UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

BACHELOR'S THESIS

By:

Student name: HOANG Viet Thanh BI12-408

Major: Information and communication technology

Title:

Investigation of IoT-Based Monitoring and Alert System For Machine Room Safety Using Raspberry Pi

Dr. PHAM Ngoc Minh Thesis supervisor

IoIT Institute of Information Technology

MSc. LE Chu Nhu Hiep Co-supervisor of the thesis

USTH University of Science and Technology of Hanoi

Hanoi, 2025

ABSTRACT

In recent years, fire accidents – particularly in enclosed and high-risk environments have become increasingly common, leading to severe damages and operational disruptions. The development of IoT fire monitoring and alarm systems has received significant attention, but challenges remain in terms of cost, real-time response, and infrastructure complexity. To solve these limitations and enable accurate, complete monitoring, this thesis investigates the use of lightweight wireless communication protocols that do not require existing infrastructure. Sensor data is securely encrypted and transmitted from the nodes to a central processing unit for real-time analysis, thereby increasing system stability and accuracy.

This thesis proposes a real-time IoT-based fire monitoring and alert platform that utilizes multiple distributed sensor nodes communicating via low-power wireless channels. Data is visualized through a web-based interface to ensure user-friendly interaction. In addition, a Fuzzy Logic algorithm is used to evaluate the potential for a fire based on temperature, humidity, and other sensor data. The technology automatically sends out warnings over email and sounds alarms, providing quick and efficient reactions to any risk of fire.

AUTHORSHIP

I have completed my bachelor's thesis, "**Investigation of IoT-Based Monitoring and Alert System for Machine Room Safety Using Raspberry Pi**," as a result of my diligent study and research under the guidance of my academic supervisors. In the reference part of this thesis, I have completely and correctly referenced all sources used during the research process.

I hereby confirm that the facts and results contained in this thesis are true and correct. I assume full responsibility for any errors or instances of academic misconduct (if any). I will accept any punishments given by the University of Science and Technology of Hanoi (USTH) to comply with its statutes.

.

Hanoi, July 4, 2025

Author

Hoang Viet Thanh

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all the professors and colleagues who have supported and guided me throughout my academic journey. My sincere thanks also to the University of Science and Technology of Hanoi (USTH) for accompanying me from the early days of university to the completion of this graduation thesis.

First and foremost, I would like to express my deepest appreciation to my thesis supervisor, Mr. PHAM Nhat Minh from the Institute of Information Technology (IOIT), for providing me with the opportunity to intern in his embedded lab. His guidance and insightful suggestions have played an important role in shaping his ideal for this project. Before the internship, I mainly focused on software. Thanks to his mentorship, I was introduced to hardware and embedded systems, expanding both my skills and interests. I also thank Mr. NGO Duy Tan from IOIT for his creative input and inspiration during the development of my thesis.

I am equally grateful to my co-supervisor, Mr. Le Chu Nhu Hiep from USTH, for his technical support in software and system deployment and for his assistance in refining my final report.

Furthermore, I would like to thank my friends Mr. HOANG Van Son, Mr. NGUYEN Trung Kien, NGUYEN Nhat Anh, and DOAN Quang for their support and companionship throughout this journey.

Finally, I wish to dedicate my deepest gratitude to my parents and sister, whose unconditional love and constant encouragement have been the foundation of my enduring strength. I am eternally thankful for their unwavering support throughout this journey.

Contents

List of Abbreviations

Abbreviation	Definition
IoT	Internet of Things

List of Figures

List of Tables

Introduction

In recent years, the risk of fire in enclosed environments, such as machine rooms, has become a growing concern, for example, due to the high density of heat-generating equipment and the presence of flammable materials. Traditional fire alarm systems often lack real-time monitoring, require constant human supervision, and have limited remote accessibility and intelligent response mechanisms. Therefore, quick detection and rapid response are crucial to minimizing damage and ensuring safety.

With the advancement of the Internet of Things (IoT), smart fire monitoring systems have emerged as a promising solution to overcome these limitations. IoT enables the integration of multiple environmental sensors, real-time data transmission, and remote control, allowing for more accurate and efficient fire detection. Moreover, the use of hardware platforms such as the Raspberry Pi makes it easier and more cost-effective to develop custom, scalable systems.

This thesis investigates the design and implementation of an IoT-based fire monitoring and alert system specifically tailored for machine room environments. The system utilizes multiple sensors, such as temperature, humidity, and flame detection modules, which are integrated into the node connected to a Raspberry Pi, which serves as the central processing unit. This system employs wireless communication protocols for data transmission and visualizes sensor data through a web-based interface. Additionally, fuzzy logic and threshold-based alert mechanisms are implemented to evaluate fire risk levels, triggering notifications via email and activating audible alarms to ensure timely alerts and responses.

Chapter 1. Overview of Monitoring &Alert System

This chapter provides an overview of the application of Internet of Things (IoT) technology in monitoring and alerting fire detection systems, primarily in the context of increasingly serious fire problems that require critical attention. It begins with an introduction to IoT and the importance of its role in improving monitoring skills and its value in fire situations. In addition, this chapter also focuses on the main objective of the thesis: to explore and employ the full potential of IoT-connected hardware devices in both IoT applications and real-world scenarios. This chapter is organized into three sections. Section 1.1 provides an overview of the IoT and its applications in fire monitoring systems. *Section 1.2* discusses the current situation and limitations of existing fire monitoring systems. *Section 1.3* presents modern fire detection methods using current technologies, along with their limitations.

1.1. Introduction to the Internet of Things (IoT) and Its Applications for Fire Monitoring Alert Systems

In recent years, fire accidents have become increasingly dangerous, causing critical damage to both human life and property. Traditional fire monitoring systems, which often rely on manual checks or local alerts, are limited in their ability to respond quickly and effectively, especially in complex or remote environments. To handle this issue, we need smart and effective solutions to improve it in time.

The IoT is one exciting technical development that resolves these issues. As its name suggests, IoT refers to a network of smart devices that can communicate and connect with each other, exchanging data. These devices can collect, transmit, and process data in real-time, thereby enabling automation and making personal decisions.

Unlike standalone devices that work independently, a true IoT system (Figure 1) is a scalable and integrated system where multiple devices work together for a common goal. In the context of fire detection, this goal is always monitoring environmental conditions, identifying early signs of fire, and sending real-time alerts, even when no one is present at that location.

By providing features such as remote monitoring, automated notifications, and log data, IoT-based systems significantly enhance responsiveness and overall safety. Therefore, using IoT for fire monitoring systems is both useful and essential for enhancing early detection and minimizing damage.

Figure 1. An IoT system with interconnected smart devices [1].

1.2. Current situation and Limitations of the monitoring fire alert system

Fire accidents are now common and can occur in various types of environments, ranging from crowded residential areas to industrial zones, forests, etc,.. In an industrial environment, the use of flammable materials, high electricity consumption, and poor ventilation are common reasons that increase the risk of fire. In a residential area, the complex and overloaded electrical network increases the chance of short circuits and electrical fires. In a forest environment, rising global temperatures make it more flammable, and also human activities, such as camping or purposeful burning of the forest, increase the risk of fire. As mentioned, the fire can occur anywhere, any time, whether in crowded urban areas or isolated natural areas.

Therefore, deploying fire monitoring systems with sensors and data-collecting modules in large areas is expensive and challenging. Traditional systems, which rely on wired connections or require continuous human monitoring, are not realistic, especially in large areas or remote and harsh environments. These systems will face problems such as processing complex data and false fire alerts. This is

really a big challenge for the fire warning system. The solution using IoT with sensor data can help provide more accurate and useful results.

Currently, traditional fire monitoring and alert systems are still widely used and have not been fully replaced by IoT-based systems. Although some improvements have been made, many issues and limitations still exist, such as detection delays, limited coverage range, and high maintenance requirements. As a result, developing a fire detection and monitoring system that uses fire, temperature, and humidity sensors combined with wireless communication technologies is a creative, possible, worthy solution to consider.

1.3. Related Works

The development of fire monitoring and alert systems based on IoT has gained increasing attention in recent years. This section presents several existing works and discusses their limitations, highlighting the gap that this project aims to address.

Currently, IoT systems typically integrate various sensors and microcontrollers, ranging from simple to advanced designs. Moreover, with the growing popularity of artificial intelligence (AI), many modern IoT applications are incorporating machine learning to improve detection accuracy. Although this thesis does not apply AI, it is planned as a direction for future work.

Regarding related works, one example is presented in [2], the authors proposed an IoT-based fire detection and monitoring system using temperature, flame, and smoke sensors, paired with an ESP32 microcontroller. Their system employed a LoRa network to detect and alert about forest and farm fires. Similarly, in [3] , the authors developed an Arduino-based home fire alarm system that uses a GSM module and temperature sensor to send SMS alerts, thereby enhancing user safety and protecting property. In another study [4], the researchers designed a smart IoT-based system that also used temperature, flame, and smoke sensors, like in [2], but combined them with an Arduino UNO and a gas sensor. Their system was implemented on the Blynk platform and used GSM communication for alerting. Meanwhile, in [5], an STM32-based wireless fire detection system was introduced,

utilizing multiple sensors and embedded wireless technologies such as Wi-Fi to detect fires in real-time. A more advanced system is shown in [6], which focuses on the use of robotics in fire monitoring. It integrates STM32 and STC89C51 microcontrollers, along with drones, to create a combined air and ground monitoring system, with communication facilitated through ZigBee. On the other hand, [7] emphasizes the alerting. This system uses an ESP32 with PIR sensors to detect both fire and movement and sends alerts through a Telegram bot to smartphones, including visual and temperature-based alarms.

As AI continues to develop, some recent papers have begun integrating it with IoT for fire detection. For instance, [8] presents a system that combines IoT devices with the YOLOv5 model to enable early and real-time forest fire detection, aiming to reduce false alarms and enhance safety during dry seasons. Building upon this, [9] proposes an improved system that uses IoT and machine learning, integrating various sensor types to enhance detection accuracy and facilitate remote monitoring. Although recent works reflect technological advancements, the core concept of IoT-based fire detection was explored much earlier. Significantly, [10] the paper describes a real-time fire alarm system developed using Raspberry Pi and Arduino Uno. This system included smoke detection, room image capture, and alerting through SMS and a web interface. Moreover, it featured a user confirmation step before notifying firefighters, effectively reducing false alarms. Despite having a basic web interface, the implementation was comprehensive and forward-thinking. Therefore, this paper provides strong motivation to support this thesis.

As stated earlier, while this thesis does not yet include AI integration, it aims to build upon these foundational ideas. Notably, many existing systems use Wi-Fi or GSM for communication. In contrast, this thesis proposes an enhanced monitoring and alert fire system by adopting radio communication as the core of an IoT-based fire detection and alert system. Ultimately, this work is made possible thanks to the pioneering efforts of these earlier researchers, upon whose contributions this thesis continues to build.

1.4. Conclusions

An IoT system is an essential technology with applications in monitoring and fire detection. Achieving real-time monitoring, high accuracy, processing all scenarios, and overcoming hardware limitations creates serious difficulties. The use of IoT brings a promising solution, enabling real-time monitoring, remote access, and automated alert notifications. This project aims to inherit and integrate the ideas from those previous works, while also contributing further improvements to develop a more responsive and reliable fire alert system. It also introduces new innovations to improve the system's flexibility and performance in real-life situations. In Chapter 2 *System Architecture Design*, a proposal will be presented, focusing on the integration of hardware and software components, including sensor modules, communication systems, and control logic. All of which will be implemented through software to build the proposed IoT-based fire monitoring solution.

Chapter 2. System Architecture Design

The first chapter of this thesis provided an overview of IoT technology and its applications in fire detection and monitoring. It also discussed the current state of fire accidents occurring in many places today, highlighting the need for more efficient solutions. Furthermore, with the current state of fire, the thesis proposed a new direction by enhancing existing ideas with the use of radio communication. While many previous systems deploy on Wi-Fi or GSM, this thesis focuses on using *low-cost, low-power* wireless communication modules. The main goal is to develop an IoT-based system that monitors, detects fires, and alerts in real-time.

Building on this ideal, *Chapter 2* presents the proposed system architecture, which consists of both hardware and software components designed to work together for efficient fire monitoring. The system includes: Raspberry Pi and STM32 microcontrollers, HC-12 modules for communication, and a variety of sensors to collect fire-related data. This chapter has three main sections: *Section 2.1 Hardware System Architecture*: In this section, an overview of the hardware components used in this thesis will be presented, including sensors, microcontrollers, and the central processing unit. Each component will be explained with the reason why it is used and a summary of its advantages and disadvantages. *Section 2.2:* Software & Firmware System Architecture: This part details how the software & firmware work across the system. It begins with sensor-side firmware, which includes collecting data, encoding, and transmission. On the server side, it explains how data is received, decoded, stored, and visualized through a Flask-based web application. This section also includes user authentication, account management, and how software sends alerts (using Fuzzy Logic and Threshold Evaluation). *Section 2.3* Conclusions*:* This final section summarizes all components introduced in *Chapter 2* and prepares for the system implementation and evaluation that will follow in later chapters, *Chapter 3: Implementation Results and Evaluation*.

2.1. Hardware System Architecture

This section introduces the hardware components used at both the *Sensor Node* and *the Central Processing Unit*.

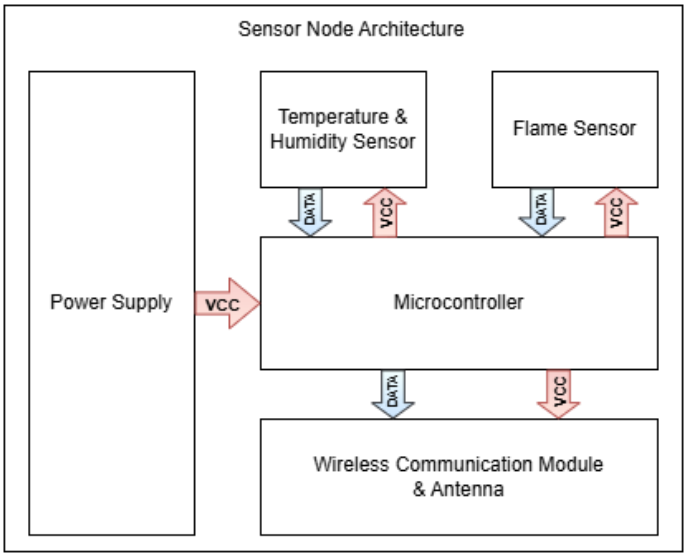


Figure 2: Sensor Node Architecture.

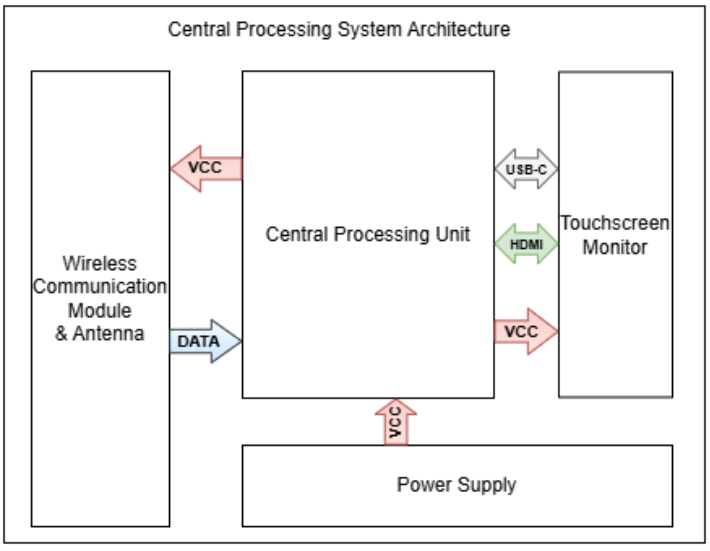


Figure 3: Central Processing Unit Architecture.

It explains what each component is, its role in the system, and the reasons behind choosing it for the demo. Subsections cover specific components, including fire sensors, temperature and humidity modules, STM32 for edge processing, and the Raspberry Pi for central control and web server hosting. The following *section 2.2* will discuss the data flow in depth, and how each part works together.

2.1.1. Sensor Node Architecture

(1) Flame Sensor using KY–026

Flame sensors are devices used to detect the presence of fire by sensing specific types of radiation emitted by flames, especially infrared light in the 760 – 1100 nm range. In this thesis, the KY-026 flame sensor is selected due to its low cost, small size, and easy to use. Despite being a basic sensor, it is well-suited for demo and educational purposes, making it a practical choice for the project's goals and budget.

Figure 4: Flame sensor KY-026 [11].

(2) Temperature & Humidity Sensor using DHT11

A temperature and humidity sensor is used to collect environmental data, which is impotant in fire detection since fires typically cause a quick increase in temperature and quick decrease in humidity. Depend only on a flame sensor may cause inaccurate results, so combining it with temperature and humidity will improves detection reliability and reduces false alarms. Although there are many types of environmental sensors, this thesis uses the DHT11 due to its low cost, low power, and availability – making it ideal for the demo goals of the project and suitable for academic applications.

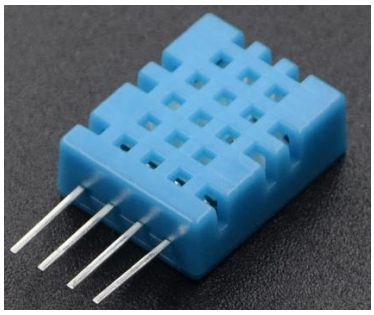


Figure 5: DHT11 - Temperature & Humidity Sensor [12].

(3) Microprocessor using STM32F103C8T6

In an IoT system, the edge processor is a lightweight microcontroller responsible for collecting data from sensors such as flame, temperature, and humidity sensors. This thesis uses the STM32F103C8T6 as the edge processor due to its suitability for the project's research and application goals. Its UART communication interface enables efficient serial data handling and transmission, making it a reliable choice for processing and transmitting sensor data in the proposed system.

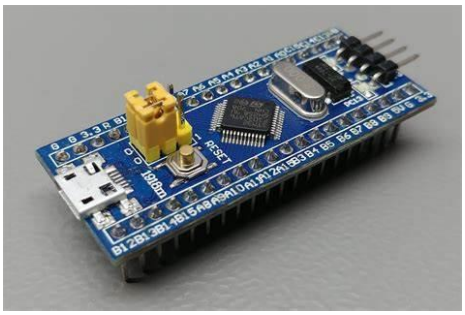


Figure 6: STM32F103C8T6 Microcontroller [13].

(4) Wireless Communication Module using HC-12

The wireless communication module is crucial in the proposed IoT system, enabling data transmission between sensor nodes and the central processing unit. While technologies like Wi-Fi, Zigbee, and Cellular require existing infrastructure, and LoRa offers long-range communication with higher cost and complexity, the HC-12 module provides a low-cost, easily configurable alternative well-suited to the project's goals. As described in Section 2.1.1, the HC-12 is integrated into the transmission block, where it wirelessly sends sensor data – processed by the STM32F103C8T6 microcontroller – through RF communication.

Figure 7: HC-12 Wireless Communication Module [14].

(5) Antenna

To enhance the wireless communication performance of the sensor node, an external 433 MHz antenna is connected to the HC-12 module through an SMA male connector. This antenna improves the range and stability of RF transmission between nodes and the central unit. Specifically, the selected antenna working at a frequency of 433 MHz, with a gain of 30 dBi, input impedance of 50 Ω, and a standing wave ratio (SWR) of ≤ 1.5. It has a height of 205 mm and supports a maximum input power of 100 W. The use of this antenna helps ensure reliable data transmission in both indoor and outdoor environments, supporting the system's goal of long-range, low-cost communication without requiring existing infrastructure.

Figure 8: 433 MHz RF Antenna [15].

2.1.2. Central Processing Unit System Architecture

(1) Wireless Communication Module using HC-12

In the *2.1.2 Central Processing Unit System Architecture*, the HC-12 module is placed in the receiver (Rx) block. After the data is transmitted from the node to the central processing unit through radio communication, the HC-12 module at the receiver receives incoming signals and transmits them to the central processor for handling.

(2) Central processing unit using Raspberry Pi 5

In this system, the central processing unit handles incoming data received wirelessly via the HC-12 module. To meet the project's goals, a Raspberry Pi 5 is used for its practical application in real-world IoT scenarios. As a single-board computer (SBC), the Pi 5 runs full operating systems like Raspberry Pi OS or Ubuntu and

supports microSD storage up to 1 TB. Powered by a quad-core ARM Cortex-A76 processor and equipped with 8 GB of RAM, it delivers strong performance comparable to standard PCs. It also offers many connectivity, including USB, Ethernet, Wi-Fi, Bluetooth, micro HDMI, and a 40-pin GPIO header for interfacing with sensors and modules like HC-12. With its power, flexibility, and ease of integration, the Raspberry Pi 5 is well-suited for modern IoT applications.

Figure 9: Single board Computer - Raspberry Pi 5 [16].

(3) Touchscreen Monitor using ASUS ZenScreen

Although this project does not focus on physical user interfaces, adding a touchscreen in the demo setup improves system visualization and user interaction. Therefore, an ASUS ZenScreen touchscreen is used. It is fully compatible with the custom monitoring software developed in this project. This addition enhances user understanding, simplifies testing, and provides a basis for future improvements.

Figure 10: Touchscreen [17].

2.2. Software & Firmware System Architecture

Section 2.1 focuses on introducing the hardware components used in this thesis, along with the reasons for their selection and their unique benefits. *Section 2.2. Software & Firmware System Architecture* discusses the working flow of the full

system implementation. *Section 2.2* is divided into two subsections: *Section 2.2.1. Firmware at the Node* explains how data is handled at the sensor node after collecting data from flame, temperature, and humidity sensors. It also provides the standard requirements for protecting the privacy and security of data during wireless transmission. *Software at the Central Processing Unit* uses software to receive, decode, and process incoming data packets. The decrypted data will be uploaded in real-time onto a web-based platform. This subsection also covers user administration, data logging, notification handling, and other system operations, which will be described in further depth.

2.2.1. Firmware at the Node

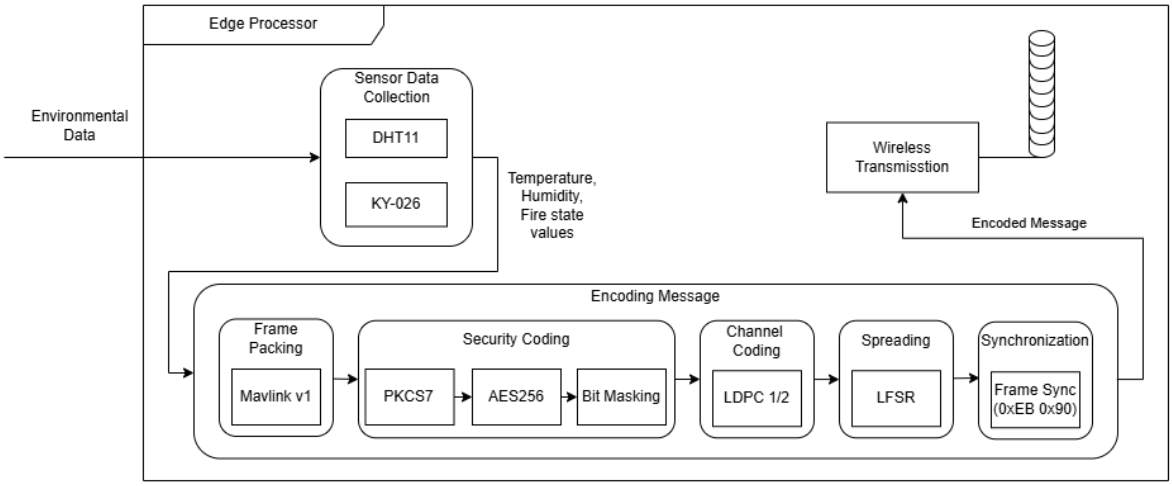


Figure 11: Firmware Architecture at the Sensor Node.

(1) Sensor Data Collection

Input: Value from the environment.

Output: Temperatue, Humidity, Fire state data.

During the Sensor Data Collection stage, environmental data is collected using two sensors: the DHT11 for temperature and humidity and the KY-026 for fire detection.

Algorithm 1: Read DHT11 Data and Fire Sensor State.

Loop forever
start DHT11, confirm **Presence** (check DHT11 response)

```
        Rh_byte1, Rh_byte2, Temp_byte1, Temp_byte2, crc ← read five bytes
        valid ← ((Rh_ byte1+Rh_ byte2+ Temp _byte1+Temp_byte2) & 0xFF) = crc
        Temperature ← float(Temp_byte1)
        Humidity    ← float(Rh _byte1)
        Fire ← readGPIO(GPIOA, PIN1)
end loop
```

The DHT11 employs a 1-Wire interface, with the STM32 initiating communication through GPIOA Pin 6 set as output with accurate time delays. After setting up, it returns 5 bytes (humidity, temperature, and checksum), and data validity is validated by checking the checksum. The KY-026 flame sensor, which is attached to GPIOA Pin 1 (digital) and, alternatively, GPIOA Pin 0 (analog via ADC), produces HIGH (1) when no flame is detected and LOW (0) when there is a fire.

(2) Decoding Message

Figure 12: Data protection sequence in wireless transmission.

This is a highly important and interesting part when transmitting messages over wireless communication. It ensures compliance with standard requirements for protecting privacy and securing data in wireless transmission. In wireless data security, the main components typically include (Figure 12): Frame Packing, Security Coding, Channel Coding, Spreading, and Synchronization.

a) <u>Frame Packing using Mavlink v1</u>

Input: Temperature, Humidity, Fire state data.

Output: MavLink frame.

In the Frame Packing step of the thesis, sensor data – including temperature, humidity, and fire state – is packed into a standardized communication format using

the MAVLink protocol (version 1). This protocol, widely used in embedded and wireless systems, ensures reliable and efficient data exchange. Each MAVLink frame comprises three sections: a 6-byte Header, a Payload of up to 255 bytes, and a 2-byte CRC.

Figure 13: Mavlink v1 Frame [18].

As in Figure 13, the Header includes the start byte (0xFE), payload length, a message sequence number, system ID, component ID, and message ID. The Payload carries the original sensor values (in hexadecimal format), while the CRC is used to verify data integrity and detect transmission errors. This structure enables robust, consistent communication between nodes in the fire monitoring system.

b) Security Coding using AES 256 in mode ECB

Input: MavLink frame**.**

Output: Ciphertext.

In the Security Coding step, to enhance the protection of MAVLink messages, this thesis applies the AES-256 block encryption algorithm, one of the most powerful and widely adopted encryption standards today. With a 256-bit key length and 14 rounds of transformation, AES-256 provides strong resistance against modern cryptographic attacks. Before using AES-256, PKCS7 padding is used to make a multiple of 16. After padding, the message is divided into fixed-length blocks and encrypted sequentially with AES-256. Additionally, to further confuse the data, this thesis uses the bit reversal, which reverses all the bits in each block.

c) Channel Coding using LDPC 1/2

Input: Despread message.

Output: Channel decode message.

At this stage, LDPC decoding is employed to detect and correct errors caused by wireless transmission. As said in Spreading, a shared parity-check matrix H is applied to the received message to compute the syndrome. If the syndrome vector contains non-zero values, it indicates the presence of errors. The decoder then tries to correct the bit error by finding and flipping the bit related to the most incorrect parity. The syndrome is recalculated by using the calculate syndrome after each bit flip, and this process continues until the syndrome is zero or a maximum of ten iterations is reached. If problems continue after the allowed rounds, the message becomes uncorrectable and is destroyed.

d) Bit scrambling using LFSR

Input: Channel-encoded message.

Output: Scrambled message**.**

In the Bit Scrambling phase, this thesis uses a Fibonacci-type Linear Feedback Shift Register (LFSR) to enhance communication safety and reliability. By generating pseudo-random sequences, the LFSR confuses the signal, making it more difficult to track, detect, and attack during wireless transmission. This step receives an LDPC-encoded message whose length has been doubled by channel coding. Each data block is XORed with the LFSR output, and the same register is shared between the transmitter and receiver for detection purposes.

e) Synchronization

Input: Scrambled message.

Output: Framed message.

After undergoing the processes of frame packing, encryption, channel coding, and spreading, the message becomes significantly confused, making it difficult for the receiver to identify the beginning or structure of the transmission. Therefore, as the final step, two header bytes (0xEB and 0x90) are added after LFSR encoding.

These specific bytes were chosen because 0xEB and 0x90 are commonly detectable in wireless environments. They frequently appear in the radio spectrum,

making them reliable markers. By adding these two bytes at the beginning of the final message, the receiver can more easily synchronize and detect the start of a valid transmission.

In the Encoding Message section, each of these components will be described in detail through corresponding steps, allowing for a clear understanding of how the message is processed and prepared for transmission.

(3) Wireless Transmission (through UART)

Input: Framed message.

Output: Message transmitted.

In the Wireless Transmission stage, the framed message is sent from the STM32 to the HC-12 module using UART communication. The STM32 delivers the data over its UART interface, while the HC-12 module handles modulation and RF transmission. To ensure communication, both the STM32 and HC-12 must be configured with the same baud rate of 115200 and operate on the same channel. These parameters are set via AT commands during initialization. Once configured, the HC-12 broadcasts the message wirelessly, completing the transmission process.

2.2.2. Data Processing at Central Processing Unit

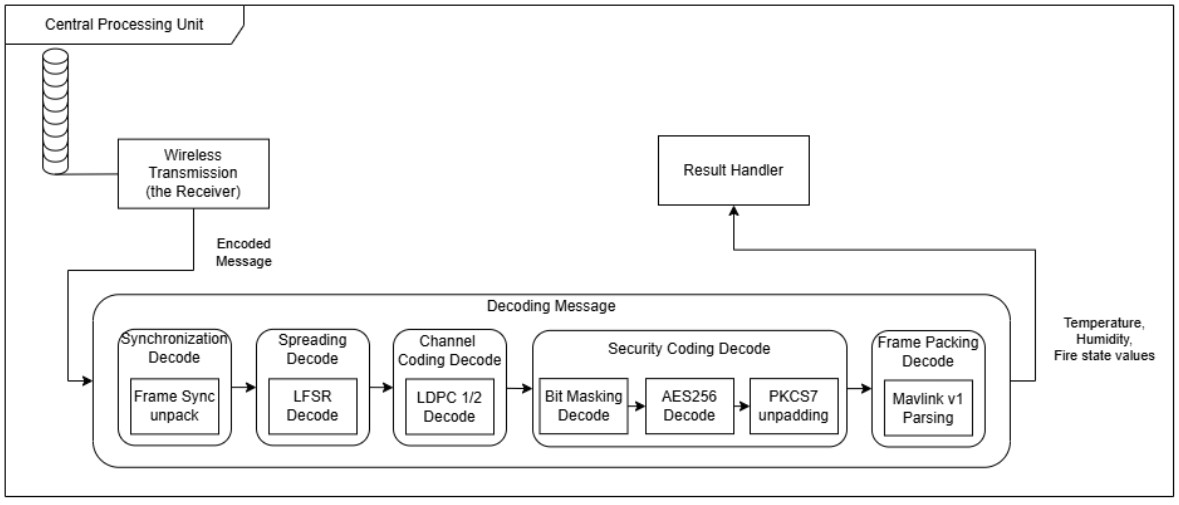


Figure 14: Software Architecture at the Central Processing Unit.

(1) Wireless Receiver

Input: Message transmitted.

Output: Message received.

On the receiver side, the HC-12 wireless module is pre-configured with the same AT command parameters, such as baud rate and channel, as described in the Wireless Transmission section. After the message is successfully transmitted, the Central Processing Unit listens for incoming data through the USB UART.

(2) Decoding Message

a) Synchronization Decode

Input: Message received.

Output: Frame message unpack.

When the message is received through USB UART at the Central Processing Unit, it checks the validity of the message by inspecting the first two bytes. If the first two bytes are 0xEB and 0x90, the message is valid. The synchronization decode step then unpacks the frame by removing these two header bytes, allowing the system to proceed with further processing.

b) Bit Scrambling Decode

Input: Frame message unpack.

Output: Message descrambling.

Because both the transmitter and the receiver use the same register after applying the LFSR Fibonacci, to decode the spreading, XOR each block again with the LFSR Fibonacci output, just as in the LFSR encoding step.

c) Channel Coding Decode

Input: Message descrambling.

Output: Channel decode message.

At this stage, LDPC decoding is employed to detect and correct errors caused by wireless transmission. As said in Spreading, a shared parity – check matrix H is

applied to the received message to compute the syndrome. If the syndrome vector contains non-zero values, it indicates the presence of errors. The decoder then tries to correct the bit error by finding and flipping the bit related to the most incorrect parity. The syndrome is recalculated by using the calculate syndrome after each bit flip, and this process continues until the syndrome is zero or a maximum of ten iterations is reached. If problems continue after the allowed rounds, the message becomes uncorrectable and is destroyed.

d) Security Coding Decode

Input: Channel decode message.

Output: Plaintext.

After LDPC decoding, if the syndrome vector contains only zeros, the message is considered error-free and proceeds to the Security Coding Decode stage. As mentioned earlier, bit masking in this system involves a simple bit-reversal operation during encoding, so decoding applies bit reversal again to restore the original bit order, preparing the message for AES-256 decryption and PKCS7 unpadding.

In the AES-256 decryption phase, the same key used during encryption is reused, but the round keys are applied in reverse order, from the last to the first, across 14 decryption rounds. Once decryption is complete, the message enters the PKCS7 unpadding stage. Here, the decoder checks the last byte to determine the number of padding bytes added. For example, if the last byte is 0x03, the decoder verifies whether the final three bytes all equal 0x03. If so, they are removed; otherwise, the message is invalid. After successful unpadding, the original MAVLink frame is fully recovered.

e) Frame Unpacking

Input: Plaintext.

Output: Temperature, Humidity, Fire state data.

In this final stage, the system verifies whether the plaintext corresponds to a valid MAVLink frame. If the first byte is 0xFE, it indicates the start of a MAVLink packet.

The decoder then checks the LEN byte to confirm that the payload length matches the actual size, and verifies the SEQ byte to ensure message order – an important factor for logging and detecting missing or out-of-sequence packets. The SYS ID and COMP ID fields help identify which system and component sent the message. Most importantly, the CRC is recalculated from byte 2 onward (not including the start byte 0xFE) and compared with the received CRC. If they match, the frame is valid; otherwise, it is discarded. Once validated, the payload is extracted to obtain the final output, which includes temperature, humidity, and fire status data.

2.2.3. Software development at Central Processing Unit

This section presents the implementation of the software system on a web platform, aiming to improve user interaction, usability, real-time monitoring, and overall system management.

a) Software Architecture

Software Design

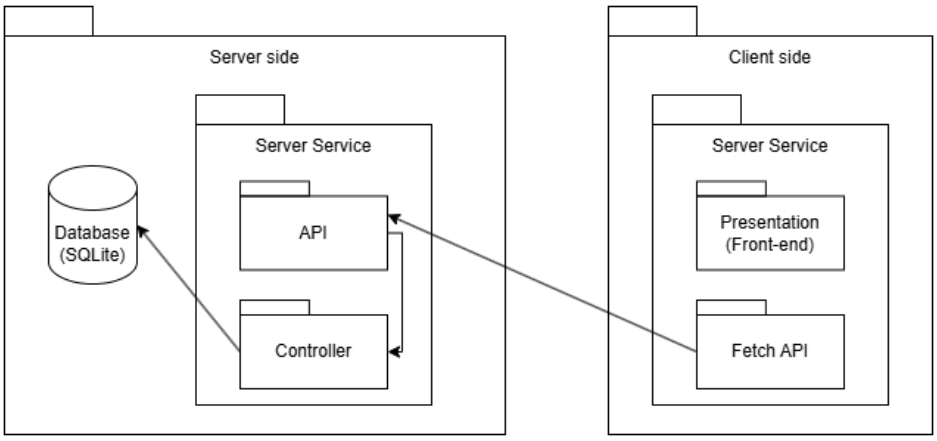


Figure 15: Software Architecture diagram.

The system architecture comprises several key components that work smoothly between the server and client sides. On the server side, a **database** serves as the central repository for user data, activity logs, fire alerts, and settings. A **server service** manages client requests and data access, while **APIs** act as communication bridges, handling requests, executing logic, and delivering functionalities. **Controllers** implement business logic, coordinating data flow between APIs and the database. On

the client side, the **user interface**, built with HTML, CSS, and JavaScript, enables dynamic interaction by calling server APIs. Using the **Fetch API**, the client can send requests, allowing smooth, real-time updates without reloading the entire page, thereby enhancing performance and user experience.

Tool & Technicals

The software system applies the MVC (Model – View – Controller) architecture, including Controller, Service, Repository, and Model layers for a clean and maintainable structure. The backend is developed using Flask, a lightweight Python framework that is ideal for interacting with GPIO and communicating through UART with the Edge Processor. SQLite is chosen as the database for its simplicity and seamless integration with Flask, making it suitable for embedded applications.

The frontend is built with HTML, CSS, and JavaScript, providing a responsive and interactive user interface. APIs bridge the frontend with the backend, enabling real-time data fetching and updates. Because the system lacks a continuously online server, a VPN solution (TailScale) is employed to allow remote access. TailScale offers secure peer-to-peer connectivity, making it a suitable choice for IoT systems that require remote monitoring and control without dedicated server infrastructure.

b) Requirement analysis

Functional requirement

The following table outlines the functional requirements of the system, based on the roles of the two main actors: Admin and User.

ID	Functionality		Description	Roles
1	Maintain User Information	Sign Up	Register a new account	Admin, User
		Password Recovery	Recover forgotten password	
		Login	Log in to the system	
		Edit Personal Information	Edit username, password, email, and phone number	
	Maintain	View Log & Fire	View daily logs and fire event history	

ID	Functionality		Description	Roles
2	Dashboard	Events		
		Real-time Data Monitoring	View real-time sensor data on the dashboard	
3	Export logs		Export the data for the days	
			Export fire events	
4	User Management	Create a new user	Add user accounts	Admin
		Delete exist user	Remove user accounts	
		Approval Sign Up queue	Approval Sign Up queue	
5	Maintain Alert & Threshold	Alert Deactivation & Notification	Send a system-wide alert if a fire is detected or the alert is deactivated	
6		Threshold Settings	Modify temperature and humidity thresholds and notify all users	

Table 1: Functional requirement.

Non-functional requirements

- **Performance:** The system must show sensor data (temperature and humidity) on the dashboard within 2 seconds after receiving it from the STM32 node.
- **Wireless Transmission Reliability:** Since the system uses HC-12 wireless modules, it must detect and handle possible transmission errors to keep data accurate and complete.
- **Data Processing Efficiency:** The process of data from node to central processing unit - encode and send sensor data than decode each message in under 1 second.
- **Data Security:** All data sent wirelessly must be encrypted to prevent tampering. User passwords must be securely hashed using reliable algorithms.
- **Email Alert Delivery:** The system must send email alerts reliably, with at least a 98% success rate when a fire is detected or thresholds are exceeded.
- **Scalability:** The system should handle multiple sensor nodes at the same time without slowing down or becoming unresponsive.

- **Cross-Platform Compatibility:** The web application must work well on major browsers like Chrome, Firefox, Microsoft Edge, and others.
- **Reliability:** Even though wireless transmission may sometimes fail, the system should remain reliable by using error checking or upgrading the wireless module if needed.
- **Usability:** The system should be easy to use. Users should be able to view sensor data, receive alerts, and export logs without needing technical knowledge.

c) System analysis and design

Use case diagram

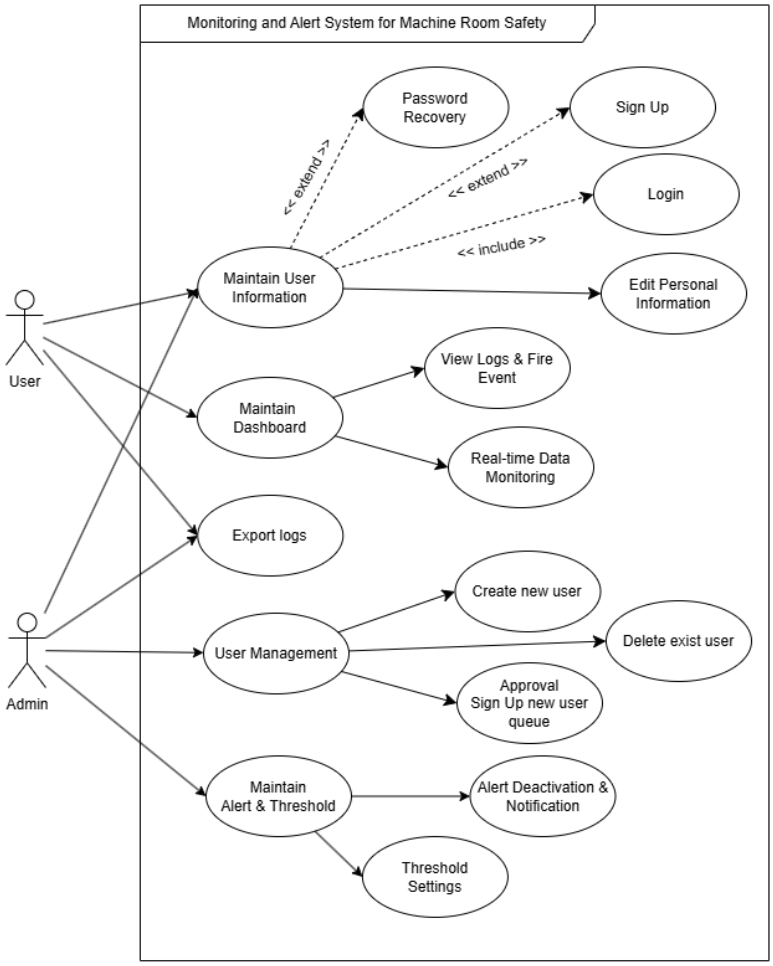


Figure 16: Use case diagram.

The use case diagram show two main actors of the system:

- User: A regular user who can sign up, log in, update personal information, monitor real-time data, receive alerts, and export logs.

- Admin: A superior user who manages accounts, approves sign-up user requests, sets alert thresholds, and sends system notifications.

The system's core functionalities are grouped into Maintain User Information, Maintain Dashboard, Export logs, User management and Maintain Alert & Threshold.

Sequence Diagrams

Export Logs

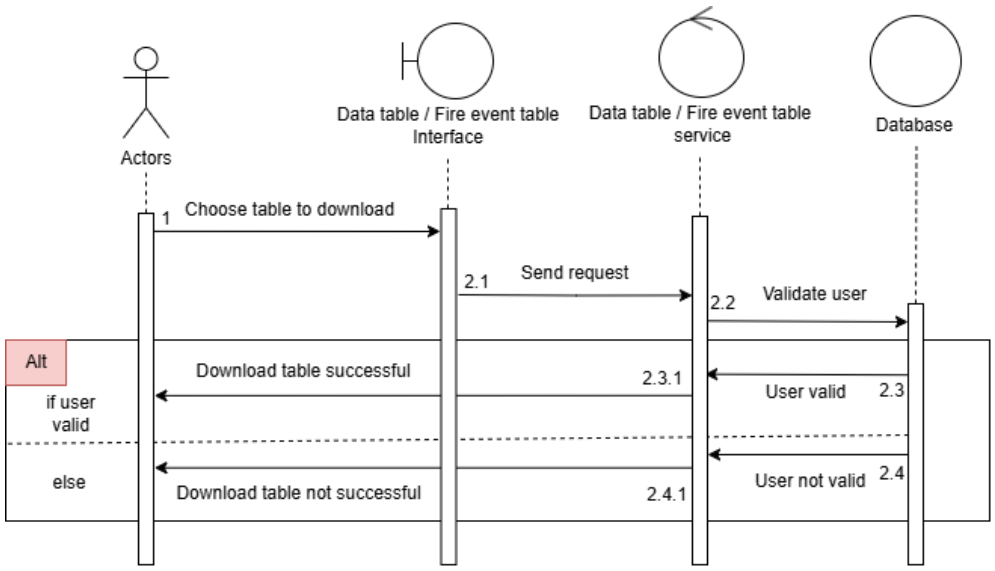


Figure 17: Export Logs Sequence Diagram.

This sequence diagram illustrates the process of exporting tabular data upon user request. The user can choose to export either daily monitoring data or fire event logs. The request is initiated via the client interface, processed through a controller, and handled by the service layer. Depending on the type selected, the system queries the corresponding data set from the SQLite database, processes it into a downloadable format (e.g., CSV), and returns the result to the client. This functionality provides flexible data access and supports user needs for reporting, analysis, or archival.

Threshold Settings

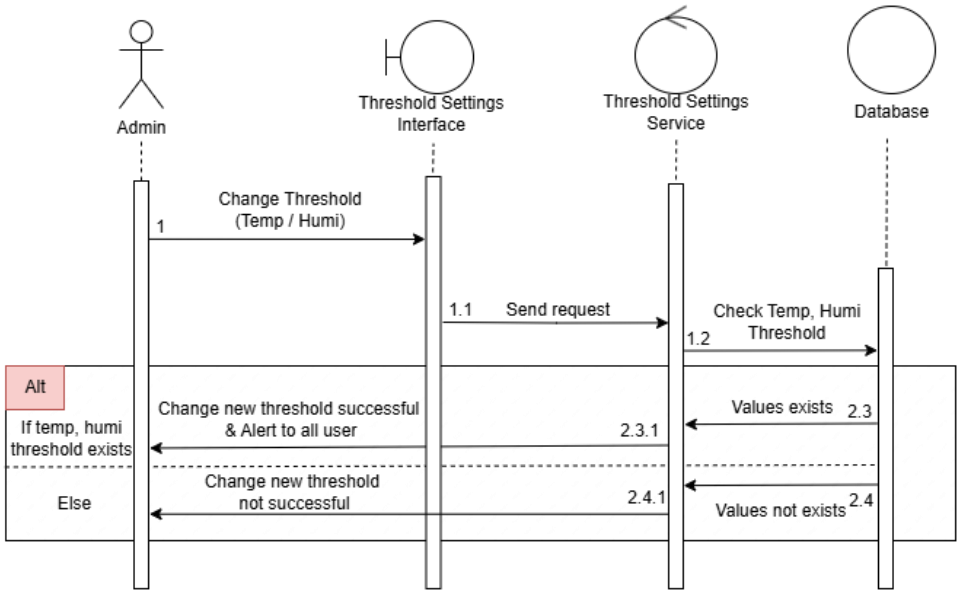


Figure 18: Threshold Setting Sequence Diagram.

This function allows the Admin to adjust the threshold values used for triggering alerts to users. The threshold includes two parameters: temperature and humidity.

Although the DHT11 sensor can theoretically return values up to 50°C and 20% humidity, using values too close to these limits may reduce the sensor's reliability and negatively impact the alert system. Therefore, in this thesis, the thresholds are intentionally set to 45°C for temperature and 40% for humidity to ensure both accurate alerts and the long-term safety of the sensor.

Deactivate Alert

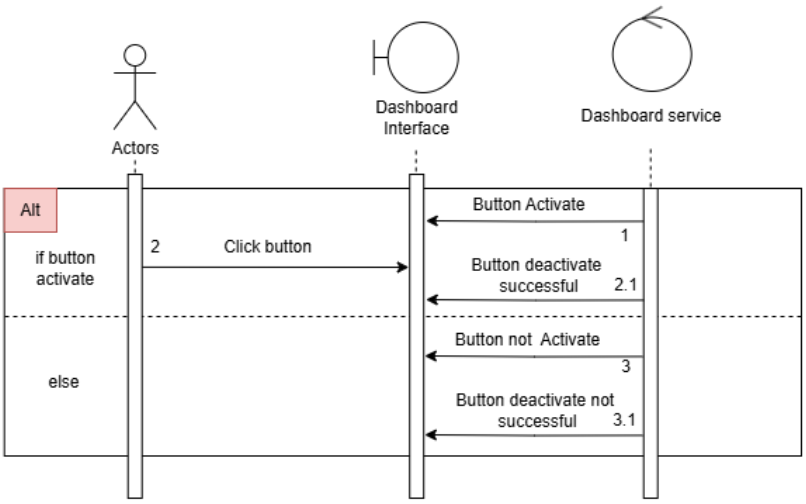


Figure 19: Deactivate Alert Sequence Diagram.

When an alert is triggered in the System Alert section (detailed in the section below), a button to dismiss the alert will appear on the Admin's dashboard. The Admin is authorized to deactivate this button, which records the end time of the fire or alert event at the time. Once the button is deactivated, an email notification will be sent to inform users that the event has ended.

2.2.4. System Alert

This is an essential part of the system that handles user notifications. This section contains warnings for Fuzzy Logic, Temperature Threshold, Humidity Threshold, and Both Temperature and Humidity Threshold.

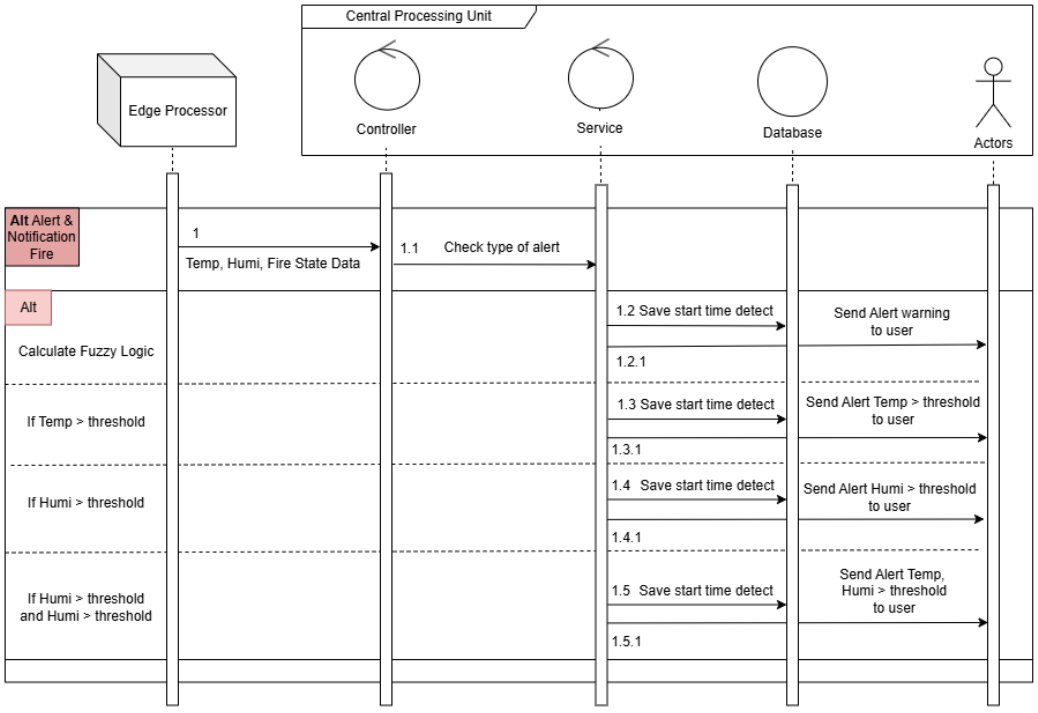


Figure 20: Alert Sequence Diagram.

a) <u>Temperature & Humidity Threshold Alerts</u>

In this section, the system determines fire alarms based on the sensor's capabilities. The DHT11 sensor operates within a temperature range of 0-50°C and a humidity range of 20-90%. However, to prevent hardware damage and enable early fire detection, this thesis does not wait for the maximum temperature to activate alarms. Instead, an alert occurs when the temperature reaches 40°C and the humidity drops below 40%. In both cases, an email notification is sent to all users.

b) ***Combined Threshold Alert for Temperature and Humidity***

When both conditions are occure together (temperature > 40°C and humidity < 40%), the system sends an alert to all users depending on a specific threshold. This combined threshold alert provides as an addition to the Fuzzy Logic-based fire detection system (which will be explained later). Although Fuzzy Logic may predict fire danger using predetermined rules, it may occasionally generate erroneous or imprecise findings. As a result, this dual-threshold alert mechanism acts as an extra precaution to improve the alert system's dependability in critical situations.

c) ***Fuzzy Logic***

Fuzzy Logic [19] [20]is a data processing approach based on approximate reasoning, allowing values to range between 0 and 1, rather than being completely binary (true = 1 or false = 0) as in traditional logic systems. Instead of assigning a crisp value like “Cold” (0) or “Hot” (1), fuzzy logic introduces transitional states—for example, a value in between, such as “Warm,” is represented by a degree of membership between 0 and 1.

As illustrated in the figure below, this flexible representation enables the system to evaluate the changes and generate risk levels ranging from low to high. Consequently, it allows for more nuanced alerts and earlier warnings, rather than waiting until a hard threshold is reached.

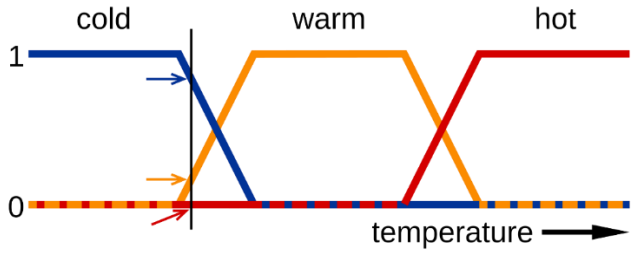


Figure 21: Fuzzy Membership Functions for Water State (Cold, Warm, Hot).

There are various types of Fuzzy Logic systems, including Rule-Based, as well as more advanced variants such as Learning-Based, Adaptive, and other forms like Fuzzy Weighted Sum.

However, this thesis adopts the Rule-Based Fuzzy Logic approach, which is also the most fundamental and representative form of fuzzy reasoning. The rule-based model consists of three main steps: Fuzzification – transform all input values into fuzzy membership functions.. Rule Evaluation – apply all applicable rules from the rule base to compute the fuzzy output sets. Defuzzification – convert the fuzzy output sets into crisp output values. This approach is chosen for its simplicity, interpretability, and effectiveness in modeling human-like reasoning in fire detection scenarios.

Fuzzification:

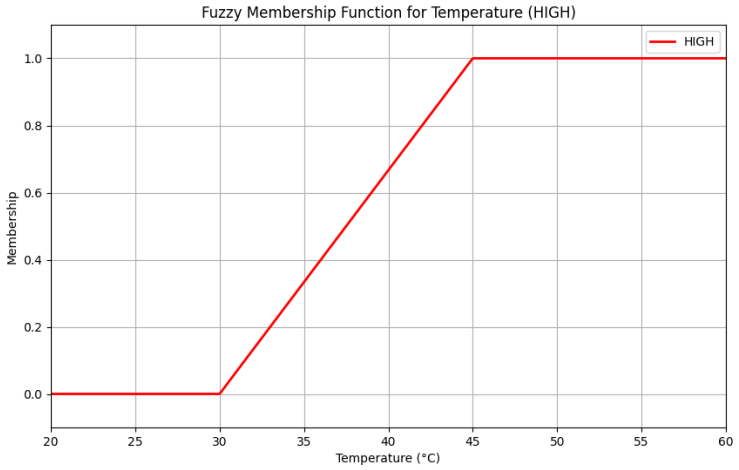


Figure 22: Membership of Temperature.

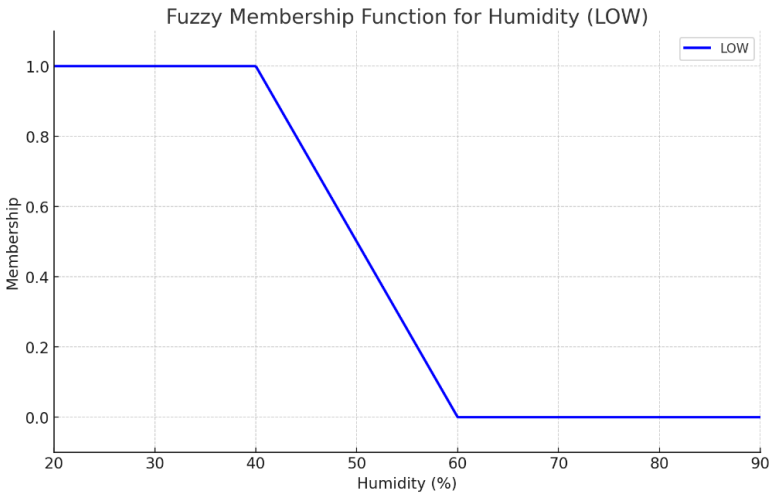


Figure 23: Membership of Humidity

To simplify the initial development process, this thesis only uses two major membership functions: Temperature High and Humidity Low. Limiting the model to

these two levels, rather than adding levels such as Low, Medium, or High for each variable, helps to reduce system complexity and prevent rule explosion. Using a larger number of fuzzy rules adds complexity, potentially leading to inconsistencies and decreased accuracy. The system's primary purpose is to detect fires quickly and reliably. As a result, the model focuses on the most important conditions commonly connected with the fire:

- High temperature: Fire causes an increase in ambient temperature. In this model, temperature is considered to gradually become “High” starting at 30°C, reaching “Fire” (1) at 45°C and above.
- Low humidity: A dry environment provides ideal conditions for fire to spread. So, humidity below 40% is considered an important risk, while levels above 60% are as safe.

This simplification enables a more efficient fuzzy rule base design without significantly compromising performance. Although the model does not account for intermediate states like Medium, it brings a high effective value to concentrate on detecting clearly high-risk fire conditions.

Rule Evaluation:

Rule	Fire status	Temperature	Humidity	Output Level
R1	Yes (1)	High (> 0.5)	Low (> 0.5)	VERY HIGH
R2	Yes (1)	High (> 0.5)	Not Low (≤ 0.5)	HIGH
R3	Yes (1)	Not High (≤ 0.5)	Low (> 0.5)	MEDIUM
R4	No (0)	High (> 0.5)	Low (> 0.5)	HIGH
R5	No (0)	Not High (≤ 0.5)	Not Low (≤ 0.5)	LOW

Table 2: Rule of Temperature, Humidity, Fire status.

The fuzzy system uses five rules to evaluate fire risk based on temperature, humidity, and fire detection status. Each rule is triggered according to the degree of membership in “high temperature” and “low humidity”. Rules with fire detected (fire state = 1) prefer higher alert levels, especially when both temperatures are high and humidity is low. Rules without fire serve as early warnings based on bad environmental conditions. Each rule contributes a weighted score based on severity,

which is averaged to produce the final fuzzy risk level. This approach ensures flexible and realistic decision-making beyond fixed thresholds.

Defuzzification:

After evaluating all activated fuzzy rules, the system applies a weighted average method to perform defuzzification. Each output label (LOW, MEDIUM, HIGH, VERY HIGH) is assigned a numeric weight reflecting its severity level (3, 5, 7, 9, respectively). The final fuzzy score is calculated by averaging these weights based on rule strength.

This crisp score is then used to analyze the fire risk level:

- A score above 7 is considered a high probability of fire, triggering an immediate fire alert.
- A score between 4 and 7 signals a potential fire risk, prompting early warnings.
- A score below or equal to 4 suggests the environment is safe.

This approach ensures that the system reacts appropriately and equally to environmental conditions, enhancing its reliability in early fire detection scenarios.

2.3. Conclusions

In the Hardware System Architecture section, the thesis introduced each component in detail, starting from the sensor nodes and edge processor to the central processing unit (CPU). The role of each hardware element in collecting data and transmission was clearly defined. Following that, the Software & Firmware System Architecture section focused on how the sensor nodes collect environmental data (e.g., temperature, humidity, fire state), then encode and transmit it through wireless communication (HC-12) to the central unit. At the CPU, the encoded packets are decoded and parsed to extract meaningful values such as temperature, humidity, and fire status. A web-based interface was developed using a Flask web server to enable real-time data monitoring. The system displays sensor data with minimal delay, allowing users to observe temperature and humidity trends in near real-time through a user-friendly website. Moreover, the web system include essential user functionalities,

such as login, signup, and profile management, among others. Users can also edit threshold values for alert conditions and update their personal information through the interface. Regarding the alert mechanism, the thesis implements a Fuzzy Logic-based algorithm to determine fire risk levels. All alert-related information is automatically sent to users via email, ensuring timely notification. In summary, Chapter 2 has provided a detailed and cohesive overview of the entire Fire Monitoring and Alert System, from sensor data acquisition and transmission to real-time visualization and intelligent alert generation, highlighting both the hardware architecture and software functionality.

Chapter 3. Results and Evaluations

If Chapter 2 presented a detailed design of the system — from both hardware and software aspects – covering data collection from sensors, data encoding, transmission, decoding, processing, and real-time visualization, then Chapter 3 *Results and Evaluations* focuses on analyzing experimental outcomes to verify the effectiveness and reliability of the IoT-based fire monitoring and alert system. This chapter is divided into three main sections: Section 3.1 Implementation: Demonstrates the system hardware to software, all features for interacting with the user, and the system alert warning user, both of which have been successfully implemented and are operating smoothly. Section 3.2 Reliability and Error Rates: Evaluates the system's reliability through analysis of error rates, packet loss, and the likelihood of false fire detections or failure to detect hazardous environmental conditions. This section also measures the end-to-end delay from sensors to the user interface and alert notifications, to assess the system's responsiveness in real time. Section 3.3 – Conclusion: Summarizes the findings presented throughout Chapter 3.

3.1. Implementation

Sensor Node Side

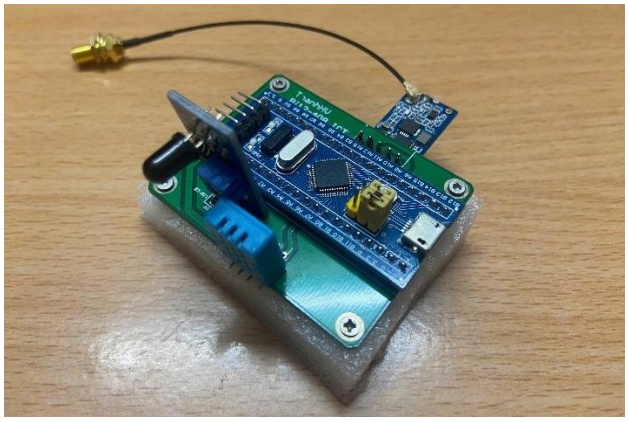


Figure 24: Edge Processor with various sensors and a wireless module.

First, on the node side, this part *Sensor Node Side* implements a printed circuit board (PCB). The pins of the STM32F103C8T6 are soldered to the DHT11, KY-026, and HC-12 modules to form a complete and functional sensor node.

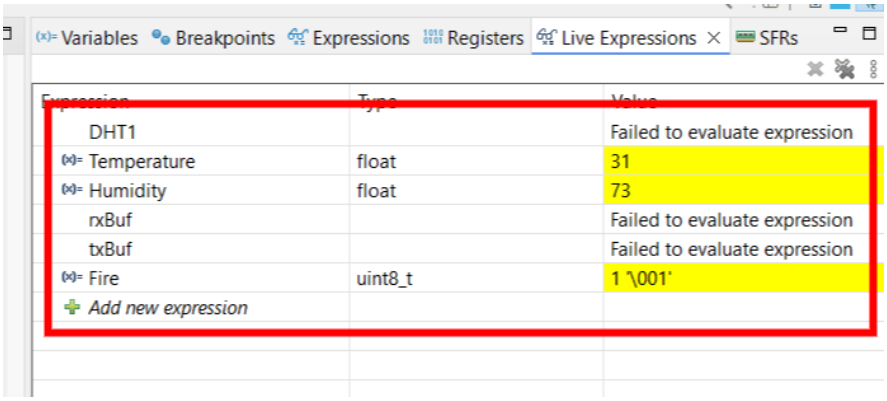


Figure 25: Debug STM32 IDE Temp, Humi, Fire state value.

Next, for the firmware, the system utilizes STM32CubeIDE to implement code that collects environmental data. These sensors then render the output values for temperature, humidity, and fire state, as previously described. And, the fire state is by default t defined as 1 when no flame is detected.

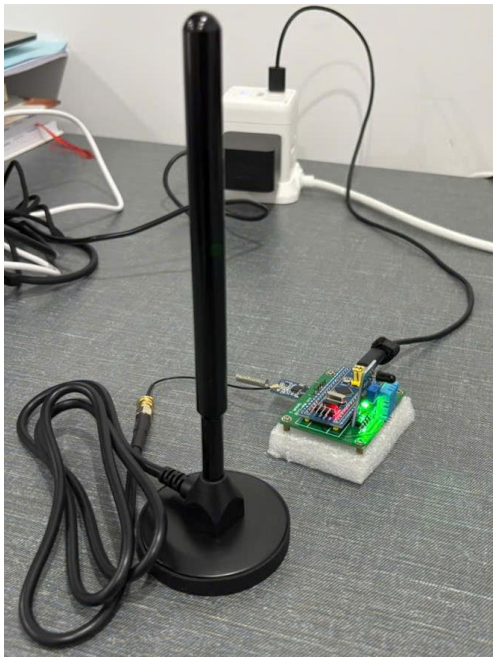


Figure 26: Sensor Node.

Finally, after the sensors collect the data, it continues to process that data, which is encoded at the edge processor (STM32F103C8T6). The HC-12 module is then connected to an antenna to improve long-range signal transmission, enabling reliable communication with the Central Processing Unit (CPU) that is awaiting incoming data.

Central Processing Unit Side

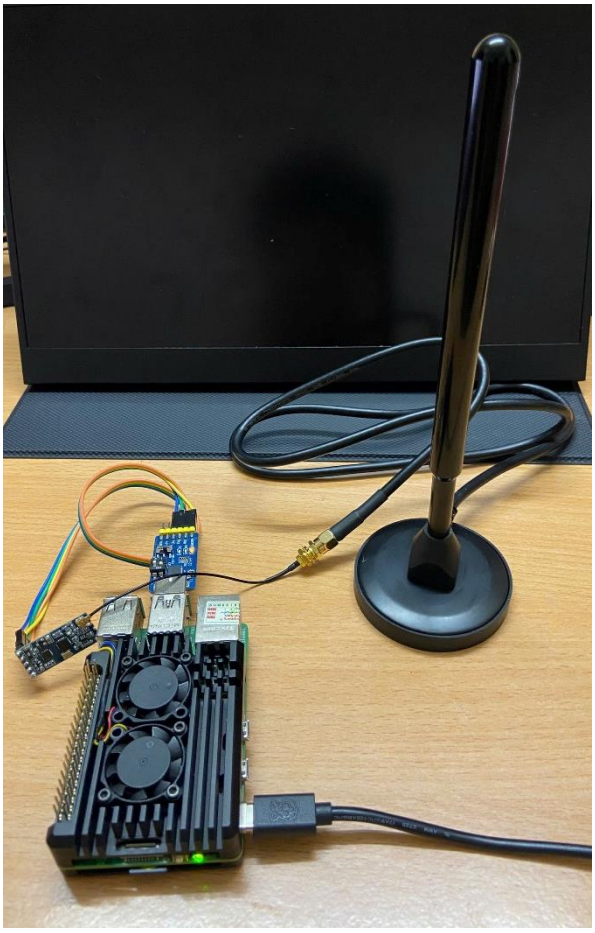


Figure 27: Central Processing Unit.

After receiving data from the node side, at the Central Processing Side, it is decoded and parsed to extract the values for temperature, humidity, and fire state. The extracted data is then displayed in real-time on a user-friendly web-based platform, which also includes various functionalities as described in the Software Development at the Central Processing Unit section in Chapter 2 (see Figure 28 below).

Figure 28: Software Interface.

Figure 29: The IoT-based monitoring and alert system.

Finally, as shown in Figure 29, the system has been fully implemented from hardware to software. All features in the Section *Software & Firmware System Architecture* that feature of the system is deployment, which enables smooth interaction with users through a touchscreen monitor. Also, system alerts from the system to the user are implemented and working promptly.

3.2. Reliability and Error Rates

For this part, “Reliability and Error Rates”, this section focuses on evaluating the reliability of environmental data collection, specifically examining packet loss during wireless transmission and the integrity of messages received by the CPU for

processing and user display. In addition, user interaction delays, such as response times from input to system feedback, are also analyzed in this section.

Firstly, Temperature and humidity data were collected in real time using the DHT11 sensor with high consistency and almost no data loss. The sensor responded correctly to changes in the environment. The KY-026 flame sensor also worked reliably, detecting heat from fire sources without delay. All data was packaged and encoded properly, then sent wirelessly to the receiver with a very low packet loss rate, thanks to channel encoding that improved data recovery. The Central Processing Unit received and decoded the data successfully.

After decoding, temperature, humidity, and fire state were shown in real time on the user interface, with no missing or delayed readings. The “Deactivate Alert” button responded immediately when a high temperature was detected, helping the admin act quickly. These results confirm the system’s reliability, fast response, and stable data transmission. With VPN (TailScale) integration, remote users could also access the system smoothly, with almost zero delay.

3.3. Conclusion

In summary, the system has been successfully implemented from hardware to software, demonstrating reliable operation across all components. The sensor node accurately collects environmental data, encodes and transmits it wirelessly with minimal packet loss, while the central processing unit decodes and displays the data in real time through a user-friendly interface. The system responds promptly to fire-related events, and all key features including alert functions and remote access via VPN are operate smoothly. These results confirm the system’s effectiveness, reliability, and real-time responsiveness, validating its applicability in practical fire monitoring scenarios such as machine rooms.

Conclusions and Perspective

In recent years, significant research and development have been conducted into real-time IoT-based fire monitoring and alarm systems. Each new implementation adds to its continuous progress by focusing on necessary design concepts, including accuracy, speed, wide coverage, and user accessibility. These characteristics are critical to any good fire detection system. This thesis investigates a multi-node sensor network technique that significantly improves coverage for monitoring and management, particularly in machine rooms. By using wireless communication technologies, the system ensures low-latency data delivery while reducing cable complexity.

Furthermore, a web-based interface was developed to improve user interaction with the system. Real-time data from sensor nodes is displayed clearly, enabling responsive supervision. The architectural design separates detection from processing by placing the CPU in a secure location, while the fire-detecting nodes are installed in high-risk zones. This structure increases safety without sacrificing detection efficiency. With this approach, the system demonstrates high potential for real-world deployment, particularly in environments where fire monitoring requires reliable and timely information.

Although fire monitoring and alarm systems are not new technologies, each innovation serves as a stepping stone toward more advanced and efficient solutions. The current prototype has many functionalities, but the system always stands at 90% so there are many things to improve. Future versions aim to integrate a broader range of sensors, such as gas, smoke, vibration, and environmental modules, into each node, and expand the system to multiple distributed nodes to cover larger areas and enhance monitoring capabilities. Additionally, improvements in data packaging and encoding algorithms are planned to enhance wireless communication, particularly in high-interference environments. While the system currently uses VPN (TailScale) for remote access, it lacks the scalability and reliability of a dedicated server. So that future updates will include an always-on server to improve system stability and

accessibility. Moreover, the integration of AI-based fire detection and prediction models is expected to enhance accuracy and response time, alongside the deployment of 360-degree thermal cameras capable of operating in complete darkness. To optimize performance in embedded systems, the software is currently written in Python; in the future, the language used may be migrated to lower-level languages like C or C++. Regarding power resilience, a backup battery will be added for the CPU, ensuring uninterrupted operation during power outages, similar to the battery of the node; the power backup will cover critical scenarios. The device casing will also be redesigned using flame-retardant materials to increase safety. Finally, the system needs to use an application to offer a more user-friendly interface, maybe such as using CMake, enabling easier installation and interaction for end users.

References

[1] "Smart Home," no. https://www.thebestsmart.homes/2022/09/smart-home-iot-smart-home-iot-solution.html.

[2] Pradeep, Aneesh; Latifov, Akmaljon; Yodgorov, Ahborkhuja; Mahkamjonkhojizoda, Nurislombek, "Hazard Detection using custom ESP32 Microcontroller and LoRa," *IEEE,* no. 26 May 2023, 2023.

[3] N. I. M. E. N. M. Z. a. K. S. S. K. M. N. N N Mahzan, "Design of an Arduino-based home fire alarm system with GSM module," *Journal of Physics: Conference Series,* vol. 1019, 2917.

[4] M. S. B. Bahrudin, "Development of Fire Alarm System using Raspberry Pi and Arduino Uno," *2013 International Conference on Electrical, Electronics and System Engineering (ICEESE),* 2013.

[5] N. I. M. E. N. M. Z. a. K. S. S. K. M. N. N N Mahzan, "Design of an Arduino-based home fire alarm system with GSM module," *Journal of Physics: Conference Series,* vol. 1019, 2017.

[6] H. Y. ·. J. L. ·. L. L. ·. Z. J. ·. Y. L. ·. D. Zhou, "Development of Four Rotor Fire Extinguishing System for Synchronized Monitoring of Air and Ground for Fire Fighting," *ICIRA,* p. 267–278, 2019.

[7] N. Komalapati, V. C. Yarra, L. A. Vyas, Kancharla and T. N. Shankar, "Smart Fire Detection and Surveillance System Using IOT," *IEEE,* 2021.

[8] A. E. H. 1. A. S. S. 1. K. 2. B. A. 1. a. I. C. y Kuldoshbay Avazov 1, "Forest Fire Detection and Notification Method Based on AI and IoT Approaches," *Future Internet (ISSN: 1999 - 5903),* vol. 15, no. 2, 2023.

[9] L. WHITE and R. AJAX, "Improved Fire Detection and Alarm Systems.," 2025.

[10] M. S. B. Bahrudin, R. A. Kassim and N. Buniyamin, "IEEE," *2013 International Conference on Electrical, Electronics and System Engineering (ICEESE),* 2013 .

[11] "SENSOR KIT X40," [Online]. Available: https://sensorkit.joy-it.net/en/sensors/ky-026.

[12] "ASAIR SENSOR," [Online]. Available: https://asairsensors.com/product/dht11-sensor/.

[13] "ePanorama.net," [Online]. Available: https://www.epanorama.net/blog/2022/06/18/introduction-to-the-stm32-blue-pill-stm32duino-and-other-stm32-boards/.

[14] "HC Information Technology Co., Ltd.," [Online]. Available: https://www.hc01.com/downloads/HC-12%20english%20datasheets.pdf.

[15] "HSHOP," [Online]. Available: https://hshop.vn/anten-433mhz-35dbi-sma-duc.

[16] "The Register," [Online]. Available: https://www.theregister.com/2023/09/28/raspberry_pi_5_revealed/?td=amp-keepreading.

[17] "ASUS," [Online]. Available: https://www.asus.com/displays-desktops/monitors/zenscreen/.

[18] "Ardupilot," [Online]. Available: https://ardupilot.org/dev/docs/mavlink-basics.html.

[19] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Fuzzy_logic.

[20] P. Bolourchi and S. Uysal, "Forest Fire Detection in Wireless Sensor Network Using Fuzzy Logic," *IEEE,* 2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks.

Appendix

List of Figure Appendix:

Login

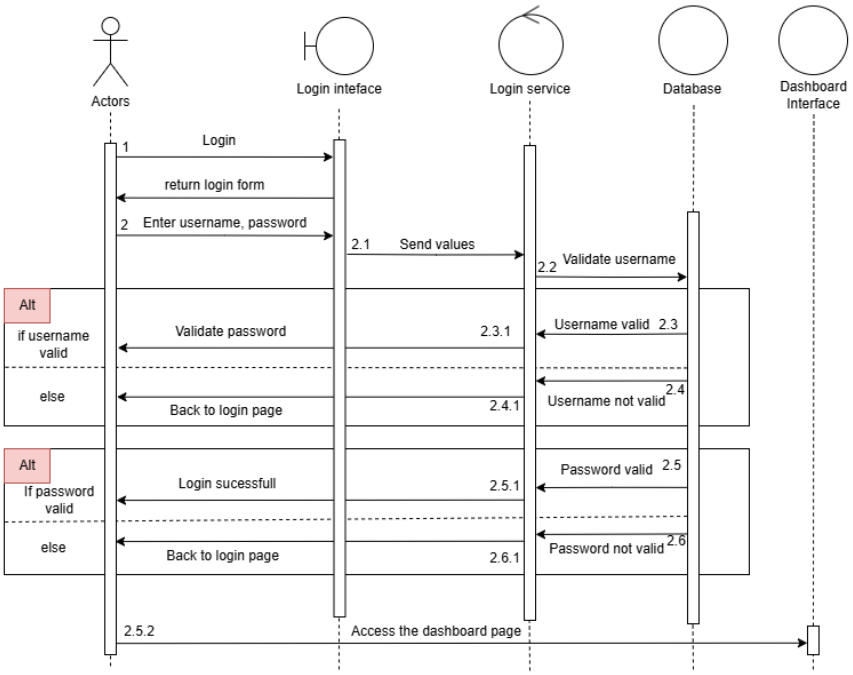


Figure 30: Login Sequence Diagram.

When the user accesses the website, the server automatically redirects them to the Login Interface. At this interface, a login form is displayed with two input fields: *Username* and *Password*. Once the user fills in both fields, the request is handled by the Controller, which forwards the login information to the Authentication Service. The service first verifies the validity of the *Username*. If the *Username* exists, it proceeds to validate the *Password*. If the *Password* is incorrect, the service returns an alert message to the user and redirects them back to the login form. If the *Password* is correct, the user is successfully authenticated and redirected to the *Dashboard Page* for further interaction with the system.

This sequence ensures that only legitimate users receive access while providing feedback on incorrect login attempts.

Sign Up

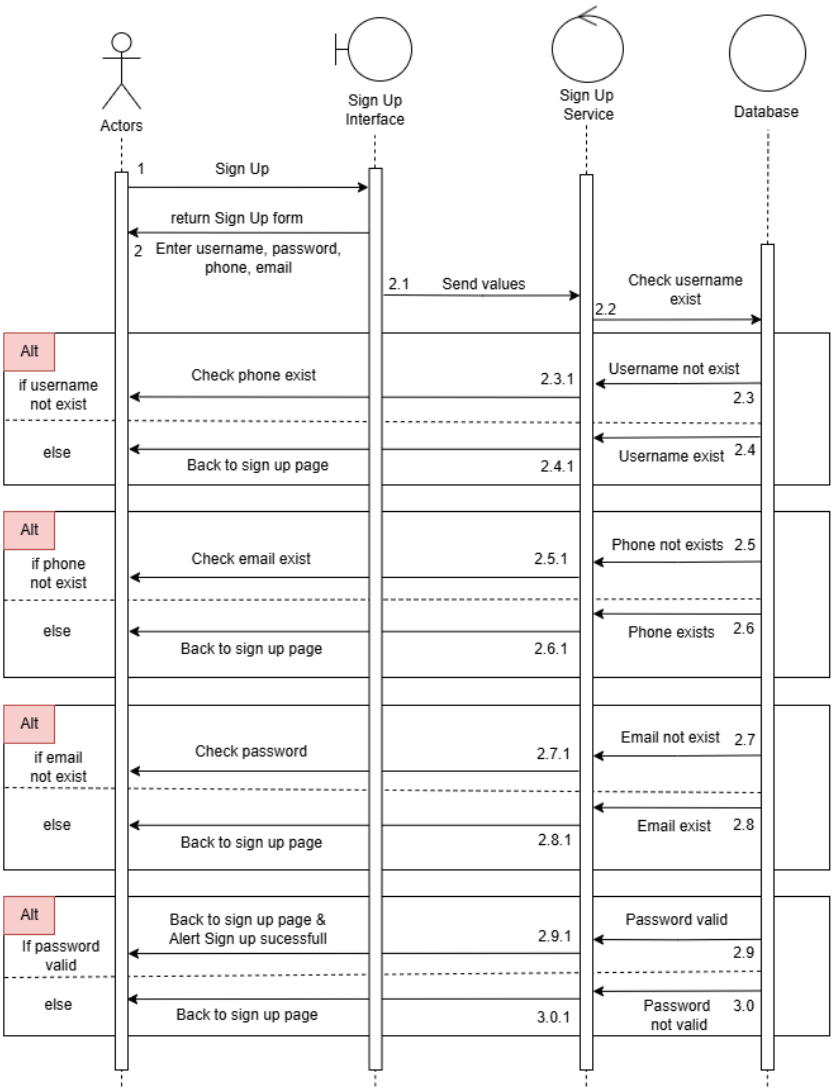


Figure 31: Sign Up Sequence Diagram.

If the user does not have an account, they can select the Sign Up option. The Controller will then redirect them to the Sign Up page, where a registration form is displayed with four fields: Username, Password, Phone Number, and Email.

The user is required to fill in all four fields. Upon submission, the Controller forwards the input data to the Service layer for validation. The service performs multiple checks, including: Ensuring the phone number contains only numeric characters. Verifying the email follows a valid email format. Checking whether the entered username, phone number, or email already exists in the system, either in the current user database or in the sign-up queue.

If all inputs are valid and do not conflict with existing user information, the sign-up request is considered successful, and a confirmation alert is displayed to the user. If any input is invalid or already in use, error messages are returned and shown as alerts to the user in Sign Up page.

Password Recovery

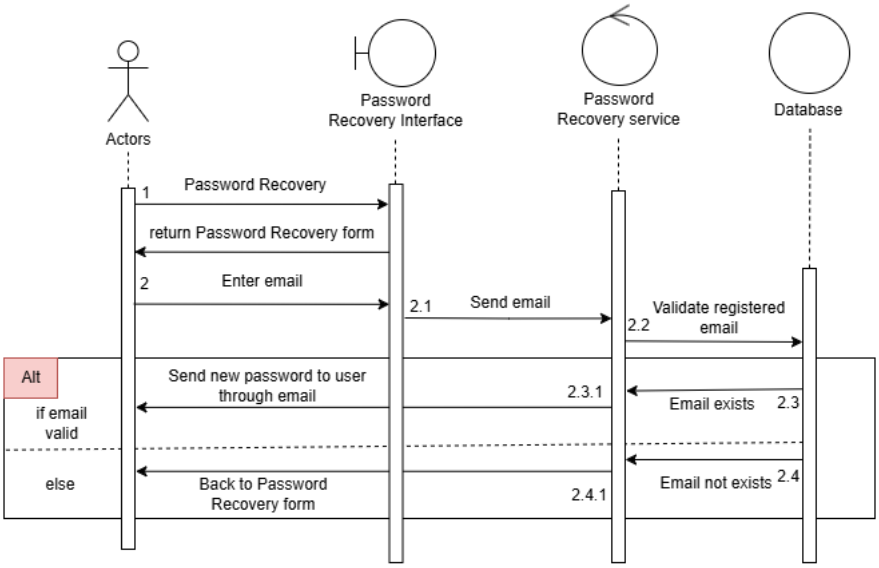


Figure 32: Password Recovery Sequence Diagram.

This sequence diagram show the password recovery process in the system. The user enters their email to request a password reset. The system verifies the email and, if valid, generates a new password. The new password is then sent to the user's email via the mail service. If the entered email is invalid or not found, the system alert the user that email not exists.

Create new user

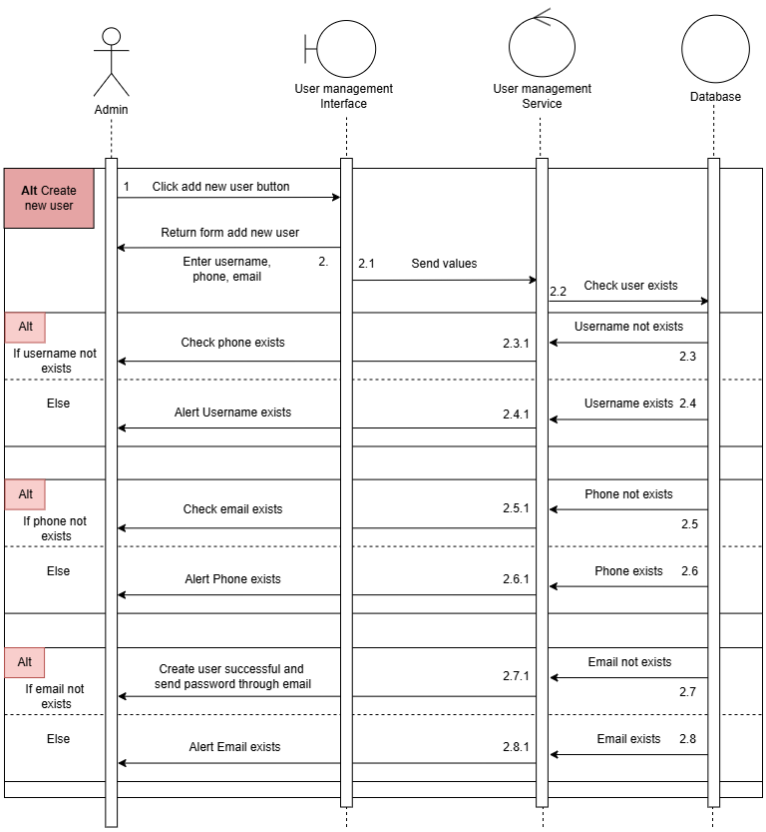


Figure 33: Create New User Sequence Diagram.

This sequence diagram shows the process of an admin creating a new user. The admin fills in three fields: **username**, **phone number**, and **email**, then submits the form to the service for validation.

If all three fields are valid, the system generates a random password and sends it to the user's email. If any of the fields are invalid, the system returns a specific alert message, as shown in the diagram.

Delete user

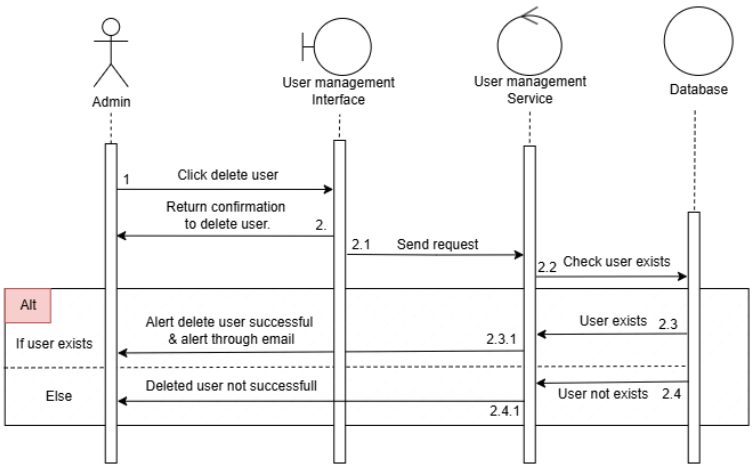
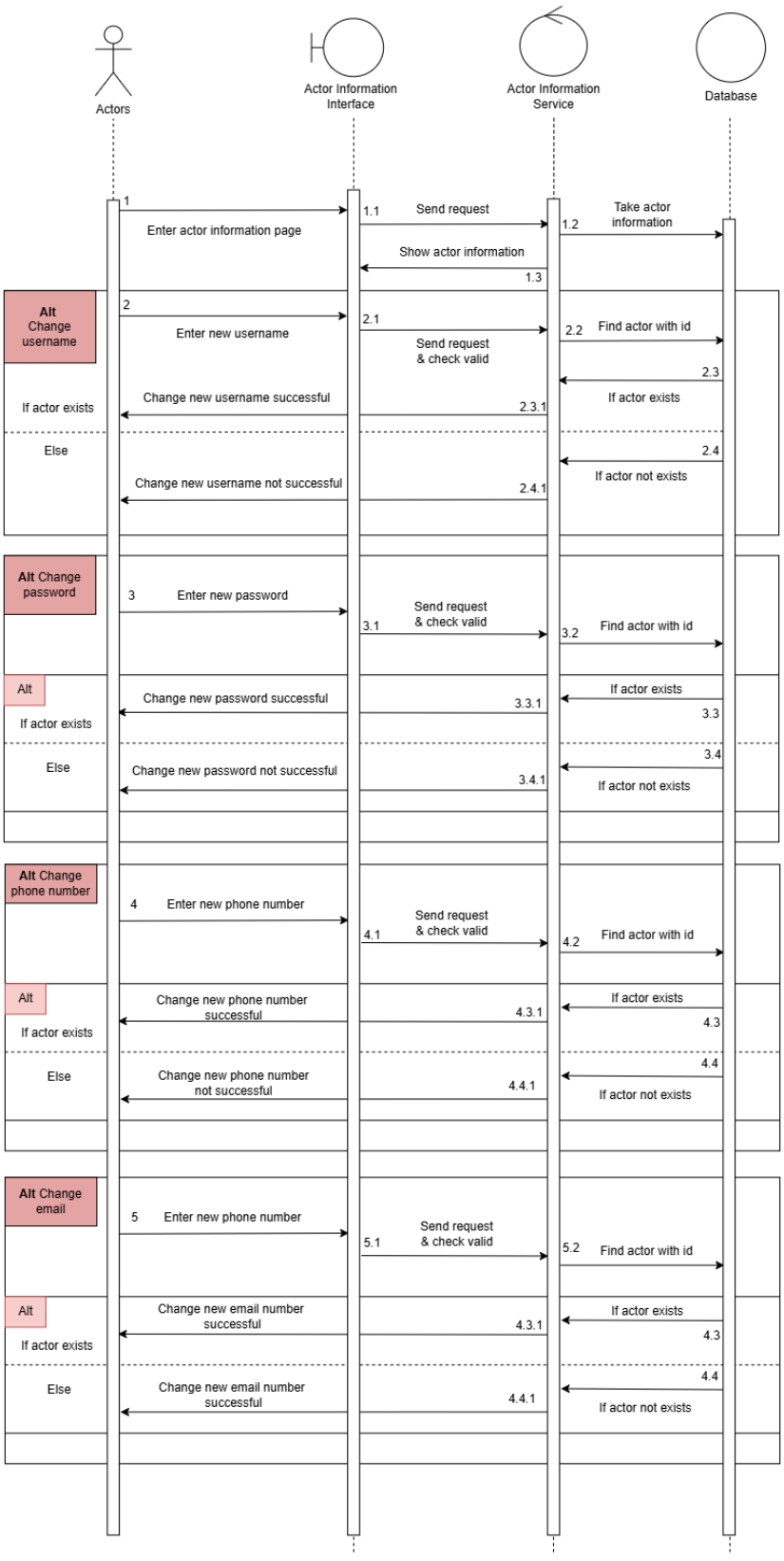


Figure 34: Delete User Sequence Diagram.

This sequence diagram illustrates the process of an admin deleting a user. After selecting the user to delete, the service verifies whether the user exists. If the user exists, they will be deleted from the system, and a notification email will be sent to inform that the account has been removed. If the user does not exist, the system returns a failure alert indicating that the deletion could not be completed.



Appendix 1: Edit User Information Diagram.