug_image = draw_point_history(debug_image, point_history)

```

```

debug_image = draw_info(debug_image, fps, mode, number)

#                                Screen                                reflection
#####

cv.imshow('Hand Gesture Recognition', debug_image)
print('Brightness:', brightness)

cap.release()
cv.destroyAllWindows()

def select_mode(key, mode):
    number = -1
    if 48 <= key <= 57: # 0 ~ 9
        number = key - 48
    if key == 110: # n
        mode = 0
    if key == 107: # k
        mode = 1
    if key == 104: # h
        mode = 2
    return number, mode

def calc_bounding_rect(image, landmarks):
    image_width, image_height = image.shape[1], image.shape[0]

    landmark_array = np.empty((0, 2), int)

    for _, landmark in enumerate(landmarks.landmark):
        landmark_x = min(int(landmark.x * image_width), image_width - 1)
        landmark_y = min(int(landmark.y * image_height), image_height - 1)

```

```
    landmark_point = [np.array((landmark_x, landmark_y))]

    landmark_array = np.append(landmark_array, landmark_point, axis=0)

x, y, w, h = cv.boundingRect(landmark_array)

return [x, y, x + w, y + h]

def calc_landmark_list(image, landmarks):
    image_width, image_height = image.shape[1], image.shape[0]

    landmark_point = []

    # Keypoint
    for _, landmark in enumerate(landmarks.landmark):
        landmark_x = min(int(landmark.x * image_width), image_width - 1)
        landmark_y = min(int(landmark.y * image_height), image_height - 1)
        # landmark_z = landmark.z

        landmark_point.append([landmark_x, landmark_y])

    return landmark_point

def pre_process_landmark(landmark_list):
    temp_landmark_list = copy.deepcopy(landmark_list)

    # Convert to relative coordinates
    base_x, base_y = 0, 0

    for index, landmark_point in enumerate(temp_landmark_list):
        if index == 0:
            base_x, base_y = landmark_point[0], landmark_point[1]
```

```

temp_landmark_list[index][0] = temp_landmark_list[index][0] - base_x
temp_landmark_list[index][1] = temp_landmark_list[index][1] - base_y

# Convert to a one-dimensional list
temp_landmark_list = list(
    itertools.chain.from_iterable(temp_landmark_list))

# Normalization
max_value = max(list(map(abs, temp_landmark_list)))

def normalize_(n):
    return n / max_value

temp_landmark_list = list(map(normalize_, temp_landmark_list))

return temp_landmark_list

def pre_process_point_history(image, point_history):
    image_width, image_height = image.shape[1], image.shape[0]

    temp_point_history = copy.deepcopy(point_history)

    # Convert to relative coordinates
    base_x, base_y = 0, 0
    for index, point in enumerate(temp_point_history):
        if index == 0:
            base_x, base_y = point[0], point[1]

        temp_point_history[index][0] = (temp_point_history[index][0] -
            base_x) / image_width
        temp_point_history[index][1] = (temp_point_history[index][1] -

```

```

        base_y) / image_height

# Convert to a one-dimensional list
temp_point_history = list(
    itertools.chain.from_iterable(temp_point_history))

return temp_point_history

def logging_csv(number, mode, landmark_list, point_history_list):
    if mode == 0:
        pass
    if mode == 1 and (0 <= number <= 9):
        csv_path = 'model/keypoint_classifier/keypoint.csv'
        with open(csv_path, 'a', newline='') as f:
            writer = csv.writer(f)
            writer.writerow([number, *landmark_list])
    if mode == 2 and (0 <= number <= 9):
        csv_path = 'model/point_history_classifier/point_history.csv'
        with open(csv_path, 'a', newline='') as f:
            writer = csv.writer(f)
            writer.writerow([number, *point_history_list])
    return

def draw_landmarks(image, landmark_point):
    if len(landmark_point) > 0:
        # Thumb
        cv.line(image, tuple(landmark_point[2]), tuple(landmark_point[3]),
            (0, 0, 0), 6)
        cv.line(image, tuple(landmark_point[2]), tuple(landmark_point[3]),
            (255, 255, 255), 2)
        cv.line(image, tuple(landmark_point[3]), tuple(landmark_point[4]),

```



```
(0, 0, 0), 6)
cv.line(image, tuple(landmark_point[3]), tuple(landmark_point[4]),
        (255, 255, 255), 2)

# Index finger
cv.line(image, tuple(landmark_point[5]), tuple(landmark_point[6]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[5]), tuple(landmark_point[6]),
        (255, 255, 255), 2)
cv.line(image, tuple(landmark_point[6]), tuple(landmark_point[7]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[6]), tuple(landmark_point[7]),
        (255, 255, 255), 2)
cv.line(image, tuple(landmark_point[7]), tuple(landmark_point[8]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[7]), tuple(landmark_point[8]),
        (255, 255, 255), 2)

# Middle finger
cv.line(image, tuple(landmark_point[9]), tuple(landmark_point[10]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[9]), tuple(landmark_point[10]),
        (255, 255, 255), 2)
cv.line(image, tuple(landmark_point[10]), tuple(landmark_point[11]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[10]), tuple(landmark_point[11]),
        (255, 255, 255), 2)
cv.line(image, tuple(landmark_point[11]), tuple(landmark_point[12]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[11]), tuple(landmark_point[12]),
        (255, 255, 255), 2)

# Ring finger
```

```
cv.line(image, tuple(landmark_point[13]), tuple(landmark_point[14]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[13]), tuple(landmark_point[14]),
        (255, 255, 255), 2)
cv.line(image, tuple(landmark_point[14]), tuple(landmark_point[15]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[14]), tuple(landmark_point[15]),
        (255, 255, 255), 2)
cv.line(image, tuple(landmark_point[15]), tuple(landmark_point[16]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[15]), tuple(landmark_point[16]),
        (255, 255, 255), 2)

# Little finger
cv.line(image, tuple(landmark_point[17]), tuple(landmark_point[18]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[17]), tuple(landmark_point[18]),
        (255, 255, 255), 2)
cv.line(image, tuple(landmark_point[18]), tuple(landmark_point[19]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[18]), tuple(landmark_point[19]),
        (255, 255, 255), 2)
cv.line(image, tuple(landmark_point[19]), tuple(landmark_point[20]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[19]), tuple(landmark_point[20]),
        (255, 255, 255), 2)

# Palm
cv.line(image, tuple(landmark_point[0]), tuple(landmark_point[1]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[0]), tuple(landmark_point[1]),
        (255, 255, 255), 2)
cv.line(image, tuple(landmark_point[1]), tuple(landmark_point[2]),
```

```

    (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[1]), tuple(landmark_point[2]),
        (255, 255, 255), 2)
cv.line(image, tuple(landmark_point[2]), tuple(landmark_point[5]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[2]), tuple(landmark_point[5]),
        (255, 255, 255), 2)
cv.line(image, tuple(landmark_point[5]), tuple(landmark_point[9]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[5]), tuple(landmark_point[9]),
        (255, 255, 255), 2)
cv.line(image, tuple(landmark_point[9]), tuple(landmark_point[13]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[9]), tuple(landmark_point[13]),
        (255, 255, 255), 2)
cv.line(image, tuple(landmark_point[13]), tuple(landmark_point[17]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[13]), tuple(landmark_point[17]),
        (255, 255, 255), 2)
cv.line(image, tuple(landmark_point[17]), tuple(landmark_point[0]),
        (0, 0, 0), 6)
cv.line(image, tuple(landmark_point[17]), tuple(landmark_point[0]),
        (255, 255, 255), 2)

```

### # Key Points

```
for index, landmark in enumerate(landmark_point):
```

```
    if index == 0: # 手首 1
```

```
        cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
                -1)
```

```
        cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
```

```
    if index == 1: # 手首 2
```

```
        cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
```

```

-1)
cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
if index == 2: # 親指: 付け根
    cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
-1)
cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
if index == 3: # 親指: 第 1 関節
    cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
-1)
cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
if index == 4: # 親指: 指先
    cv.circle(image, (landmark[0], landmark[1]), 8, (255, 255, 255),
-1)
cv.circle(image, (landmark[0], landmark[1]), 8, (0, 0, 0), 1)
if index == 5: # 人差指: 付け根
    cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
-1)
cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
if index == 6: # 人差指: 第 2 関節
    cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
-1)
cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
if index == 7: # 人差指: 第 1 関節
    cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
-1)
cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
if index == 8: # 人差指: 指先
    cv.circle(image, (landmark[0], landmark[1]), 8, (255, 255, 255),
-1)
cv.circle(image, (landmark[0], landmark[1]), 8, (0, 0, 0), 1)
if index == 9: # 中指: 付け根
    cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),

```

```

-1)
cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
if index == 10: # 中指: 第 2 關節
    cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
-1)
cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
if index == 11: # 中指: 第 1 關節
    cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
-1)
cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
if index == 12: # 中指: 指先
    cv.circle(image, (landmark[0], landmark[1]), 8, (255, 255, 255),
-1)
cv.circle(image, (landmark[0], landmark[1]), 8, (0, 0, 0), 1)
if index == 13: # 薬指: 付け根
    cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
-1)
cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
if index == 14: # 薬指: 第 2 關節
    cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
-1)
cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
if index == 15: # 薬指: 第 1 關節
    cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
-1)
cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
if index == 16: # 薬指: 指先
    cv.circle(image, (landmark[0], landmark[1]), 8, (255, 255, 255),
-1)
cv.circle(image, (landmark[0], landmark[1]), 8, (0, 0, 0), 1)
if index == 17: # 小指: 付け根
    cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),

```

```

        -1)
        cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
    if index == 18: # 小指: 第 2 關節
        cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
            -1)
        cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
    if index == 19: # 小指: 第 1 關節
        cv.circle(image, (landmark[0], landmark[1]), 5, (255, 255, 255),
            -1)
        cv.circle(image, (landmark[0], landmark[1]), 5, (0, 0, 0), 1)
    if index == 20: # 小指: 指先
        cv.circle(image, (landmark[0], landmark[1]), 8, (255, 255, 255),
            -1)
        cv.circle(image, (landmark[0], landmark[1]), 8, (0, 0, 0), 1)

    return image

def draw_bounding_rect(use_brect, image, brect):
    if use_brect:
        # Outer rectangle
        cv.rectangle(image, (brect[0], brect[1]), (brect[2], brect[3]),
            (0, 0, 0), 1)

    return image

def draw_info_text(image, brect, handedness, hand_sign_text,
    finger_gesture_text):
    cv.rectangle(image, (brect[0], brect[1]), (brect[2], brect[1] - 22),
        (0, 0, 0), -1)

```

```

info_text = handedness.classification[0].label[0:]
if hand_sign_text != '':
    info_text = info_text + ':' + hand_sign_text
cv.putText(image, info_text, (brect[0] + 5, brect[1] - 4),
    cv.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 1, cv.LINE_AA)

if finger_gesture_text != '':
    cv.putText(image, "Finger Gesture:" + finger_gesture_text, (10, 60),
    cv.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 0), 4, cv.LINE_AA)
    cv.putText(image, "Finger Gesture:" + finger_gesture_text, (10, 60),
    cv.FONT_HERSHEY_SIMPLEX, 1.0, (255, 255, 255), 2,
    cv.LINE_AA)

return image

def draw_point_history(image, point_history):
for index, point in enumerate(point_history):
    if point[0] != 0 and point[1] != 0:
        cv.circle(image, (point[0], point[1]), 1 + int(index / 2),
        (152, 251, 152), 2)

return image

def draw_info(image, fps, mode, number):
    cv.putText(image, "FPS:" + str(fps), (10, 30), cv.FONT_HERSHEY_SIMPLEX,
    1.0, (0, 0, 0), 4, cv.LINE_AA)
    cv.putText(image, "FPS:" + str(fps), (10, 30), cv.FONT_HERSHEY_SIMPLEX,
    1.0, (255, 255, 255), 2, cv.LINE_AA)

mode_string = ['Logging Key Point', 'Logging Point History']
if 1 <= mode <= 2:
    cv.putText(image, "MODE:" + mode_string[mode - 1], (10, 90),
    cv.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 1,
    cv.LINE_AA)

```

```

if 0 <= number <= 9:
    cv.putText(image, "NUM:" + str(number), (10, 110),
               cv.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 1,
               cv.LINE_AA)
return image

if __name__ == '__main__':
    main()

Below is for Face Recognition
import tensorflow as tf
from tensorflow import keras
import numpy as np
import cv2
from keras.models import load_model
import numpy as np
import subprocess

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

cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
font = cv2.FONT_HERSHEY_COMPLEX

model = load_model('keras_model.h5')

def get_className(classNo):
    if classNo == 0:
        return "Host"
    elif classNo == 1:
        return "Unknown"

```



```

def detect_host(imgOriginal):
    faces = facedetect.detectMultiScale(imgOriginal, 1.3, 5)
    for x, y, w, h in faces:
        crop_img = imgOriginal[y:y + h, x:x + h]
        img = cv2.resize(crop_img, (224, 224))
        img = img.reshape(1, 224, 224, 3)
        prediction = model.predict(img)
        classIndex = np.argmax(prediction, axis=-1) # Using argmax to get the class
index
        probabilityValue = np.amax(prediction)
        if classIndex == 0:
            cap.release()
            cv2.destroyAllWindows() # Close the current window
            subprocess.run(['python', 'app.py']) # Run app.py using subprocess
            exit() # Exit the script after running app.py
        return probabilityValue # Return probabilityValue from the function

while True:
    success, imgOriginal = cap.read()
    probabilityValue = detect_host(imgOriginal) # Get probabilityValue from
detect_host function
    if probabilityValue is not None:
        cv2.putText(imgOriginal, str(round(probabilityValue * 100, 2)) + "%", (180,
75), font, 0.75, (255, 0, 0), 2, cv2.LINE_AA)
        cv2.imshow('Result', imgOriginal)
        k = cv2.waitKey(1)
        if k == ord('q'):
            break

cap.release()
cv2.destroyAllWindows()

```

## FINAL YEAR PROJECT WEEKLY REPORT WEEK 2

*(Project II)*

<b>Trimester, Year: January 2024</b>	<b>Study week no.: 2</b>
<b>Student Name &amp; ID: TAN TECK SHENG (20ACB03875)</b>	
<b>Supervisor: MS. TSEU KWAN LEE</b>	
<b>Project Title: HANDGESTURESENSE: HAND GESTURE RECOGNITION-CONTROLLED SMART HOME AUTOMATION</b>	

<p><b>1. WORK DONE</b></p> <p>Make a 3D scatter plot for extract brightness.</p>
<p><b>2. WORK TO BE DONE</b></p>
<p><b>3. PROBLEMS ENCOUNTERED</b></p>
<p><b>4. SELF EVALUATION OF THE PROGRESS</b></p> <p>Not satisfied</p>



\_\_\_\_\_  
Supervisor's signature



\_\_\_\_\_  
Student's signature

**FINAL YEAR PROJECT WEEKLY REPORT WEEK 4***(Project II)*

<b>Trimester, Year: January 2024</b>	<b>Study week no.: 4</b>
<b>Student Name &amp; ID: TAN TECK SHENG (20ACB03875)</b>	
<b>Supervisor: MS. TSEU KWAN LEE</b>	
<b>Project Title: HANDGESTURESENSE: HAND GESTURE RECOGNITION-CONTROLLED SMART HOME AUTOMATION</b>	

<p><b>1. WORK DONE</b></p> <p>Make a function that can manual and auto control brightness.</p>
<p><b>2. WORK TO BE DONE</b></p>
<p><b>3. PROBLEMS ENCOUNTERED</b></p>
<p><b>4. SELF EVALUATION OF THE PROGRESS</b></p> <p>Not satisfied</p>



\_\_\_\_\_  
Supervisor's signature



\_\_\_\_\_  
Student's signature

**FINAL YEAR PROJECT WEEKLY REPORT WEEK 6***(Project II)*

<b>Trimester, Year: January 2024</b>	<b>Study week no.: 6</b>
<b>Student Name &amp; ID: TAN TECK SHENG (20ACB03875)</b>	
<b>Supervisor: MS. TSEU KWAN LEE</b>	
<b>Project Title: HANDGESTURESENSE: HAND GESTURE RECOGNITION-CONTROLLED SMART HOME AUTOMATION</b>	

<p><b>1. WORK DONE</b></p> <p>Train the hand gestures recognition using 3 model, CNN, NN and RandomForest</p>
<p><b>2. WORK TO BE DONE</b></p>
<p><b>3. PROBLEMS ENCOUNTERED</b></p>
<p><b>4. SELF EVALUATION OF THE PROGRESS</b></p> <p>Not satisfied</p>



\_\_\_\_\_  
Supervisor's signature



\_\_\_\_\_  
Student's signature

**FINAL YEAR PROJECT WEEKLY REPORT WEEK 8**

*(Project II)*

<b>Trimester, Year: January 2024</b>	<b>Study week no.: 8</b>
<b>Student Name &amp; ID: TAN TECK SHENG (20ACB03875)</b>	
<b>Supervisor: MS. TSEU KWAN LEE</b>	
<b>Project Title: HANDGESTURESENSE: HAND GESTURE RECOGNITION-CONTROLLED SMART HOME AUTOMATION</b>	

<b>1. WORK DONE</b> Make a face recognition system.
<b>2. WORK TO BE DONE</b>
<b>3. PROBLEMS ENCOUNTERED</b>
<b>4. SELF EVALUATION OF THE PROGRESS</b> Not satisfied



\_\_\_\_\_  
Supervisor's signature




\_\_\_\_\_  
Student's signature


**FINAL YEAR PROJECT WEEKLY REPORT WEEK 10**

*(Project II)*

<b>Trimester, Year: January 2024</b>	<b>Study week no.: 10</b>
<b>Student Name &amp; ID: TAN TECK SHENG (20ACB03875)</b>	
<b>Supervisor: MS. TSEU KWAN LEE</b>	
<b>Project Title: HANDGESTURESENSE: HAND GESTURE RECOGNITION-CONTROLLED SMART HOME AUTOMATION</b>	

<p><b>1. WORK DONE</b></p> <p>Try using increase brightness function to tune the frame</p>
<p><b>2. WORK TO BE DONE</b></p>
<p><b>3. PROBLEMS ENCOUNTERED</b></p>
<p><b>4. SELF EVALUATION OF THE PROGRESS</b></p> <p>Not satisfied</p>

  
 \_\_\_\_\_  
 Supervisor's signature

  
 \_\_\_\_\_  
 Student's signature

**FINAL YEAR PROJECT WEEKLY REPORT WEEK 12***(Project II)*

<b>Trimester, Year: January 2024</b>	<b>Study week no.: 12</b>
<b>Student Name &amp; ID: TAN TECK SHENG (20ACB03875)</b>	
<b>Supervisor: MS. TSEU KWAN LEE</b>	
<b>Project Title: HANDGESTURESENSE: HAND GESTURE RECOGNITION-CONTROLLED SMART HOME AUTOMATION</b>	

**1. WORK DONE**

Work done on reports and send to supervisor.

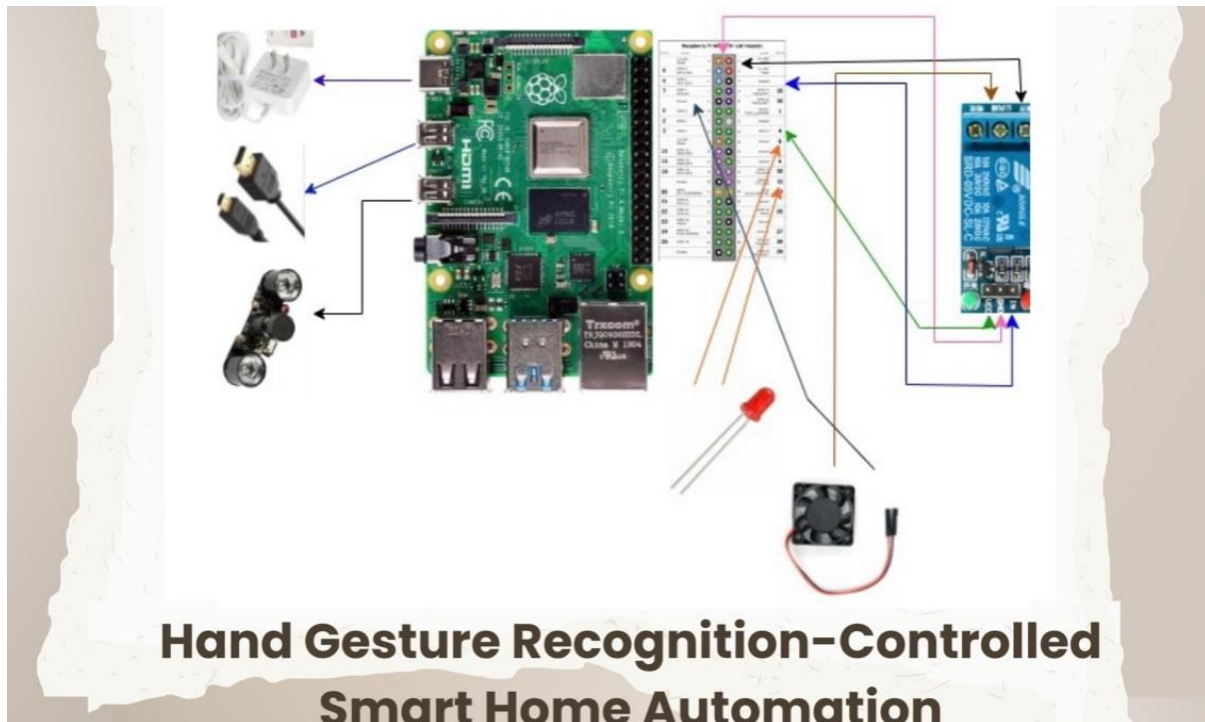
**2. WORK TO BE DONE****3. PROBLEMS ENCOUNTERED****4. SELF EVALUATION OF THE PROGRESS**

Not satisfied


\_\_\_\_\_  
Supervisor's signature

\_\_\_\_\_  
Student's signature

## POSTER



## Hand Gesture Recognition-Controlled Smart Home Automation

### INTRODUCTION

In recent years, the concept of smart home automation has emerged as a promising solution to enhance convenience, efficiency, and overall comfort for residents. However, traditional methods of managing smart devices, such as switches and mobile apps, often lack seamless interaction between humans and technology.

### OBJECTIVE

- Investigate existing smart home and gesture detection technologies for automatic hand detection.
- Analyze advantages and disadvantages of previous work to inform project improvements.

### MOTIVATION

#### Motivation 1 - Gesture-Controlled Smart Home:

- Emphasis on user experience.
- Allows control of lighting, temperature, entertainment, and security systems with hand gestures.
- Simplifies daily routines, enhances efficiency.

#### Motivation 2 - User Feedback:

- Clear feedback upon recognizing and executing gestures.
- Visual, auditory, or haptic cues reinforce the user's sense of control.





## PLAGIARISM CHECK RESULT

Forturnitin3.docx

## ORIGINALITY REPORT

<b>11</b> %	<b>10</b> %	<b>5</b> %	<b>0</b> %
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

## PRIMARY SOURCES

<b>1</b>	<a href="http://eprints.utar.edu.my">eprints.utar.edu.my</a> Internet Source	<b>2</b> %
<b>2</b>	<a href="http://www.mdpi.com">www.mdpi.com</a> Internet Source	<b>1</b> %
<b>3</b>	<a href="http://www.papercamp.com">www.papercamp.com</a> Internet Source	<b>1</b> %
<b>4</b>	<a href="http://www.coursehero.com">www.coursehero.com</a> Internet Source	<b>1</b> %
<b>5</b>	<a href="http://fastercapital.com">fastercapital.com</a> Internet Source	<b>1</b> %
<b>6</b>	<a href="http://github.com">github.com</a> Internet Source	<b>&lt;1</b> %
<b>7</b>	<a href="http://ia902201.us.archive.org">ia902201.us.archive.org</a> Internet Source	<b>&lt;1</b> %
<b>8</b>	"Inventive Communication and Computational Technologies", Springer Science and Business Media LLC, 2023 Publication	<b>&lt;1</b> %

[www.smallake.kr](http://www.smallake.kr)

9	Internet Source	<1 %
10	<a href="http://www.analyticsvidhya.com">www.analyticsvidhya.com</a> Internet Source	<1 %
11	<a href="http://fict.utar.edu.my">fict.utar.edu.my</a> Internet Source	<1 %
12	<a href="http://philpapers.org">philpapers.org</a> Internet Source	<1 %
13	<a href="http://peerj.com">peerj.com</a> Internet Source	<1 %
14	<a href="http://www.uctronics.com">www.uctronics.com</a> Internet Source	<1 %
15	<a href="http://coek.info">coek.info</a> Internet Source	<1 %
16	<a href="http://arxiv.org">arxiv.org</a> Internet Source	<1 %
17	S. H. Karamchandani, U. B. Desai, S. N. Merchant, G. D. Jindal. "Parallel support vector architectures for taxonomy of radial pulse morphology", Signal, Image and Video Processing, 2012 Publication	<1 %
18	<a href="http://nanopdf.com">nanopdf.com</a> Internet Source	<1 %
	<a href="http://dokumen.pub">dokumen.pub</a>	

19	Internet Source	<1 %
20	"Advances in Data and Information Sciences", Springer Science and Business Media LLC, 2024 Publication	<1 %
21	hal.archives-ouvertes.fr Internet Source	<1 %
22	123dok.com Internet Source	<1 %
23	Muslem Al-Saidi, Áron Ballagi, Oday Ali Hassen, Saad M. Saad. "Cognitive Classifier of Hand Gesture Images for Automated Sign Language Recognition: Soft Robot Assistance Based on Neutrosophic Markov Chain Paradigm", Computers, 2024 Publication	<1 %
24	eudl.eu Internet Source	<1 %
25	www.ijraset.com Internet Source	<1 %
26	www.ncbi.nlm.nih.gov Internet Source	<1 %
27	Du, C.J.. "Recent developments in the applications of image processing techniques	<1 %

for food quality evaluation", Trends in Food  
Science & Technology, 200405

Publication

28	Hritwik Ghosh, Irfan Sadiq Rahat, Sachi Nandan Mohanty, Janjhyam Venkata Naga Ramesh. "Microbial Image Deciphering: Navigating Challenges with Machine and Deep Learning", Research Square Platform LLC, 2023	<1 %
29	Wafae Abbaoui, Sara Retal, Soumia Ziti, Brahim El Bhiri. "Automated Ischemic Stroke Classification from MRI Scans: Using a Vision Transformer Approach", Journal of Clinical Medicine, 2024	<1 %
30	www.freedomtoascend.com	<1 %
31	mdpi.com	<1 %
32	ethesis.nitrkl.ac.in	<1 %
33	0-www-mdpi-com.brum.beds.ac.uk	<1 %
34	es.scribd.com	<1 %

35	<a href="https://issuu.com">issuu.com</a> Internet Source	<1 %
36	<a href="http://www.iotforall.com">www.iotforall.com</a> Internet Source	<1 %
37	"Image Processing and Capsule Networks", Springer Science and Business Media LLC, 2021 Publication	<1 %
38	Armitage, Andy. "EBOOK: Teaching in Post-14 Education & Training", EBOOK: Teaching in Post-14 Education & Training, 2016 Publication	<1 %
39	Divya Alok Gupta, C.Y.N. Dwith, B. Aditya Vighnesh Ramakanth. "Wireless Home Automation using social networking websites", 20th Annual International Conference on Advanced Computing and Communications (ADCOM), 2014 Publication	<1 %
40	Raghad K. Mohammed, Azmi Tawfeq Hussein Alrawi, Ali Jbaeer Dawood. "U-Net for genomic sequencing: A novel approach to DNA sequence classification", Alexandria Engineering Journal, 2024 Publication	<1 %
41	<a href="https://assets.researchsquare.com">assets.researchsquare.com</a> Internet Source	<1 %

---

42	<a href="https://community.dfrobot.com">community.dfrobot.com</a> Internet Source	<1 %
43	<a href="https://medium.com">medium.com</a> Internet Source	<1 %
44	<a href="https://mohitagr18.github.io">mohitagr18.github.io</a> Internet Source	<1 %
45	<a href="https://umldiagramtutorial.blogspot.com">umldiagramtutorial.blogspot.com</a> Internet Source	<1 %
46	<a href="https://www.researchsquare.com">www.researchsquare.com</a> Internet Source	<1 %
47	<a href="https://www.scopus.com">www.scopus.com</a> Internet Source	<1 %
48	Jeff Cicolani. "Beginning Robotics with Raspberry Pi and Arduino", Springer Science and Business Media LLC, 2021 Publication	<1 %
49	Yu, . "System Prototype and Performance Evaluation", Series in Electrical and Computer Engineering, 2011. Publication	<1 %
50	Bayan Ibrahim Alabdullah, Hira Ansar, Naif Al Mudawi, Abdulwahab Alazeb, Abdullah Alshahrani, Saud S. Alotaibi, Ahmad Jalal. "Smart Home Automation-Based Hand Gesture Recognition Using Feature Fusion	<1 %

and Recurrent Neural Network", Sensors,  
2023

Publication

---

**51** Takehiro Niikura, Yoshihiro Watanabe,  
Masatoshi Ishikawa. "Anywhere surface  
touch", Proceedings of the 5th Augmented  
Human International Conference, 2014

Publication

<1%

---

**52** [staff.utar.edu.my](http://staff.utar.edu.my)

Internet Source

<1%

---

**53** [www.hackster.io](http://www.hackster.io)

Internet Source

<1%

---

Exclude quotes On

Exclude matches Off

Exclude bibliography On

<b>Universiti Tunku Abdul Rahman</b>			
<b>Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)</b>			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date:	Page No.: 1 of 1



**FACULTY OF INFORMATION AND COMMUNICATION  
TECHNOLOGY**

<b>Full Name(s) of</b>	TAN TECK SHENG
<b>ID Number(s)</b>	20ACB03875
<b>Programme / Course</b>	CS
<b>Title of Final Year Project</b>	HANDGESTURESENSE: HAND GESTURE RECOGNITION- CONTROLLED SMART HOME AUTOMATION

<b>Similarity</b>	<b>Supervisor's Comments (Compulsory if parameters of originality)</b>
<b>Overall similarity index: _____</b> <b>11 _____ % Similarity by source</b> Internet Sources: _____10_____ % Publications: _____ 5 _____ % Student Papers: _____ 0 _____ %	OK
<b>Number of individual sources listed</b>	
<b>Parameters of originality required and limits approved by UTAR are as Follows:</b> (i) <b>Overall similarity index is 20% and below, and</b> (ii) <b>Matching of individual sources listed must be less than 3% each, and</b>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute



CHAPTER 7

*Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.*



\_\_\_\_\_  
Signature of Supervisor

Name: Tseu Kwan Lee

Date:

2024.04.26

\_\_\_\_\_  
Signature of Co-Supervisor

Name:

\_\_\_\_\_

Date:

\_\_\_\_\_



**UNIVERSITI TUNKU ABDUL RAHMAN**

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR  
CAMPUS)

**CHECKLIST FOR FYP2 THESIS SUBMISSION**

Student Id	20ACB03875
Student Name	TAN TECK SHENG
Supervisor Name	MS TSEU KWAN LEE

<b>TICK (√)</b>	<b>DOCUMENT ITEMS</b>
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
√	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster

CHAPTER 7

√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

\*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.



\_\_\_\_\_  
(Signature of Student)

Date:20/4/2024