REAL TIME SYSTEM AND INTERNET OF THINGS FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA

ShieldWatchIOT : Your Smart Home Guardian

GROUP A1

Rafie Amandio Fauzan	**2106731232**
Rizal Ab’daan	**2006577441**
Zefanya Christira Deardo	**2106637214**
Muhammad Fathan Muhandis	**2106731623**

PREFACE

The value of clever home automation and security systems is immeasurable in our current lives, where technology permeates every aspect of our everyday lives. In this context, we are excited to present "ShieldWatch: Smart Home Guardian". This project represents a significant advancement in the search for a complete and intuitive system that not only protects our houses but also enhances the idea of a smart networked living environment.

ShieldWatch was created in response to the difficulties that conventional automation and security systems encountered. Widespread adoption is frequently hampered by the complexity, expense, and lack of integration, leaving houses vulnerable and automation experiences fragmented. Having a thorough awareness of these drawbacks, the initiative aims to completely change the game by skillfully fusing cutting-edge technology with usefulness.

ShieldWatch is primarily intended to tackle two essential facets of modern life: automation and security. The system guards against possible intruders with PIR sensors, sending out alerts in real time and capturing images. In addition, it adds a new level of functionality to home monitoring by enabling users to monitor things like room temperature and automatic lamp based on current light intensity.

Depok, December 10, 2023

Group A1

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

The ShieldWatch project is initiated to address several challenges in home security by integrating Internet of Things (IoT) technology. The main challenges faced by households today include the increasing risk of crimes such as burglary and theft. Therefore, this project aims to provide a more efficient and intelligent security solution through the utilization of cutting-edge technology. In designing a smart home security solution, we identified the need to detect intruders early on. To achieve this, the project integrates PIR Sensors, effective in detecting suspicious movements around the house. Additionally, the presence of the DHT11 sensor provides the capability to monitor the temperature inside the home, while the LDR sensor is used to regulate lighting based on natural light intensity.

One of the key innovations of this project is the use of ESP32 as the system's brain, capable of activating the camera and transmitting related photos through the MQTT protocol with base64 encoding. This allows homeowners to receive real-time visual monitoring, providing an additional layer of security. The project also considers the need to identify intruders more accurately through the use of facial recognition technology. By leveraging the face recognition library on the backend, the system can provide more reliable identification data. Furthermore, the project addresses the need to provide quick notifications to users when intruders are detected. Through email notification services, homeowners can receive information directly, enabling faster and timely responses.

1.2 PROPOSED SOLUTION

The proposed solution for the ShieldWatch project involves the integration of various components to create a comprehensive and intelligent home security system. To address the challenge of detecting intruders, PIR sensors are strategically placed to capture and analyze movements around the house, providing an early warning system. Temperature monitoring inside the home is facilitated by the DHT11 sensor, offering homeowners insights into the environmental conditions. Additionally, the LDR sensor contributes to energy efficiency by adjusting lighting based on the surrounding natural light intensity.

The ESP32 microcontroller serves as the central processing unit, orchestrating the activation of the camera and transmitting images through the MQTT protocol with base64 encoding. This real-time visual data allows homeowners to remotely monitor their homes, adding a valuable layer of security. Facial recognition technology is incorporated into the backend to enhance the identification of intruders. This feature aims to provide more accurate and reliable data, improving the overall effectiveness of the security system.

To ensure homeowners receive timely information, the system utilizes email notification services, alerting them when an intrusion is detected. This quick response mechanism enables homeowners to take immediate action, contributing to an enhanced sense of security. By combining these innovative technologies, the ShieldWatch project offers an intelligent and proactive home security solution. The integration of IoT components and advanced features empowers homeowners with real-time information, contributing to a safer and more secure living environment.

1.3 ACCEPTANCE CRITERIA

The acceptance criteria of this project are as follows:

1. PIR sensors promptly detect and report intrusions, triggering immediate alerts.
2. DHT11 and LDR sensors ensure accurate temperature readings and efficient lighting control.
3. ESP32 activates the camera and transmits images securely via MQTT with base64 encoding.
4. Backend system achieves high-precision facial recognition, continuously improving through updates.
5. Frontend displays real-time data from MQTT topics for homeowners' up-to-date security status.
6. System sends timely and clear email notifications with intrusion details and images.

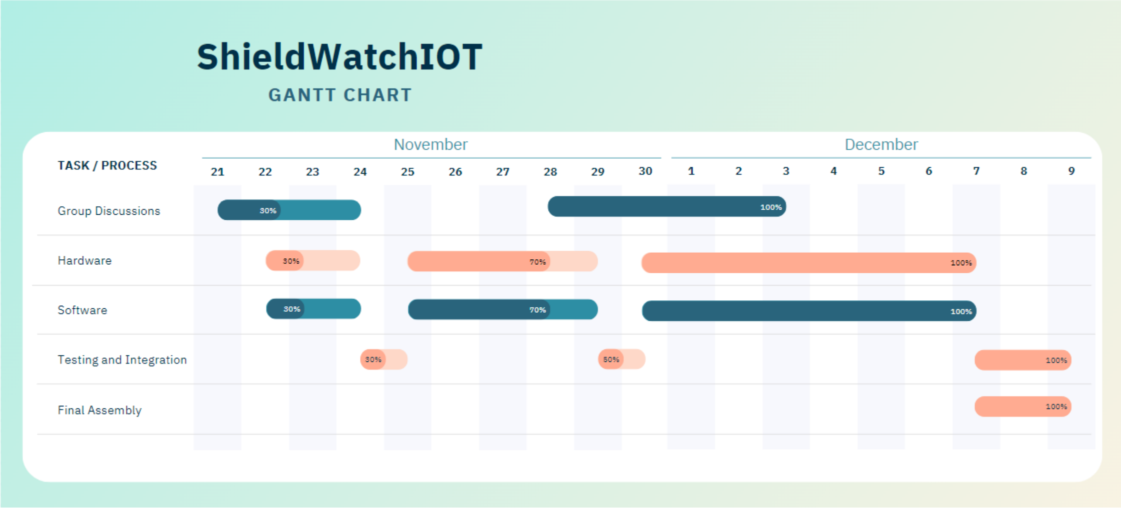
1.4 ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
Frontend Developer	Design and implement the user interface and frontend of the ShieldWatch control website.	Muhammad Fathan Muhandis
Backend Developer	Develop the backend of the website used for ShieldWatch device control.	Rafie Amandio Fauzan
IoT Monitoring Developer	Develop the code and circuitry for the ShieldWatch device's monitoring functionalities	Rizal Ab'daan
IoT Camera Developer	Develop the code and circuitry for the ShieldWatch device's camera functionalities.	Zefanya Christira Deardo

Table 1. Roles and Responsibilities

1.5 TIMELINE AND MILESTONES

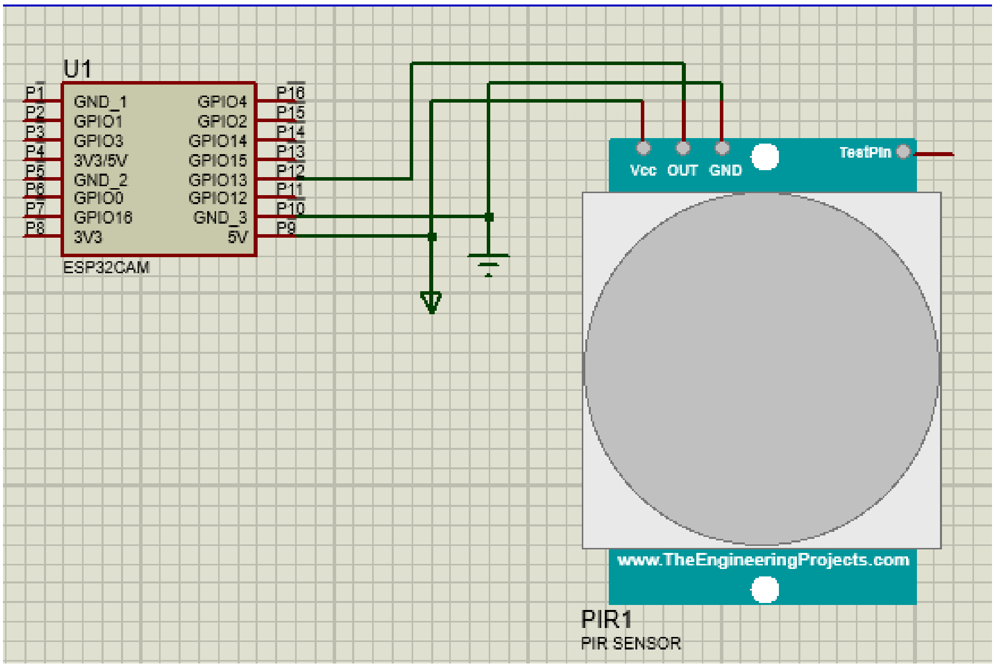

ShieldWatchIOT
GANTT CHART
TASK / PROCESS
November
December
21
22
23
24
25
26
27
28
29
30
1
2
3
4
5
6
7
8
9
Group Discussions
30%
100%
Hardware
30%
70%
100%
Software
30%
70%
100%
Testing and Integration
30%
50%
100%
Final Assembly
100%

CHAPTER 2

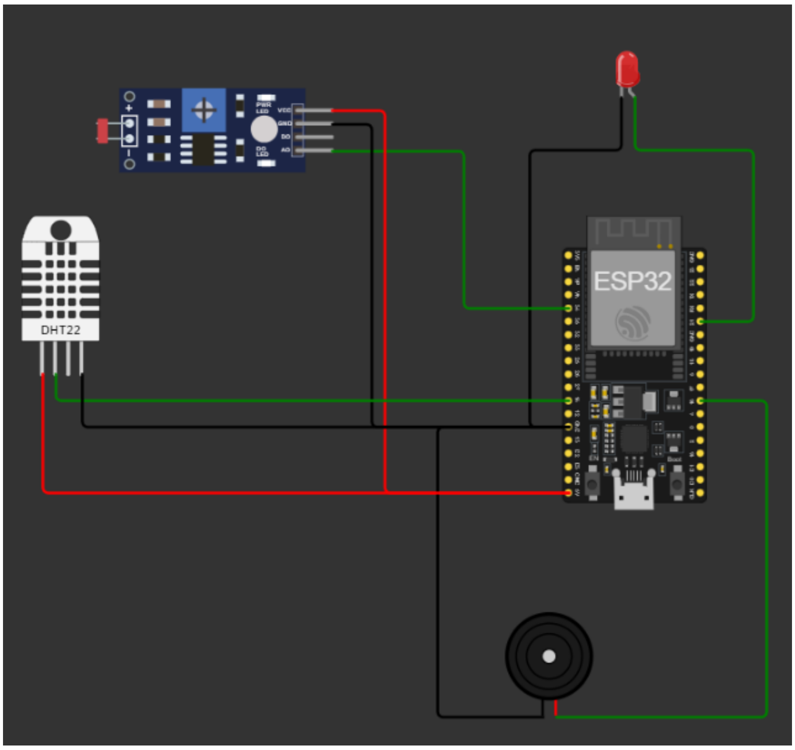
IMPLEMENTATION

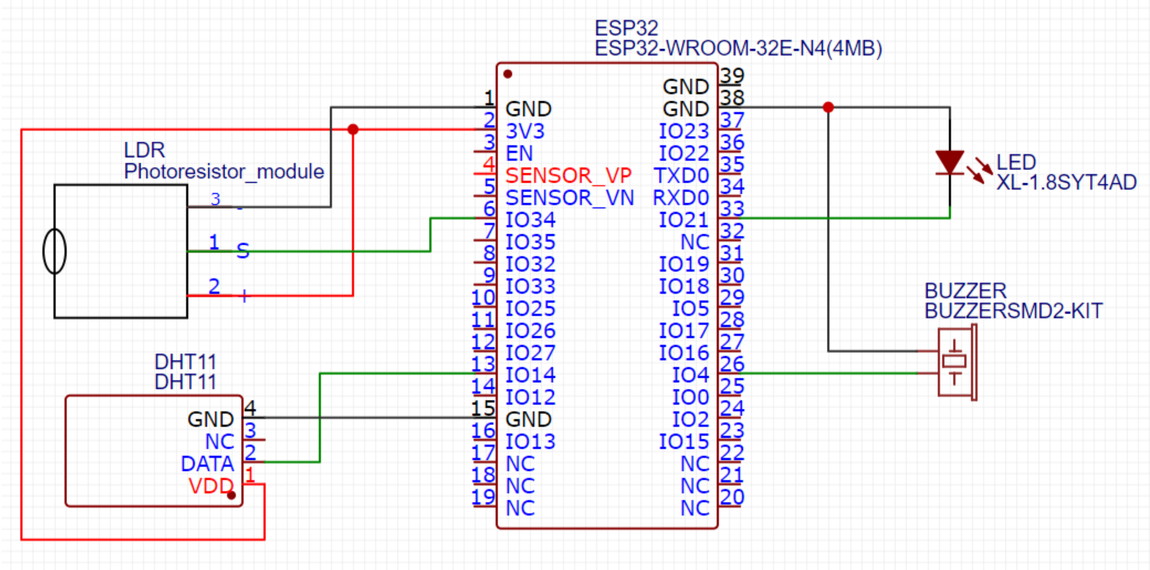
2.1 HARDWARE DESIGN AND SCHEMATIC

A) ESP CAM



B) ESP32





2.2 SOFTWARE DEVELOPMENT

2.2.1 Frontend Software

The frontend system plays a pivotal role, leveraging the power of React to provide an intuitive and dynamic interface for monitoring all devices connected to a user's account. React, known for its efficiency and flexibility, serves as the backbone for creating a responsive and engaging user experience.

The primary function of the frontend is to serve as a centralized hub for monitoring devices. Through React components, users can seamlessly navigate and visualize real-time data from their connected IoT devices. The system employs REST API calls to establish a secure and efficient connection with the backend, facilitating essential functionalities like user authentication, device registration, and management.

To establish communication with IoT devices, the frontend utilizes MQTT (Message Queuing Telemetry Transport) protocol. MQTT's lightweight and efficient nature makes it an ideal choice for real-time data exchange with IoT devices. This connection enables the frontend to receive and display crucial information from the devices, ensuring users have immediate access to the latest updates and warnings.

Reasoning for using MQTT and REST API:

- MQTT (Message Queuing Telemetry Transport): We chose MQTT for efficient and real-time communication between the monitoring device and the Frontend. MQTT's lightweight protocol ensures rapid data transfer, making it ideal for monitoring applications where timely updates are crucial. Its publish-subscribe model allows for an organized and responsive data flow.

- REST API (Representational State Transfer Application Programming Interface): The Frontend interacts with the backend through REST API endpoints for user-related functionalities, such as login, registration, and device management. REST API provides a standardized and stateless communication approach, enhancing interoperability and simplifying integration. This choice ensures a secure and scalable foundation for managing user accounts and devices.

2.2.2 Backend Software

The backend system, powered by Node.js and Express, serves as the brains orchestrating key functionalities.

Node.js and Express:

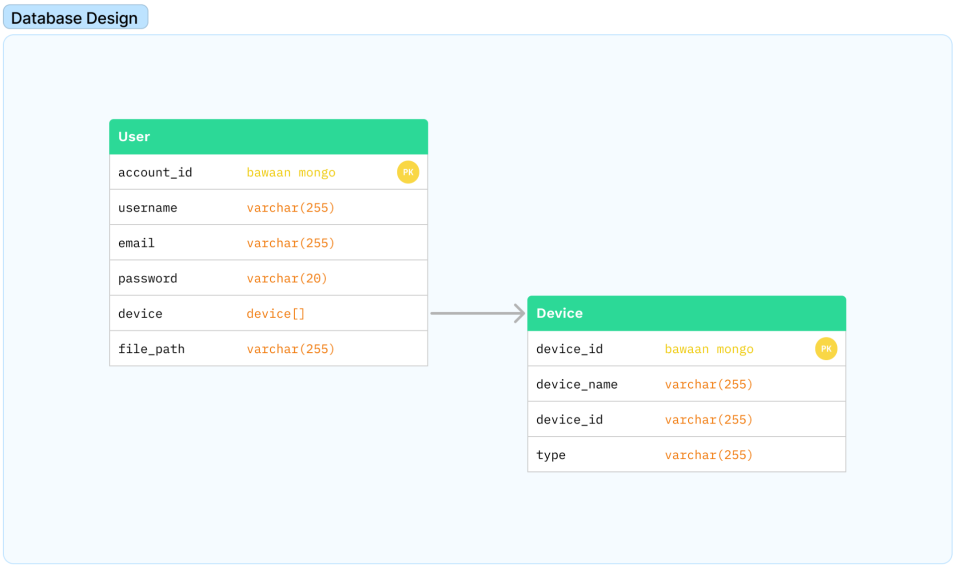
The backend leverages the efficiency and scalability of Node.js in tandem with the expressive and minimalist framework Express. This combination facilitates the creation of robust APIs and efficient handling of HTTP requests, ensuring a seamless communication channel with the frontend.

TensorFlow.js for Facial Comparison:

The integration of TensorFlow.js elevates the security aspect by enabling facial comparison. The backend employs this machine learning library to analyze faces captured by the system and compare them with entries in the database. This enhances the system's ability to detect intruders and provides an added layer of protection.

The TensorFlow.js-powered facial comparison module allows the backend to discern intruders by matching faces captured by the system with those stored in the database. This process ensures the system's ability to accurately identify and respond to potential security threats. The backend seamlessly connects with the database, establishing a secure link for quick retrieval and comparison of facial data.

Database Design



Database Design

User		
account_id	bawaan mongo	PK
username	varchar(255)	
email	varchar(255)	
password	varchar(20)	
device	device[]	
file_path	varchar(255)	

Device		
device_id	bawaan mongo	PK
device_name	varchar(255)	
device_id	varchar(255)	
type	varchar(255)	

The project's database relies on MongoDB, a NoSQL solution chosen for its scalability and flexibility. The database comprises two essential collections: User and Device.

- User Collection

  The User collection serves as the repository for user-centric information, encompassing credentials, preferences, and personalized details. MongoDB's flexible schema accommodates evolving user needs, ensuring seamless user authentication, registration, and a personalized experience.

- Device Collection

  The Device collection centralizes information related to connected IoT devices, housing details such as unique identifiers, status, and metadata. MongoDB's document-oriented structure aligns naturally with managing device data, facilitating straightforward queries and updates.

2.2.3 IoT Software

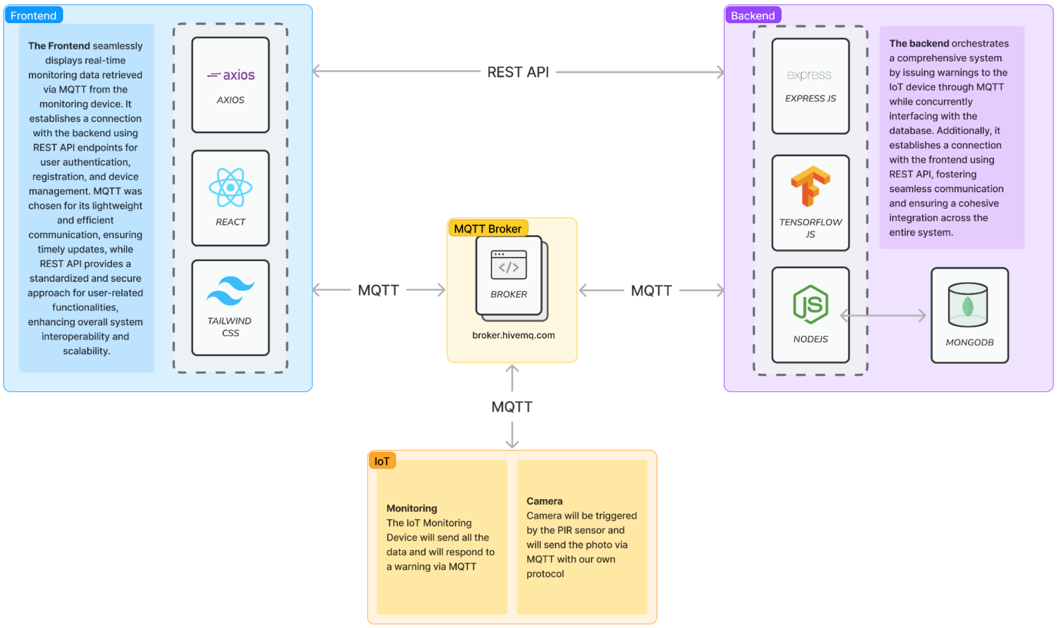
ESP32 CAM

The ESP32 CAM's software functionality is primarily focused on image capture, motion detection, and facial recognition. The integration with TensorFlow.js in the backend enhances its facial comparison capabilities, contributing to the system's robust security features. The software running on the ESP32 CAM manages the image capture process, communicates effectively with the backend for facial analysis, and ensures the seamless integration of visual data into the larger smart home security system.

ESP32 Monitoring:

The software running on the ESP32 microcontroller is multifaceted, handling tasks related to environmental data monitoring and MQTT communication. Its ability to interface with sensors for real-time data collection and transmit this information to the frontend demonstrates efficient software design. Moreover, the software on the ESP32 microcontroller ensures it can receive and interpret warning signals from the backend via MQTT, enabling a swift response to security alerts. The versatility of its software contributes to the seamless integration of the IoT devices within the broader smart home automation system.

2.3 HARDWARE AND SOFTWARE INTEGRATION



The Frontend seamlessly displays real-time monitoring data retrieved via MQTT from the monitoring device. It establishes a connection with the backend using REST API endpoints for user authentication, registration, and device management. MQTT was chosen for its lightweight and efficient communication, ensuring timely updates, while REST API provides a standardized and secure approach for user-related functionalities, enhancing overall system interoperability and scalability.

The backend orchestrates a comprehensive system by issuing warnings to the IoT device through MQTT while concurrently interfacing with the database. Additionally, it establishes a connection with the frontend using REST API, fostering seamless communication and ensuring a cohesive integration across the entire system.

Monitoring
The IoT Monitoring Device will send all the data and will respond to a warning via MQTT

Camera
Camera will be triggered by the PIR sensor and will send the photo via MQTT with our own protocol

In the architecture of our smart home security and automation system, the integration of software and hardware components is seamlessly achieved through the utilization of the HiveMQ MQTT broker and REST API. Acting as a central communication hub, the MQTT broker facilitates real-time data exchange among IoT devices, including the ESP32 CAM and ESP32 microcontroller, ensuring decentralized yet efficient interactions. On the software side, the REST API serves as the bridge between the frontend and backend, enabling user-centric functionalities such as authentication, registration, and device management. This RESTful architecture ensures standardized and stateless communication, fostering interoperability and user-friendly interactions. The synergy between the MQTT broker and REST API forms a cohesive and responsive system, where devices communicate swiftly, and users have seamless control over their smart home environment. This integrated approach not only enhances reliability and responsiveness but also provides a scalable foundation for future feature enhancements and adaptations to evolving smart home requirements.

CHAPTER 3

TESTING AND EVALUATION

3.1 TESTING

3.1.1. ESP32-CAM

The testing for ESP32-CAM module is to check the PIR Sensor is able to trigger the camera, and then send the taken picture to the MQTT broker. The acceptance criteria for this test is verifying the pir sensor and camera's functionality, ensuring proper image capture, and validating the successful transmission of images to the specified MQTT topic.

3.1.2. ESP32

The testing for ESP32 module is validate the accurate readings from DHT 11 sensor, validate the reading of the LDR sensor and turn on/off the LED based on the reading, validating the buzzer activation upon receiving alerts from the backend system, and sending the data read to MQTT broker.

3.1.3. Back End

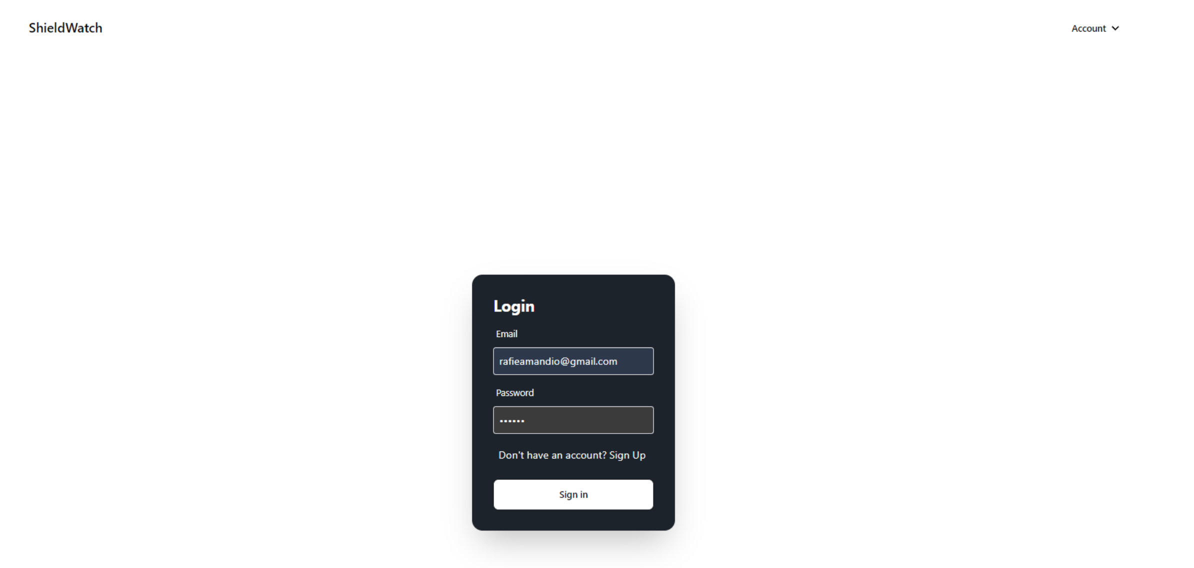
The testing for Backend involve getting the image from MQTT server that was sent by ESP32-CAM, and checking it if it is an intruder or not and send an alert accordingly. The testing ensured seamless bidirectional communication between the hardware devices and the backend.

3.1.4. Front End

The testing for Frontend involve properly displaying the sensor reading that was sent by ESP32 and the image that was sent by ESP32-CAM

3.2 RESULT

3.2.1 Frontend



Email: rafieamandio@gmail.com

Password: ••••••

Don't have an account? Sign Up

The frontend interface proves its mettle by displaying real-time data sourced from MQTT topics. Homeowners can conveniently access up-to-date security status information through an intuitive and visually engaging interface. The frontend not only serves as a dashboard for monitoring but also facilitates user-friendly interactions, such as login, registration, and device management, ensuring a seamless user experience.

The frontend of the smart home security and automation system successfully implements a range of essential features, contributing to a user-friendly and comprehensive interface. Here's an overview of each feature:

- Login:

  The Login feature provides users with a secure and authenticated entry point to access the system. Users can confidently log in to the frontend, ensuring that their personal and security-related information is protected.

- Register:

  The Register feature allows new users to seamlessly create accounts within the system. It provides a straightforward and intuitive registration process, ensuring that users can quickly set up their profiles and gain access to the smart home security and automation functionalities.

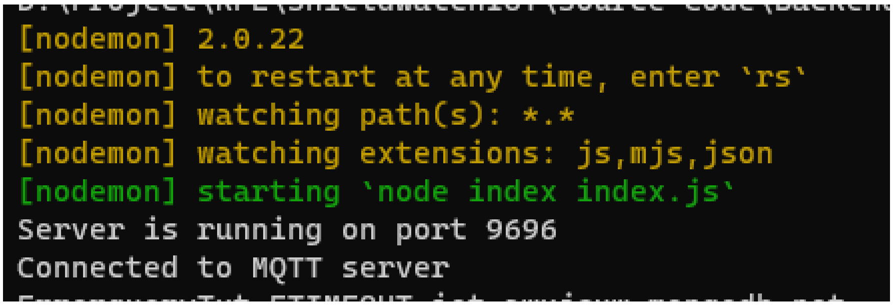
- Add New Device:

  The Add New Device feature enhances the system's scalability by allowing users to effortlessly integrate new IoT devices into their smart home ecosystem. This feature streamlines the process of expanding the system, ensuring that users can easily incorporate additional devices to meet their evolving needs.

- Monitoring:

  The Monitoring feature stands as the central component of the frontend, offering users real-time insights into their smart home's security status. Through an intuitive and visually engaging dashboard, users can monitor the data generated by connected devices, staying informed about intrusions, environmental conditions, and overall system status.

3.2.2 Backend



```
[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
Server is running on port 9696
Connected to MQTT server
```

The backend of the smart home security and automation system incorporates critical features to ensure robust functionality and efficient management. Here's an overview of each feature:

- Connecting to Database:

  The Connecting to Database feature establishes a secure and reliable link between the backend and the MongoDB database. This integration enables efficient data storage, retrieval, and management, supporting critical functionalities such as user and device information storage.

- Face Comparison:

  The Face Comparison feature, powered by TensorFlow.js, showcases advanced facial recognition capabilities. The backend efficiently compares facial data captured by the

ESP32 CAM with entries in the database, enhancing the system's ability to detect and respond to intruders with high precision.

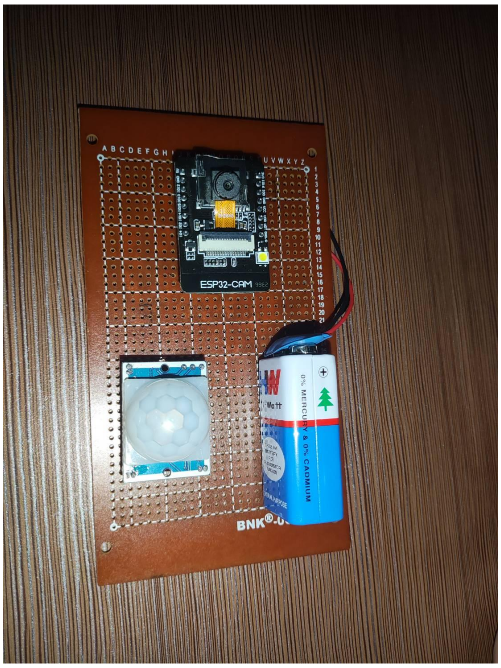
- Auto Email Warning:

  The Auto Email Warning feature adds an extra layer of security by automatically generating and sending timely email notifications in response to detected intrusions. This feature ensures that homeowners receive clear and detailed alerts, complete with intrusion details and images, allowing for swift awareness and response to potential security threats.

- Device and User Management:

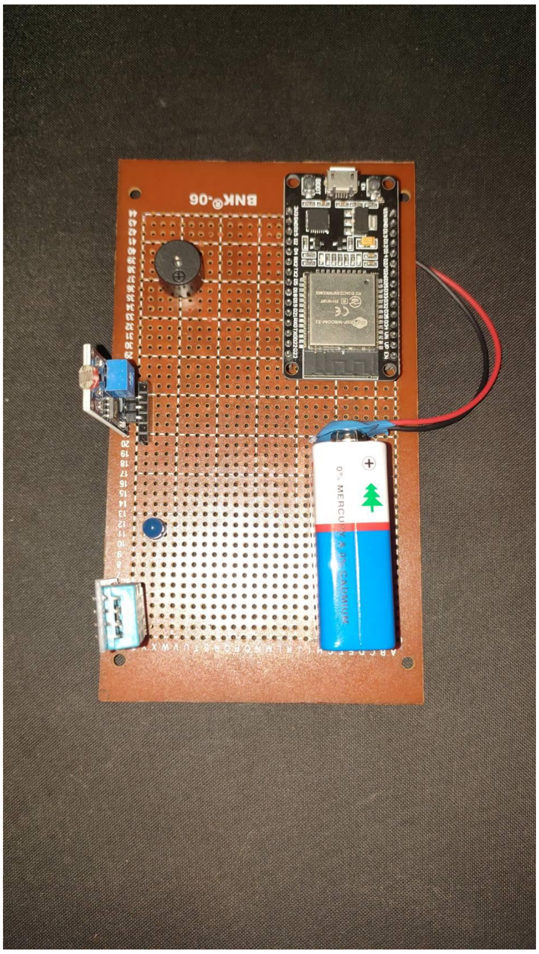
  Device and User Management features empower homeowners with control over their smart home ecosystem. Users can seamlessly manage their connected devices, ensuring a scalable and personalized experience. Additionally, robust user management functionalities provide administrators with the ability to oversee and control access, enhancing security measures.

3.2.3 IoT Devices



ESP32 CAM:

- Intruder Detection with PIR Sensors: The ESP32 CAM stands as the first line of defense, promptly detecting intrusions through its sophisticated Passive Infrared (PIR) sensors. This capability triggers immediate alerts, bolstering the overall security measures in place.



ESP32 Monitoring

- Environmental Data Monitoring: The ESP32 microcontroller assumes a multifaceted role by connecting to DHT11 and LDR sensors. This integration enables the monitoring of environmental conditions, ensuring accurate temperature readings for comfort and efficient lighting control for energy optimization.

- Communication via MQTT: Serving as a communication hub, the microcontroller establishes seamless connections with the MQTT broker. This facilitates real-time data exchange with the frontend, enabling homeowners to monitor environmental conditions and security alerts promptly. Furthermore, it receives warning signals from the backend through MQTT, ensuring a swift and synchronized response to potential security threats.

- Emergency Warning Feature: In a strategic enhancement to its capabilities, the ESP32 microcontroller now incorporates an Emergency Warning feature. This feature empowers the device to generate and transmit urgent warning signals in critical situations. By doing so, the system becomes equipped to respond swiftly to emergency scenarios, ensuring the safety and well-being of the homeowners.

3.3 EVALUATION

The project report provides a comprehensive overview of the smart home security and automation system, highlighting its features, functionalities, and the successful implementation of various components. However, as with any project, there are areas that warrant further evaluation and potential improvement:

- Facial Recognition Processing Speed:

  Observation: The evaluation reveals that the current hardware for facial recognition, particularly in the Backend, might benefit from faster processing capabilities.

  Recommendation: Consider exploring hardware upgrades or optimizations to enhance the facial recognition processing speed. This could involve assessing more powerful server GPU or dedicated hardware for facial recognition tasks.

- Email Limitations

  Observation: The report notes that the email functionality is limited to 300 emails per day due to the constraints of the free mailer service.

  Recommendation: Explore alternative email services or consider upgrading to a premium mailer service to remove email limitations. This ensures a more robust and scalable email notification system without constraints.

- MQTT Topic Organization:

  Observation: The report suggests that MQTT topics are currently organized in a way that aggregates messages for all users into a single topic.

  Recommendation: Consider reorganizing MQTT topics to a one-topic-per-person structure. This improves scalability and ensures a more organized and efficient communication flow, especially as the number of users and devices increases.

Overall Assessment:

The project demonstrates a commendable integration of various technologies and functionalities to create a smart home security and automation system. The evaluation highlights areas for improvement, focusing on enhancing facial recognition speed, addressing email limitations, and optimizing MQTT topic organization for improved scalability and efficiency. These recommendations aim to further elevate the system's performance and user experience.

CHAPTER 4

CONCLUSION

In conclusion, the "Smart Home Guardian" project has successfully addressed the need for an intelligent home security solution by leveraging IoT technology. The integration of PIR sensors, DHT11, LDR, and ESP32 ensures comprehensive security coverage, from intruder detection to environmental monitoring and camera activation. The backend, utilizing MQTT communication and face recognition libraries, enhances the system's intelligence and adaptability. The frontend provides users with a user-friendly interface, offering real-time monitoring and efficient device management. Additionally, the email notification feature adds an extra layer of alerting to keep users informed of any detected intrusions. The project's successful implementation of the proposed solution demonstrates its capability to enhance home security through advanced technology. Looking forward, further refinements and expansions could optimize the system's performance and broaden its applicability in creating secure and smart living spaces.

REFERENCES

[1] “Getting Started With ESP32-CAM: A Beginner’s Guide,” *Last Minute Engineers*, Mar. 02, 2023. https://lastminuteengineers.com/getting-started-with-esp32-cam/

[2] Components101, “DHT11 Sensor Pinout, Features, Equivalents & Datasheet,” *Components101*, Jul. 16, 2021. https://components101.com/sensors/dht11-temperature-sensor

[3] circuitgeeks, “Arduino Buzzer Tutorial,” *Circuit Geeks*, Mar. 31, 2021. https://www.circuitgeeks.com/arduino-buzzer-tutorial/

[4] “Arduino with PIR Motion Sensor | Random Nerd Tutorials,” *Random Nerd Tutorials*, Aug. 18, 2014. https://randomnerdtutorials.com/arduino-with-pir-motion-sensor/

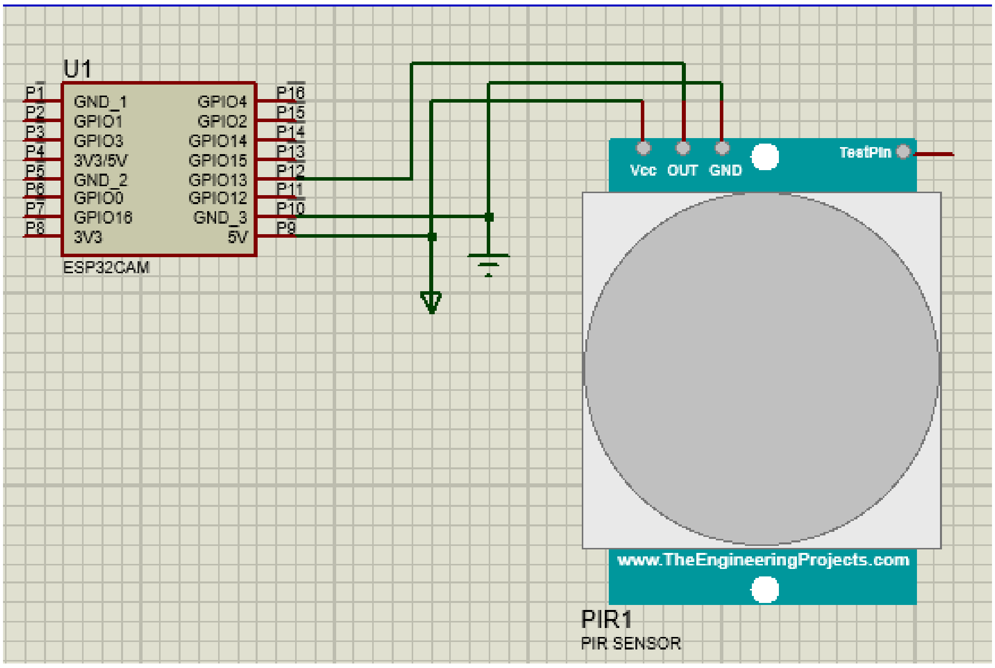
[5] “How LDR Sensor Works - Working & Application,” *Robocraze*, Dec. 08, 2023. https://robocraze.com/blogs/post/how-ldr-sensor-works#:~:text=The%20Light%20Dependent%20Resistor%20(LDR (accessed Dec. 10, 2023).

[6] DIY Engineer, "ESP32-Cam – Complete Guide." diyengineers.com. https://www.diyengineers.com/2023/04/13/esp32-cam-complete-guide (accessed Nov. 24, 2023).

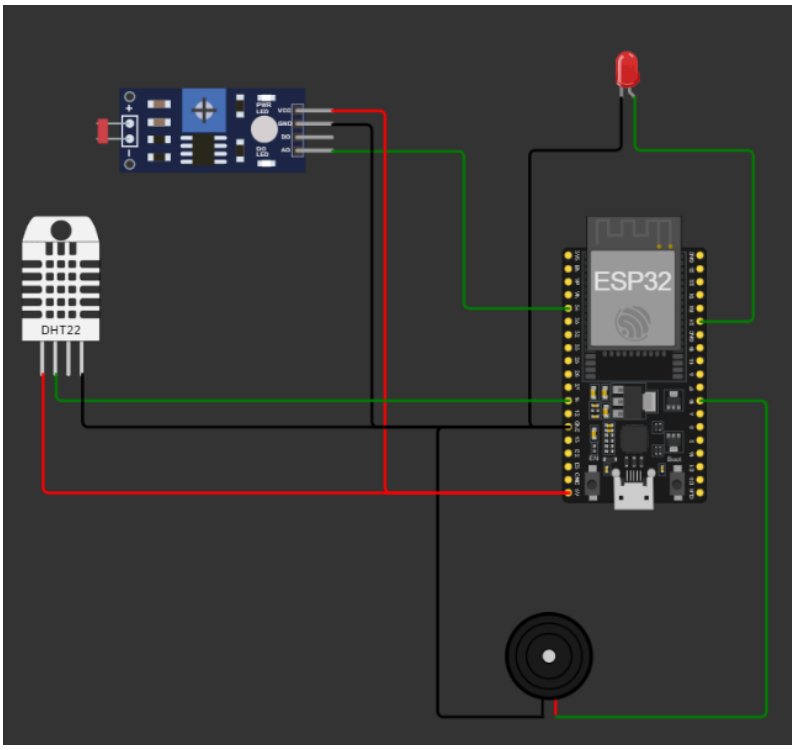
APPENDICES

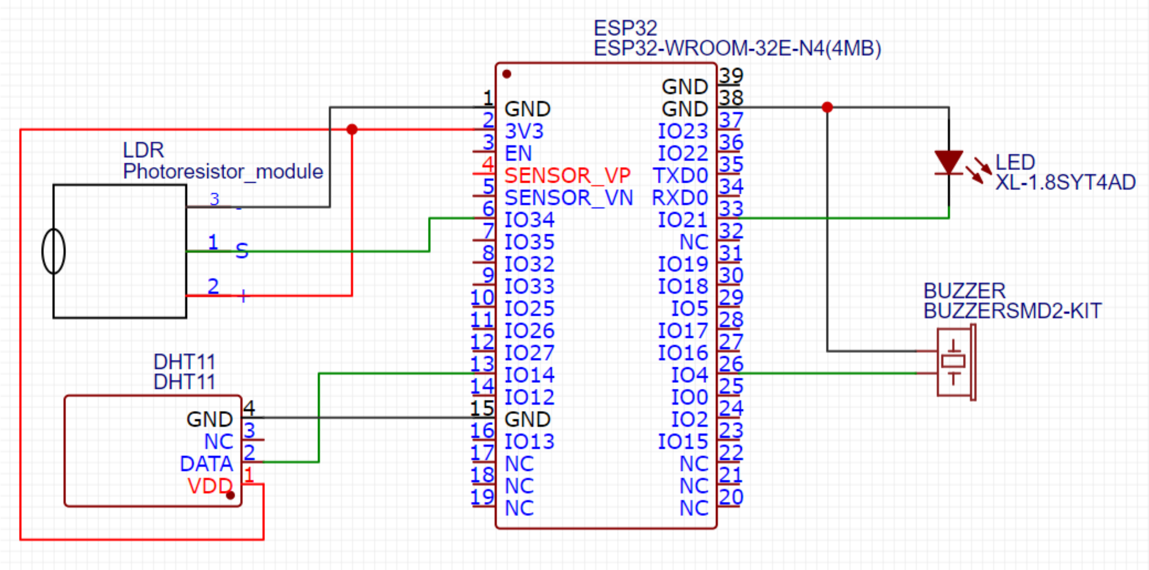
Appendix A: Project Schematic

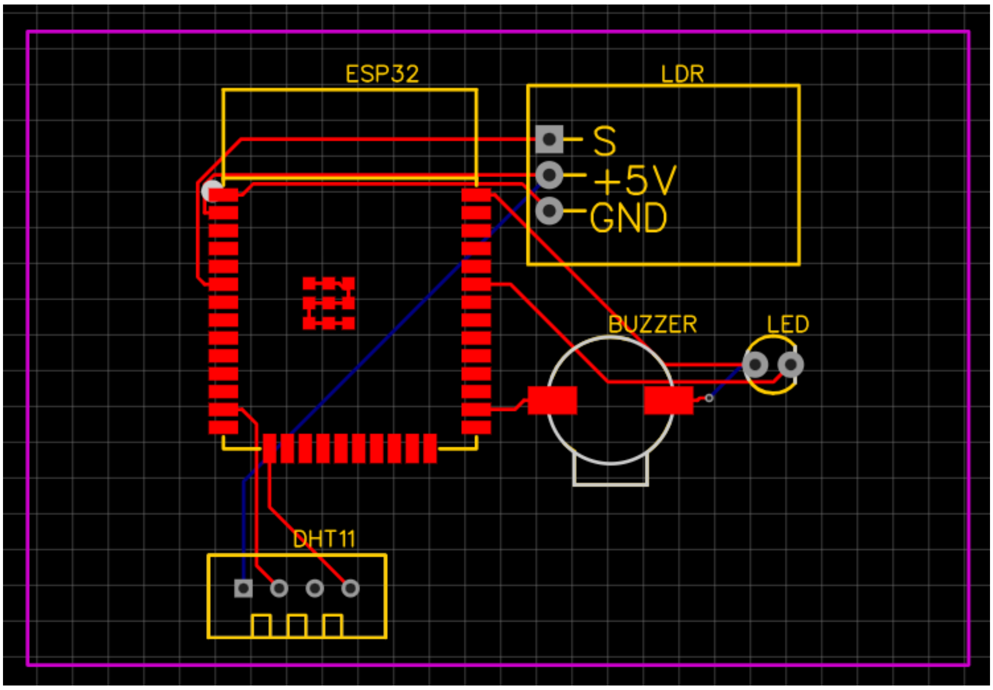
A) ESP CAM



B) ESP32




ESP32
ESP32-WROOM-32E-N4(4MB)
LDR
Photoresistor_module
3
1 S
2 +
DHT11
DHT11
GND 4
NC 3
DATA 2
VDD 1
1 GND
2 3V3
3 EN
4 SENSOR_VP
5 SENSOR_VN
6 IO34
7 IO35
8 IO32
9 IO33
10 IO25
11 IO26
12 IO27
13 IO14
14 IO12
15 GND
16 IO13
17 NC
18 NC
19 NC
GND 39
GND 38
IO23 37
IO22 36
TXD0 35
RXD0 34
IO21 33
NC 32
IO19 31
IO18 30
IO5 29
IO17 28
IO16 27
IO4 26
IO0 25
IO2 24
IO15 23
NC 22
NC 21
NC 20
LED
XL-1.8SYT4AD
BUZZER
BUZZERSMD2-KIT


ESP32
LDR
S
+5V
GND
BUZZER
LED
DHT11