

**HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY OF INTERNATIONAL EDUCATION**



**HCMUTE**

**GRADUATION PROJECT**

**FACIAL RECOGNITION AND HAND GESTURES TO  
CONTROL HOME APPLIANCES**

**LÊ THỊ THANH THU'**

**Student ID: 18119044**

**NGUYỄN TRỌNG HẢI**

**Student ID: 19119028**

**Major: COMPUTER ENGINEERING**

**Advisor: ĐỖ DUY TÂN, PhD.**

**Ho Chi Minh City, June 2024**

**HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION**  
**FACULTY OF INTERNATIONAL EDUCATION**



**GRADUATION PROJECT**

**FACIAL RECOGNITION AND HAND GESTURES TO  
CONTROL HOME APPLIANCES**

**LÊ THỊ THANH THU'**

**Student ID: 18119044**

**NGUYỄN TRỌNG HẢI**

**Student ID: 19119028**

**Major: COMPUTER ENGINEERING**

**Advisor: ĐỖ DUY TÂN, PhD.**

**Ho Chi Minh City, June 2024**



THE SOCIALIST REPUBLIC OF VIETNAM  
**Independence – Freedom – Happiness**

-----  
Ho Chi Minh City, June 20, 2024

## GRADUATION PROJECT ASSIGNMENT

Student name: Lê Thị Thanh Thu	Student ID: 18119044
Student name: Nguyễn Trọng Hải	Student ID: 19119028
Major: Computer Engineering Technology	Class: 18119CLA
Advisor: PhD. Đỗ Duy Tân	Email: tandd@hcmute.edu.vn
Date of assignment: February 28 <sup>th</sup> , 2024	Date of submission: July 14 <sup>th</sup> , 2024

1. Project title: Facial Recognition And Hand Gestures To Control Home Appliances.
2. Initial materials provided by the advisor: Documents such as paper about application of IoT based home appliance control system.
3. Content of the project:
  - Evaluate the project's challenges. Explore the technical specifications, guiding principles, and theoretical foundations of the hardware components.
  - Propose the model and provide an overview of the entire system. Create a block diagram and outline the operating principle. Configure the system and design the hardware.
  - Test and evaluate the performance.
4. Final product: A realistic home system model, system control software, final report, and demo video.

**CHAIR OF THE PROGRAM**  
(Sign with full name)

**ADVISOR**  
(Sign with full name)



THE SOCIALIST REPUBLIC OF VIETNAM  
Independence – Freedom – Happiness  
-----

Ho Chi Minh City, June 20, 2024

## ADVISOR'S EVALUATION SHEET

Student name: Lê Thị Thanh Thu

Student ID: 18119044

Student name: Nguyễn Trọng Hải

Student ID: 19119028

Major: Computer Engineering Technology

Project title: Facial Recognition And Hand Gestures To Control Home Appliances

Advisor: PhD. Đỗ Duy Tân

### EVALUATION

1. Content of the project:

- Evaluate the project's challenges. Explore the technical specifications, guiding principles, and theoretical foundations of the hardware components.
- Propose the model and provide an overview of the entire system. Create a block diagram and outline the operating principle. Configure the system and design the hardware.
- Test and evaluate the performance.

2. Strengths:

- Design and implement a system model for controlling home appliances through facial recognition and hand gestures.

3. Weaknesses:

- Consider more measurements with respect to different users and testing conditions.
- Need to improve the explanation of evaluation results in Chapter 4.

4. Approval for oral defense? **Approved**

5. Overall evaluation: (Excellent, Good, Fair, Poor)

FAIR

6. Mark: 8.5 (*in words*: eight point five)

Ho Chi Minh City, June 20, 2024

**ADVISOR**

(Sign with full name)

Đỗ Duy Tân



THE SOCIALIST REPUBLIC OF VIETNAM  
Independence – Freedom – Happiness  
-----

Ho Chi Minh City, July 2, 2024

## PRE-DEFENSE EVALUATION SHEET

Student name: Le Thi Thanh Thu      Student ID: 18119044

Student name: Nguyen Trong Hai      Student ID: 19119028

Major: Computer Engineering

Project title: Face recognition and hand gestures to control home appliances

Name of Reviewer: Truong Ngoc Son

### EVALUATION

1. Content and workload of the project

*Meets the requirements of a Bachelor's thesis*

2. Strengths:

.....

3. Weaknesses:

Some of the following points need to be improved or corrected

- Citation must be in increasing order

- Some of theories should be removed since they are fundamental such as Google Firebase, Roboflow, Led, fan,

- Flowchart presented in Fig. 3.7, 3.10, and 3.13: there exist parallel blocks, these flowcharts need to be corrected.

- how to achieve statistic presented in table 4.1, 4.2 need to be clearly stated clearly

4. Approval for oral defense? (Approved or denied)

*Approved for oral defense*

5. Overall evaluation: (Excellent, Good, Fair, Poor)

*Good*

6. Mark:.....(in words:.....)

Ho Chi Minh City, month day, year

**REVIEWER**

(Sign with full name)

Truong Ngoc Son



THE SOCIALIST REPUBLIC OF VIETNAM  
**Independence – Freedom – Happiness**

-----

Ho Chi Minh City, June 20, 2024

## EVALUATION SHEET OF DEFENSE COMMITTEE MEMBER

Student name: Lê Thị Thanh Thu

Student ID: 18119044

Student name: Nguyễn Trọng Hải

Student ID: 19119028

Major: Computer Engineering Technology

Project title: Facial Recognition And Hand Gestures To Control Home Appliances

Advisor: PhD. Đỗ Duy Tân

### EVALUATION

1. Content and workload of the project

.....  
.....  
.....  
.....

2. Strengths:

.....  
.....  
.....

3. Weaknesses:

.....  
.....  
.....

4. Overall evaluation: (*Excellent, Good, Fair, Poor*)

.....

5. Mark: .....(*in words* ..... )

Ho Chi Minh City, month day, year

**COMMITTEE MEMBER**

(*Sign with full name*)

## **DISCLARATION**

This project has been thoroughly researched and implemented with diligence. No content has been directly copied from existing projects. Proper citations have been provided for all references used. We take full responsibility for any potential violations that may arise.

### **Authors**

LE THI THANH THU

NGUYEN TRONG HAI

## **ACKNOWLEDGEMENTS**

We extend our deepest gratitude to PhD. Do Duy Tan for his invaluable contributions to our project. Mr. Tan's unwavering dedication, insightful guidance, and boundless enthusiasm have played a pivotal role in shaping the trajectory of our work. His profound expertise and generosity in sharing knowledge have not only enriched our project but also inspired us to explore new realms of creativity and innovation.

Throughout the implementation process, despite our best efforts, we acknowledge that challenges and occasional mistakes are inevitable. However, PhD. Tan's steadfast support and mentorship have empowered us to overcome obstacles with resilience and determination, transforming setbacks into valuable learning opportunities. His unwavering belief in our capabilities has motivated us to push the boundaries of our creativity and strive for excellence in all facets of our work.

Lastly, we wish to express our heartfelt appreciation to all who have supported our group's endeavors, whether through words of encouragement, practical assistance, or simply by being a listening ear. Your steadfast support has been a constant source of motivation and inspiration, fueling our determination to succeed.

Together, we have embarked on a journey of innovation and collaboration, and it is through the collective efforts of individuals like Mr. Tan and our broader support network that we have been able to achieve our objectives and make meaningful progress towards our shared vision.

Thank you all for your invaluable contributions, dedication, and camaraderie. Your support has been deeply valued, and we eagerly anticipate continuing our collaborative efforts as we work towards our common goals.



## TABLE OF CONTENTS

<b>DISCLARATION .....</b>	<b>i</b>
<b>ACKNOWLEDGEMENTS .....</b>	<b>ii</b>
<b>TABLE OF CONTENTS .....</b>	<b>iii</b>
<b>LIST OF FIGURES .....</b>	<b>vi</b>
<b>LIST OF TABLES .....</b>	<b>viii</b>
<b>LIST OF ABBREVIATIONS .....</b>	<b>ix</b>
<b>ABSTRACT.....</b>	<b>x</b>
<b>Chapter 1. INTRODUCTION .....</b>	<b>1</b>
<b>1.1 INTRODUCTION.....</b>	<b>1</b>
<b>1.2 OBJECTIVES .....</b>	<b>2</b>
<b>1.3 LIMITATION .....</b>	<b>2</b>
<b>1.4 SCOPE .....</b>	<b>2</b>
<b>1.5 DOMESTIC RESEARCH.....</b>	<b>4</b>
<b>1.6 INTERNATIONAL RESEARCH .....</b>	<b>4</b>
<b>1.7 THE NOVELTY OF TOPIC .....</b>	<b>5</b>
<b>1.8 OUTLINE .....</b>	<b>5</b>
<b>Chapter 2. LITERATURE REVIEW .....</b>	<b>6</b>
<b>2.1 INTERNET OF THINGS.....</b>	<b>6</b>
<b>2.2 WI-FI.....</b>	<b>7</b>
<b>2.3 AI Technology.....</b>	<b>7</b>
<b>2.3.1 CNN .....</b>	<b>7</b>
<b>2.3.2 YOLO .....</b>	<b>9</b>
<b>2.3.3 MEDIAPIPE.....</b>	<b>11</b>
<b>2.4 GOOGLE FIREBASE .....</b>	<b>13</b>
<b>2.5 ROBOTFLOW .....</b>	<b>14</b>
<b>2.6 HARDWARE.....</b>	<b>15</b>
<b>2.6.1 Camera .....</b>	<b>15</b>
<b>2.6.2 Raspberry Pi.....</b>	<b>15</b>
<b>2.6.3 Display .....</b>	<b>16</b>
<b>2.6.4 Power Supply .....</b>	<b>17</b>
<b>2.6.5 ESP32-WROOM-32 .....</b>	<b>17</b>
<b>2.6.6 Relay Module 4.....</b>	<b>17</b>
<b>2.6.7 Home Appliance .....</b>	<b>18</b>
<b>2.6.7.1 Fan.....</b>	<b>18</b>

2.6.7.2 Led.....	18
2.6.7.3 Solenoid Lock.....	18
<b>Chapter 3. DESIGN AND IMPLEMENTATION .....</b>	<b>19</b>
<b>3.1 SYSTEM REQUIREMENTS.....</b>	<b>19</b>
<b>3.2 BLOCK DIAGRAM .....</b>	<b>21</b>
<b>3.3 OPERATING PRINCIPLE.....</b>	<b>22</b>
<b>3.4 SOFTWARE DESIGN.....</b>	<b>23</b>
<b>3.4.1 IoT System .....</b>	<b>23</b>
3.4.1.1 Building Database .....	23
3.4.1.2 Raspberry Pi transmits data to Database .....	27
3.4.1.3 ESP32 receives data to Database.....	32
<b>3.4.2 Image Processing Block .....</b>	<b>39</b>
3.4.2.1 Facial Recognition.....	44
3.4.2.1.1 Building Custom Dataset .....	44
3.4.2.1.2 Set Up train YOLO model .....	48
3.4.2.2 Hand Gestures Recognition.....	51
3.4.2.2.1 Configure Parameter of Hand Gestures .....	51
3.4.2.2.2 Set Up MediaPipe .....	52
<b>3.4.3 Control Block.....</b>	<b>54</b>
<b>3.4.4 GUI Interface.....</b>	<b>56</b>
<b>Chapter 4. RESULT .....</b>	<b>58</b>
<b>4.1 IoT Systems.....</b>	<b>58</b>
<b>4.1.1 System Design .....</b>	<b>58</b>
<b>4.1.2 Control Home Appliances via Camera.....</b>	<b>60</b>
4.1.2.1 Control Door.....	60
4.1.2.2 Control Fan.....	61
4.1.2.3 Control Led.....	63
<b>4.1.3 Remote Home Appliances via GUI App.....</b>	<b>64</b>
<b>4.2 Evaluation .....</b>	<b>67</b>
<b>4.2.1 Performance Assessment of IoT System .....</b>	<b>67</b>
<b>4.2.2 Performance of the AI model .....</b>	<b>68</b>
4.2.2.1 YOLO Model .....	68
4.2.2.2 MediaPipe.....	70
<b>Chapter 5. CONCLUSIONS AND FUTURE WORKS.....</b>	<b>72</b>
<b>5.1 CONCLUSIONS .....</b>	<b>72</b>
<b>5.2 FUTURE WORKS .....</b>	<b>73</b>

<b>REFERENCES.....</b>	<b>75</b>
------------------------	-----------

## LIST OF FIGURES

Figure 2.1. Architecture of CNN network [1 - Figure 5].....	8
Figure 2.2. Development time of YOLO network [2 – Figure 1].....	9
Figure 2.3. The structure of YOLO network [3 – Figure 1] .....	9
Figure 2.4. Detailed illustration of YOLOv8 model architecture. The Backbone, Neck, and Head are the three parts of our model, and C2f, ConvModule, DarknetBottleneck, and SPPF are modules [26 – Figure 4].....	10
Figure 2.5. Landmarks positions detected by MediaPipe [4 – Figure 5].....	13
Figure 2.6. The mobile app development process [5].....	13
Figure 3.1. Block diagram of system.....	21
Figure 3.2. The sequence diagram of system.....	23
Figure 3.3. The flowchart of building database .....	24
Figure 3.4. The structure of database.....	26
Figure 3.5. API of database .....	27
Figure 3.6. The block diagram of Raspberry Pi transmit data to Firebase .....	27
Figure 3.7. The flowchart of deploy API on Raspberry Pi .....	29
Figure 3.8. The flowchart depicting the process of data uploading to the database after facial recognition .....	30
Figure 3.9. The flowchart depicting the process of data uploading to the database after hand gestures recognition .....	31
Figure 3.10. The block diagram of ESP32 receives data to Firebase .....	32
Figure 3.11. The flowchart of deploy API on ESP32.....	34
Figure 3.12. The flowchart for the process of receiving data from the database to control door .....	36
Figure 3.13. The flowchart for the process of receiving data from the database to control fan ....	37
Figure 3.14. The flowchart for the process of receiving data from the database to control led ....	38
Figure 3.15. Processing Block .....	39
Figure 3.16. The flowchart of image processing block .....	40
Figure 3.17. The flowchart of facial recognition process on Rasp .....	42
Figure 3.18. The flowchart of hand gestures recognition process on Rasp .....	43
Figure 3.19. Labelling each face of custom dataset.....	45
Figure 3.20. The custom dataset after labelling.....	45
Figure 3.21. Choose type of dataset to export dataset .....	46
Figure 3.22. The custom dataset after export from Robotflow .....	47
Figure 3.23. The sample image of custom dataset with bounding box.....	47
Figure 3.24. Information about example bounding box .....	47
Figure 3.25. The flowchart of YOLO model installation on Raspberry Pi.....	48

Figure 3.26. The resulted images after test model .....	49
Figure 3.27. The flowchart of deploy YOLOv8 model on Rasp .....	50
Figure 3.28. The flowchart of set up MediaPipe on Rasp .....	52
Figure 3.29. Control Block .....	54
Figure 3.30. The flowchart of control block .....	55
Figure 3.31. The flowchart of GUI .....	56
Figure 4.1. Result of external system design. ....	58
Figure 4.2. Results of internal system design .....	59
Figure 4.3. Result of facial recognition on Rasp then transmit data to database .....	60
Figure 4.4. Result of control solenoid lock after ESP32 receives data from database .....	61
Figure 4.5. Result of index finger recognition on Rasp then transmit data to database .....	61
Figure 4.6. Result of thumb finger recognition on Rasp then transmit data to database .....	62
Figure 4.7. Result of control fan after ESP32 receives data from database.....	62
Figure 4.8. Result of pinky finger recognition on Rasp then transmit data to database .....	63
Figure 4.9. Result of ring finger recognition on Rasp then transmit data to database .....	64
Figure 4.10. Result of control light after ESP32 receives data from database.....	64
Figure 4.11. Results of remote door via GUI app.....	65
Figure 4.12. Results of remote Fan via GUI app .....	66
Figure 4.13. Results of remote Led via GUI app .....	66
Figure 4.14. Results of controlling home appliances via the GUI app .....	67

## **LIST OF TABLES**

Table 2.1. Camera comparison table.....	15
Table 2.2. Mini computers comparison table .....	16
Table 3.1. Hand Gestures for User Manual .....	51
Table 3.2. The state table of hand gestures recognition .....	53
Table 4.1. Evaluate average processing time in daylight condition .....	68
Table 4.2. Evaluate average processing time in nightlight condition .....	68
Table 4.3. Comparison table of between YOLOv5 and YOLOv8.....	69
Table 4.4. Comparison table of between MediaPipe and Handtrack.js.....	70

## **LIST OF ABBREVIATIONS**

<b>AI</b>	Artificial intelligence
<b>YOLO</b>	You Only Look Once
<b>CV</b>	Computer Vision
<b>CNN</b>	Convolutional Neural Network
<b>IoT</b>	Internet of Things
<b>GUI</b>	Graphical User Interface
<b>WI-FI</b>	Wireless Fidelity
<b>API</b>	Application Programming Interface
<b>URL</b>	Uniform Resource Locator

## **ABSTRACT**

This study presents an innovative system that combines advanced facial recognition technology for door access with precise hand gesture recognition for controlling lighting and ventilation systems. The system's design aims to address the dual objectives of enhancing security measures and streamlining user interactions within controlled environments. Facial recognition technology serves as the primary means of access control, allowing authorized individuals to gain entry to restricted areas by simply presenting their face to the system's camera. This streamlined authentication process eliminates the need for physical keys or access cards, enhancing convenience and efficiency for users. In parallel, hand gesture recognition technology provides an intuitive interface for controlling lighting and ventilation systems. By recognizing specific hand gestures, users can seamlessly adjust lighting levels and fan speeds without the need for physical switches or controls. This intuitive interaction mechanism enhances user experience and promotes energy efficiency by facilitating precise control over environmental conditions. The integration of facial recognition and hand gesture recognition technologies offers a comprehensive approach to access control and environmental management in diverse settings. Beyond traditional security applications, the system's versatility makes it suitable for various environments, including residential, commercial, and industrial spaces. Implementation of this integrated system is anticipated to yield significant benefits, including improved security, enhanced user convenience, and optimized energy utilization.

Furthermore, the insights provided in this report serve as valuable resources for students and professionals in Computer Engineering, Telecommunication Electronic Engineering, and related fields, offering detailed perspectives on system architecture, implementation strategies, and potential applications in real-world scenarios.



## **Chapter 1. INTRODUCTION**

The rapid evolution of IoT technology has profoundly reshaped daily life, with home automation emerging as a pivotal advancement. These systems empower users to effortlessly manage household appliances, bolstering comfort, security, and energy efficiency. Integrating facial recognition and hand gestures into IoT home automation offers a novel way to personalize access and streamline control methods. This research explores developing an IoT-based home appliance control system using these technologies to enhance security, efficiency, and convenience in everyday living.

### **1.1 INTRODUCTION**

The incorporation of face and hand gesture detection AI technologies into IoT-based smart home systems adds a number of novel features and benefits that set this project apart from other smart home solutions. This idea takes use of many major areas of originality. First, it uses a dual recognition system: face recognition for individualized access and seamless verification, as well as hand gesture detection for contactless, intuitive operation. This combination provides a frictionless, sanitary, and easily accessible interface that improves user convenience and security. The improved user interface offers smooth interaction and an adaptable environment, making device operation intuitive and adapted to individual preferences. Users may manage their home equipment using natural gestures and facial expressions, such as finger up to turn on lights or fan, removing the need for physical connection with the devices.

Furthermore, this initiative focuses on enhanced interoperability, connecting with current IoT devices to form a coherent, networked home environment that may adapt in response to future technology advances. The system's ability to integrate with a variety of smart home devices guarantees a comprehensive and integrated experience. This interoperability enables a uniform control interface, which simplifies user interactions across many devices and platforms, hence improving the overall user experience.

Improved security and privacy are also important, with face recognition offering safe access control and on-device processing protecting data privacy. By utilizing on-device processing for both facial and gesture detection, the solution reduces the danger of data breaches while also protecting sensitive user information. This safe, non-intrusive approach of managing home settings meets rising privacy concerns in the digital era. The system's architecture also includes strong security mechanisms to prevent unwanted access and cyber threats, which boosts user trust and confidence.

Overall, this project revolutionizes IoT-based smart home solutions by offering a dynamic, intuitive, and adaptive approach to home automation. By focusing on enhancing user interface, security, interoperability, and operating efficiency, the system promises to deliver an unprecedented level of convenience, efficiency, and interactivity. Unlike existing solutions that often rely on more conventional methods of control, this project introduces a sophisticated and user-friendly way of managing home environments, paving the way for truly intelligent living spaces that cater to individual needs and preferences. The integration of cutting-edge AI technologies ensures that the system is not only convenient and efficient but also future-proof, capable of evolving with ongoing technological advancements. This innovative approach transforms our living spaces into truly intelligent environments, significantly elevating the overall quality of life.

## **1.2 OBJECTIVES**

The aim of this thesis is to design and implement an Internet of Things (IoT)-based Smart Home system that leverages facial recognition and hand gestures to control home devices via cameras. Additionally, it will facilitate remote control of these devices through an internet-based application. The objectives of this research will be guided by the following specific guidelines:

- **Collect Real-World Data and Reinforce Acquired Knowledge:** During the implementation phase, the system will gather real-world data, which will be used to enhance and reinforce existing knowledge, ensuring the system's effectiveness and reliability.
- **Understand and Articulate the Operational Concept of the System:** The research aims to provide a clear and detailed explanation of how the smart home system functions, including the mechanisms and processes involved in its operation.
- **Explore Component Integration:** This study will investigate and experiment with the integration of various components, such as the Camera Raspberry Pi 4, ESP32, Solenoid Lock, LED, and Fan, to ensure seamless operation and functionality of the smart home system.
- **Define Specific Development Strategies:** The research will provide clear and precise development strategies to ensure the practical application of the product, outlining the steps and methodologies required for successful implementation.
- **Design a Simple, Cost-Effective, and Practical Solution for Daily Life:** The system will be designed to be straightforward, cost-effective, and practical, ensuring that it can be easily implemented and utilized in everyday life. This will involve focusing on user-friendly interfaces and affordable components without compromising on functionality.

## **1.3 LIMITATION**

The project has the following limitations:

- The project mainly focuses on face and hand gesture detection and recognition performed under good lighting conditions.
- The demo IoT system is deployed in indoor conditions avoiding direct sunlight.
- The display software can only be used execution file on Raspberry Pi.
- The object detection speed of the camera on Raspberry Pi is relatively slow.

## **1.4 SCOPE**

This research will focus on developing a multifunctional system capable of recognizing faces and hand gestures, interacting with devices such as fans and light bulbs in the home environment. The specific scope of the project will include the following aspects:

### **Wi-Fi Connectivity for IoT System:**

**Wireless Communication:** The system will establish a Wi-Fi connection to enable seamless communication between the Raspberry Pi 4, ESP32, and the home appliances.

This setup allows for remote control and monitoring, enhancing user convenience and flexibility.

**IoT Integration:** The Wi-Fi-enabled communication will integrate the system into the broader IoT ecosystem, allowing for interoperability with other smart home devices and platforms.

### **Integration with ESP32 for Database Connectivity:**

**Database Interaction:** The ESP32 microcontroller will be used to connect to a database. This connection will enable the system to store and retrieve user preferences, device statuses, and other relevant data, facilitating efficient and centralized control of home appliances.

**Home Appliance Control:** The ESP32 will manage the control signals to home appliances based on the processed data from the Raspberry Pi 4. This division of tasks ensures optimal performance and responsiveness.

### **Face and Hand Gesture Recognition:**

**Algorithm Development:** Research will focus on developing effective face and hand gesture recognition algorithms. These algorithms will be optimized for interaction and object recognition, ensuring high accuracy and responsiveness in real-time.

**Image Processing on Raspberry Pi 4:** The image processing tasks will be handled by the Raspberry Pi 4, utilizing its processing power to perform complex computations needed for facial and gesture recognition. This setup ensures that the system remains both powerful and flexible for various home environments.

### **User Interface Development:**

**User-Friendly Design:** A easy to use and intuitive user interface will be developed, allowing users to easily interact with the system. This interface will provide access to device controls, status monitoring, and user preferences.

**Database Connectivity:** The user interface will connect to the database, offering flexible information and management options to users. This feature ensures that users can configure and control their home environment with ease.

By focusing on these detailed aspects, the research promises to deliver an integrated smart home system that prioritizes user needs. The combination of advanced technologies like facial recognition and gesture control, along with robust database connectivity and Wi-Fi communication, will enable seamless interaction with home devices. Emphasizing real-time data processing and centralized control, the system will ensure efficiency and reliability. Utilizing a modular design for scalability, it is adaptable to future technological advancements. By addressing practical concerns and enhancing convenience and security, the system has the potential to significantly improve the management of living environments, providing a sophisticated and intuitive approach to smart home automation.

## **1.5 DOMESTIC RESEARCH**

In Vietnam, numerous studies have explored IoT systems and image processing applications, aiming to evaluate their performance across various contexts.

Some researchers have focused on applying IoT to control household devices, emphasizing the design and installation of a compact central controller that meets technological requirements for monitoring and managing smart home devices. They have also developed mobile apps and computer interfaces to streamline system control and monitoring. Experiments using a smart home model assessed system stability and determined the maximum control range of the LoRa wireless network.

“Design and implementation of smart home system” [24] - The project aims to create and deploy a smart home system, unlocking possibilities for intelligent and practical solutions in everyday life. It also promotes thoughtful considerations about human-machine interactions, security, and the future of smart living.

“Designing the smart locking door by using image processing” [25] - The purpose of this project is to develop a face recognition system that requires minimal data and training yet meets all necessary requirements. Additionally, the system includes liveness detection and a user-friendly interface for enhanced usability.

These studies contribute to Vietnam's research in domestic IoT systems and image processing technologies, assessing their effectiveness using hardware standards and data transmission methods, typically employing simulations and real-world testing.

## **1.6 INTERNATIONAL RESEARCH**

Outside of Vietnam, there have been many international studies that have evaluated IoT systems and image processing technology in different fields and applications. Here are some interesting studies:

"An IoT-Based Home Automation System with Hand Gesture Recognition" [18] - The paper discusses Home Automation Systems commonly controlled via smartphones and microcontrollers for convenience, efficiency, and security. IoT cloud services enable remote control, optimizing water usage through parameter thresholds. This design minimizes water waste, ensuring efficient irrigation as needed.

“Smart Control of Home Appliances Using Hand Gesture Recognition in an IoT-Enabled System” [22] - Research focuses on developing a smart home device control system using hand gesture recognition. Experimental results show that the system recognizes gestures from numbers 0 to 9 with high accuracy and resilience, helping to control home devices through a low-cost IoT system.

"IoT Enabled Virtual Home Assistant Using Raspberry Pi" [23] - The study includes experimental data analysis to assess the feasibility of the virtual assistant, focusing on real-world testing of user interaction within home automation systems related to AC power supply. It compares response times with existing solutions and evaluates user satisfaction.

The aforementioned studies serve as examples and do not encompass all international research on IoT systems and image processing technology. Nevertheless,

researchers have observed and investigated their effectiveness and outstanding performance across a range of applications in different fields.

## **1.7 THE NOVELTY OF TOPIC**

The topic **"FACIAL RECOGNITION AND HAND GESTURES TO CONTROL HOME APPLIANCES"** is newer than traditional research that is often limited to exploring IoT systems or image processing separately. Building upon our accumulated knowledge and insights gleaned from prior research endeavors, we undertook a project aimed at seamlessly integrating advanced image processing techniques into the domain of IoT. This approach represents a departure from the conventional silos of research, seeking to harness synergies between these two domains to innovate and enhance the functionality of IoT applications. In addition, this topic also has the potential to contribute to the development of smart home IoT systems, especially in the context of advancing AI technologies and the next technologies. In short, this project not only provides practical satisfaction in enhancing and improving the efficiency of processing blocks but also contributes to the formation and development of many technologies for IoT systems in the future.

## **1.8 OUTLINE**

Chapter 1 Introduction: A brief introduction about our project as well as our objectives and scopes.

Chapter 2 Literature Review: Presenting background knowledge about the methods and technologies we used in this project.

Chapter 3 Design and implementation: Explanation for the block diagram and flowchart generated after considering the system requirements.

Chapter 4 Result: Presenting the results of hardware and software construction, evaluating their operation.

Chapter 5 Conclusions and future works: Our conclusion on the project and the direction of development in the future.

## **Chapter 2. LITERATURE REVIEW**

In this chapter, we provide an overview of the key technologies: IoT, Wi-Fi, CNN, YOLOv8, MediaPipe, Google Firebase, and Robotflow and background hardware utilized in our project. These technologies, along with our hardware setup, form the foundation of our project's architecture, providing the necessary tools and frameworks for development and implementation.

### **2.1 INTERNET OF THINGS**

As reported in the article [19], the Internet of Things (IoT) is a rapidly developing field that has sparked significant interest in academic and industrial circles. This network of interconnected devices, equipped with sensors, software, and network connections, is transforming how data is gathered, processed, and used across various industries. The core premise of IoT is its ability to connect diverse objects and systems, allowing seamless data interaction and exchange, leading to more efficient and automated operations.

The literature on IoT highlights its numerous applications and impacts. In the industrial sector, known as the Industrial Internet of Things, IoT technologies are crucial for optimizing manufacturing processes through smart factories. Real-time data from machines and equipment enable predictive maintenance, operational optimization, and downtime reduction, resulting in significant cost savings and productivity gains. Additionally, IoT enhances supply chain management by providing end-to-end visibility and item monitoring, thereby improving logistics efficiency and reducing losses.

In the field of smart homes and cities, IoT helps to create intelligent settings that improve people's quality of life. Residents benefit from smart home equipment like linked thermostats, lighting systems, and security cameras, which provide convenience, energy efficiency, and increased safety. On a bigger scale, smart city efforts use IoT technology to manage urban infrastructure, such as traffic, garbage, and public safety. These applications lead to more effective resource usage and better urban living circumstances.

Despite its evolving potential, IoT faces numerous challenges, particularly concerning security and privacy. The extensive connectivity and data exchange inherent in IoT devices increase the risk of cyberattacks and data breaches. The literature underscores the importance of robust security measures, including encryption, authentication procedures, and secure communication channels, to safeguard sensitive data and uphold user confidence. Moreover, integrating diverse IoT devices requires standardized protocols and interoperability frameworks to ensure seamless connectivity and optimal performance.

Initial IoT research focused on merging AI and Machine Learning to successfully use the massive data volumes generated by IoT devices. These developments boost analytics, predictive analysis, and autonomous decision-making, increasing the range of IoT applications. Furthermore, there is a growing focus on edge computing, which allows for data processing closer to the data source. This method reduces latency and conserves bandwidth, which is critical for real-time applications.

To summarize, the Internet of Things is an essential part of the current digital transformation, having the potential to profoundly affect a variety of industries by improving efficiency, productivity, and quality of life. However, realizing its full potential involves addressing significant security, interoperability, and data management issues.

Continued research and development, as well as collaborative efforts across sectors and regulatory authorities, are required to create a safe, interoperable, and efficient IoT environment.

## **2.2 WI-FI**

As reported in the article [20], Wi-Fi, a core component of wireless communication technology, has spawned a significant body of literature that examines its varied influence across several fields. Since its debut, Wi-Fi has transformed how people access and exchange information by converting traditional wired networks into seamless, ubiquitous connection options. To ensure reliable and efficient wireless communication, initial research efforts concentrated on fundamental factors such as signal propagation, interference reduction, and protocol improvement. As Wi-Fi became more ubiquitous, researchers looked at its economic consequences, including its function in bridging the digital gap, promoting economic development, and enabling new kinds of social contact and collaboration.

Furthermore, Wi-Fi's creation into mobile devices has accelerated research into mobile computing, with a particular emphasis on mobility management, energy economy, and smooth transition between Wi-Fi and cellular networks. The expansion of Wi-Fi-enabled IoT devices has created new difficulties and possibilities for study into resource restrictions, network scalability, and security risks in IoT installations. Moreover, developments in Wi-Fi technology, such as Wi-Fi 6 (802.11ax), promise to meet the growing need for faster data rates, lower latency, and increased spectrum efficiency in crowded wireless situations. Security and privacy remain significant areas of exploration, with researchers looking into encryption methods, authentication systems, and intrusion detection approaches to protect Wi-Fi networks from cyber-attacks and unwanted access.

Additionally, the introduction of Wi-Fi-based positioning systems has contributed to research into internal localization, context-aware services, and location-based advertising, creating new opportunities for personalized and location-based experiences.

Overall, the literature on Wi-Fi emphasizes its transformational influence on communication, computers, and society as a whole while also noting current research efforts aimed at improving its performance, security, and adaptability to suit the changing requirements of an increasingly linked world.

## **2.3 AI Technology**

### **2.3.1 CNN**

Convolutional Neural Networks (CNNs) are specialized artificial neural networks widely used in Deep Learning for image and object identification, as well as classification applications. CNNs have transformed how Deep Learning models perceive objects in pictures, making them crucial in various applications.

CNNs excel in identifying and categorizing objects within images, ranging from basic household items to complex medical images. They can precisely locate items within a picture (localization) and segment them into distinct zones. CNNs analyze and interpret video data, enabling applications such as activity recognition, video summarization, and anomaly detection. Due to their significant contributions to these rapidly evolving

domains, CNNs are highly popular in Deep Learning, providing robust tools for analyzing and interpreting visual data.

Figure 2.1 shows the architecture of CNNs, or Convolutional Neural Networks, which are structured with three main layers: input, hidden, and output layers, which collectively extract features from images. However, what sets CNNs apart are additional layers such as convolutional, pooling, and fully connected layers, all of which contribute to the network's architecture.

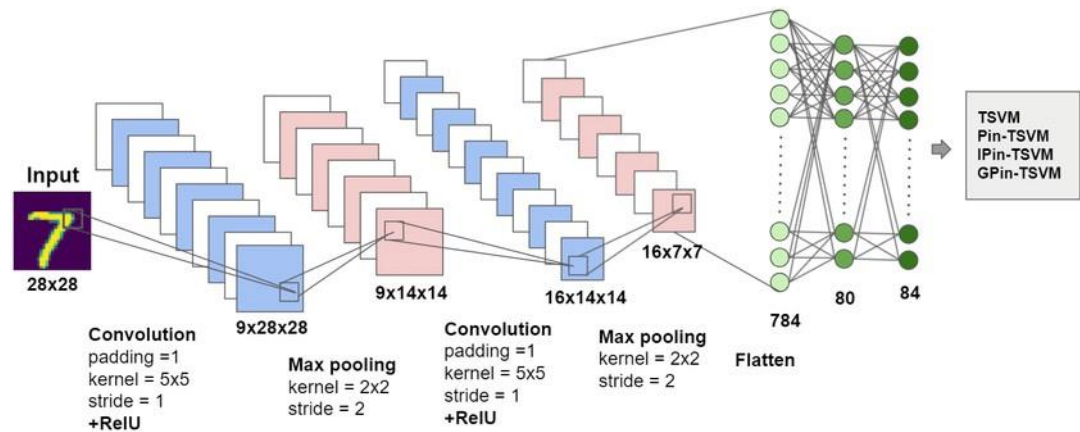


Figure 2.1. Architecture of CNN network [1 - Figure 5]

**Input Layer:** The input layer accepts raw data, such as photos, and sends it to the following layers for processing.

#### Hidden Layers:

- **Convolutional Layers:** Convolutional layers extract crucial elements from input data, including edges, textures, and patterns.
- **Pooling Layers:** Pooling layers reduce the spatial dimensions of convolutional feature maps while preserving crucial information.
- **Fully Connected Layers:** Fully connected layers connect all neurons from previous layers to the next, enabling the network to understand complex patterns and correlations in the data.

**Output Layer:** The output layer generates final findings or predictions based on processed data from the hidden layers.

CNNs can extract sequential characteristics from pictures by stacking these layers in a hierarchical fashion, enabling tasks such as image categorization, object recognition, and segmentation. This design allows CNNs to learn complex patterns and structures within images, making them particularly effective in a variety of computer vision applications.



### 2.3.2 YOLO

The You Only Look Once (YOLO) method advocates for the use of an end-to-end neural network to concurrently predict bounding boxes and class probabilities. This approach is distinct from traditional object detection algorithms, which typically use classifiers adapted for detection tasks. Figure 2.2 below illustrates the architecture of the YOLO network.

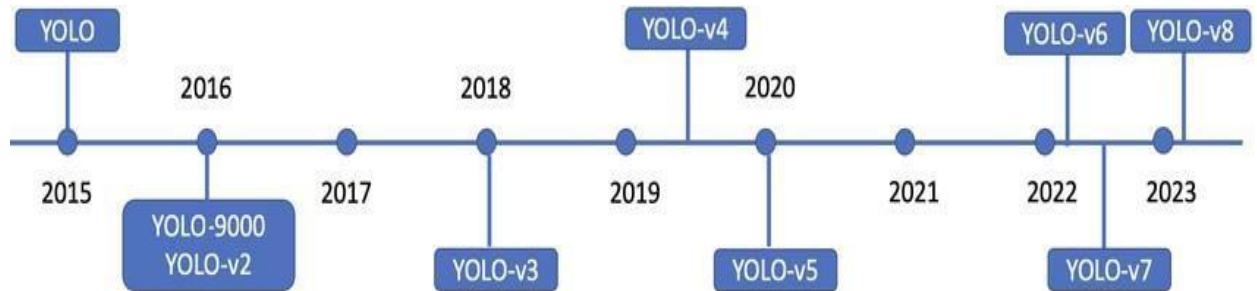


Figure 2.2. Development time of YOLO network [2 – Figure 1]

The YOLO technique begins by taking an image as input and then uses a basic deep convolutional neural network (CNN) to recognize objects within the image. Figure 2.3 below depicts the fundamental architecture of the CNN model required for YOLO.

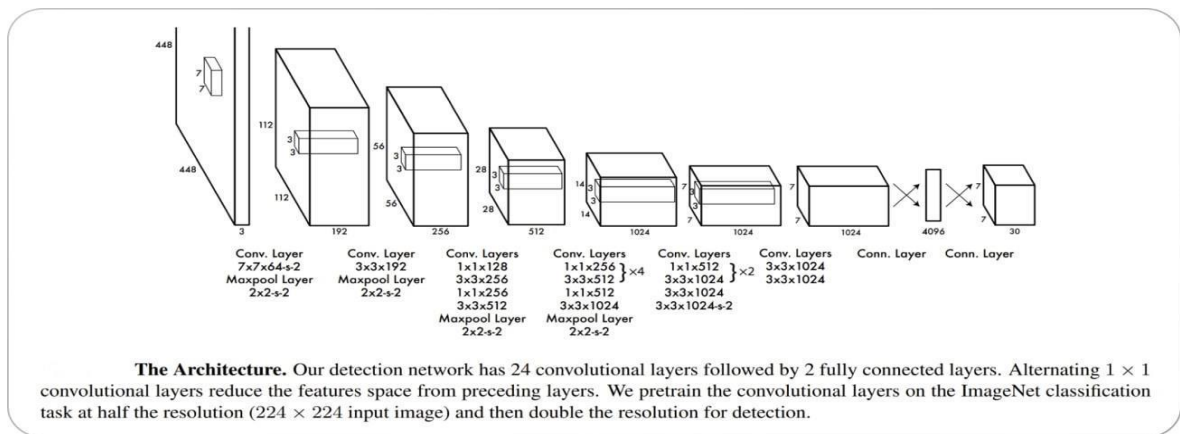


Figure 2.3. The structure of YOLO network [3 – Figure 1]

The initial 24 convolution layers of the YOLO model are pre-trained on ImageNet, using temporary average pooling and fully connected layers. Following this, the pre-trained model is modified for detection tasks, as prior research has shown that adding convolution and connected layers to a pre-trained network enhances performance. The final fully connected layer in YOLO is responsible for predicting both class probabilities and bounding box coordinates.

YOLO divides the input image into an  $S \times S$  grid, with each grid cell responsible for detecting an object if the object's center falls within it. Within each grid cell, YOLO predicts  $B$  bounding boxes and assigns confidence scores to these boxes. These confidence scores indicate the model's certainty about the presence of an object and the accuracy of the predicted bounding box.

During training, the objective is to assign one bounding box predictor to each object. YOLO achieves this by assigning the predictor with the highest Intersection over Union (IOU) with the ground truth as the "responsible" predictor. This approach helps the model specialize in predicting objects of specific sizes, aspect ratios, or classes, thereby improving the overall recall score.

A crucial technique used in YOLO models is non-maximum suppression (NMS). NMS is a post-processing step that refines the accuracy and efficiency of object detection. Since multiple bounding boxes may be generated for a single object, potentially overlapping or located at different positions, NMS identifies and removes redundant or incorrect boxes. The result is a streamlined output with a single bounding box representing each object in the image.

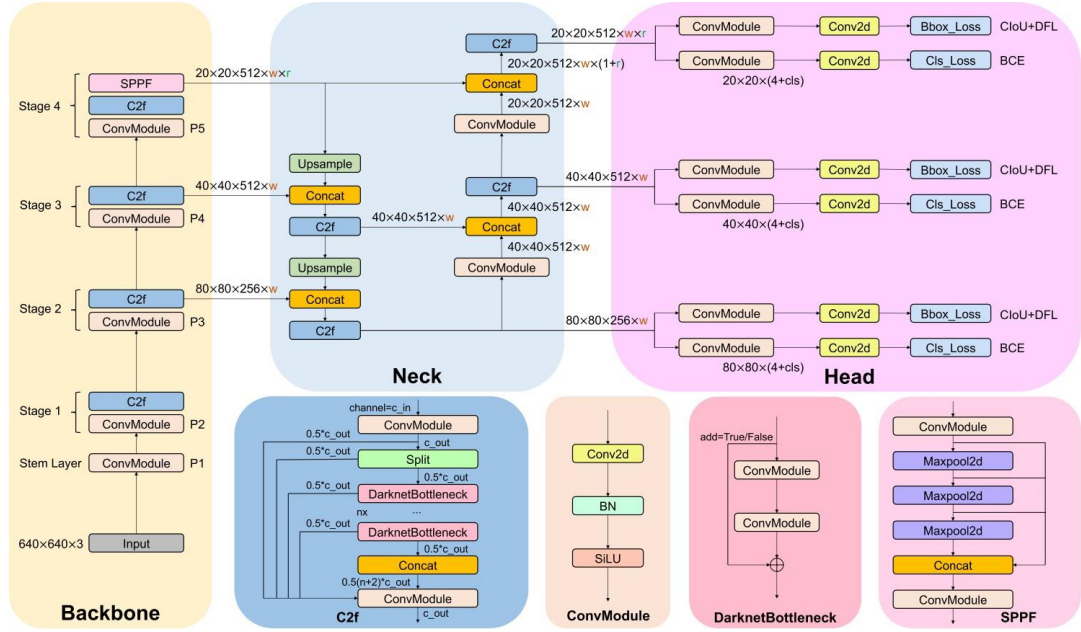


Figure 2.4. Detailed illustration of YOLOv8 model architecture. The Backbone, Neck, and Head are the three parts of our model, and C2f, ConvModule, DarknetBottleneck, and SPPF are modules [26 – Figure 4]

Our model YOLOv8 architecture comprises three main components: the backbone, the neck, and the head, as illustrated in Figure 2.4. The subsequent subsections will detail the design concepts and modules associated with each part of the model architecture.

#### Backbone:

The backbone of our model utilizes the Cross Stage Partial (CSP) architecture, which divides the feature map into two segments. The first segment undergoes convolution operations, while the second segment is concatenated with the output from the first segment. This CSP architecture enhances the learning capability of convolutional neural networks (CNNs) and reduces the model's computational cost.

YOLOv8 introduces the C2f module, which combines the C3 module and the Efficient Layer Aggregation Networks (ELAN) concept from YOLOv7. This combination allows the model to capture richer gradient flow information. The C3 module comprises three ConvModules and  $n$  DarknetBottleNecks, whereas the C2f module includes two ConvModules and  $n$  DarknetBottleNecks connected via Split and Concat operations, as shown in Figure 4. Each ConvModule consists of Conv-BN-SiLU layers, with  $n$  representing the number of bottlenecks. Unlike YOLOv5, we employ the C2f module instead of the C3 module.

To further reduce computational cost, we decrease the number of blocks in each stage compared to YOLOv5, specifically reducing the blocks to 3, 6, 6, and 3 in Stages 1 through 4, respectively. Additionally, we incorporate the Spatial Pyramid Pooling - Fast (SPPF) module in Stage 4, an enhancement over the Spatial Pyramid Pooling (SPP)

module, to improve the model's inference speed. These modifications result in a model with enhanced learning ability and shorter inference times.

*Neck:*

While deeper networks typically capture more feature information, resulting in improved dense prediction, excessively deep networks can lose location information and suffer from information loss for small objects due to numerous convolution operations. Therefore, it is essential to use Feature Pyramid Network (FPN) and Path Aggregation Network (PAN) architectures for multi-scale feature fusion. The Neck part of our model architecture employs multi-scale feature fusion to integrate features from different network layers. The upper layers acquire more feature information, while the lower layers retain location information due to fewer convolution layers.

Inspired by YOLOv5, which uses FPN to upsample from top to bottom and increase feature information in the lower layers, and PAN to downsample from bottom to top to obtain more information in the upper layers, our model merges these feature outputs to ensure precise predictions for images of various sizes. We adopt the FP-PAN (Feature Pyramid-Path Aggregation Network) in our model and eliminate convolution operations in upsampling to reduce computational cost.

*Head:*

Unlike the YOLOv5 model, which uses a coupled head, our model employs a decoupled head, where the classification and detection heads are separated. As illustrated in Figure 4, our model removes the objectness branch and retains only the classification and regression branches. Anchor-based methods use numerous anchors in the image to determine the four offsets of the regression object from the anchors and adjust the precise object location using the corresponding anchors and offsets. In contrast, we adopt an anchor-free approach, which identifies the object's center and estimates the distance between the center and the bounding box.

*Loss:*

In our model training, we use the Task Aligned Assigner from Task-aligned One-stage Object Detection (TOOD) for positive and negative sample assignment. This method selects positive samples based on the weighted scores of classifications and regression, as represented in Equation below:

$$t = s^\alpha + u^\beta$$

where  $s$  is the predicted score corresponding to the labeled class, and  $u$  is the Intersection over Union (IoU) of the prediction and the ground truth bounding box.

### **2.3.3 MEDIAPIPE**

MediaPipe, developed and investigated by Google, is a versatile library offering a wide range of cross-platform machine learning applications. It is open-source and free, allowing users to customize it to fit their specific needs. MediaPipe supports various platforms, including mobile, desktop, cloud, online, and IoT devices.

- **Modular Pipeline Design:** MediaPipe allows developers to construct complex pipelines by linking "calculators." These calculators can perform diverse functions such as image processing, feature extraction, and machine learning inference.
- **Cross-Platform Support:** MediaPipe is compatible with multiple platforms such as Android, iOS, Linux, macOS, and Windows, facilitating the development of applications across different environments.
- **Pre-Trained Models:** MediaPipe provides pre-trained machine learning models for tasks like face identification, hand tracking, pose estimation, and object detection. These models can be seamlessly integrated into applications to enhance perceptual capabilities.
- **Customization and Extension:** Developers have the flexibility to customize and expand MediaPipe by adding new calculators or integrating their own machine learning models, enabling the creation of tailored applications for specific use cases.
- **Real-Time Performance:** MediaPipe is optimized for real-time performance, efficiently processing video streams and camera input with minimal latency. This capability makes it well-suited for interactive and augmented reality applications.

This library supports numerous areas of computer vision, such as face detection, face mesh, hand detection, human pose estimation, object detection, and more. Specifically, MediaPipe includes a hand detection solution designed to identify and locate hands within images. This solution comprises two models: the palm detection model and the hand landmarks detection model. The palm detection model identifies hand regions in the input image, while the hand landmarks detection model extracts specific points within those regions. The key points extracted consist of 21 landmark coordinates, each representing the width and height of a key point based on its position in the image.

Figure 2.5 provides information about the 21 hand landmarks that MediaPipe identifies and extracts. For each key point, MediaPipe returns its location on the image, corresponding to the width and height relative to the image's size. The reference point is located at the top left corner of the image. Each unique hand gesture results in a different set of values, making it suitable for training a model to recognize hand gestures using these output values.

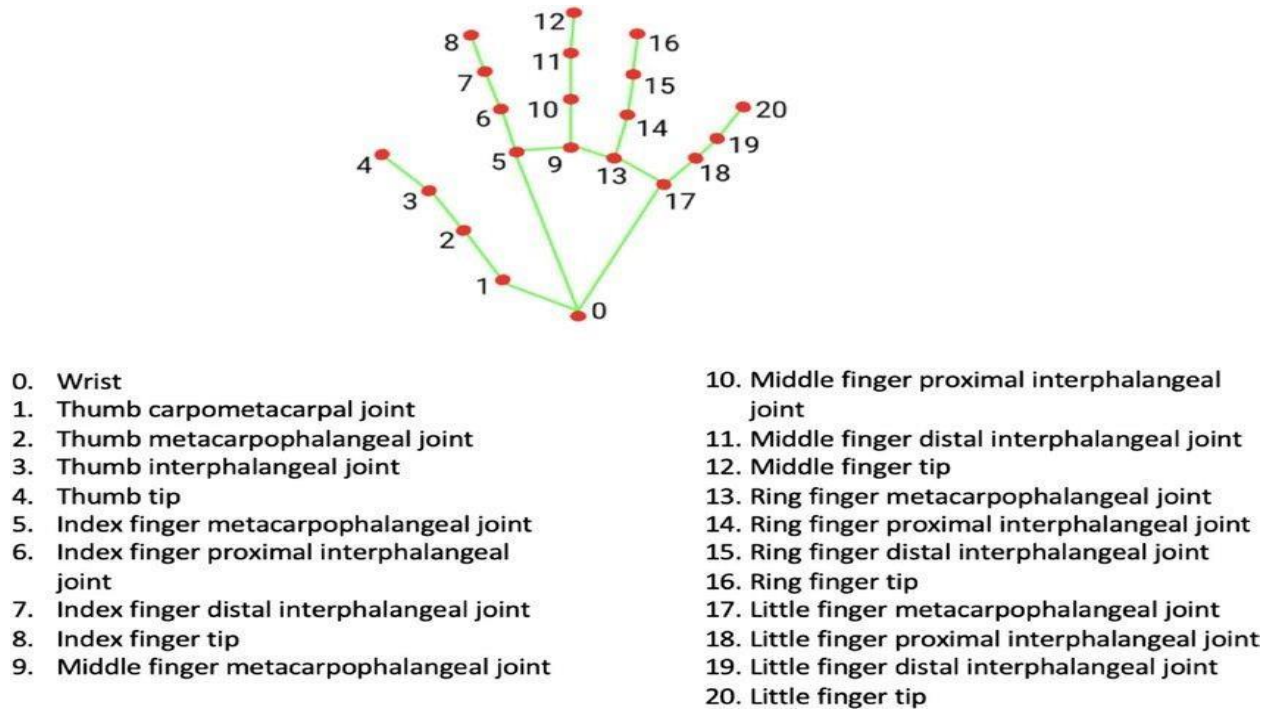


Figure 2.5. Landmarks positions detected by MediaPipe [4 – Figure 5]

## 2.4 GOOGLE FIREBASE

Firebase by Google comprises a suite of cloud-based development tools designed to aid mobile and web app developers in creating, deploying, and scaling their applications. Figure 2.6 illustrates the development process of the mobile app.



Figure 2.6. The mobile app development process [5]

As reported in the article [5], Firebase offers a variety of functionalities, including:

- **Authentication:** Enables users to sign into programs securely and easily. Developers may utilize Firebase Authentication to allow logins by email and password, Google Sign-In, Facebook Login, and other authentication methods.
- **Realtime Database:** A cloud-hosted NoSQL database enables real-time data synchronization across all user devices. This feature makes it easier to create apps that keep updated even while users are offline.
- **Cloud Messaging:** Firebase Cloud Messaging (FCM) allows businesses to deliver messages to consumers' devices even when they aren't using the app. FCM enables developers to deliver push alerts, change app information, and do other communication activities.
- **Crashlytics:** A service for tracking and addressing app problems. Firebase Crashlytics generates thorough crash reports, allowing for rapid root cause detection and problem resolution.
- **Performance Monitoring:** Provides insights into app performance. To guarantee peak performance, organizations can monitor indicators such as CPU utilization, memory consumption, and network traffic.
- **Test Lab:** A cloud-based tool for developers to test apps on different devices and setups. This feature assists in ensuring that the app works properly across several devices and under a variety of network situations.

## **2.5 ROBOFLOW**

Roboflow is a complete computer vision platform that makes it easier to develop computer vision models. It is a comprehensive and diverse benchmark that includes 100 datasets covering 7 different image domains, totaling 224,714 images and 805 class labels. These datasets are compiled from over 90,000 public datasets and 60 million public images, actively curated and labeled by computer vision experts via the Roboflow Universe web application. The goal of releasing Roboflow is to give researchers a benchmark for testing the generalizability of their object detection models with real-world data. Additionally, Roboflow functions as a flow-based visual programming language that allows users to program mobile manipulation tasks. It ensures robust low-level implementation on a mobile manipulator while restricting high-level programming to reduce user errors. Roboflow has been successfully implemented on a PR2 mobile manipulator, demonstrating its generalizability and error-handling capabilities in everyday tasks within human environments.

The following are the advantages of using Roboflow:

- Roboflow provides sophisticated annotation tools for labeling and annotating data in machine vision projects. It supports a variety of annotation types, including bounding boxes, polygons, keypoints, and segmentation masks.
- Roboflow offers many data augmentation approaches to improve the diversity and quality of training data. Random cropping, rotation, flipping, scaling, and other methods all contribute to improved model performance and resilience.
- Roboflow allows for efficient dataset versioning and administration, including tracking changes, team collaboration, and annotation history. This ensures repeatability and streamlines model iteration.



- Roboflow seamlessly interfaces with popular deep learning frameworks, including TensorFlow and PyTorch. It provides export options for various frameworks, such as code snippets or pre-configured project templates, to simplify integration into your development workflow.
- The platform simplifies training computer vision models by providing pre-configured pipelines for data preparation, training, and evaluation. Once trained, models can be deployed across various platforms, including the cloud, edge devices, and APIs.
- Roboflow offers comprehensive performance data and analytic tools for analyzing and optimizing models. You can track metrics such as accuracy, recall, and mAP (mean Average Precision) to gain insights into model performance and view model predictions.
- The platform supports transfer learning, allowing you to leverage pre-trained models on large datasets. This significantly reduces training time and computational resources required to achieve satisfactory results.
- Roboflow has a robust user community and provides support through forums, documentation, and tutorials. Users can access tools and guidance for a wide range of computer vision tasks, share work, and seek community assistance as needed.

## 2.6 HARDWARE

### 2.6.1 Camera

At the behest of the system, the project requires a camera capable of capturing clear, high-resolution images to enhance image quality for processing. Various camera modules are suitable for AI projects. Table 2.1 below lists some appropriate cameras.

Camera name	Megapixel	Price
Raspberry Pi Camera Module V3	12	790,000 VND
Raspberry Pi Camera Module V2	8	750,000 VND
Logitech Webcam C310 HD	5	650,000 VND
Logitech Webcam C270 HD	3	400,000 VND

Table 2.1. Camera comparison table

Comparing the results from Table 2.1, the Logitech Webcam C270 HD emerges as the most suitable camera for this project. Its 3-megapixel resolution is sufficient, and it is also the most cost-effective option among all cameras listed in the table, which helps save project costs. Additionally, this webcam offers a high resolution of 720p at 30fps, which significantly aids in continuous image capture, minimizing frame loss.

### 2.6.2 Raspberry Pi

There is an application for a facial recognition model in the project, so the team needs to select a mini computer that meets the system's requirements and is powerful enough for image processing. There are several mini computers suitable for AI projects. Table 2.2 lists some suitable options.

Mini computer name	CPU	RAM	Wi-Fi	Price
Raspberry Pi 4 B	Cortex-A72	1GB	Yes	1,300,000VND
Raspberry Pi 4 B	Cortex-A72	2GB	Yes	1,500,000VND
Raspberry Pi 4 B	Cortex-A72	4GB	Yes	2,200,000VND
Raspberry Pi 4 B	Cortex-A72	8GB	Yes	2,800,000VND
NVIDIA Jetson Nano B01	ARM A57	4GB	None	5,000,000VND

Table 2.2. Mini computers comparison table

After comparing results of Table 2.2, the Raspberry Pi 4 B model with 8 GB RAM is the most suitable mini-computer because image processing will require a lot of RAMS [8]. Choosing the Raspberry Pi 4 Model B with 8GB RAM for image processing blocks offers several advantages. Firstly, the increased RAM capacity allows for handling larger datasets and more complex image processing tasks efficiently. This is particularly beneficial when processing high-resolution images or performing computationally intensive tasks such as object detection, image classification, or real-time video processing.

Secondly, the Raspberry Pi 4 Model B with 8GB RAM provides better multitasking capabilities. It can simultaneously handle multiple processes related to image processing without experiencing significant slowdowns or memory constraints. This capability is crucial for applications where continuous processing and responsiveness are required.

Additionally, the Raspberry Pi 4 Model B offers improved computational power compared to its predecessors, making it more suitable for demanding image processing algorithms and applications. The quad-core ARM Cortex-A72 processor and enhanced GPU (Graphics Processing Unit) ensure faster processing and efficient handling of image manipulation tasks.

Moreover, the Raspberry Pi ecosystem supports a variety of software libraries and frameworks for image processing, machine learning, and computer vision. This includes popular tools like OpenCV, TensorFlow, and PyTorch, which leverage the Pi's hardware capabilities effectively, especially when paired with the expanded RAM of the 8GB model.

Overall, choosing the Raspberry Pi 4 Model B with 8GB RAM for image processing blocks ensures robust performance, versatility, and compatibility with a wide range of image processing applications, making it a suitable choice for hobbyists, educators, and professionals alike.

### 2.6.3 Display

As per the system's request, we require a screen for monitoring employee information. Since interaction with the screen is not necessary, a regular screen will suffice;



there is no need for a touchscreen or an expensive display. Alternatively, we can use an online application like VNC Viewer on a computer or laptop to monitor information via the Internet.

As indicated above, the Raspberry Pi functions as a mini computer capable of displaying the operating system interface on any screen.

#### **2.6.4 Power Supply**

A power supply that converts 220V AC to 5V DC is known as a step-down converter or AC-DC adapter. It works by receiving 220V AC from the mains electricity and passing it through a transformer to reduce the voltage. The lower AC voltage is then converted to DC using a rectifier. After rectification, the DC voltage is filtered to smooth out any fluctuations and regulated to ensure a constant 5V output. These power supplies are commonly used for charging USB devices, powering microcontrollers, and providing power to other low-voltage electronics. They can be found as wall adapters, DC-DC converters, or internal power supplies in various devices.

#### **2.6.5 ESP32-WROOM-32**

The ESP32 is chosen for the control block of IoT systems due to its versatile dual-core microcontroller, integrated Wi-Fi, and Bluetooth capabilities. These features enable seamless connectivity for both local and wireless communication needs essential in IoT applications. Its cost-effectiveness and energy efficiency are particularly beneficial for battery-powered devices, supporting prolonged operation in low-power modes when idle. Moreover, the ESP32's extensive peripheral interfaces, such as GPIOs, ADCs, and UART, facilitate easy integration with various sensors and actuators commonly used in IoT environments. Supported by a robust developer community and accessible programming environments like Arduino IDE and MicroPython, the ESP32 offers a straightforward development process, making it a preferred choice for diverse IoT applications. ESP32-WROOM-32 Bluetooth and Wi-Fi transceiver RF kit is integrated with antenna and RF, energy saving, stable operation, good anti-interference, this is the lowest cost solution for a project with a circuit using 2.4Ghz Wi-Fi and Bluetooth TSMC low power 40nm technology [8].

#### **2.6.6 Relay Module 4**

A 4-channel relay module is a versatile electronic component used to control multiple high-power devices using low-power signals from microcontrollers or other control systems. Each channel on the module includes a relay switch capable of handling significant currents and voltages. These relay channels provide electrical isolation between the control (low voltage) and load (high voltage) sides, ensuring the safety of sensitive electronics from potential voltage spikes. The module simplifies wiring and integration into projects with screw terminals or headers for secure connections. It finds applications in home automation, industrial automation, robotics, and various IoT projects, offering a cost-effective solution for managing and switching multiple devices independently and simultaneously. Module 4 Relay H/L 5VDC trigger uses 5VDC source to power the circuit,

the trigger signal can optionally trigger high level (High - 5VDC) or low level (Low - 0VDC) via Jumper on each relay. Suitable for devices using 5VDC signal level [9].

## **2.6.7 Home Appliance**

### **2.6.7.1 Fan**

5VDC 40mm Sunon Maglev/Vapo Bearing Cooling Fan (#ME40100V1-0000-A99) with a thermoplastic frame is designed for long-life and minimal noise operation. Sunon® fans are manufactured to a very high standard and hold CSA and UL listings. Altronics has been a Sunon distributor for many years and has found them to be of excellent overall quality and reliability [10].

### **2.6.7.2 Led**

The 5mm LED is a versatile and widely used component known for its energy efficiency, durability, and versatility in various lighting and indication applications [11].

### **2.6.7.3 Solenoid Lock**

The uxcell DC 5V 1A 30g 3mm Mini Electromagnetic Solenoid Lock Pull Type. It offers a compact, low-voltage solution for securing cabinet doors or similar applications where space and power constraints are important considerations [12].

## **Chapter 3. DESIGN AND IMPLEMENTATION**

This chapter illustrates the integration of advanced technologies to develop a secure and convenient smart home system. By utilizing facial recognition, the system enhances home security by permitting only authorized individuals to unlock the door. Additionally, hand gesture control offers a touch-free and intuitive method for managing home appliances, thereby enhancing convenience and the overall user experience. This successful integration of technologies results in a responsive and intelligent home automation solution that prioritizes both security and user comfort.

### **3.1 SYSTEM REQUIREMENTS**

#### **Hardware Components:**

**Main Processing Unit:** Raspberry Pi 4 serves as the central processing unit for image processing, facial recognition, hand gesture recognition, and managing system functionalities.

**ESP32 Microcontroller:** Interfaces with the Raspberry Pi and connects to the database via Wi-Fi for controlling the relay module.

**Power Supply:** Provides electrical power to all system components, ensuring uninterrupted operation.

**Camera:** Captures images for facial recognition and hand gesture recognition.

**Relay Module:** Interfaces between the ESP32 and electrical devices, enabling remote control over household appliances.

**Electrical Devices:** Appliances and devices within the household that can be controlled through the system.

#### **Software Components:**

**Wi-Fi Communication Protocol:** Implemented on both the Raspberry Pi and ESP32, this protocol establishes a connection with the cloud database service and facilitates remote control over home appliances. It ensures secure and reliable communication between the system components via Wi-Fi connectivity.

**Google Firebase:** A cloud-based database service used to store information about household devices and system states.

**Image Processing Module:** This module, executed on the Raspberry Pi, is responsible for processing images captured by the camera.

- **Facial Recognition Algorithm:** Running on the Raspberry Pi, this module utilizes machine learning algorithms to perform facial recognition. It analyzes the processed image data from the Image Processing Module to identify faces, extract facial features, and authenticate access to the smart home system.

- **Hand Gesture Recognition Algorithm:** Also executed on the Raspberry Pi, this module recognizes hand gestures captured by the camera and translates them into commands for controlling household devices. It processes gesture data to provide an intuitive and hands-free control mechanism for users.

**Graphical User Interface (GUI):** An application managed by the homeowner, accessible via the internet. The GUI displays family member information and provides device control functionalities.

**Home Appliance Control Module:** Implemented on the ESP32 to control home appliances based on the state stored in the database.

### **System Features:**

**Data Transmission via Wi-Fi:** The Raspberry Pi sends processed data, including member information and device control commands, to the cloud database via Wi-Fi.

**Remote Appliance Control via Wi-Fi:** The ESP32 receives control commands from the real-time database via Wi-Fi and interfaces with the relay module to control home appliances accordingly.

**Facial Recognition Door Unlock:** Family members can unlock the door by looking at the camera. When the GUI switches to face recognition mode, the system will automatically recognize the face via the Raspberry Pi. The result is then pushed to Google Firebase, which sends data to the ESP32 over Wi-Fi to control the door lock.

**Hand Gestures Control Home Appliance:** The system has an automatic mode to recognize hand gestures without face recognition. Members can control household devices by making hand gestures in front of the camera. The system uses the MediaPipe library to recognize gestures (THUMB, INDEX, MIDDLE, RING, PINKY) and execute the desired device commands.

**User Interface:** An app accessible by the homeowner that displays family member information and allows remote device control, enhancing convenience and accessibility.

### **Conclusion:**

This smart home system integrates face and gesture recognition technology to enhance security and convenience. By using a Raspberry Pi for image processing, an ESP32 for device control via Wi-Fi, and Firebase for data storage, the system offers an effective and accessible remote-control solution through a user-friendly app. The system ensures seamless data transmission and device control, creating an intelligent and convenient user experience.

### 3.2 BLOCK DIAGRAM

Figure 3.1 below illustrates the architecture of a smart home system integrating facial recognition for door unlocking and hand gesture control for home appliances.

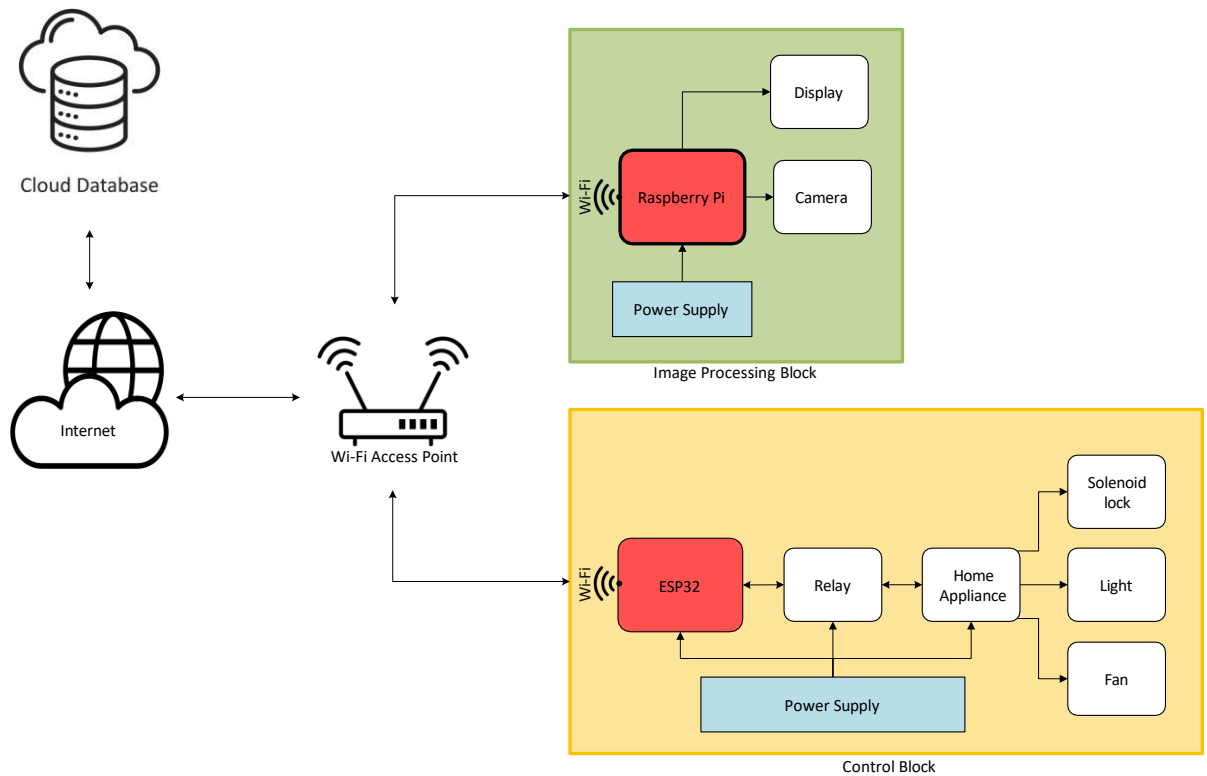


Figure 3.1. Block diagram of system

**Wi-Fi:** This is a wireless connection protocol used by both the ESP32 and Raspberry Pi. These devices connect to the Wi-Fi network for seamless data transmission and command retrieval.

**Camera:** This module operates as a continuous peripheral camera, capturing facial and hand gesture images. These images are then sent to the processing unit for recognition and analysis. Subsequently, the processed data is transmitted to Google Firebase via Wi-Fi.

**Raspberry Pi:** Serving as the central processing unit, the Raspberry Pi handles image processing and communicates with the database via API and Wi-Fi.

**Display:** This module serves as the display screen responsible for presenting all result information.

**Cloud Database:** Acting as a central repository for processed data, the cloud database can be a cloud-based service like Google Firebase.

**ESP32:** This microcontroller interacts with the database to control a relay, making it suitable for various IoT applications.

**Power Supply:** This component provides voltage and current to the entire system, ensuring flawless operation.

**Relay:** The relay enables remote control and automation of electrical devices.

**Home Appliances:** These devices connect to and are controlled by the system, such as solenoid locks, lights, fans, etc.

### **3.3 OPERATING PRINCIPLE**

The system's operating principle is designed to enhance home automation and security through facial recognition and hand gesture control. Below is a detailed description of how the system operates:

- **Image Capture and Processing: Facial Recognition for Door Unlocking:** A camera captures images of individuals at the door. These images are processed by a Raspberry Pi using the YOLOv8 model to detect and recognize faces. If the recognized face matches an authorized user, the door unlocks.
- **Hand Gesture Recognition for Appliance Control:** The same or a different camera captures hand gestures within the home. The images are processed by the Raspberry Pi using the MediaPipe framework to identify specific hand gestures. Each recognized gesture corresponds to a control command for various home appliances (e.g., turning lights on/off, adjusting fans, etc.).
- **Data Transmission:**
  - **Processing Data:** The Raspberry Pi processes the image data to determine the presence of individuals and the specific hand gestures made. Based on this processing, the Raspberry Pi generates relevant commands (e.g., unlock door, turn on light).
  - **Database Communication:** The Raspberry Pi transmits the processed data and commands to a central database via Wi-Fi. This database serves as an intermediary, ensuring that data and commands are available to other connected devices in the system.
- **ESP32 Control:**
  - **Retrieving Data:** The ESP32 microcontroller connects to the central database via Wi-Fi. It retrieves the latest commands and data processed by the Raspberry Pi.
  - **Executing Commands:** Based on the retrieved data, the ESP32 controls connected relays and home appliances. For example, if the database indicates a recognized hand gesture to turn on the lights, the ESP32 activates the corresponding relay to power the lights.
- **Relay and Appliance Control:**
  - **Relay Activation:** The ESP32 sends signals to the relay module, which in

turn controls the power supply to various home appliances. Relays act as switches that can turn appliances on or off based on the commands received from the ESP32.

- **Home Appliance Management:** Home appliances such as light, fan, and solenoid lock are controlled through the relays. The system ensures that these devices operate according to the user's gestures and the processed commands from the Raspberry Pi.

### 3.4 SOFTWARE DESIGN

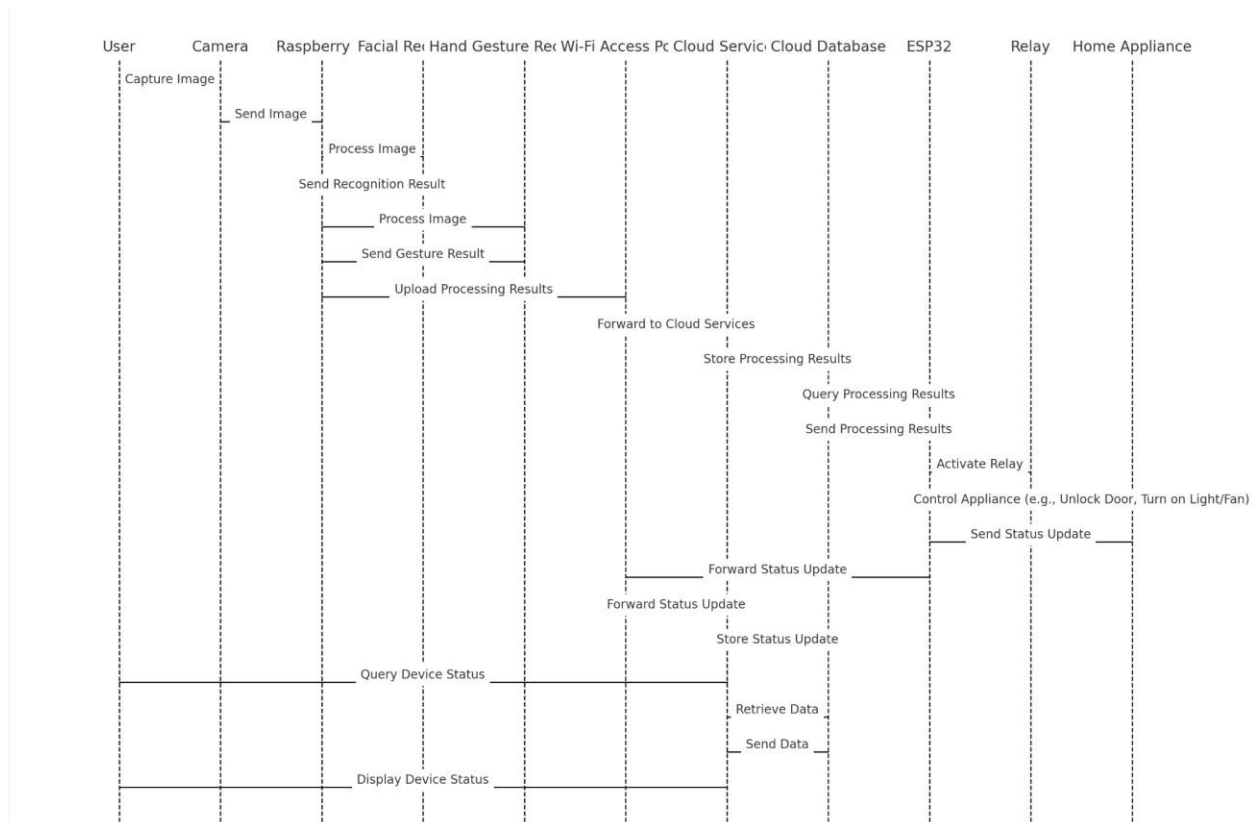


Figure 3.2. The sequence diagram of system

Figure 3.2 illustrates the sequence diagram, clearly depicting the flow of data and control signals between the components. It includes the step where image processing results are uploaded to the Cloud Database via the Wi-Fi Access Point.

#### 3.4.1 IoT System

##### 3.4.1.1 Building Database

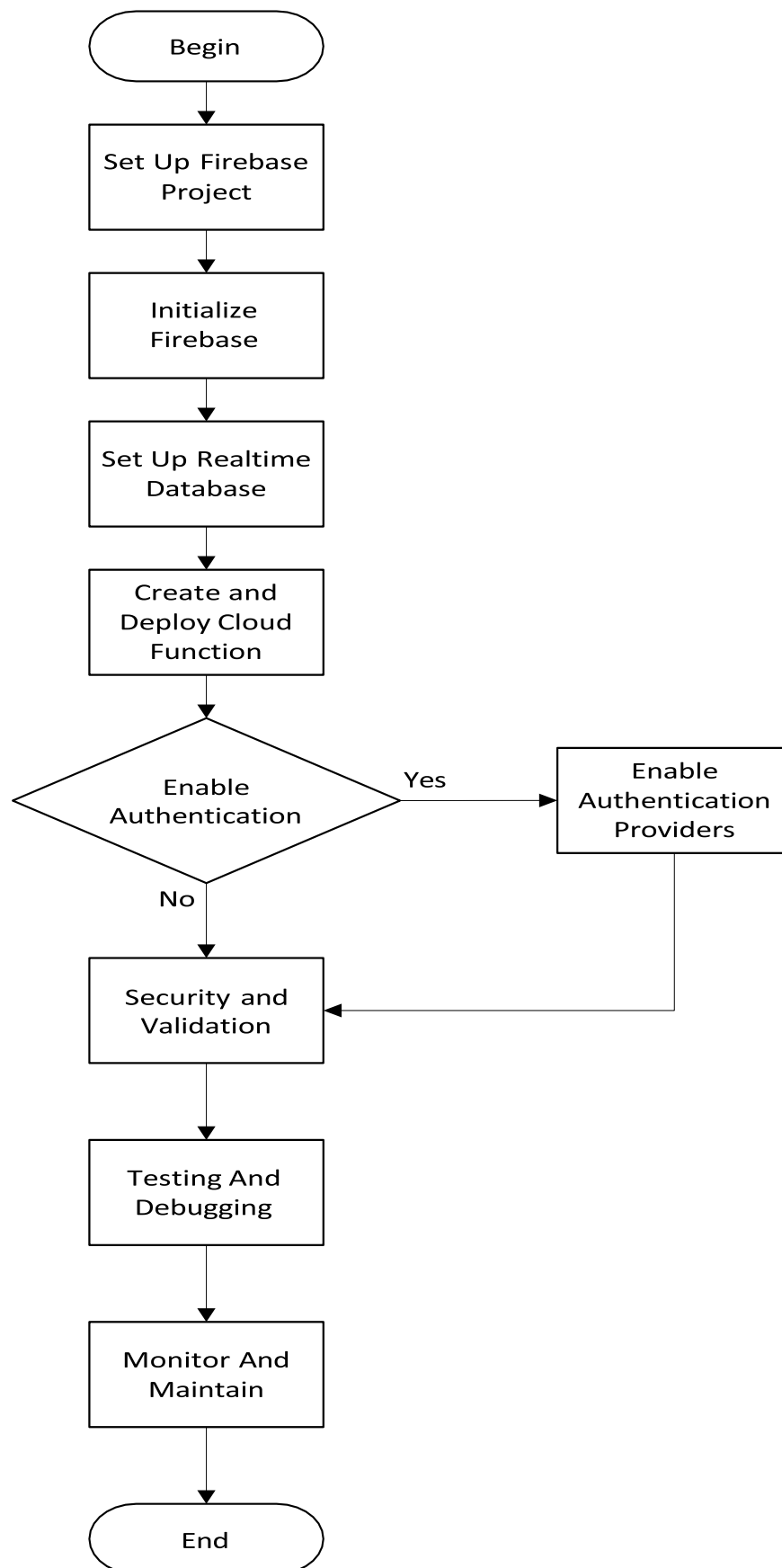


Figure 3.3. The flowchart of building database.

Figure 3.3 presents a comprehensive flow diagram detailing the steps required to set up and integrate a Firebase-powered application. This diagram systematically outlines the



process from initial setup to deployment, ensuring a clear understanding of each stage involved.

The process of setting up and integrating a Firebase-powered application involves several distinct phases, each crucial for successful implementation. First, in the Setup Firebase Project phase, a new Firebase project is created in the Firebase console where project settings are configured and necessary credentials are obtained.

Next, during the Initialize Firebase phase, Firebase is initialized using these credentials, enabling the application to interact with various Firebase services. In the Setup Realtime Database phase, a decision is made to use Realtime Database as the data storage solution. The chosen database service is then configured, including the establishment of necessary data structures and security rules.

Following this, the Create and Deploy Cloud Functions phase involves the development and deployment of custom server-side logic using Firebase Cloud Functions, allowing the application to handle complex operations on the server side. The Enable Authentication phase represents the decision to enable user authentication functionality. If required, the necessary authentication providers are integrated and configured; otherwise, the process skips directly to the Security and Validation phase.

Here, appropriate security rules and data validation mechanisms are established to ensure that only authorized users can access and modify data, and that the data conforms to expected structures and constraints. During the Testing and Debugging phase, the Firebase-integrated application is thoroughly tested to ensure all components function as expected. Any issues that arise are identified and addressed.

Finally, the Monitor and Maintain phase involves ongoing monitoring and maintenance of the application. This includes implementing logging, error reporting, and monitoring mechanisms to quickly identify and resolve issues. Regular reviews and updates are performed to keep the application secure and up-to-date..

### **Database location**

Firebase Realtime Database is a cloud-based database service provided by Google. This service ensures high reliability and scalability by storing data on Google's servers. Utilizing Firebase Realtime Database eliminates the need for managing physical servers or complex infrastructure, as all backend operations are managed by Google.

Designing a database for IoT devices such as fans, door locks, LEDs, cam detection, and hand detection requires a structured approach to ensure data integrity, security, and performance.

After building the database, we design the structure of the database as Figure 3.4 shown below.

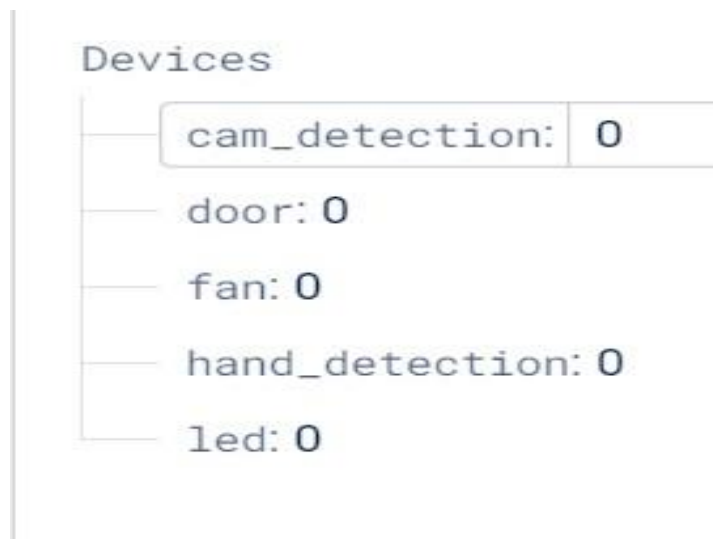


Figure 3.4. The structure of database

Figure 3.4 describes the structure of the database we have built. We define variables upload to the database so that Raspberry Pi and ESP32 can communicate data to easily control the devices. The "Devices" node contains several key-value pairs representing the state of different devices connected to the system:

cam\_detection: 0

door: 0

fan: 0

hand\_detection: 0

led: 0

These entries indicate the current states of various devices, such as the camera detection status, door lock status, fan, hand gesture detection, and LED light status. The values (0 or 1) signify whether each device is off (0) or on (1). This real-time database setup enables synchronized control and monitoring of home automation devices using the ESP32 microcontroller.

Finally, we need to creating API key.

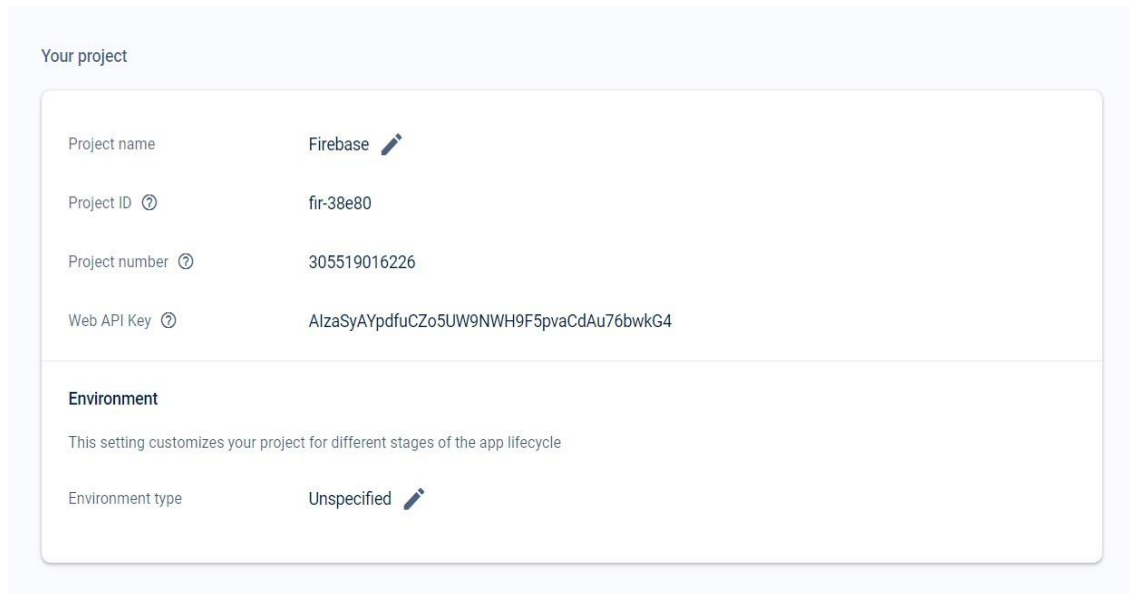


Figure 3.5. API of database

Figure 3.5 shows the API after initialization on Firebase.

#### 3.4.1.2 Raspberry Pi transmits data to Database

##### Raspberry Pi transmits data to Firebase

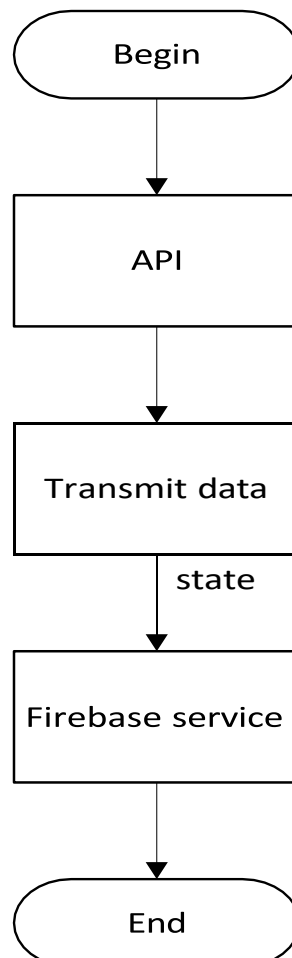


Figure 3.6. The block diagram of Raspberry Pi transmit data to Firebase

Figure 3.6 illustrates a streamlined block diagram detailing the process of transmitting data from a Raspberry Pi using Firebase services. The flowchart starts by initiating the process, marking the beginning of a systematic sequence aimed at efficiently sending data to Firebase.

The next step involves an "API" where the data interface is established, allowing for interaction with the data source. Following this, the "Transmit data" step takes place, where data is sent to the Firebase service. A specific "state" parameter is managed during this transmission to ensure proper handling and synchronization of the data.

Subsequently, the data is received and processed by the "Firebase service," which is responsible for storing and managing the transmitted data. Finally, the process signifies the successful completion of the data transmission to Firebase.

### **Configuration API on Raspberry Pi**

Here is a breakdown of important field in the JSON file:

- `private_key_id`: The ID of the private key used for authentication.
- `private_key`: The private key itself, used to authenticate requests.
- `auth_uri`: The URI to which authentication requests are sent.
- `token_uri`: The URI to which token requests are sent.

The main purpose of the JSON file is to initialize and set up the appropriate parameters to connect to the firebase service using the API.

### **Deploy API on Raspberry Pi**

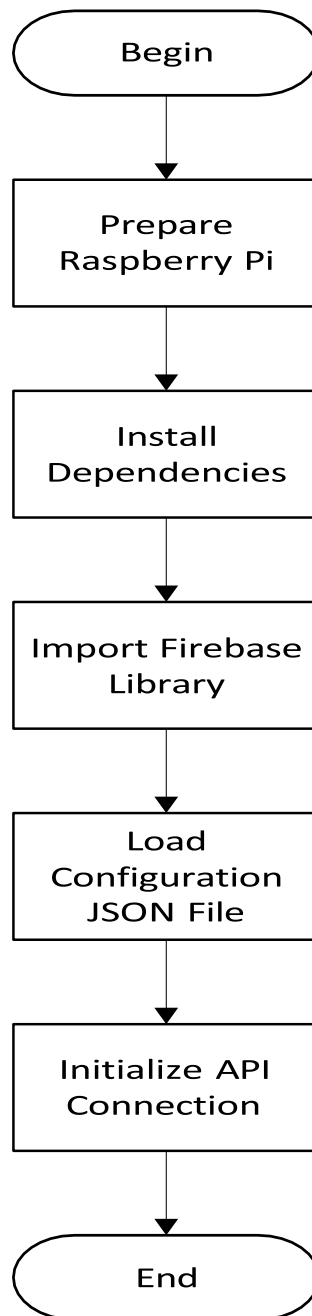


Figure 3.7. The flowchart of deploy API on Raspberry Pi

Figure 3.7 displays an algorithm flow chart for delivering API on Raspberry Pi. To set up a Firebase API connection in Python on a Raspberry Pi, first prepare the device by connecting it to electricity and the internet and installing any necessary software, including OS updates. Next, install Python and the Firebase Admin SDK. Once the prerequisites are installed, you may import the appropriate libraries into your Python script. After you've imported the libraries, load your Firebase setup JSON file, making sure to use the correct location. Finally, set up the Firebase API connection. This will establish a connection to your Firebase project, enabling your Python application to communicate with Firebase services.

### **The process of data uploading to the database**

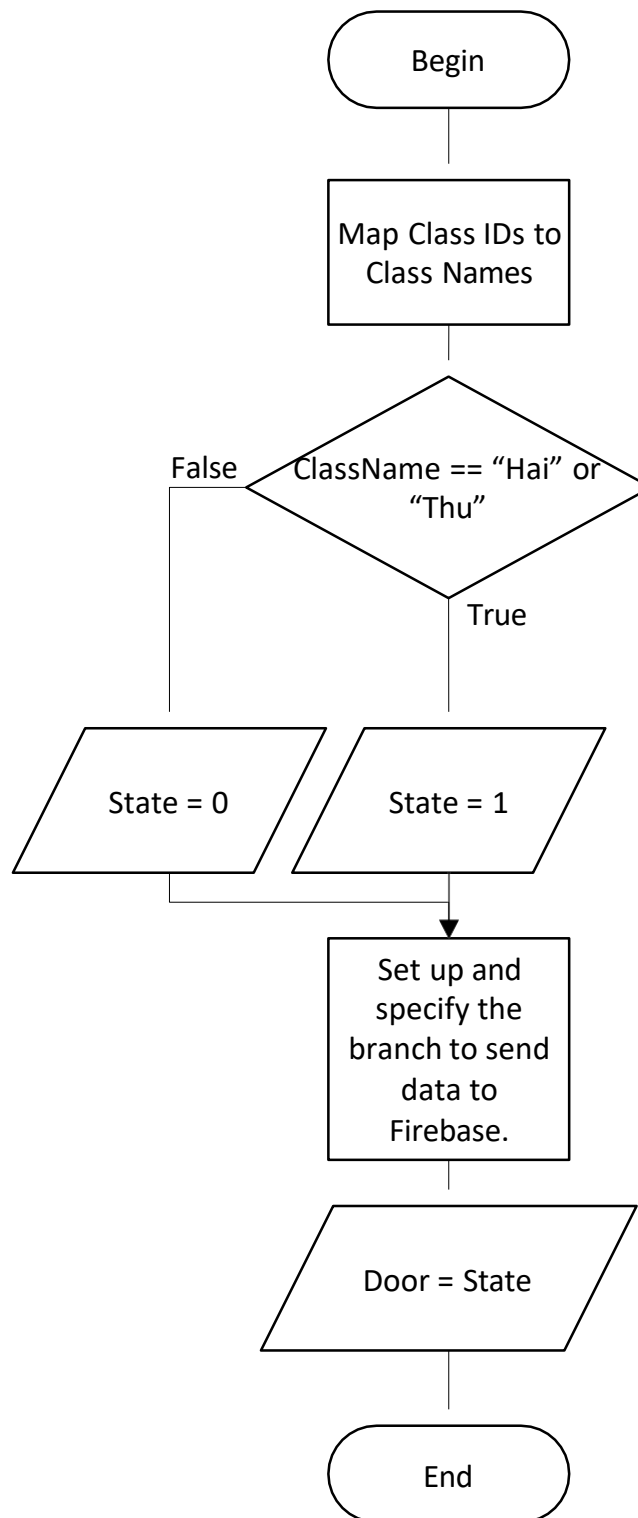


Figure 3.8. The flowchart depicting the process of data uploading to the database after facial recognition

Figure 3.8 outlines the process of setting up and uploading data to Firebase based on facial recognition. The flowchart handles facial recognition with YOLO. It starts by mapping class IDs to class names. A decision is then made based on whether the class name is "Hai" or "Thu". If the class name is "Hai" or "Thu", the state is set to 1; otherwise, it is set to 0. The process then sets up the branch to send this state data to Firebase and assigns the state value to the "Door" variable, indicating whether the door should be opened or closed based on the facial recognition result. The process ultimately converges at the

"End" node, completing the process. This flowchart provides a clear depiction of the process of facial recognition, along with the corresponding actions for data transmission to Firebase.

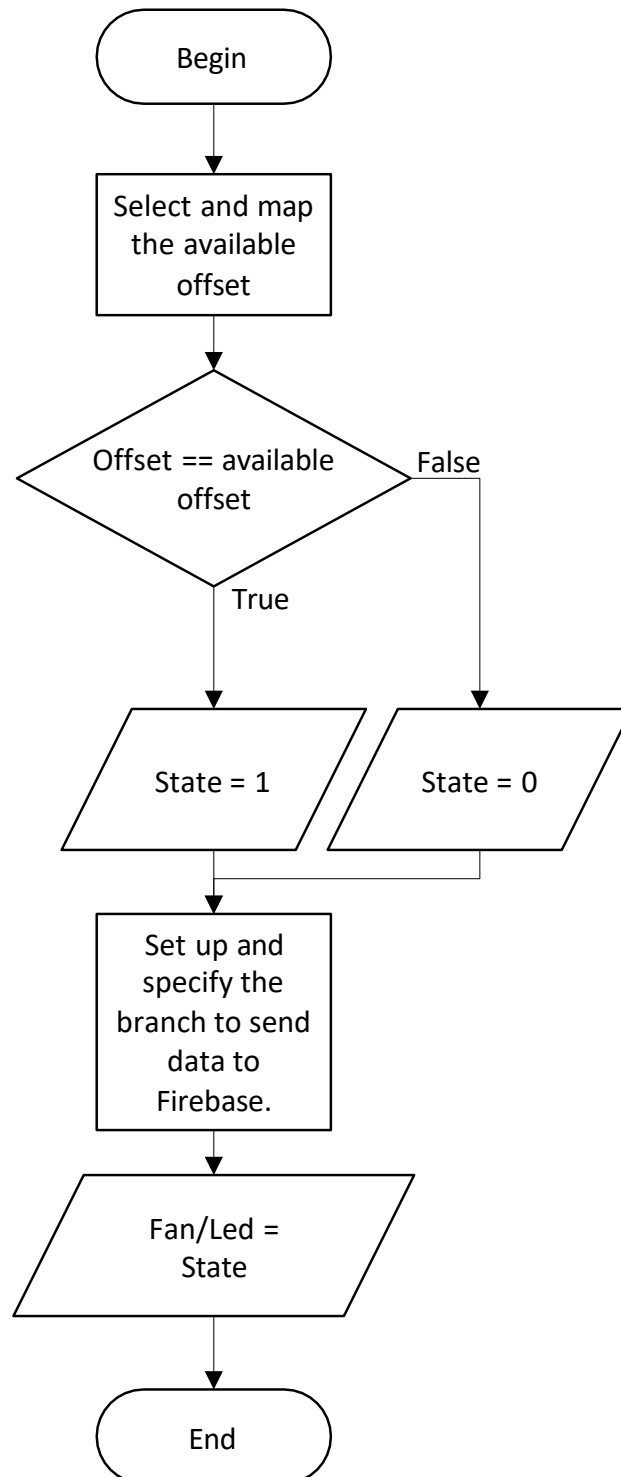


Figure 3.9. The flowchart depicting the process of data uploading to the database after hand gestures recognition

Figure 3.9 outlines the process of setting up and uploading data to Firebase based on hand gestures recognition. The flowchart handles hand gesture recognition with MediaPipe. It begins by selecting and mapping the available offset. A decision is then made

based on whether the offset matches the available offset. If it matches, the state is set to 1; otherwise, it is set to 0. The process sets up this branch to send the state data to Firebase and assigns the state value to the "Fan/LED" variable, indicating whether the fan or LED should be turned on or off based on the hand gesture recognition. The process ultimately converges at the "End" node, completing the process. This flowchart provides a clear depiction of the process of hand gestures recognition, along with the corresponding actions for data transmission to Firebase.

#### 3.4.1.3 ESP32 receives data to Database

##### ESP32 receives data to Firebase

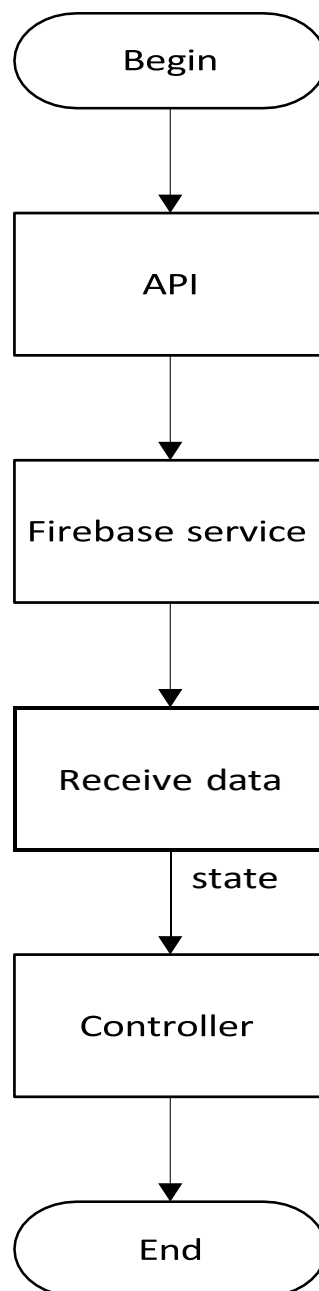


Figure 3.10. The block diagram of ESP32 receives data to Firebase

Figure 3.10 depicts a flowchart outlining a process involving data handling from an ESP32 device through Firebase services. This flowchart systematically illustrates the steps



involved in transmitting and managing data between the ESP32 microcontroller and Firebase.

The process begins by indicating the initiation of the workflow. The next step is the "API," where the data interface is established to interact with the data source or client. Following this, the data is managed by the "Firebase service," which stores and processes the incoming data.

The "Receive data" step follows, where the processed data is received and prepared for further actions. A "state" parameter is involved to maintain the status and integrity of the data throughout the process. Subsequently, the "Controller" step manages and directs the received data to its intended destination or application.

Finally, the process signifies the successful completion of the data handling and processing workflow.

### **Configuration API on ESP32**

Configuring an API on an ESP32 involves setting up the ESP32 microcontroller to communicate with an API over Wi-Fi. This typically includes connecting to a Wi-Fi network, making HTTP requests to the API, and handling the responses.

### **Deploy API on ESP32**

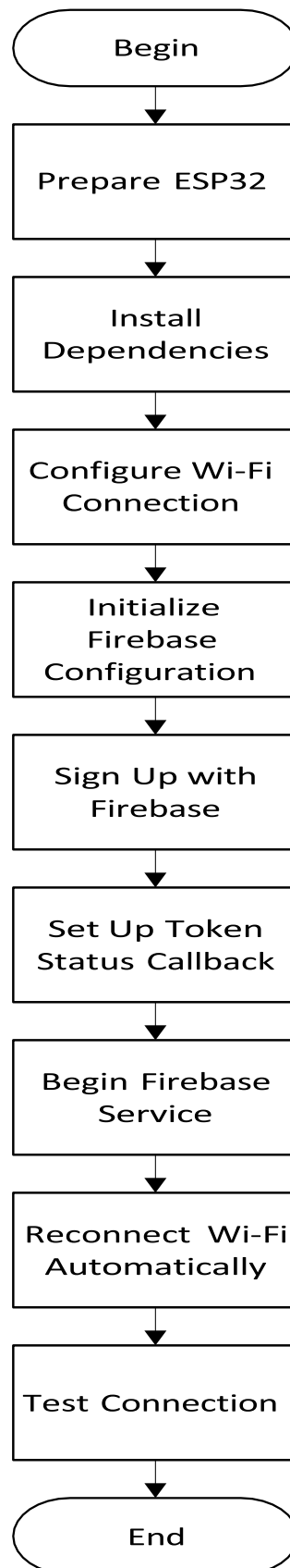


Figure 3.11. The flowchart of deploy API on ESP32

Figure 3.11 displays the algorithm flowchart for launching an API on the ESP32. To publish an API on an ESP32 using Firebase, follow these steps:

First, prepare your device by downloading the Arduino IDE and enabling ESP32 board support. Next, use the Arduino IDE's Library Manager to install the necessary libraries. Configure your ESP32 to connect to a Wi-Fi network by setting the SSID and password, and then establish the connection.

Initialize the configuration object using your Firebase API key and Real-Time Database URL. Sign up for Firebase and handle the response to set a flag based on whether the sign-up procedure succeeded or failed. Assign a callback function to manage token status.

Initialize the Firebase service with the configured settings and authentication. Enable automatic Wi-Fi reconnection using `Firebase.reconnectWiFi(true)`. Finally, test the connection by ensuring the ESP32 can connect to Firebase and interact with the database, printing messages to the serial monitor for debugging.

Here are the important fields and their roles in the Firebase configuration:

- `API_KEY`: Essential for authenticating requests to Firebase.
- `DATABASE_URL`: Your actual Firebase Realtime Database URL.
- `config.token_status_callback`: Assigns a callback function that handles the status of token generation. Useful for monitoring and reacting to token generation processes.
- `Firebase.begin(&config, &auth)`: Initializes the Firebase library with the provided configuration and authentication details, setting up the connection to Firebase services.
- `Firebase.reconnectWiFi(true)`: Configures Firebase to automatically reconnect to Wi-Fi if the connection drops, ensuring continuous connectivity.

The main purpose of the provided code snippet is to configure and initialize Firebase services on an embedded system, such as an ESP32.

### **The process of receiving data from the database**

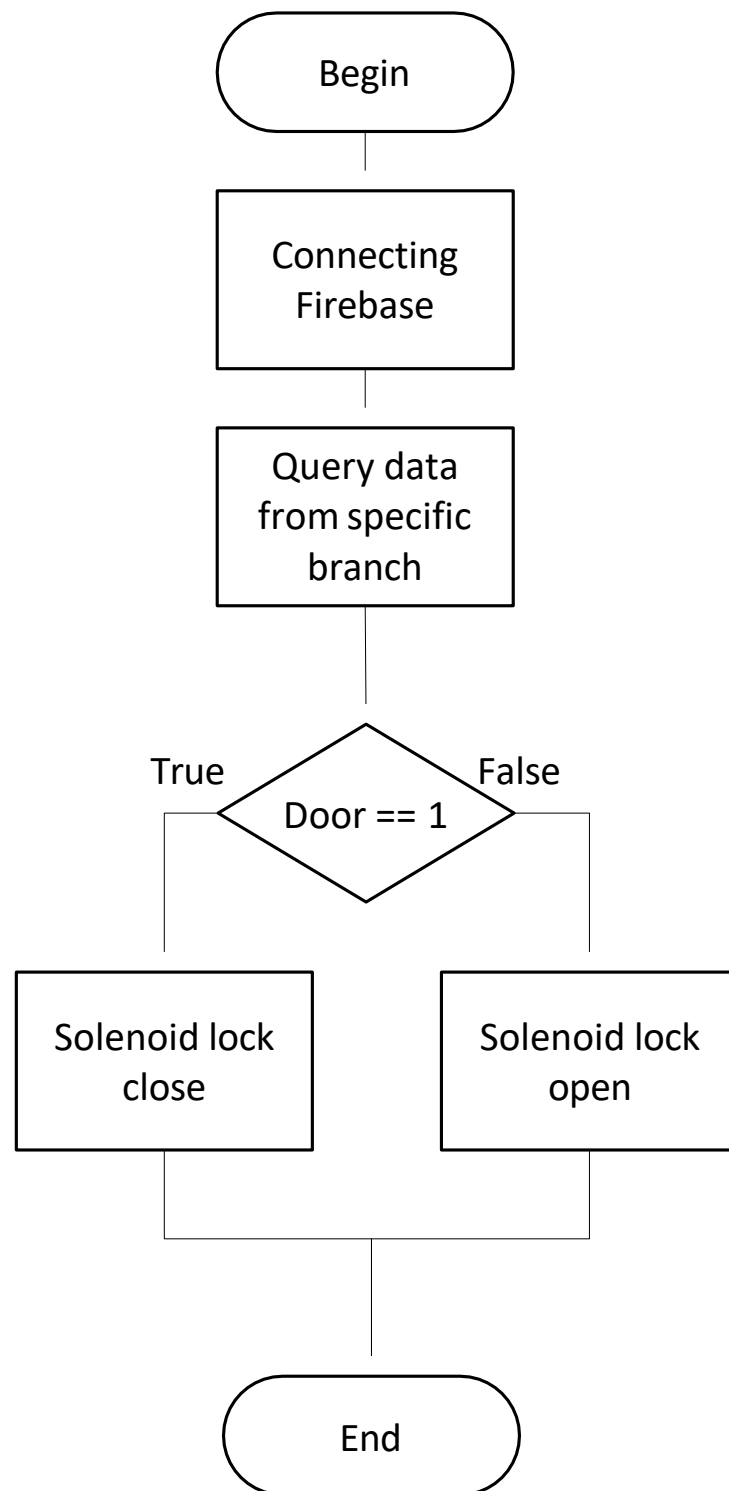


Figure 3.12. The flowchart for the process of receiving data from the database to control door

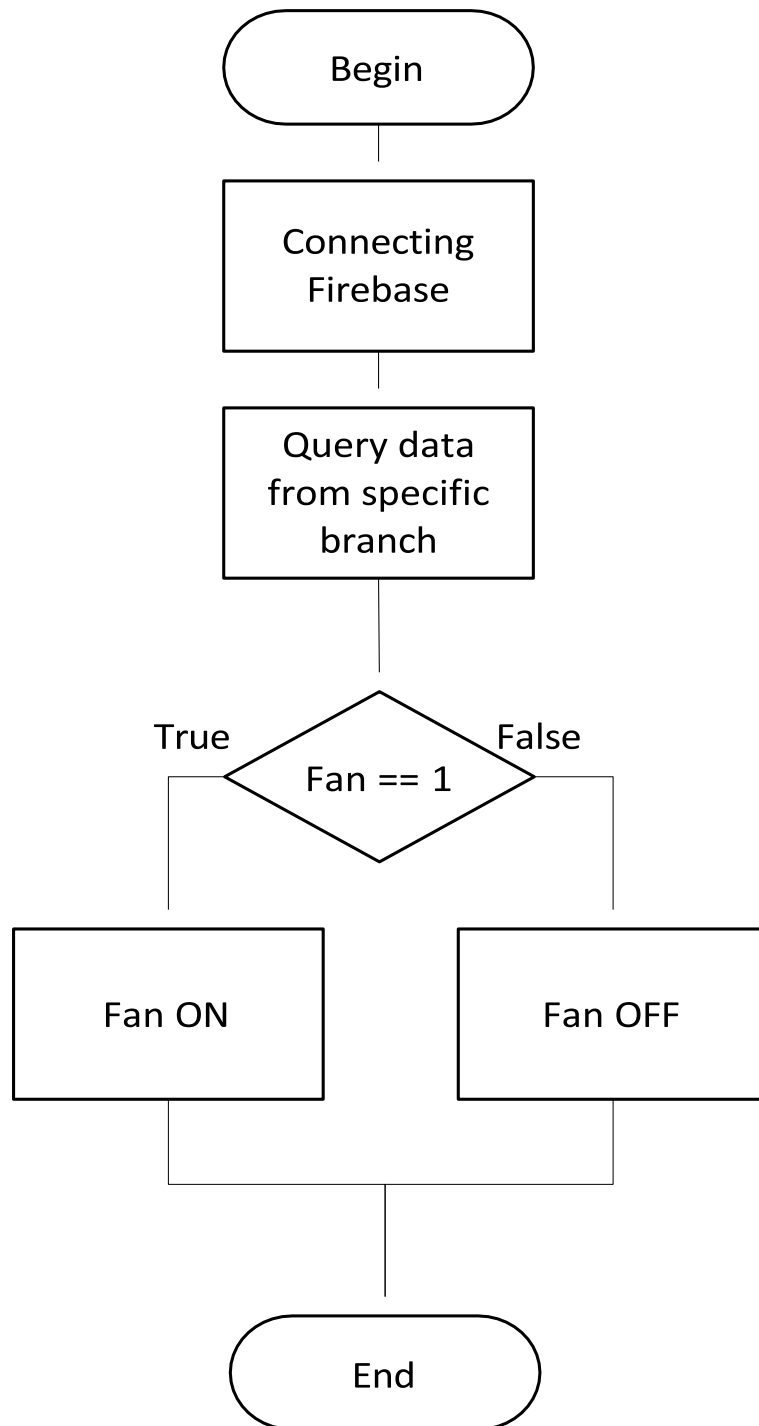


Figure 3.13. The flowchart for the process of receiving data from the database to control fan

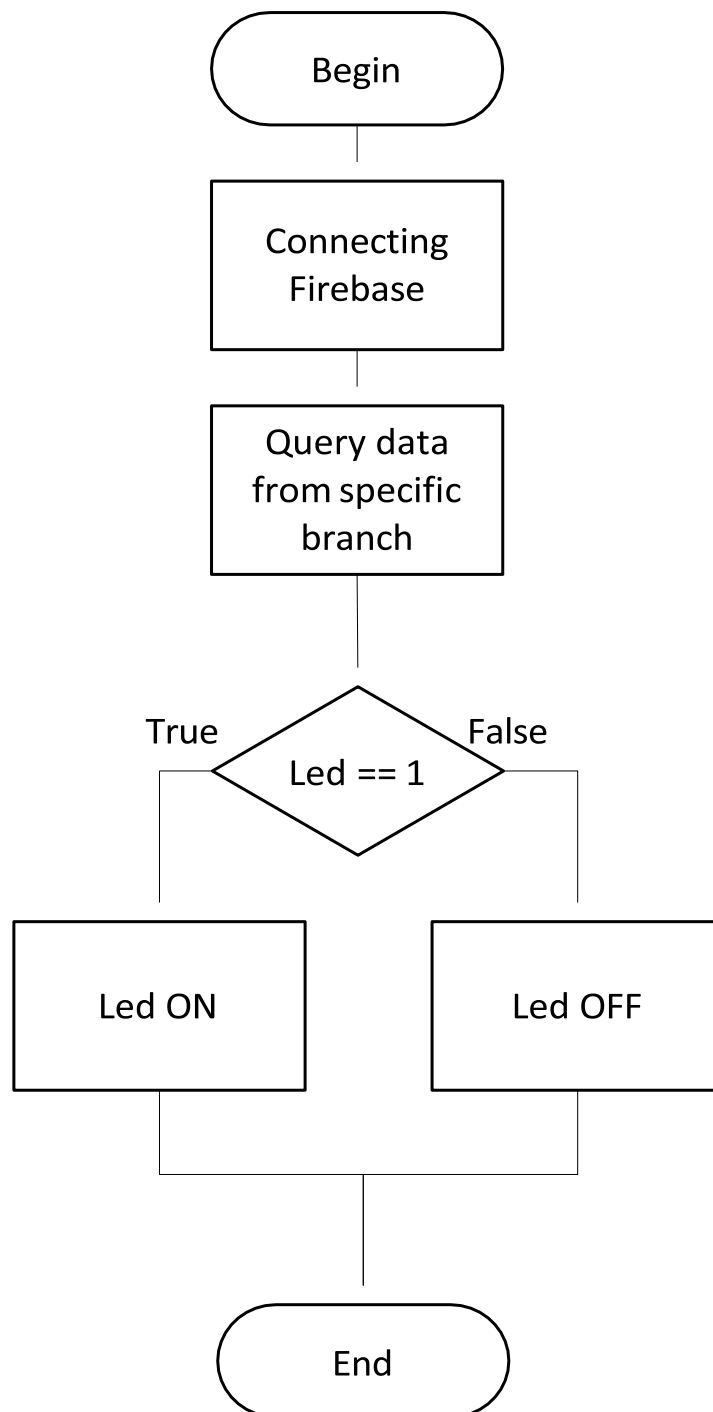


Figure 3.14. The flowchart for the process of receiving data from the database to control led

Figure 3.12, Figure 3.13 and Figure 3.14 illustrate flowcharts representing algorithms for controlling a system. It begins with a "Start" step that establishes a connection to the Firebase database, followed by a step that retrieves data from each branch. Next, the diagram evaluates the conditions related to the "Door," "Fan," and "LED" states and executes corresponding actions such as locking/unlocking the solenoid, turning the fan on/off, and switching the LED on/off. Finally, the process concludes at the "End" step. Overall, this flowchart illustrates the control and data processing logic for an automation system, without referencing any specific individuals.

### 3.4.2 Image Processing Block

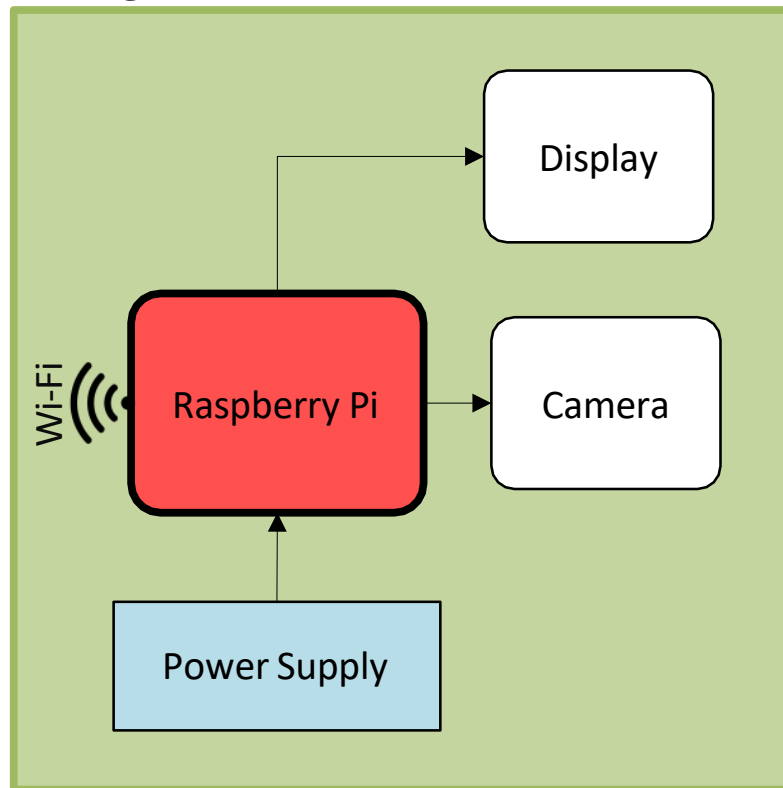


Figure 3.15. Processing Block

Figure 3.15 above depicts a smart home system centered around a Raspberry Pi with an integrated Wi-Fi module, which uses advanced image processing techniques for home automation and security. The system includes a camera, a display, and a power supply, all interfacing with the Raspberry Pi. The camera captures video footage and sends it to the Raspberry Pi for processing. The Raspberry Pi employs YOLO for facial recognition to unlock a door, ensuring secure access control. Additionally, it uses MediaPipe to recognize hand gestures, enabling users to control various home appliances through intuitive gestures.

The processed data, including facial recognition results and hand gesture commands, is then sent to a cloud-hosted database. This database can store and manage the data, allowing for further actions such as logging access events or adjusting appliance settings based on user preferences. The display connected to the Raspberry Pi can show real-time feedback, such as confirming recognized faces or detected hand gestures, enhancing user interaction with the system. The entire setup is powered by a dedicated power supply, ensuring consistent operation. This smart system not only enhances home security by using facial recognition but also improves convenience by allowing gesture-based control of home appliances, all while maintaining a seamless flow of data to the database for efficient management and control.

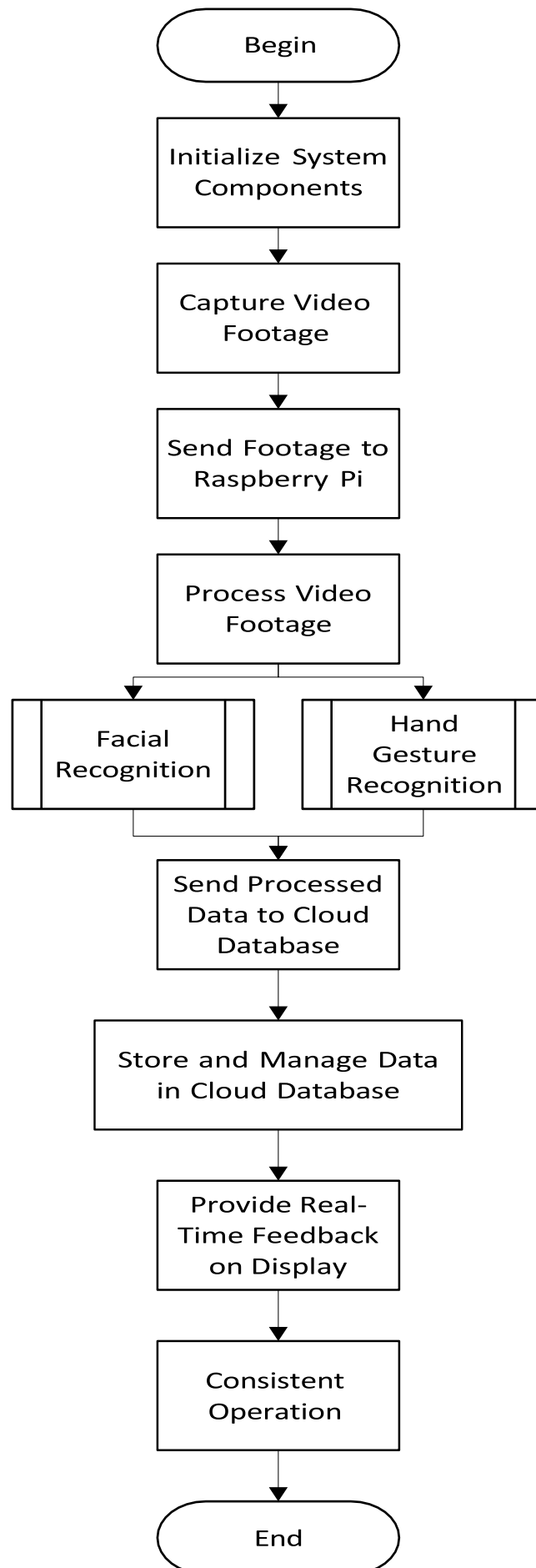


Figure 3.16. The flowchart of image processing block



Figure 3.16 outlines a flowchart for deploying an API on an ESP32 microcontroller using Firebase, starting with the initialization of system components, including the power supply, camera, display, and Wi-Fi module. The process starts with the camera capturing video footage, which is then sent to the Raspberry Pi for processing. The Raspberry Pi employs YOLO for facial recognition and MediaPipe for hand gesture recognition. Facial recognition results are used to unlock the door, ensuring secure access control, while recognized hand gestures allow users to control various home appliances intuitively. The processed data, including facial recognition outcomes and hand gesture commands, is transmitted to a cloud-hosted database. This database stores and manages the data, enabling further actions such as logging access events and adjusting appliance settings based on user preferences. The display connected to the Raspberry Pi provides real-time feedback, confirming recognized faces and detected hand gestures, which enhances user interaction with the system. Throughout this process, the power supply ensures consistent operation of all components.

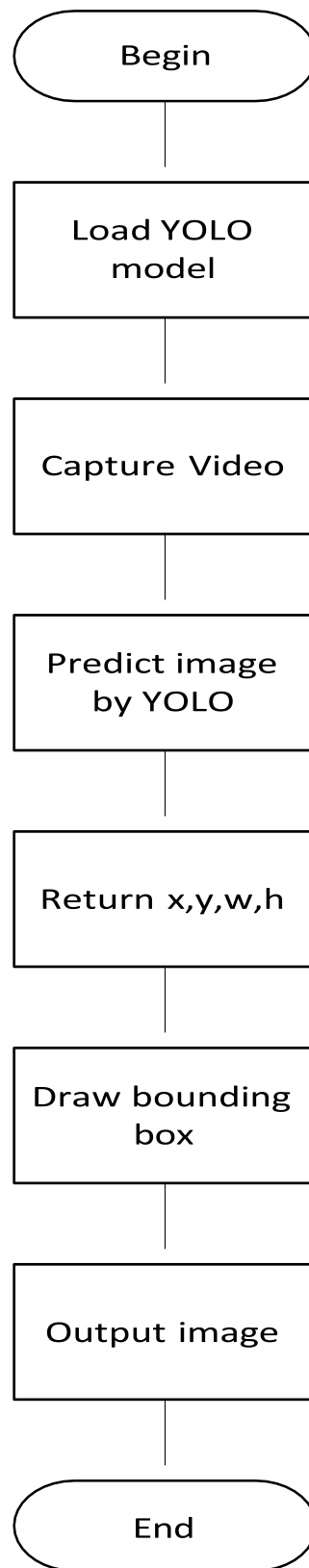


Figure 3.17. The flowchart of facial recognition process on Rasp

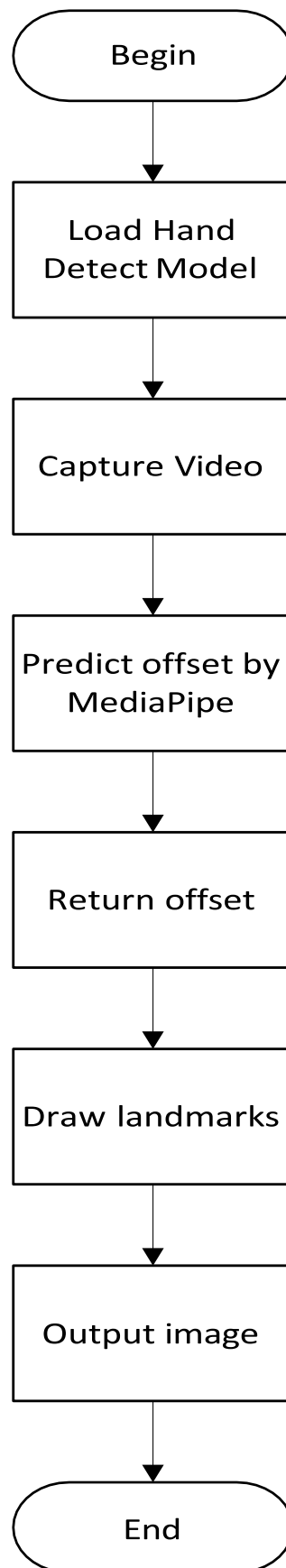


Figure 3.18. The flowchart of hand gestures recognition process on Rasp

Figure 3.17 and Figure 3.18 illustrate the algorithm flowchart for image processing of facial and hand gestures recognition on Raspberry Pi. The image processing process for

face and hand gesture recognition involves distinct steps. For face recognition using YOLO, it starts with image acquisition, pre-processing and face detection using YOLO. For hand gesture recognition using MediaPipe, it begins with image acquisition, pre-processing, and hand detection.

### **3.4.2.1 Facial Recognition**

#### **3.4.2.1.1 Building Custom Dataset**

To build a comprehensive and diverse facial recognition dataset, you can follow these steps:

- **Recording Videos**

**Different Angles:** Record videos from various angles of the face, including straight-on, side profiles, top-down, and bottom-up views.

**Various Lighting Conditions:** Record videos under different lighting conditions such as daylight, low light, and bright light to ensure data diversity.

**Different Distances:** Record videos at varying distances, from close-up shots to distances of about 1 to 2 meters, to enhance the system's ability to recognize faces in different real-world scenarios.

- **Extracting Frames from Videos**

**Frame Extraction:** Use software or programming scripts to extract frames from the recorded videos. This helps in generating multiple facial images from each video.

**Frame Selection:** Ensure the extracted frames include various facial poses, expressions, and lighting conditions to increase the dataset's diversity.

- **Data Preprocessing:**

1. **Upload Images to Roboflow:** First, upload the images extracted from the videos to Roboflow. Roboflow is a powerful tool that makes it easy to crop and label faces in images.

2. **Use Roboflow to Create Labels:**

**Register and Log In:** Visit Roboflow, register, and log in to your account.

**Create a New Project:** Create a new project on Roboflow.

**Upload Images:** Upload the images extracted from the videos.

**Label Faces:** Use Roboflow's labeling tool to create bounding boxes around the faces in each image and assign labels to each face. Labels can include the names of individuals (e.g., "Hai," "Thu," "Vy").

3. **Label Each Face**

**Labeling Accuracy:** Ensure that you accurately label each face. This helps the machine learning model to correctly recognize each individual. Figure 3.19 illustrates an example of labeling each face.

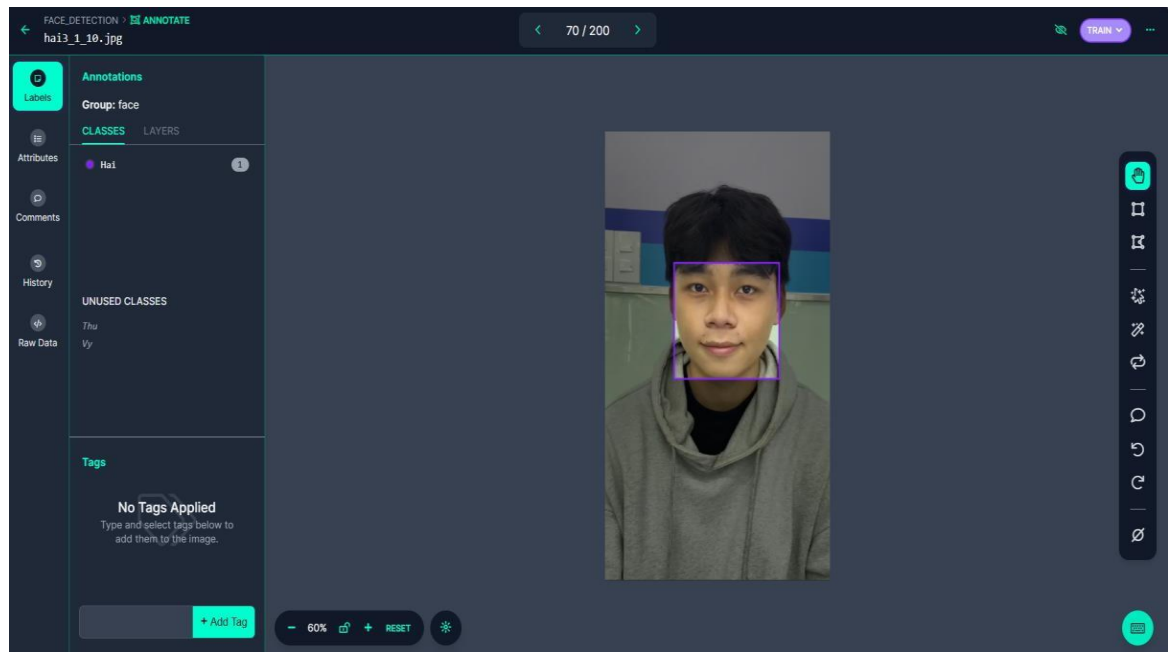


Figure 3.19. Labelling each face of custom dataset

- Creating a Dataset for YOLOv8:

Annotated Dataset: Utilize all 1,898 annotated images for training. Figure 3.20 below shows the results of our dataset using Roboflow. This figure illustrates the performance metrics and visual outcomes of the dataset processed through the Roboflow platform.

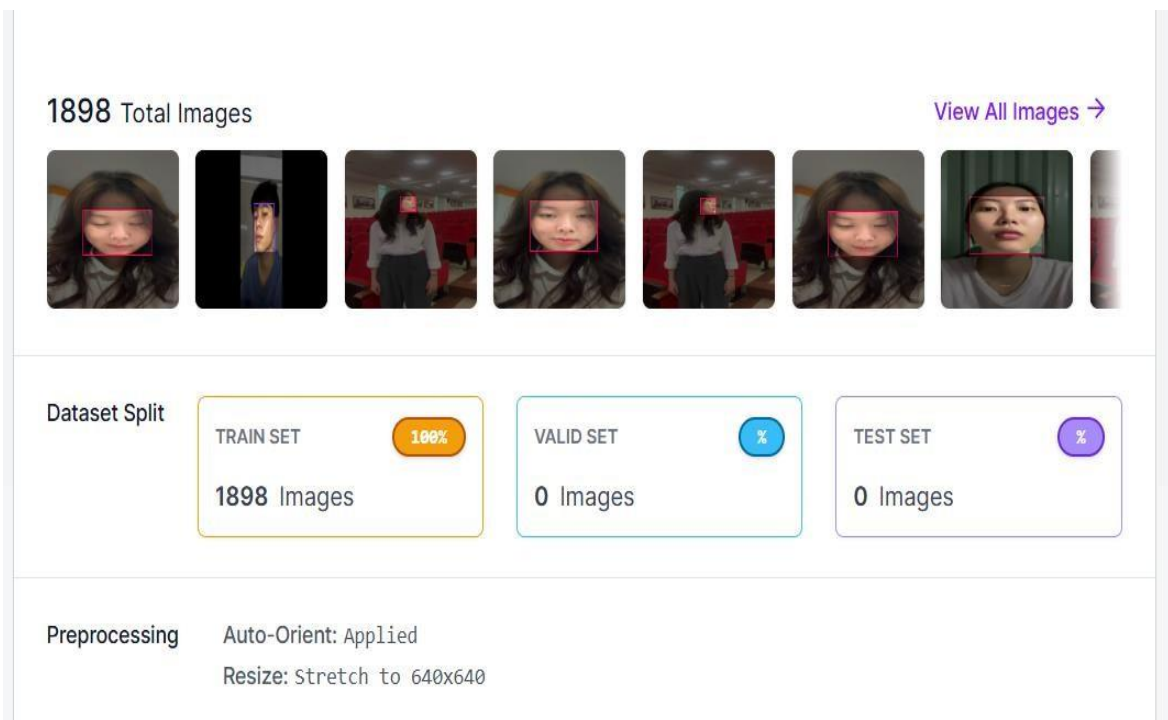


Figure 3.20. The custom dataset after labelling

After creating the labels and splitting the dataset for the training set, export the dataset from Roboflow in a format suitable for YOLO (e.g., YOLOv5 PyTorch, YOLOv4

Darknet, YOLOv8). Here, we choose the format for YOLOv8. Figure 3.21 shows the format that the dataset can export from Roboflow.

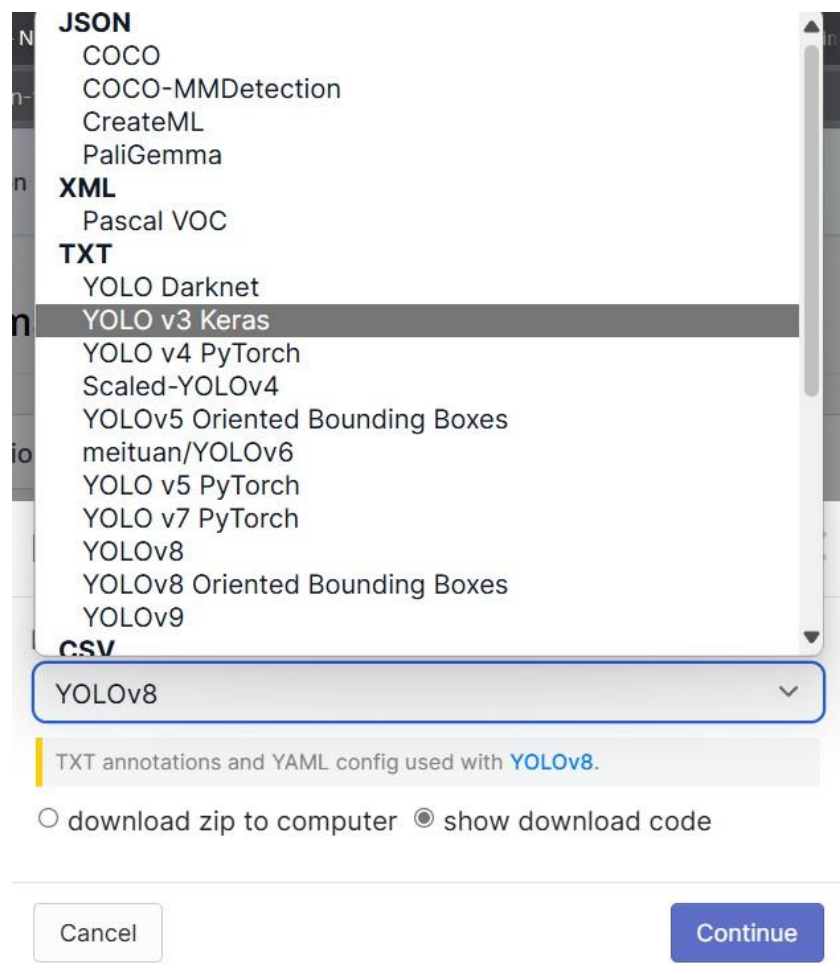


Figure 3.21. Choose type of dataset to export dataset

To sample from an image dataset on Roboflow, follow these steps: Define the sampling purpose, which is for research or experimentation, specifically to randomly sample images named "Hai" and "Thu" to analyze their diversity. Utilize Roboflow's powerful tools for filtering, sorting, and sampling images conveniently and effectively. After selecting the sample of "Hai" and "Thu" images, export them as files for use in your analysis and testing. Ensure that the sampled subset accurately represents the entire original dataset to maintain the accuracy and broad applicability of your analysis results. After sampling image files from the dataset on Roboflow and exporting them as files, store them in personal folder directory on Google Drive.

The dataset should contain image files (.jpg, .png) and corresponding label files (.txt). Below is inside custom dataset folder. Figure 3.22 provides a detailed depiction of the structure of a dataset folder after downloading from Roboflow and store on Google Drive. This structure is crucial for organizing data used in machine learning and computer vision projects.



Figure 3.22. The custom dataset after export from Robotflow

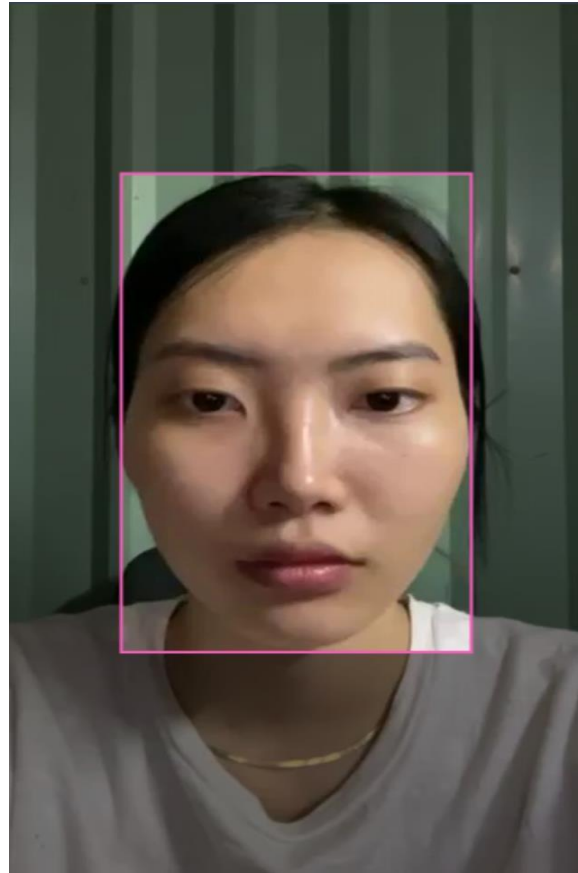


Figure 3.23. The sample image of custom dataset with bounding box

Figure 3.23 is a sample of a .jpg file taken from the 'images' folder in the 'dataset' folder, it shows a face within a bounding box, indicating the detected area of the face. This bounding box is a result of using facial recognition algorithms such as YOLOv8, which identify and isolate the face from the rest of the image for further processing and authentication.

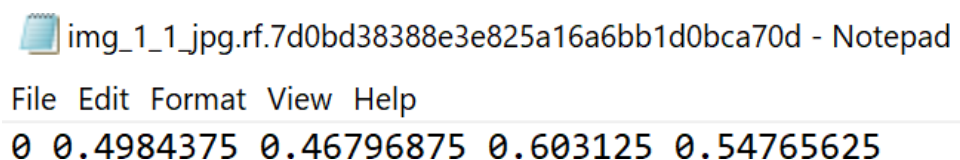


Figure 3.24. Information about example bounding box

Figure 3.24 above is a sample .txt file taken from the 'label' folder in the 'dataset' folder, it containing normalized coordinates for the bounding box around the detected face. The values are likely in the format used for object detection annotations:

- 0: the number of classes
- 0.4984375: x center (following by width of bounding box)
- 0.46796875: y center (following by height of bounding box)
- 0.603125: width of bounding box
- 0.54765625: height of bounding box

#### 3.4.2.1.2 Set Up train YOLO model

For Raspberry Pi, lighter versions like YOLOv8, we choose YOLOv8n is the best choice to ensure performance and processing speed.

**Setting up the environment:**

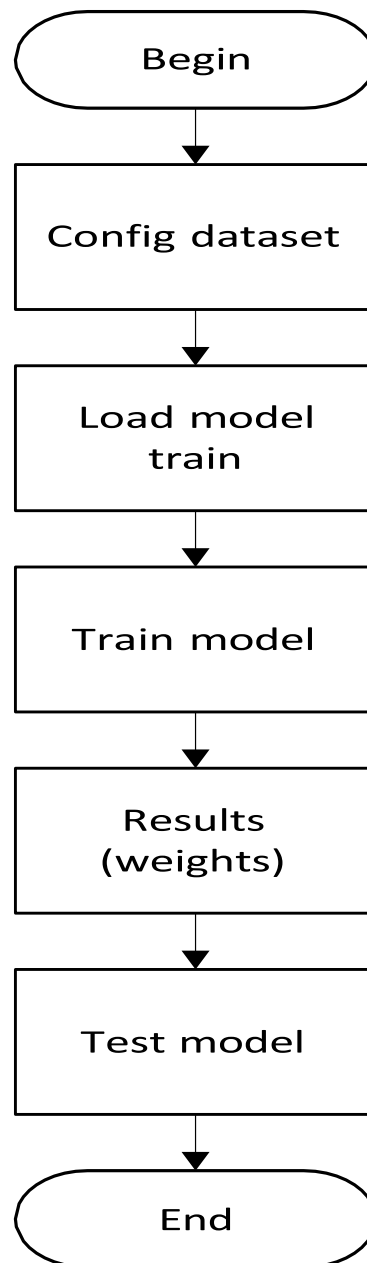


Figure 3.25. The flowchart of YOLO model installation on Raspberry Pi

Figure 3.25 depicts the flowchart for setting up the environment of the YOLO model on Raspberry Pi.



Firstly, we need to configure the dataset, which involves preparing the data, labeling images, and splitting the data into training, validation, and test sets.

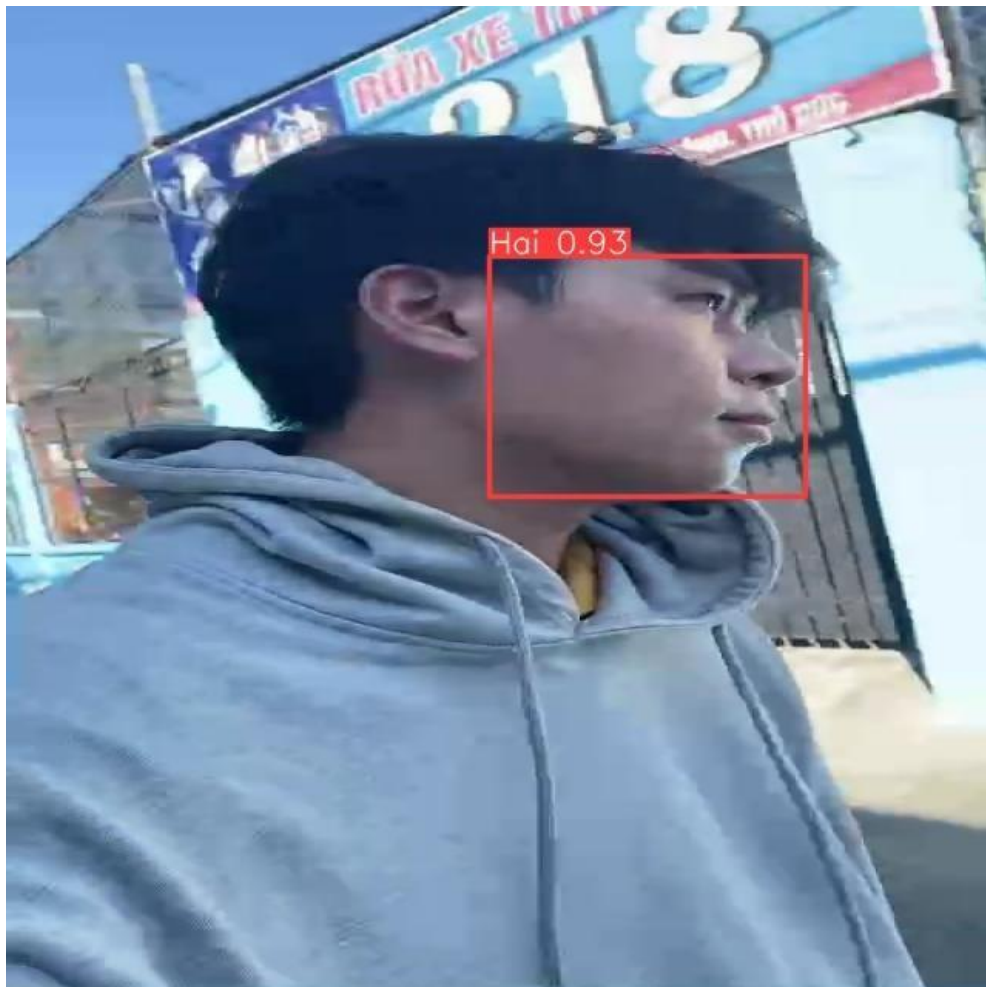
Secondly, load the YOLOv8 model for training, which involves setting up the model architecture and loading any pre-trained weights if necessary.

Thirdly, the model is trained using the configured dataset.

Next, results (weights): After training, we receive two files such as "best.pt" and "last.pt".

Finally, the trained model is tested on a separate test dataset to evaluate its performance. This involves running the model on the test data and measuring its accuracy, precision, recall, and other relevant metrics. The process concludes here, completing the model training and testing cycle.

**Test YOLOv8 model train to evaluate the trained model:** After training, use test images or videos to evaluate the model.



*Figure 3.26. The resulted images after test model*

**Performance Evaluation:** After testing the detection results on the test data, we proceed to evaluate the model's accuracy. Figure 3.26 shows that the model achieves high accuracy in detecting faces with scores above 0.93. This indicates that the YOLOv8 model not only accurately locates the positions of faces in the images but also has the capability to distinguish and recognize labeled faces accurately. This demonstrates the effectiveness and reliability of the model for real-world face detection applications.

**Deploy YOLOv8 model on Raspberry Pi:**

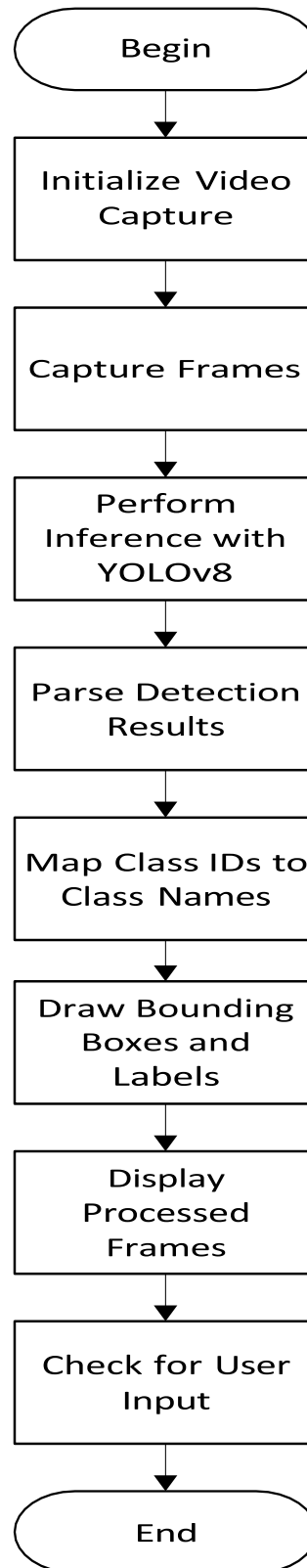


Figure 3.27. The flowchart of deploy YOLOv8 model on Rasp

Figure 3.27 depicts the flowchart for deploying YOLOv8, a deep learning model for object recognition, on a Raspberry Pi. It begins with installing necessary dependencies such as OpenCV. After acquiring the pre-trained YOLOv8 model, it's loaded into memory for real-time object detection from video frames captured by a connected camera. Detected objects are annotated with bounding boxes using OpenCV and displayed on a connected

screen, enabling continuous monitoring until manually stopped. This setup integrates YOLOv8's capabilities into practical applications like automation and surveillance on Raspberry Pi.

### 3.4.2.2 Hand Gestures Recognition

#### 3.4.2.2.1 Configure Parameter of Hand Gestures

The MediaPipe Gesture Recognizer identifies hand motions in real-time and provides results along with landmarks of the detected hands. This capability can be utilized to detect specific hand motions from users and activate corresponding application features. Configure the parameters for hand gestures as shown in Table 3.1 in the user manual.





Hand Gestures	Function
	<b>Action:</b> Turn off the fan. <b>Description:</b> Close your fist and extend your thumb upward.
	<b>Action:</b> Turn on the fan. <b>Description:</b> Close your fist and extend your index finger upward.
	<b>Action:</b> Turn off the lights. <b>Description:</b> Close your fist and extend your ring finger to the side.
	<b>Action:</b> Turn on the lights. <b>Description:</b> Close your fist and extend your pinky finger to the side.

Table 3.1. Hand Gestures For User Manual

#### 3.4.2.2.2 Set Up MediaPipe

Setting up the environment:

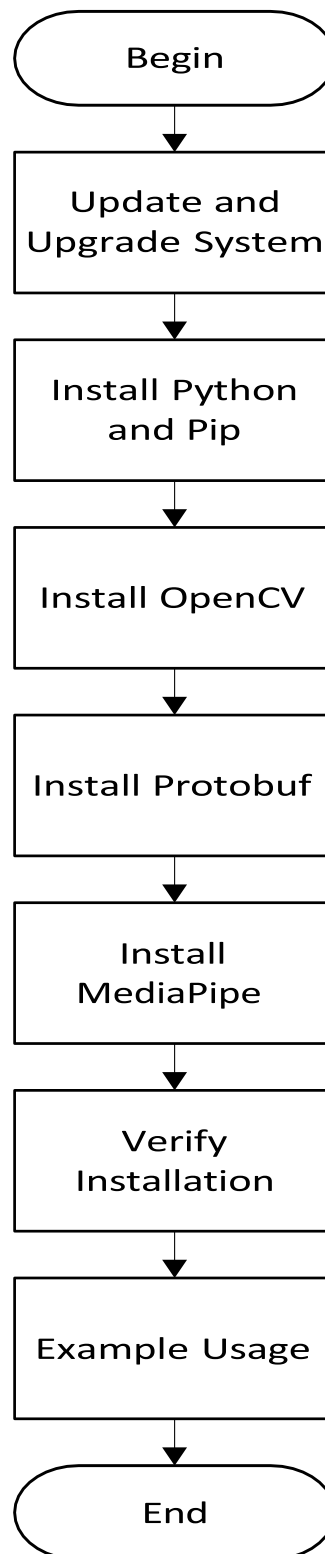


Figure 3.28. The flowchart of set up MediaPipe on Rasp

Figure 3.28 depicts a detailed flowchart outlining the methodical steps to configure MediaPipe on a Raspberry Pi, enabling advanced image processing and computer vision tasks. It begins with updating and upgrading the system to ensure all packages are current, followed by the installation of Python 3 and Pip to manage software dependencies.

OpenCV, a critical library for image processing, is then installed along with necessary dependencies to enable functionalities like camera input handling. Next, Protocol Buffers (protobuf) are installed to facilitate efficient data serialization, essential for MediaPipe's operations. The flowchart proceeds with the installation of MediaPipe itself using pip, ensuring the environment is equipped with the tools needed for leveraging MediaPipe's machine learning models and algorithms. A verification step is included to confirm the successful installation of MediaPipe by running a test script, which serves to validate functionality and readiness for subsequent development tasks. Optionally, an example usage scenario, such as hand detection using MediaPipe, demonstrates practical application of the installed setup. This structured approach ensures a comprehensive setup of MediaPipe on the Raspberry Pi, enabling developers to harness its capabilities effectively for diverse computer vision applications.

**Analyzing the Results:** Based on the detected landmark points, it is possible to identify specific hand gestures, as detailed in Table 3.2. This table categorizes and describes various gestures inferred from the landmark data, providing a comprehensive reference for understanding the interpreted actions and commands. Each gesture corresponds to a unique combination or sequence of landmark positions, facilitating the recognition and classification of user intentions in applications such as sign language recognition, gesture-based control systems, or interactive interfaces.

Offset	Functions
[1,0,0,0,0]	Turn off Fan
[0,1,0,0,0]	Turn on Fan
[0,0,0,1,0]	Turn off LED
[0,0,0,0,1]	Turn on LED

*Table 3.2. The state table of hand gestures recognition*

Based on the state table and the values given, we can define specific hand gestures to control devices such as fans and LED lights. Here's how to represent the status of the fingers and their respective functions: [Thumb, Index, Middle, Ring, Pinky]

1 means the finger is open.

0 means the finger is retracted.

Based on the parameter table:

- [1,0,0,0,0]: Only the Thumb is open then turn off the fan.
- [0,1,0,0,0]: Only the Index finger is open then turn on fan.
- [0,0,0,1,0]: Only the Ring finger is open then turn off LED.
- [0,0,0,0,1]: Only the Little finger (Pinky) is open then turn on LED.

### 3.4.3 Control Block

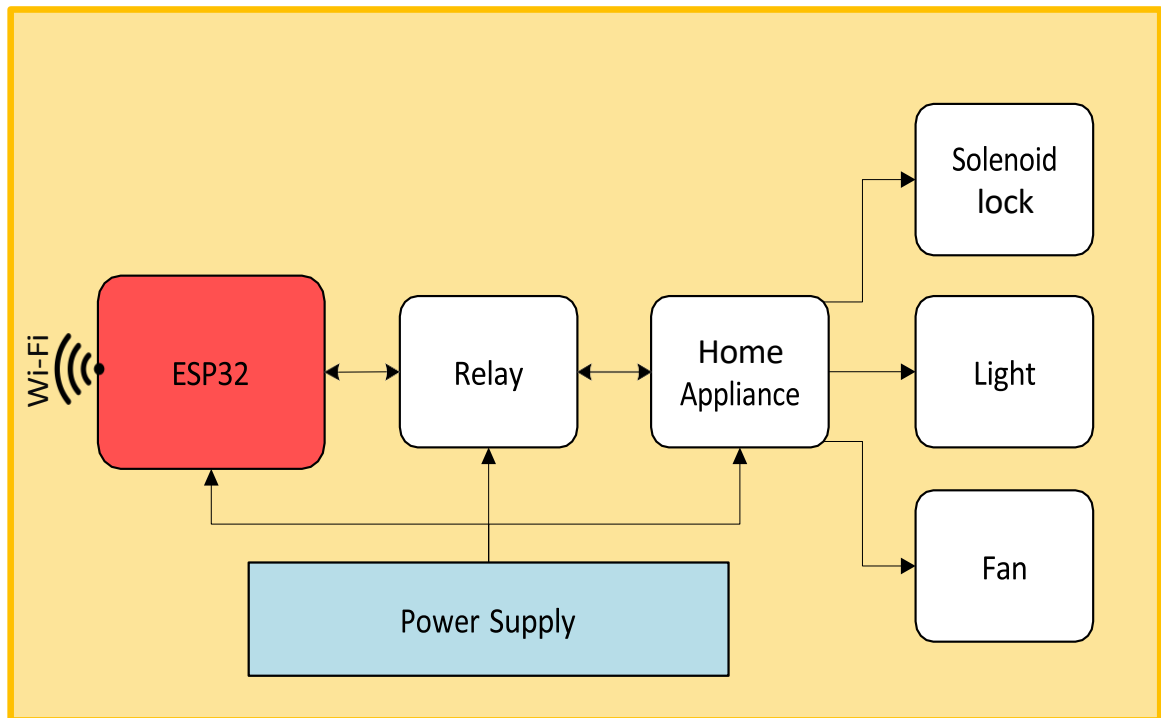


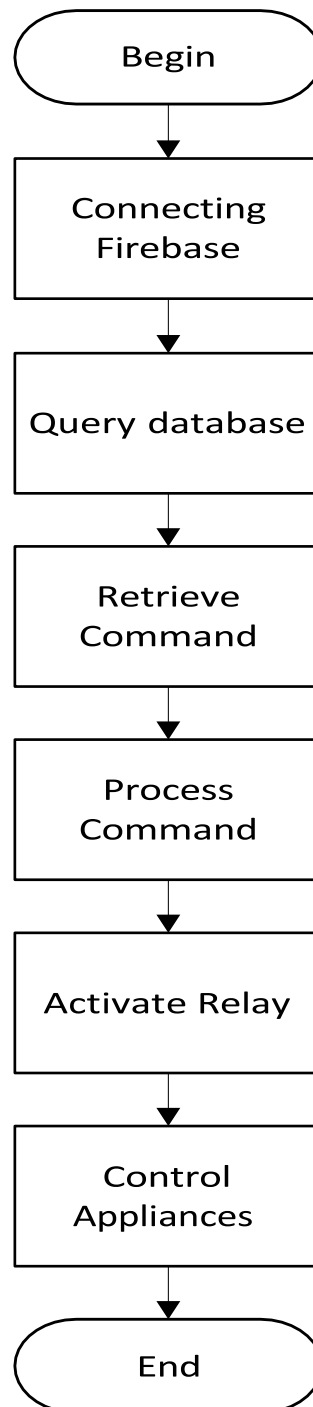
Figure 3.29. Control Block

Figure 3.29 presents a block diagram illustrating the control block within a smart home system. We control three main devices as follows: a solenoid lock, light, and fan. To control these devices, we use a relay to receive input signals from the ESP32. The output of the relay sends signals to the home appliances.

Figure 3.29 above illustrates a smart home automation system utilizing an ESP32 microcontroller to manage various home appliances through a relay. The system comprises several key components, starting with a cloud-hosted database such as Firebase, which stores and retrieves the status and commands for the home appliances. The ESP32 microcontroller connects to this database via the internet, allowing it to receive commands or send data. Upon receiving data from the database, the ESP32 controls the relay module accordingly. The relay, an electromechanical switch, enables the ESP32 to turn home appliances on and off by controlling their power supply. Connected appliances, including a solenoid lock, light, and fan, are managed through this relay mechanism.

The power supply component ensures that the ESP32, relay, and home appliances receive the necessary power to function. The operational workflow begins with the ESP32 communicating with the cloud database over the internet to retrieve commands sent by the user via a mobile or web application. The ESP32 processes these commands and activates the corresponding relay channel to control the power supply to the connected appliances. For example, if the command is to turn on the light, the ESP32 will activate the relay channel linked to the light, allowing current to flow and thereby turning on the light. When the relay is deactivated, it cuts off the current, turning off the appliance.

In a typical scenario, a user wanting to remotely turn on a light sends a command through a mobile app, updating the status in the cloud database. The ESP32 module, which continuously checks the database, receives this updated command and activates the appropriate relay channel, resulting in the light being turned on. This setup enables remote and automated control of home appliances using internet connectivity, making it an efficient and modern smart home automation system.



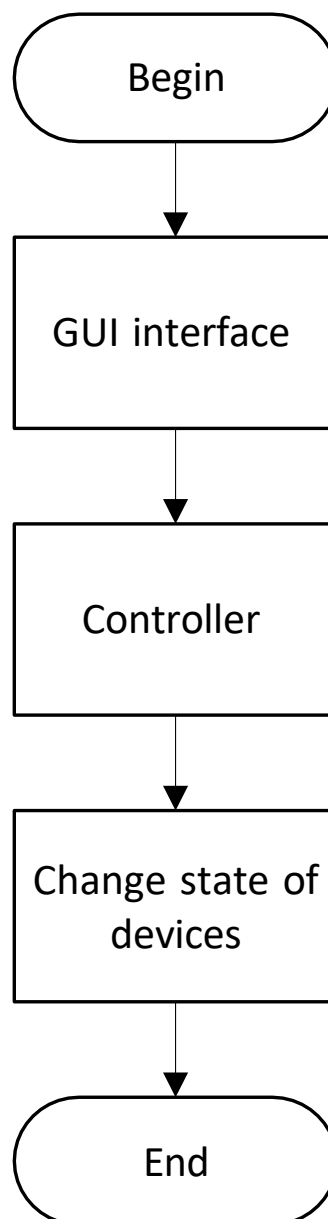
*Figure 3.30. The flowchart of control block*

Figure 3.30 illustrates a flowchart process likely embedded within an IoT or home automation system. The sequence begins with initialization, followed by "Connecting Firebase" to establish a link with the Firebase platform for managing app data. Next, the

system queries the database to retrieve stored commands, then proceeds to the "Retrieve Command" step to identify the specific action required. Once the command is processed, the system activates relays, which are crucial for controlling electrical appliances. In the "Control Appliances" phase, the appliances are managed accordingly, performing tasks such as turning devices on or off. The process concludes, highlighting a structured approach to leveraging Firebase for command-based interaction and appliance control, characteristic of modern smart home and IoT applications.

#### **3.4.4 GUI Interface**

Figure 3.31 is a flowchart that outlines a straightforward process beginning with a user interacting with a GUI. This interaction is processed by a controller, which then alters the state of various devices, ultimately concluding the process.



*Figure 3.31. The flowchart of GUI*



The process begins with the commencement of the sequence. It then moves to a step labeled "GUI Interface," where user interaction through a graphical user interface (GUI) occurs. Following this, the flow proceeds to a "Controller," which is responsible for managing and processing the input received from the GUI. The subsequent step involves "Change State of Devices," where the process alters the state of various devices, such as adjusting settings, activating, or deactivating hardware components. Finally, the sequence signifies the completion of the process.

## **Chapter 4. RESULT**

In this chapter, we detail the implementation of our IoT-based facial recognition and hand gesture control system, following the design from Chapter 3. We present the results from rigorous testing of the integrated YOLOv8 and MediaPipe models and explore hand gesture control for home appliances within an IoT framework. By discussing user experience and system responsiveness, this chapter provides insights into the effectiveness and practicality of our IoT system. The process also starts with user interaction through a GUI, proceeds to a controller that processes input, and results in changes to the state of various IoT-connected devices, concluding the sequence.

### **4.1 IoT Systems**

#### **4.1.1 System Design**

The image below shows a real-life IoT system simulation of a smart home we have completed. Inside the house there is a camera installed that uses facial and hand gestures recognition to control home appliances such as door, light and fan. Figure 4.1 shows the results of the external system design.



*Figure 4.1. Result of external system design.*

Figure 4.2 illustrates the internal system design for a smart home appliance control system, divided into two main sections.



*Figure 4.2. Results of internal system design*

The section framed in red on the left represents the control block for indoor devices, consisting of several key components: a power source that supplies electricity to the entire system, a relay module for switching electrical devices on and off, and an ESP32 microcontroller responsible for processing and communication tasks. Additionally, this section includes a lock solenoid mechanism for controlling locks, a fan for ventilation or cooling, and an LED for visual indicators or lighting.

On the other hand, the blue-framed section on the right is dedicated to the image processing block. It features a camera for capturing real-time images, which are processed by a Raspberry Pi. The Raspberry Pi not only handles image processing but also manages data transmission to other parts of the system or to a cloud database. A separate power source ensures both the camera and Raspberry Pi operate efficiently. Together, these components collaborate to control home appliances through real-time image processing and data management.

#### 4.1.2 Control Home Appliances via Camera

This section presents the results of real-time images captured directly by the camera connected to the Raspberry Pi. After image processing, the data will be transmitted to the cloud database via an internet connection, which is used to control devices within the home system.

##### 4.1.2.1 Control Door

Figure 4.3 shows the return results of face recognition after image processing on the left and the database after receiving data from image processing on the right. The camera mounted directly on the Raspberry Pi will capture facial images. This system detects class\_name "Hai" and "Thu" from the pre-trained dataset. Once the image data has been processed by the image processing block, it will be uploaded to the database. Upon correct detection, it uploads the result to the database for review. When cam\_detection is 1, if the detection is correct, it sets the 'door' variable to 1.

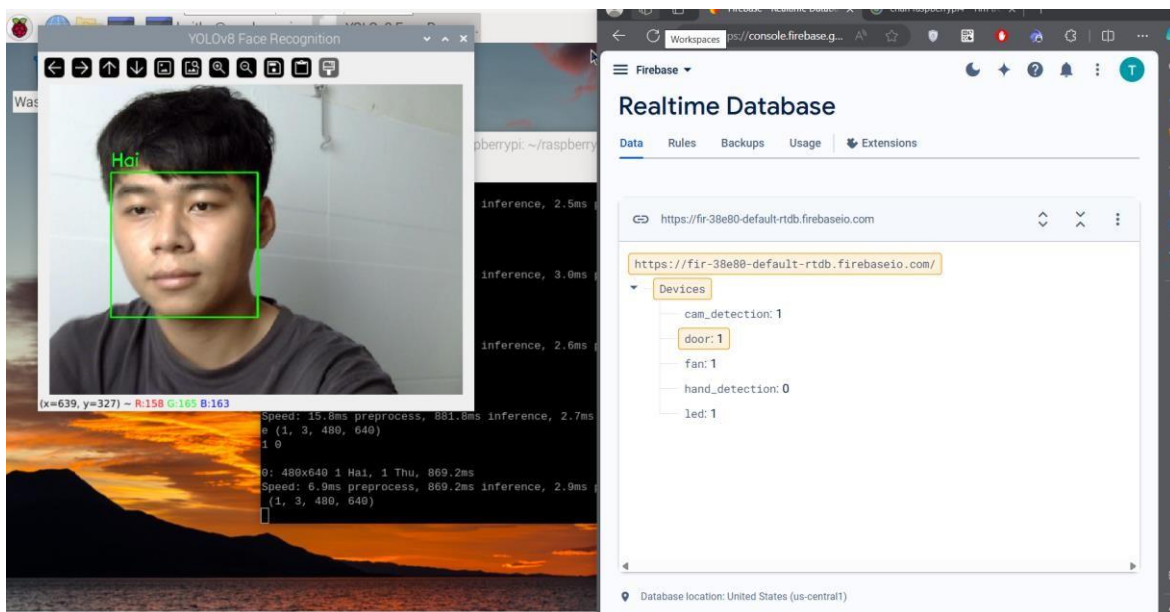


Figure 4.3. Result of facial recognition on Rasp then transmit data to database

After the data is stored in the database, if cam\_detection is 1 and door is 1, it indicates that the door is unlocked. The ESP32 will automatically retrieve this data from the database via Wi-Fi connection to control the relay, thereby closing the solenoid to open the door. Conversely, if the door value is 0, the solenoid lock will open. Figure 4.4 shows the result of control solenoid lock after ESP32 receives data from database.





Figure 4.4. Result of control solenoid lock after ESP32 receives data from database

#### 4.1.2.2 Control Fan

Figure 4.5 shows the return result of the hand gesture with the index finger raised after image processing on the left and the database after receiving data from image processing on the right. The camera mounted directly on the Raspberry Pi will capture hand gestures images. This system is capable of recognizing hand gestures and hand\_detection is 1, specifically when the index finger is raised (value of 1) while the other fingers remain lowered (value of 0). In this scenario, the fan will automatically turn on. Simultaneously, Firebase will receive a signal from the Raspberry Pi, and the corresponding 'fan' variable in Firebase will be updated to 1. This allows the smart system to promptly respond to hand gestures, offering convenience and modern functionality for users.

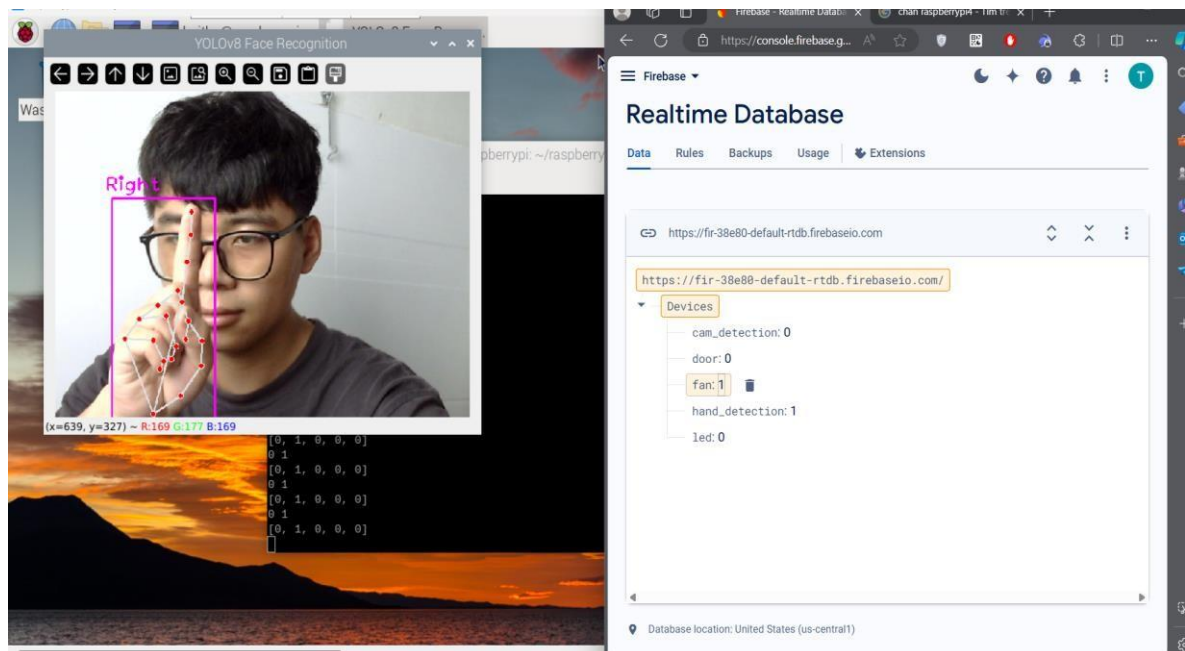


Figure 4.5. Result of index finger recognition on Rasp then transmit data to database

Figure 4.6 shows the return result of the hand gesture with the thumb finger raised after image processing on the left and the database after receiving data from image processing on the right. When the thumb is raised (value of 1) and the other fingers are down (value of 0), the fan will automatically turn off. At the same time, Firebase will receive a signal from the Raspberry Pi and update the 'fan' variable to 0, reflecting that the fan is off.

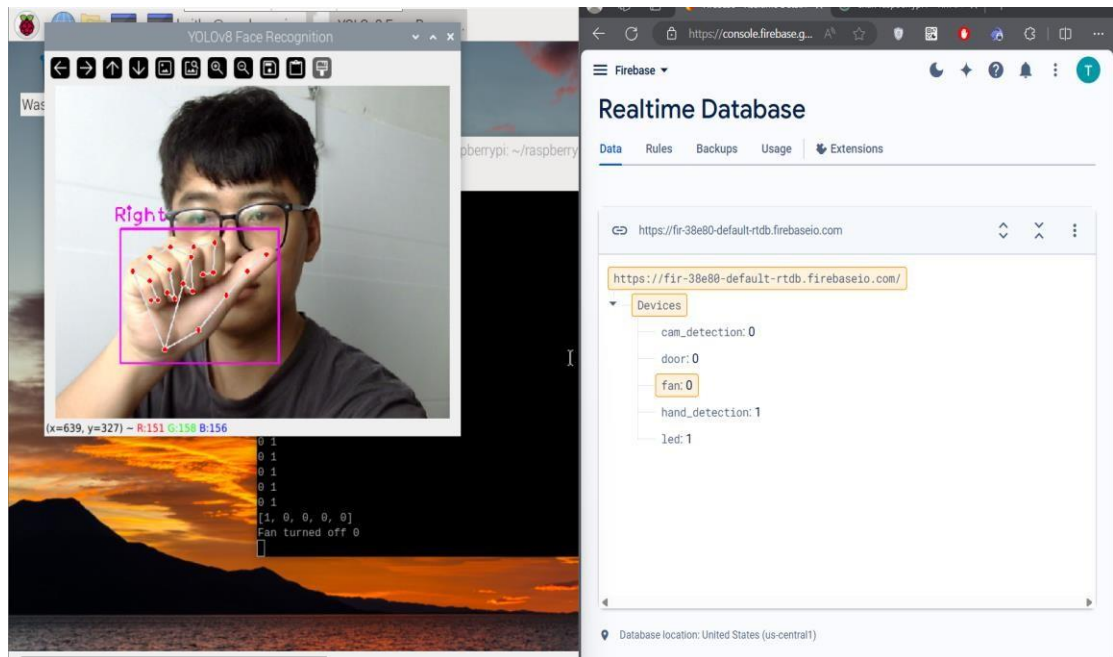


Figure 4.6. Result of thumb finger recognition on Rasp then transmit data to database

After the data is stored in the database, if hand\_detection is 1 and fan is 1 it means the fan is turned on. Then, the ESP32 will receive the signal from the database via Wi-Fi and pulse the fan to 1 to turn it on. Conversely, if the fan value is 0, the fan will turn off. Figure 4.7 shows the result of control fan after ESP32 receives data from database.



Figure 4.7. Result of control fan after ESP32 receives data from database

#### 4.1.2.3 Control Led

Figure 4.8 shows the return result of the hand gesture with the pinky finger raised after image processing on the left and the database after receiving data from image processing on the right. The camera mounted directly on the Raspberry Pi will capture hand gestures images. When the little finger (pinky) is raised (value of 1) and the other fingers are down (value of 0), the LED light will automatically turn on. At the same time, Firebase will receive a signal from the Raspberry Pi and update the 'led' variable to 1, indicating that the LED is on.

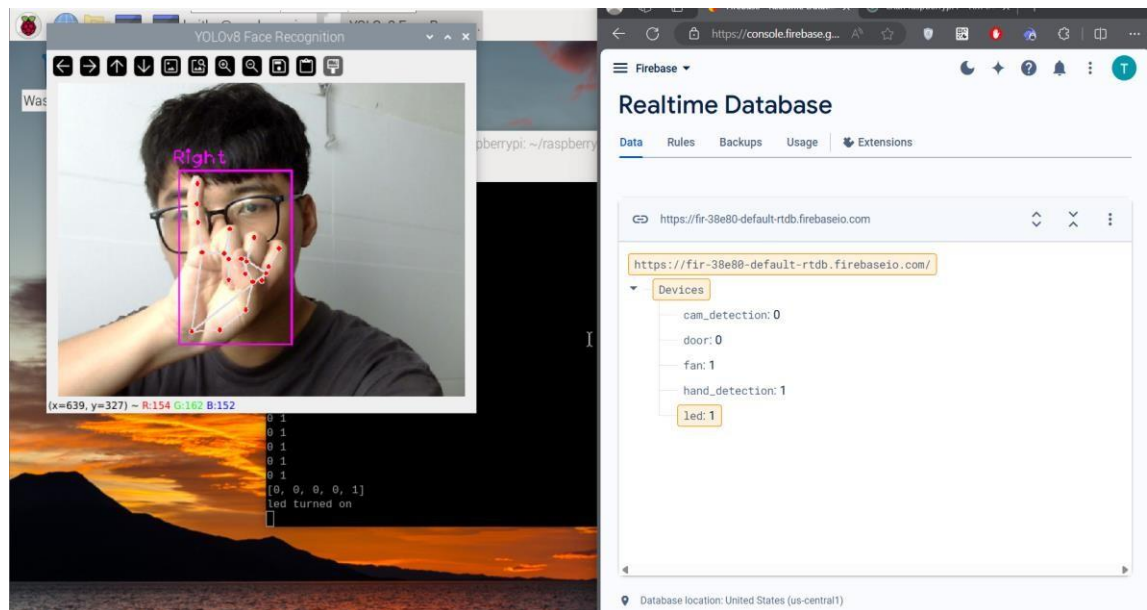


Figure 4.8. Result of pinky finger recognition on Rasp then transmit data to database

Figure 4.9 shows the return result of the hand gesture with the ring finger raised after image processing on the left and the database after receiving data from image processing on the right. When the ring finger is raised (value of 1) and the other fingers are down (value of 0), the LED light will automatically turn off. At the same time, Firebase will receive a signal from the Raspberry Pi and update the 'led' variable to 0, indicating that the LED is off.

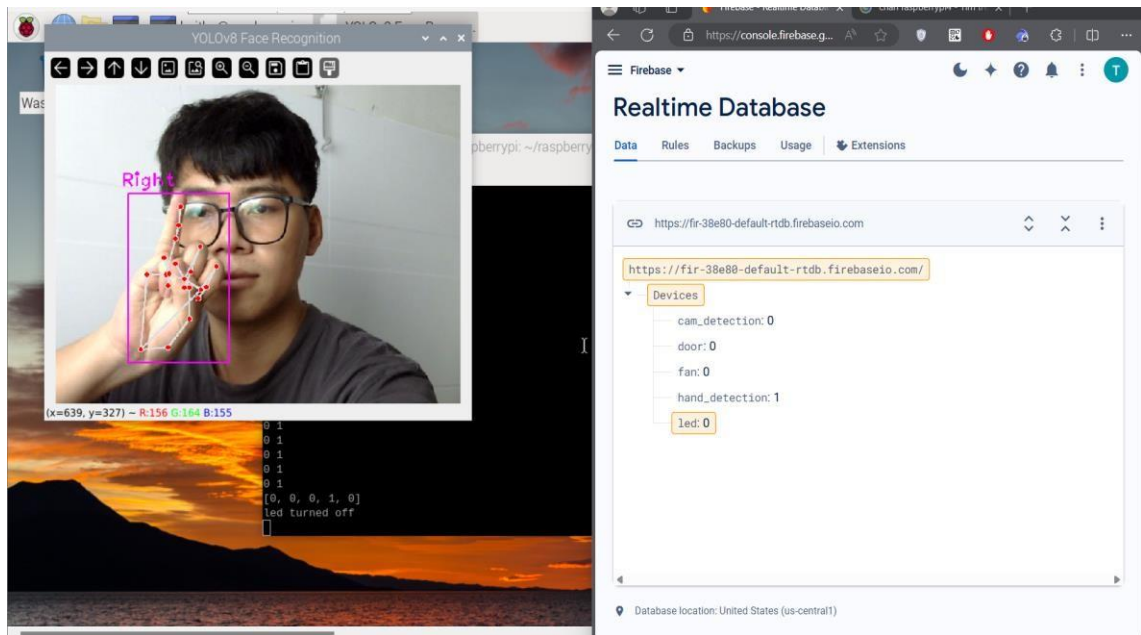


Figure 4.9. Result of ring finger recognition on Rasp then transmit data to database

After the data is stored in the database, if hand\_detection is 1 and led is 1, it indicates that the light is on. The ESP32 will automatically retrieve this data from the database via Wi-Fi connection to control the relay, thereby turning on the light. Conversely, if the led value is 0, the light will turn off. Figure 4.10 shows the result of control light after ESP32 receives data from database.

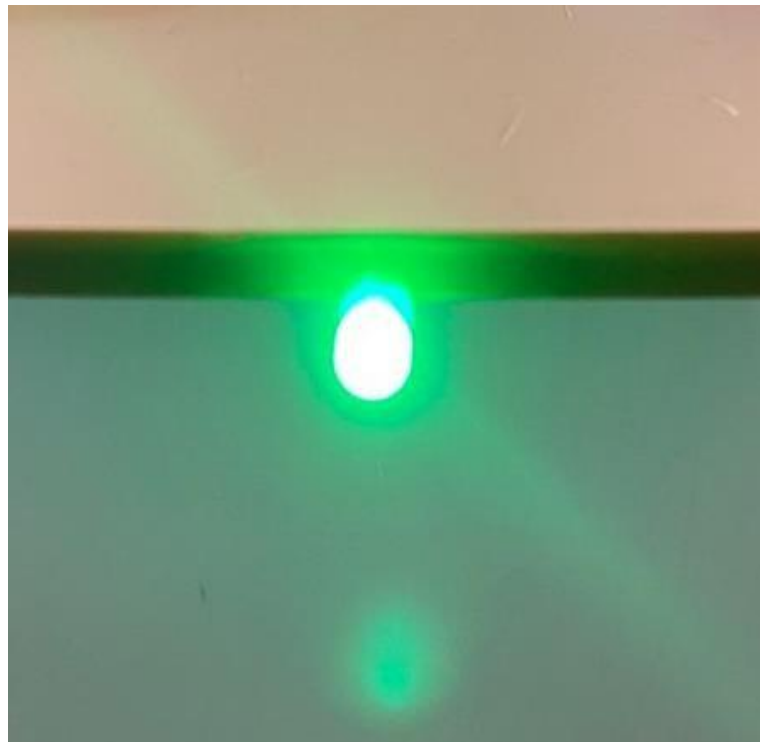


Figure 4.10. Result of control light after ESP32 receives data from database

### 4.1.3 Remote Home Appliances via GUI App

This section showcases the results of remotely controlling home devices via an Internet connection through the GUI app.



## Remote Door

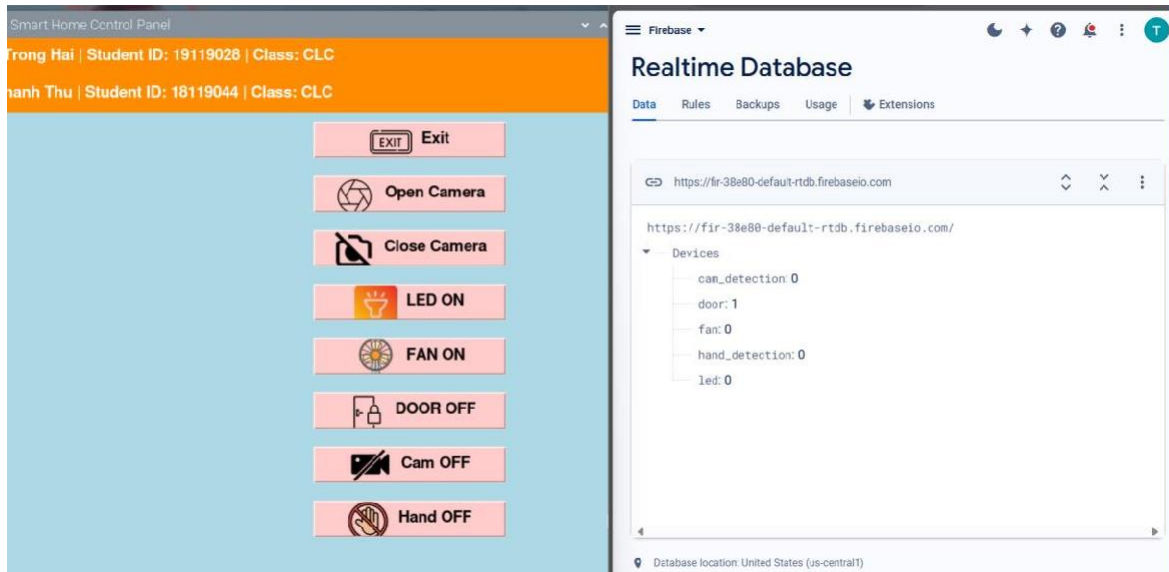


Figure 4.11. Results of remote door via GUI app

Figure 4.11 above is the interface of the GUI app and database when controlling the door. When the user directly presses the "DOOR" button on the GUI interface, a series of actions will occur. First, the signal from the button will be transmitted through the WiFi network to the central database. Here, the system will update the status of the "door" variable to a value of 1. As soon as this status is updated, a command will be sent to the relay to activate the door lock mechanism.

Specifically, this process unfolds as follows:

- The user presses the "DOOR" button on the GUI interface.
- A signal is sent from the GUI through the WiFi network to the server or central database.
- At the server, the system receives the signal and updates the status of the "door" variable to a value of 1
- This update triggers an automated process that sends a command from the server to the relay.
- The relay receives the command and activates the door lock mechanism, allowing the door to open.

## Remote Fan and Led

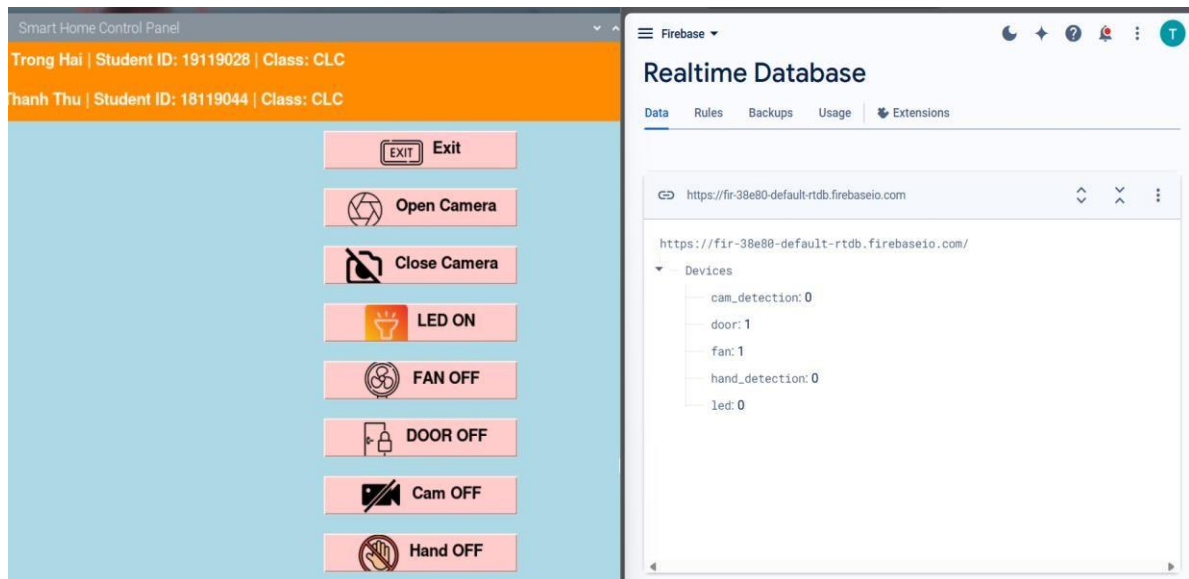


Figure 4.12. Results of remote Fan via GUI app

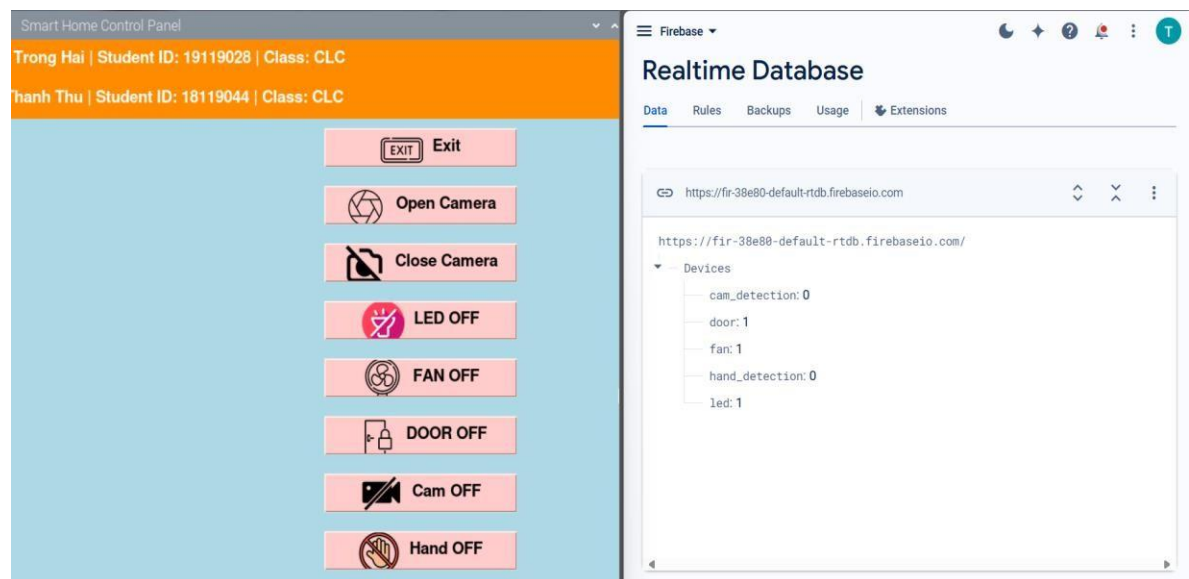


Figure 4.13. Results of remote Led via GUI app

Figure 4.12 and Figure 4.13 are the interfaces of the GUI app and database when controlling Fan and Led. When a user directly presses the "FAN" or "LED" button on the GUI interface, a series of actions occur to control the respective fan or LED light. Here is a detailed description of the operation process for each button:

"FAN" Button:

- The user presses the "FAN" button on the GUI interface.
- A signal is sent from the GUI through the WiFi network to the central server or database.
- At the server, the system receives the signal and updates the status of the "fan" variable to the corresponding value (1 to turn on or 0 to turn off).
- This update triggers an automated process that sends a command from the server to the relay controlling the fan.
- The relay receives the command and activates or deactivates the mechanism to turn the fan on or off as requested.

"LED" Button:

- The user presses the "LED" button on the GUI interface.
- A signal is sent from the GUI through the WiFi network to the central server or database.
- At the server, the system receives the signal and updates the status of the "led" variable to the corresponding value (1 to turn on or 0 to turn off).
- This update triggers an automated process that sends a command from the server to the relay controlling the LED light.
- The relay receives the command and activates or deactivates the mechanism to turn the LED light on or off as requested.

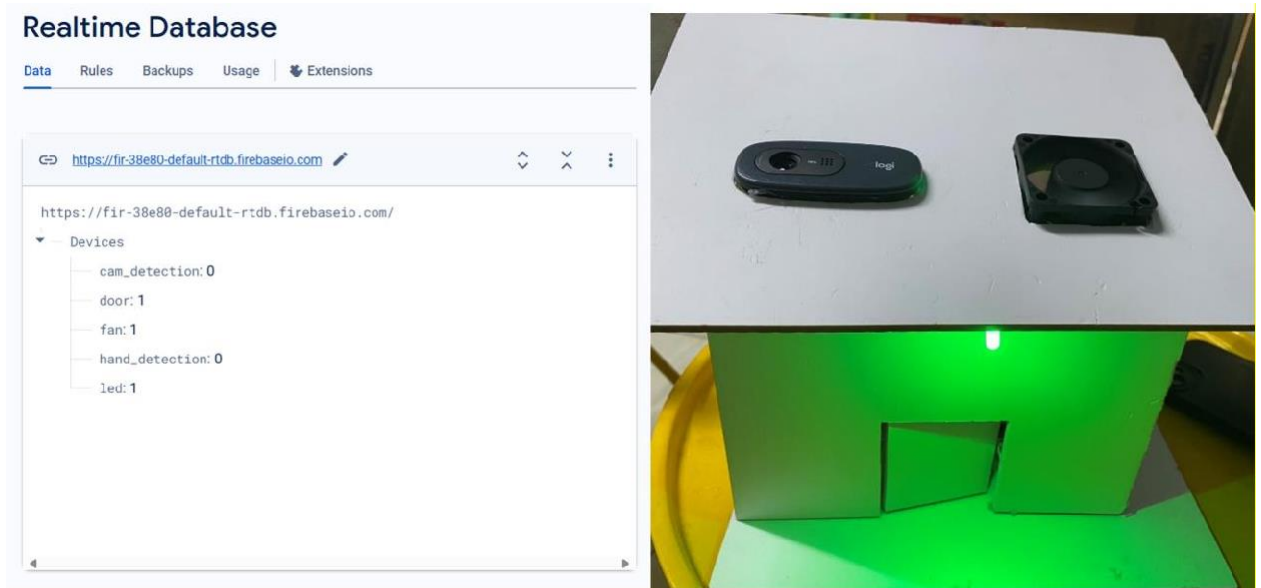


Figure 4.14. Results of controlling home appliances via the GUI app

Figure 4.14 depicts the outcomes when all DOOR, FAN, and LED buttons are engaged. Activating these buttons prompts the IoT home system to activate the corresponding devices.

Specifically, pressing the DOOR button transmits a signal that updates the central database and activates the door lock mechanism, thereby opening the door. Similarly, pressing the FAN button initiates a process that issues a command to activate the fan, while pressing the LED button illuminates the LED light. This integrated response guarantees precise and effective control of each device, showcasing the seamless integration and functionality of the IoT home system.

## 4.2 Evaluation

### 4.2.1 Performance Assessment of IoT System

Real-time evaluation of the system spans from receiving and processing images to responding to results. The lower the average processing time returned, the faster the program's processing speed and responsiveness. Additionally, the effectiveness of the recognition system depends on its accuracy; higher accuracy enhances recognition capabilities, while lower accuracy diminishes them.

In daylight conditions:

<b>Avg. Processing Time (s)</b>	<b>Accuracy (%)</b>
0.81	80.5
0.85	81.8
0.90	84.0
0.95	87.0
1.00	90.0

Table 4.1. Evaluate average processing time in daylight condition

In nightlight conditions:

<b>Avg. Processing Time (s)</b>	<b>Accuracy (%)</b>
0.87	63.0
0.88	64.0
0.93	68.0
0.97	72.0
0.99	75.0

Table 4.2. Evaluate average processing time in nightlight condition

Table 4.1 and Table 4.2 show the system's accuracy and average response time.

We opted for the Wi-Fi communication method, a widely adopted standard in IoT systems due to its broad device compatibility. Simultaneously, we implemented the latest version of an image processing model to achieve improved efficiency and accuracy compared to earlier iterations.

In contrast to projects utilizing Microcontroller Units (MCUs) like Arduino, which are capable of basic image processing but are limited in computational power compared to full computing devices such as the Raspberry Pi, we also explored the ESP32. This robust MCU features integrated Wi-Fi capabilities, facilitating rapid deployment in IoT applications.

Overall, our system design features cost-effective hardware and optimized software, delivering more stable processing speeds compared to previous studies. The software design has been optimized to achieve higher accuracy than the designs of previous studies.

## 4.2.2 Performance of the AI model

Assessing this section enables us to select the optimal model for integration into the system.

### 4.2.2.1 YOLO Model

We chose YOLOv8 because the system requirements required high accuracy and detailed hand tracking. The Table 4.3 below is a comparison table between YOLO models.

	YOLOv5	YOLOv8
<b>Epoch</b>	100	100
<b>Train/Obj Loss</b>	0.0042958	0.18483
<b>Train/Cls Loss</b>	0.00017987	0.82858
<b>Metrics/Precision</b>	0.99539	0.9954
<b>Metrics/mAP_0.5:0.95</b>	0.91488	0.93813

Table 4.3. Comparison table of between YOLOv5 and YOLOv8

The comparison between YOLOv5 and YOLOv8 based on the latest data reveals that YOLOv5 has significantly lower training and validation losses across all metrics, indicating more efficient training and potentially better generalization. Specifically, YOLOv5 shows lower box, object, and classification losses. Despite this, YOLOv8 achieves similar precision and recall scores, with YOLOv8 slightly outperforming YOLOv5 in terms of mAP@0.5:0.95, suggesting better overall detection performance. YOLOv8's higher losses indicate a more complex training process, but its superior mAP@0.5:0.95 highlights its effectiveness in object detection tasks. Consequently, the choice between YOLOv5 and YOLOv8 depends on whether the priority is minimizing training losses or maximizing detection accuracy in practical applications.

#### **Difficulty choosing YOLOv8 model:**

Using YOLOv8 can pose several significant challenges. Firstly, the YOLOv8 model requires powerful hardware for training and deployment due to its large and complex size. Secondly, to achieve optimal performance, YOLOv8 demands a large and diverse training dataset. Additionally, the intricate structure of the model makes understanding and adjusting it difficult, especially for beginners in object detection.

Fine-tuning and optimizing YOLOv8's hyperparameters is a time-consuming and labor-intensive process. Moreover, the model requires high computational resources, particularly during training and deployment. Therefore, deploying YOLOv8 on devices with limited or embedded resources can present numerous challenges.

In summary, to effectively use YOLOv8, users need deep knowledge of the model, computational capabilities, and experience in fine-tuning model parameters to achieve optimal results.

#### **Running YOLOv8 on a Raspberry Pi presents specific challenges:**

**Processing Speed:** Although the Raspberry Pi 4 Model B boasts relatively robust processing power among Single Board Computers, it is not the most powerful in its category. The device's performance heavily relies on factors such as how well the code is optimized, the CPU workload, and how efficiently YOLOv8 utilizes the available RAM.

**Feasibility of Execution:** YOLOv8 requires significant computational resources due to its intricate model and large size. The Raspberry Pi 4 with 8GB RAM offers a more

conductive environment compared to models with less RAM but may still encounter challenges, especially in real-time object detection tasks.

**Conducting Experiments:** It is recommended to optimize the code and leverage efficient libraries to improve YOLOv8's performance on the Raspberry Pi 4. These steps are crucial for mitigating the Raspberry Pi 4's inherent processing limitations and enhancing the practicality of deploying YOLOv8 applications.

In essence, while the Raspberry Pi 4 Model B with 8GB RAM improves YOLOv8's potential compared to earlier versions, achieving optimal performance necessitates careful consideration and optimization of the device's computational capabilities.

#### 4.2.2.2 MediaPipe

We chose MediaPipe because the system requirements required high accuracy and detailed hand tracking. The Table 4.4 below is a comparison table between Mediapipe and other hand tracking libraries.

Feature	MediaPipe	Handtrack.js
<b>Precision</b>	High precision, sub-millimeter accuracy	Moderate precision, sufficient for basic tasks
<b>Accuracy</b>	High accuracy, detailed landmark detection (21 points per hand)	Moderate accuracy, general hand presence detection
<b>mAP@0.99</b>	High mAP@0.99, strong overlap with ground-truth	Lower mAP@0.99, simpler model architecture
<b>Real-time Performance</b>	Optimized for real-time applications	Optimized for real-time in-browser performance
<b>Ease of Use</b>	More complex setup, but highly customizable	Easy to set up and use with simple API
<b>Landmark Detection</b>	Yes (21 points per hand)	No detailed landmark detection, basic bounding box
<b>Customization</b>	Highly customizable with additional models	Limited customization, primarily pre-trained models

Table 4.4. Comparison table of between MediaPipe and Handtrack.js

Summary:

- **Precision & Accuracy:** MediaPipe excels in precision and accuracy, making it suitable for detailed and advanced applications. Handtrack.js offers moderate precision and accuracy, suitable for simpler tasks.
- **mAP@0.99:** MediaPipe is designed to achieve high mAP@0.99 scores, indicating robust performance even at stringent IoU thresholds. Handtrack.js has lower mAP@0.99 due to its simpler approach.
- **Platform Support:** MediaPipe is cross-platform, supporting a wide range of environments. Handtrack.js is primarily designed for web applications.

- **Real-time Performance:** Both libraries are optimized for real-time performance, but MediaPipe's detailed tracking can handle more complex scenarios.
- **Ease of Use:** Handtrack.js is easier to set up and use, ideal for quick implementation. MediaPipe requires more setup but offers extensive customization and capabilities.

## **Chapter 5. CONCLUSIONS AND FUTURE WORKS**

In this final chapter, we present conclusions based on the outcomes of our study. We summarize the key insights derived from the system's implementation and evaluation, emphasizing its strengths and limitations. Furthermore, we discuss potential future research directions and enhancements to improve the system's performance and usability. By offering a comprehensive conclusion and identifying paths for future work, this chapter aims to advance the field of biometric authentication and gesture-based interaction in smart environments.

### **5.1 CONCLUSIONS**

This project demonstrates the powerful integration of IoT technologies to create an advanced home automation system. By leveraging IoT principles, the system effectively utilized interconnected devices and sensors to enhance user interaction and control over household appliances. Key aspects highlighted in this project include:

**Real-Time Data Transfer:** Wi-Fi facilitated real-time data transfer between the Raspberry Pi 4, ESP32 microcontrollers, and Firebase cloud services, ensuring prompt updates on device status and operations. This was crucial for maintaining the system's responsiveness and reliability.

**Scalability and Adaptability:** The project's architecture and the use of cloud-based solutions like Firebase showcased the scalability of IoT. It laid a foundation for future expansion and integration of additional smart devices or functionalities, adapting to evolving user needs and technological advancements. Wi-Fi's ubiquity and compatibility with various devices and platforms supported this scalability.

**Challenges and Considerations:** While Wi-Fi offered benefits such as high-speed data transfer and widespread availability, it also posed challenges related to network security, reliability in crowded environments, and potential connectivity issues. Addressing these challenges required robust security measures and optimized network configurations to ensure stable operation.

**Hardware Control with ESP32 and Relays:** The ESP32 microcontroller interfaced with relay modules connected to various devices such as the door lock, LED lights, and fan. The GPIO pins of the ESP32 were used to control the relays, enabling seamless switching of connected devices.

**Data Handling and Display with Firebase:** Firebase Realtime Database was used to store and retrieve data regarding the status of connected devices. A GUI interface was developed to display the status of devices and enable remote control, providing users with real-time updates and control capabilities.

**System Integration and Testing:** All components (YOLOv8 for face recognition, MediaPipe for gesture recognition, ESP32 for device control, and Firebase for data



management) were successfully integrated to create a cohesive and functional system. Extensive testing ensured the system's reliability and responsiveness in real-world scenarios.

**Face Recognition with YOLOv8:** A face recognition system using the YOLOv8 model was implemented. The model efficiently detects and recognizes faces, providing secure access control for door opening. The system demonstrated high accuracy and response times, making it suitable for real-time security applications.

**Hand Gesture Recognition with MediaPipe:** MediaPipe was integrated for real-time hand gesture recognition, enabling the system to recognize specific gestures for controlling the light and fan. This provided an intuitive, contactless method for interacting with home automation devices, enhancing user convenience and safety.

**Overall System Performance:** The project successfully integrated advanced technologies to create a sophisticated home automation system. By employing YOLOv8 for facial recognition and MediaPipe for hand gesture recognition on the Raspberry Pi 4, the system achieved remarkable precision in identifying users and their actions. The incorporation of ESP32 facilitated hardware control, allowing for the manipulation of various home appliances and devices based on recognized facial and hand gestures. Real-time data handling capabilities provided by Firebase ensured swift communication and synchronization between different components, enhancing the system's overall responsiveness and reliability.

**Practical Application:** Users could unlock doors or activate appliances simply by presenting their faces or making predefined hand gestures, offering a seamless and intuitive home automation experience.

**Future Directions:** Moving forward, the project aims to optimize system performance, refine algorithms for more accurate recognition, and explore additional features to enhance user convenience and security. This may involve implementing machine learning techniques for continuous improvement and expanding compatibility with a broader range of devices and platforms, ultimately aiming for a fully integrated and adaptable smart home ecosystem.

## **5.2 FUTURE WORKS**

While the project achieved its primary goals, several areas for improvement and expansion have been identified:

### **Optimization of Recognition Algorithms:**

Enhance the YOLOv8 and MediaPipe models to increase accuracy and reduce latency, particularly in low-light or complex environments.

Investigate alternative machine learning models and techniques to boost the robustness of face and gesture recognition.

### **Enhanced Security Measures:**

Add extra security layers, such as multi-factor authentication that combines face recognition with PIN codes or mobile authentication.

Encrypt communication between the Raspberry Pi 4, ESP32, and Firebase to safeguard user data and device control commands from potential security threats.

### **Scalability and Flexibility:**

Design a modular system architecture to facilitate the addition of new devices and control features.

Extend the system to recognize a wider range of gestures, offering users more diverse control options.

### **User Interface Improvements:**

Improve the GUI with more interactive features and a better user experience design.

Develop companion mobile applications to give users additional control options and real-time notifications.

### **Advanced Data Analytics:**

Implement data analytics to monitor usage patterns and enhance system performance based on user behavior.

Apply machine learning to predict user actions and automate device control based on learned preferences, enhancing the user experience.

### **Integration with Other IoT Platforms:**

Explore integration with other IoT platforms and services to expand the system's functionality and interoperability.

Ensure compatibility with popular smart home ecosystems to deliver a more cohesive smart home experience.

## **REFERENCES**

- [1] Panup, Wanida & Ratipapongton, Wachirapong & Wangkeeree, Rabian. (2022). A Novel Twin Support Vector Machine with Generalized Pinball Loss Function for Pattern Classification. *Symmetry*. 14. 289. 10.3390/sym14020289.
- [2] Zhang, Youshan. (2023). Stall Number Detection of Cow Teats Key Frames.
- [3] Zhang, Xinliang & Wang, Wanru & Zhao, Yunji & Xie, Heng. (2020). An improved YOLOv3 model based on skipping connections and spatial pyramid pooling. *Systems Science & Control Engineering*. 9. 1-8. 10.1080/21642583.2020.1824132.
- [4] Altayeb, Muneera. (2023). Hand Gestures Replicating Robot Arm based on MediaPipe. *Indonesian Journal of Electrical Engineering and Informatics (IJEI)*. 11. 727-737. 10.52549/ijeei.v11i3.4491.
- [5] Hanna, K. T., & Rosencrance, L. (2023, May 19). Google Firebase. Mobile Computing. [Online]. Available: <https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase>
- [6] Logitech HD Webcam C270 Technical Specifications. (n.d.). Logitech Support + Download. [Online]. Available: <https://support.logi.com/hc/en-my/articles/360023462093-Logitech-HD-Webcam-C270-Technical-Specifications>
- [7] “Raspberry Pi 4 Model B Datasheet,” June 2019. [Online]. Available: <https://docs.rs-online.com/0e7b/A700000006696790.pdf>
- [8] “ESP32 WROOM 32 Datasheet,” V3.4, (2023 Feb 13). [Online]. Available: [https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf)
- [9] 5VDC Relay Module 4 Channel – H/L Trigger | Ampere Electronics . (2024, June 5). Ampere Electronics. [Online]. Available: <https://ampere-electronics.com/product/5vdc-relay-module-4-channel-h-l-trigger/>
- [10] Sunon 40mm 5VDC Maglev Bearing Cooling Fan - Altronics. (n.d.). Altronics. [Online]. Available: <https://www.altronics.com.au/p/f1110-sunon-40mm-5vdc-maglev-bearing-fan/>
- [11] Green 5mm LED Specs, Data Sheet 500TSG4DA from lc-led.com. (n.d.). [Online]. Available: <https://www.lc-led.com/products/500tsg4da.html>
- [12] uxcell DC 5V 1A 30g 3mm Mini Electromagnetic Solenoid Lock Pull Type for Electric Lock Cabinet Door Lock. Amazon. [Online]. Available: <https://www.amazon.com/uxcell-Electromagnetic-Solenoid-Electirc-Cabinet/dp/B07TKS6L7C>.
- [13] YOLOv8 Object Detection Model: What is, How to Use. (n.d.). [Online]. Available: <https://roboflow.com/model/yolov8>
- [14] Sapkota, Ranjan & Ahmed, Dawood & Karkee, Manoj. (2023). Comparing YOLOv8 and Mask RCNN for object segmentation in complex orchard environments. 10.32388/ZB9SB0.
- [15] Ganesan, K., Narmadha, A. S., & Nandhini, J. (2022). Hand Gesture Recognition for Home Automation. *International Journal of Advance Research*

- in Computer Science and Management, 2, 312-321.
- [16] Shinde, P., Bhosale, A., Kulkarni, A., & Deshmukh, A. (2019). IoT-based home automation and security system using Raspberry Pi. *International Journal of Engineering and Advanced Technology*, 8(5), 5784-5788.
  - [17] Deshmukh, S., Singhal, S., & Pandey, R. (2021). Development of Smart Home Automation and Security System using Internet of Things (IoT). *International Journal of Recent Technology and Engineering*, 9(1), 5070-5076.
  - [18] Hasan, M., Akter, F., & Islam, M. (2022). An IoT-Based Home Automation System with Hand Gesture Recognition. *International Journal of Advanced Computer Science and Applications*, 13(1), 291-296.
  - [19] Berte, Dan-Radu. (2018). Defining the IoT. *Proceedings of the International Conference on Business Excellence*. 12. 118-128. 10.2478/picbe-2018-0013.
  - [20] Guo, Yinan & Zhang, Shuguo & Xiao, Dawei. (2012). Overview of Wi-Fi Technology. 10.2991/iccasm.2012.330.
  - [21] Politiek, R. (2024, January 29). Install OpenCV on Raspberry Pi. Q-engineering. [Online]. Available: <https://qengineering.eu/install-opencv-on-raspberry-pi.html>
  - [22] Yang, Cheng-Ying, Yi-Nan Lin, Sheng-Kuan Wang, Victor R.L. Shen, Yi-Chih Tung, Frank H.C. Shen, and Chun-Hsiang Huang. 2023. "Smart Control of Home Appliances Using Hand Gesture Recognition in an IoT-Enabled System." *Applied Artificial Intelligence* 37 (1). doi:10.1080/08839514.2023.2176607.
  - [23] Islam, Md Tarekul & Azad, Md & Ahammed, Md & Rahman, Wahidur & Azad, Dr & Nasir, Mostofa. (2022). IoT Enabled Virtual Home Assistant Using Raspberry Pi. 10.1007/978-981-19-2281-7\_52.
  - [24] Design and implementation of smart home system: Graduation project of Computer engineering technology/ Le Ba Minh Dat, Nguyen Van Quang Huy; Phan Van Ca (Advisor)--Ho Chi Minh City: Ho Chi Minh City University of Technology and Education, 2024 Call no.: KMT-19 629.89 L433-D232
  - [25] Designing the smart locking door by using image processing: Faculty of high quality graduation's thesis of the Automation and Control Engineering/ Pham Tuan Quang Huy, Nguyen Ngoc Bao; Tran Vu Hoang (Supervisor). -- Ho Chi Minh city: HCM university of Technology and Education, 2022 72p.; 30cm
  - [26] Ju, Rui-Yang & Cai, Weiming. (2023). Fracture detection in pediatric wrist trauma X-ray images using YOLOv8 algorithm. *Scientific Reports*. 13. 10.1038/s41598-023-47460-7.