|  |
| --- |
| 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 incidents—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 (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, for his technical support in software and system deployment and for his assistance in refining my final report.

Furthermore, I would like to thank Mr. HOANG Van Son and Mr. NGUYEN Trung Kien from VTX, and also my colleagues: NGUYEN Duc Hanh, HO Quang Trung, NGUYEN Nhat Anh from USTH, and DOAN Quang from EPU, 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

[ABSTRACT i](#_Toc201758878)

[AUTHORSHIP ii](#_Toc201758879)

[ACKNOWLEDGEMENT iii](#_Toc201758880)

[List of Abbreviations vi](#_Toc201758881)

[List of Figures viii](#_Toc201758882)

[List of Tables ix](#_Toc201758883)

[Introduction x](#_Toc201758884)

[Chapter 1. Real-time Object Detection System 1](#_Toc201758885)

[1.1. Introduction to the Internet of Things (IoT) and Its Applications for Fire Monitoring Alert Systems 1](#_Toc201758886)

[1.2. Current situation and Limitations of the monitoring fire alert system 2](#_Toc201758887)

[1.3. Related Works 3](#_Toc201758888)

[1.4. Conclusions 5](#_Toc201758889)

[Chapter 2. System Architecture Design 6](#_Toc201758890)

[2.1. Hardware System Architecture 7](#_Toc201758891)

[2.1.1. Sensor Node Architecture 8](#_Toc201758892)

[2.1.2. Central Processing Unit System Architecture 11](#_Toc201758893)

[2.2. Software & Firmware System Architecture 12](#_Toc201758894)

[2.2.1. Firmware at the Node 13](#_Toc201758895)

[2.2.2. Data Processing at Central Processing Unit 17](#_Toc201758896)

[2.2.3. Software development at Central Processing Unit 20](#_Toc201758897)

[2.2.4. System Alert 32](#_Toc201758898)

[2.3. Conclusions 37](#_Toc201758899)

[Chapter 3. Results and Evaluations 38](#_Toc201758900)

[3.1. Implementation 38](#_Toc201758901)

[3.2. Reliability and Error Rates 39](#_Toc201758902)

[3.3. Conclusion 40](#_Toc201758903)

[Chapter 4. Conclusions and Perspective 41](#_Toc201758904)

[4.1. Conclusion 41](#_Toc201758905)

[4.2. Future work 41](#_Toc201758906)

[References 43](#_Toc201758907)

# List of Abbreviations

|  |  |
| --- | --- |
| Abbreviation | Definition |
| IoT | Internet of Things |
|  | Advanced Driver Assistance Systems |
|  | Artificial Intelligence |
|  | Advanced Micro Devices |
|  | Mean Average Precision |
|  | Application Specific Integrated Circuit |
|  | Advanced Video Coding |
|  | Convolutional Block Attention Module |
|  | Convolutional Neural Network |
|  | Common Objects in Context |
|  | Central Processing Unit |
|  | Direct Memory Access |
|  | Field Programmable Gate Array |
|  | Frames Per Second |
|  | Gigabyte |
|  | Gate equivalents |
|  | Graphics Processing Unit |
|  | High Definition |
|  | Histogram of Oriented Gradients |
|  | Long Term Support |
|  | Motion JPEG |
|  | Multiple Object Tracking Benchmark 2015 |
|  | Multiple Object Tracking Challenge |
|  | MultiProcessor System On Chip |
|  | Non-Maximum Suppression |
|  | Operating System |
|  | Pattern Analysis Statistical Modelling and Computational Learning |
|  | Performance Evaluation of Tracking and Surveillance 2009 |
|  | Pan-Tilt-Zoom |
|  | Random Access Memory |
|  | Regional Convolutional Neural Network |
|  | Red Green Blue |
|  | Scale-Invariant Feature Transform |
|  | Spatial Pyramid Pooling |
|  | Static Random Access Memory |
|  | Single Shot Detector |
|  | Support Vector Machine |
|  | Taiwan Semiconductor Manufacturing Company |
|  | Ultra High Definition |
|  | Vietnam National University |
|  | Visual Object Classes |
|  | You Only Look Once |
|  | Zynq UltraScale+ ZCU106 Evaluation Kit |

# List of Figures

[Figure 1. An IoT system with interconnected smart devices [1]. 2](#_Toc201762354)

[Figure 2: Hardware System Architecture. 7](#_Toc201762355)

[Figure 3: Central Processing Unit Architecture. 7](#_Toc201762356)

[Figure 4: Flame sensor KY-026 [10]. 8](#_Toc201762357)

[Figure 5: DHT11 - Temperature & Humidity Sensor [11]. 9](#_Toc201762358)

[Figure 6: STM32F103C8T6 Microcontroller [12]. 9](#_Toc201762359)

[Figure 7: HC-12 Wireless Communication Module [13]. 10](#_Toc201762360)

[Figure 8: 433 MHz RF Antenna [14]. 11](#_Toc201762361)

[Figure 9: Single board Computer - Raspberry Pi 5 [15]. 12](#_Toc201762362)

[Figure 10: Touchscreen [16]. 12](#_Toc201762363)

[Figure 11: Firmware Architecture at the Sensor Node. 13](#_Toc201762364)

[Figure 12: Data protection sequence in wireless transmission. 14](#_Toc201762365)

[Figure 13: Mavlink v1 Frame [17]. 15](#_Toc201762366)

[Figure 14: Software Architecture at the Central Processing Unit. 17](#_Toc201762367)

[Figure 15: Software Architecture diagram. 20](#_Toc201762368)

[Figure 16: Use case diagram. 25](#_Toc201762369)

[Figure 17: Login Sequence Diagram. 26](#_Toc201762370)

[Figure 18: Sign Up Sequence Diagram. 27](#_Toc201762371)

[Figure 19: Password Recovery Sequence Diagram. 28](#_Toc201762372)

[Figure 20: Create New User Sequence Diagram. 29](#_Toc201762373)

[Figure 21: Delete User Sequence Diagram. 30](#_Toc201762374)

[Figure 22: Export Logs Sequence Diagram. 30](#_Toc201762375)

[Figure 23: Threshold Setting Sequence Diagram. 31](#_Toc201762376)

[Figure 24: Deactivate Alert Sequence Diagram. 32](#_Toc201762377)

[Figure 25: Alert Sequence Diagram. 33](#_Toc201762378)

[Figure 26: Fuzzy Membership Functions for Water State (Cold, Warm, Hot). 34](#_Toc201762379)

[Figure 27: Membership of Temperature. 35](#_Toc201762380)

[Figure 28: Membership of Humidity 35](#_Toc201762381)

# List of Tables

[Table 1: Functional requirement. 23](#_Toc201758840)

[Table 2: Rule of Temperature, Humidity, Fire status. 36](#_Toc201758841)

# Introduction

# Real-time Object Detection 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*](#Section1_2) discusses the current situation and limitations of existing fire monitoring systems. [*Section 1.3*](#Section1_3) presents modern fire detection methods using current technologies, along with their limitations.

## 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.

[](equipment-services.blogspot.com)

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

## 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.

## 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 notable example is presented in [2], where 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 via ZigBee. On the other hand, [7] emphasizes the alerting aspect. This system uses an ESP32 with PIR sensors to detect both fire and movement and sends alerts via 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 that, [9] proposes an improved system using IoT and machine learning, integrating various sensor types to boost detection accuracy and support remote monitoring. Although recent works reflect technological advancements, the core concept of IoT-based fire detection was explored much earlier. For example, [9] describes a real-time fire alarm system developed using Raspberry Pi and Arduino Uno. This system included smoke detection, room image capture, and alerting via SMS and a web interface. Notably, 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, even including features like user login.

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 approach 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.

## 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.

# 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*.

## Hardware System Architecture

This section introduces the hardware components used at both the *sensor node* and *the central processing unit*.

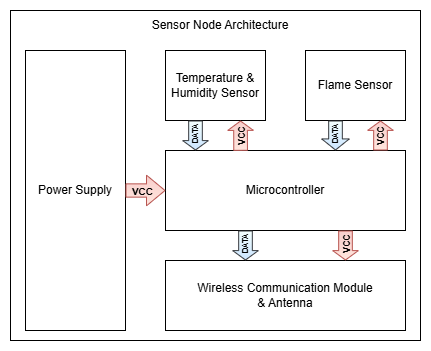


Figure : Hardware System Architecture.

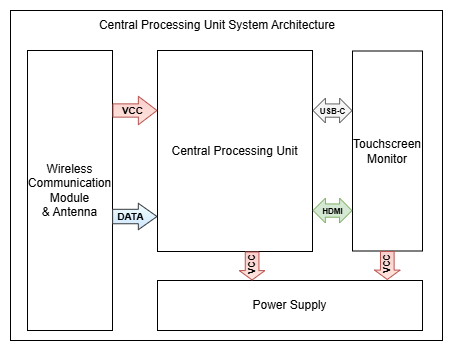


Figure : Central Processing Unit Architecture.

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.

### Sensor Node Architecture

#### 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.

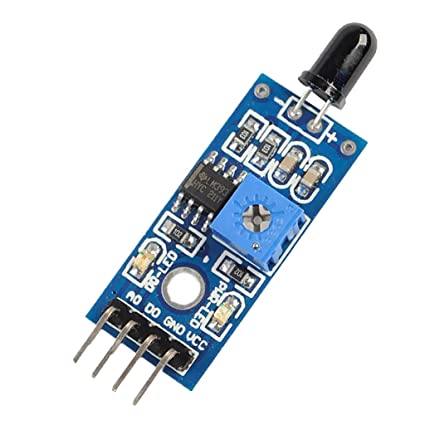


Figure : Flame sensor KY-026 [10].

#### 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.

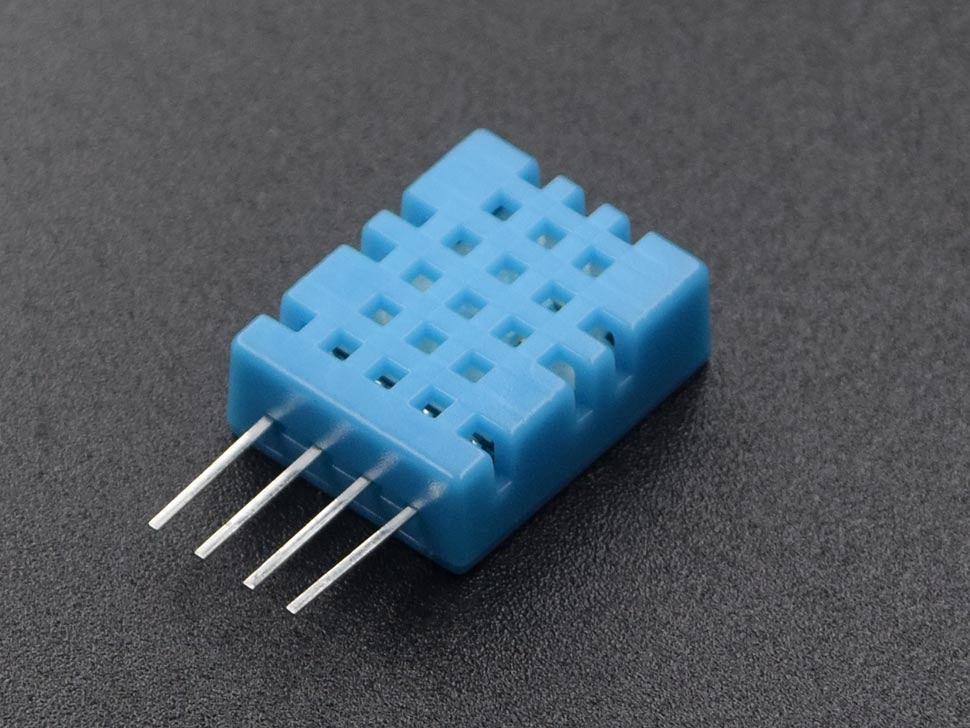
[](https://circuit.rocks/products/sensors-temperature-and-humidity-dht11-sensor-html)

Figure : DHT11 - Temperature & Humidity Sensor [11].

#### 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.

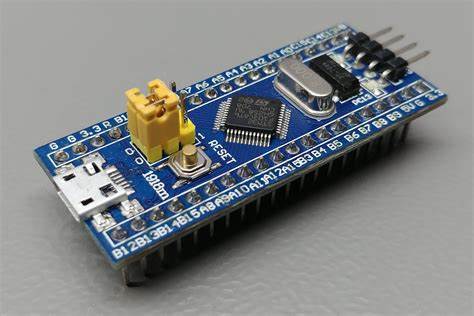
[](https://www.epanorama.net/blog/2022/06/18/introduction-to-the-stm32-blue-pill-stm32duino-and-other-stm32-boards/)

Figure : STM32F103C8T6 Microcontroller [12].

#### 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.

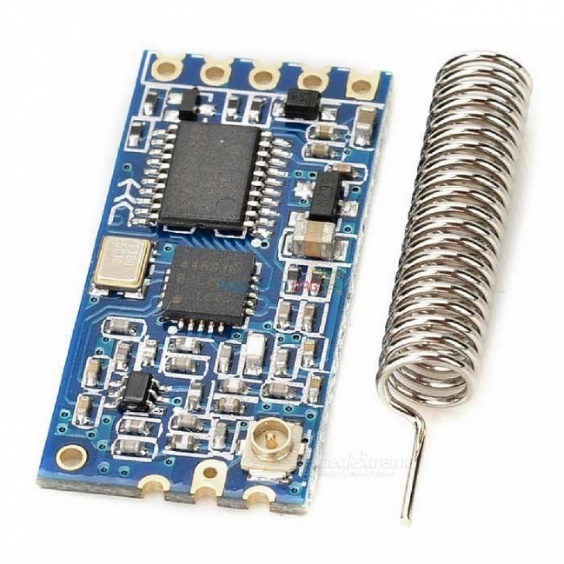
[](https://www.flyrobo.in/hc-12-433-si4463-wireless-serial-module)

Figure : HC-12 Wireless Communication Module [13].

#### 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-costcommunication without requiring existing infrastructure.



Figure : 433 MHz RF Antenna [14].

### Central Processing Unit System Architecture

#### 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.

#### 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.

[](https://www.theregister.com/2023/09/28/raspberry_pi_5_revealed/?td=amp-keepreading)

Figure : Single board Computer - Raspberry Pi 5 [15].

#### 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 : Touchscreen [16].

## 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.

### Firmware at the Node

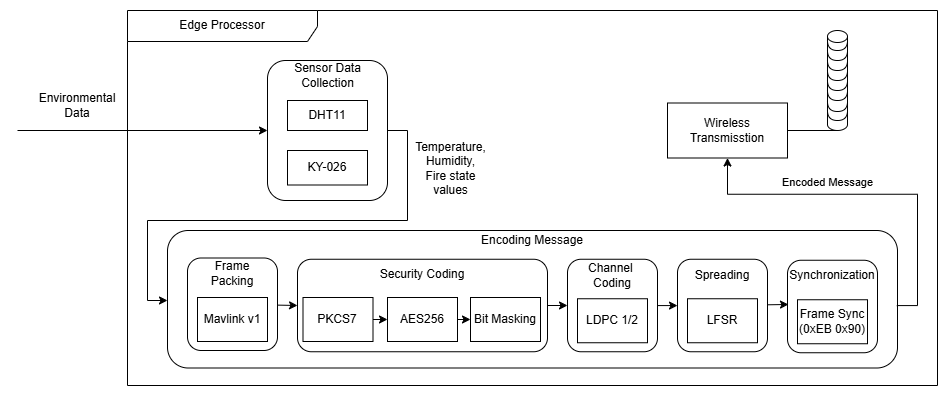


Figure : Firmware Architecture at the Sensor Node.

#### 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 : 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.

#### Decoding Message



Figure : 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.

* 1. ***Frame Packing using Mavlink v1***

**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.

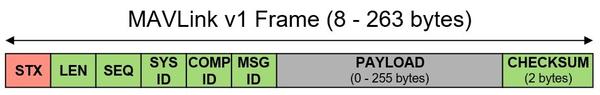


Figure : Mavlink v1 Frame [17].

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.

* 1. ***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. Because *AES-256* works with 16-byte blocks, the MAVLink message must be padded accordingly. The *PKCS7* padding is used: if the message length is not a multiple of 16, the missing bytes are filled with the number of padding bytes; if the message is already a multiple of 16, a full block of padding with the value 0x10 is appended. 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 masking technique – specifically, bit reversal, which reverses all the bits in each block.

* 1. ***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.

* 1. ***Spreading using LFSR***

**Input:** Channel-encoded message.

**Output:** Spread message.

In the Spreading 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.

* 1. ***Synchronization***

**Input:** Channel-encoded 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.

### Data Processing at Central Processing Unit

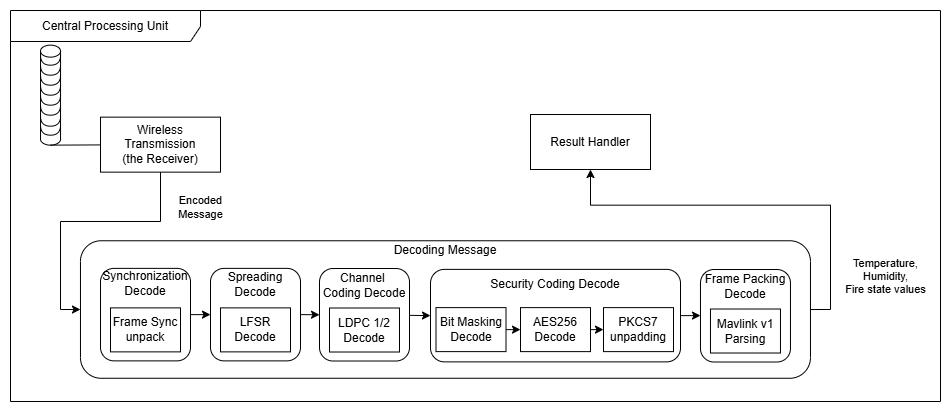


Figure 14: Software Architecture at the Central Processing Unit.

#### 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.

#### Decoding Message

1. ***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.

1. ***Spreading Decode***

**Input:** Frame message unpack.

**Output:** Despread message.

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.

1. ***Channel Coding Decode***

**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.

1. ***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.

1. ***Frame Packing Decode***

**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.

### 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.

1. ***Software Architecture***

***Software Design***

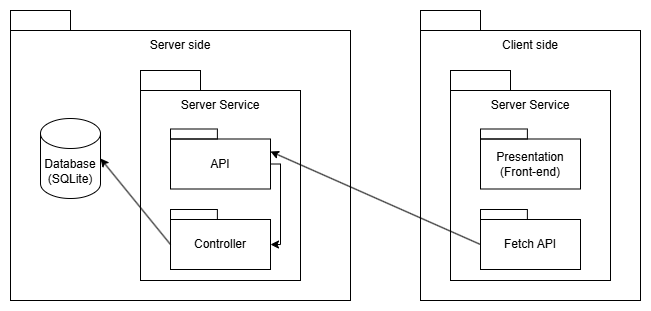
******

Figure : Software Architecture diagram.

Several key components are essential for the seamless operation of the system on the server side:

**Database**: Acts as a central repository, storing critical data including user information, daily activity logs, fire event alerts, threshold settings, etc.

**Server Service:** Serves as the bridge between the client and the database, responsible for handling user requests and coordinating data access.

**API:** APIs act as the communication interface between the client and the server through various endpoints. They receive and process requests from the client, query the database, execute business logic, and provide system functionalities.

**Controllers**: handle the business logic of the application. They interact with APIs to retrieve or update data from the database, manage data flow, and ensure the overall system's functionality.

On the client side, the presentation layer is developed using HTML, CSS, and JavaScript.:

* + **User Interface:** This layer enables user interaction with the system. It can invoke server APIs to retrieve or update data, allowing dynamic and responsive content to be presented to the user.
  + **Fetch API:** Fetch API is a built-in JavaScript feature that enables the client to send HTTP requests to the server flexibly. It allows web content to be updated without reloading the entire page, enhancing performance and user experience.

***Tool & Technicals***

The MVC (Model–View–Controller) architecture is adopted in the software system of this thesis. It includes the full structure of Controller, Service, Repository, and Model, offering a clean and maintainable design pattern.

**T**he backend is built using Flask, a lightweight and flexible web framework in Python. Python’s versatility makes it ideal for interacting with GPIOs and peripherals, such as opening the UART COM port to receive data from the Edge Processor. In addition, the framework is easy to maintain, user-friendly for developers, and highly accessible due to the abundance of learning resources available, thanks to its widespread popularity.

SQLite is used as the database system. While it is a relatively old system, it is lightweight, simple to use, and integrates easily with Flask, making it ideal for embedded systems or lightweight applications.

The user interface (Frontend) is implemented using HTML, CSS, and JavaScript – the three fundamental technologies in web development. Despite their simplicity, they remain powerful, maintainable, and developer-friendly, enabling responsive and interactive UI design.

APIs serve as the communication bridge between the Frontend and the Server’s Controller, Service, and Database. From the user interface, APIs can be called to retrieve or update data, enabling dynamic and efficient information display and system interaction.

Because the system currently doesn't have a continuous online server, a VPN solution is used to allow remote users to access the web interface and interact with system components. TailScale is the VPN used, which is popular in IoT systems because of its simplicity of setup, secure peer-to-peer connections, and reliable performance. This technique ensures that remote users may successfully monitor, operate, and reply to the system, even without the support of a dedicated always-on server.

1. ***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 |
| 2 | Password Recovery | Recover forgotten password |
| 3 | Login | Log in to the system |
| 4 | EditPersonal Information | Edit username, password, email, and phone number |
| 5 | Maintain  Dashboard | View Log & Fire Events | View daily logs and fire event history |
| 6 | Real-time Data Monitoring | View live sensor data (temperature, humidity) on the dashboard |
| 7 | Export logs | | Export the data of the days |
| Export fire events |
| 8 | User Management | Create a new user | Add user accounts | Admin |
| Delete exist user | Remove user accounts |
| 9 | Approval Sign Up queue | Approval Sign Up queue |
| 10 | Maintain Alert & Threshold | Alert Deactivation & Notification | Send a system-wide alert if a fire is detected or the alert is deactivated |
| 11 | Threshold Settings | Modify temperature and humidity thresholds and notify all users |

Table : 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 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.

1. ***System analysis and design***

**Use case diagram**

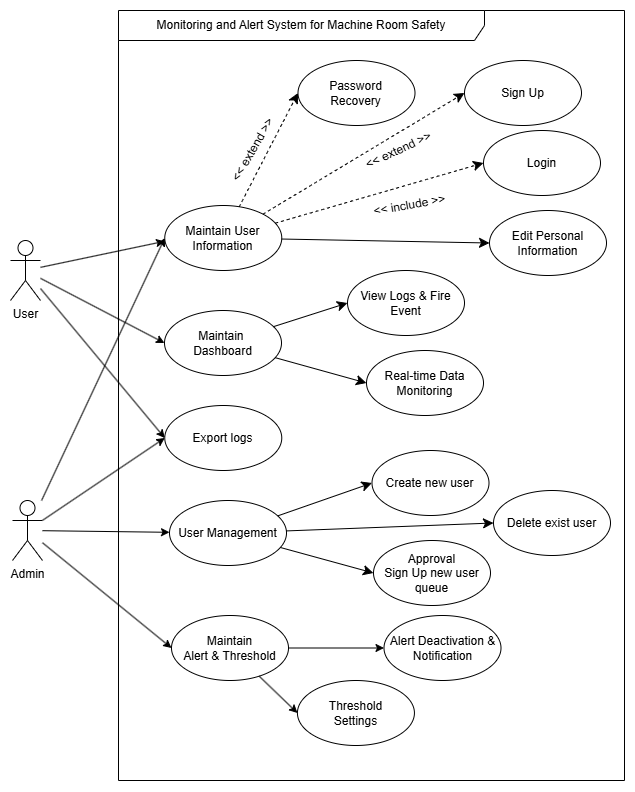


Figure : Use case diagram.

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**

***Login***

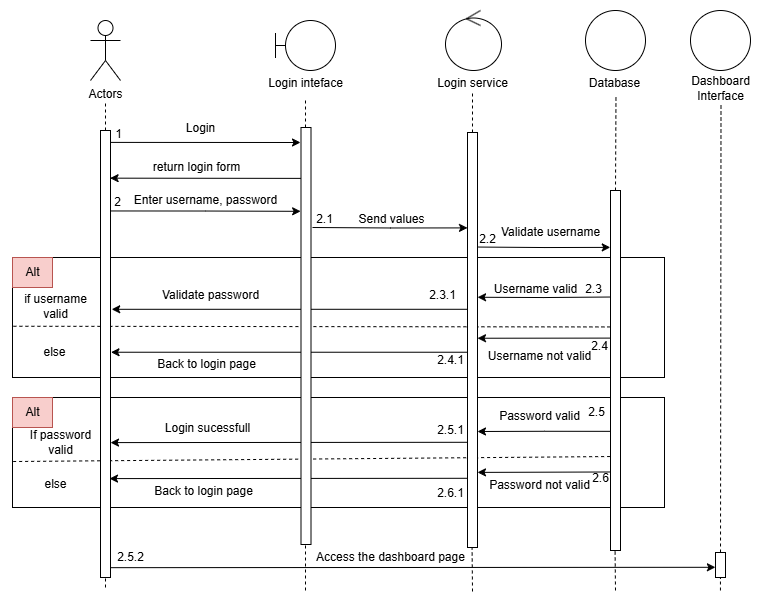
******

Figure : 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***

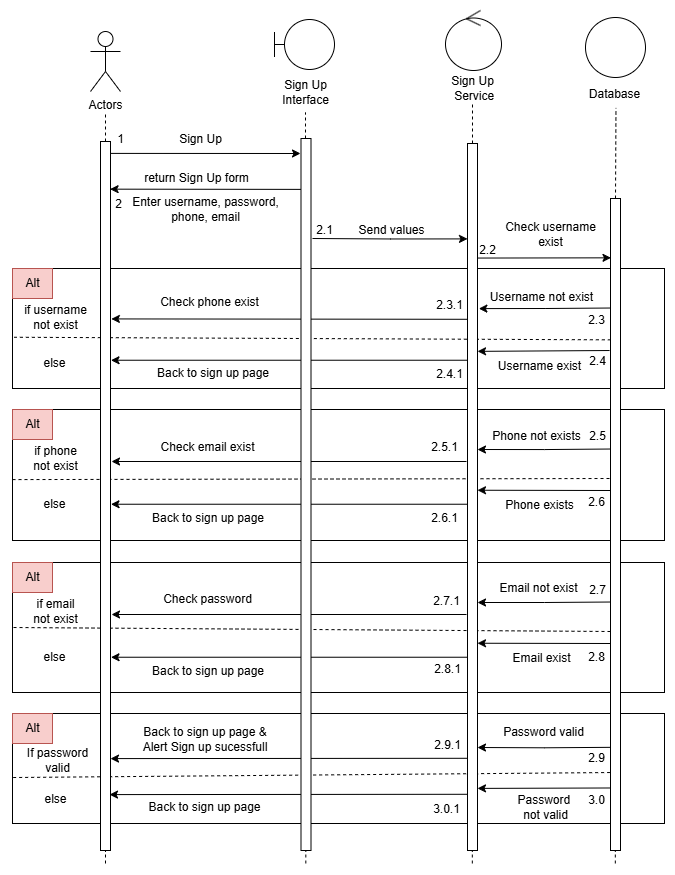
******

Figure : 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***

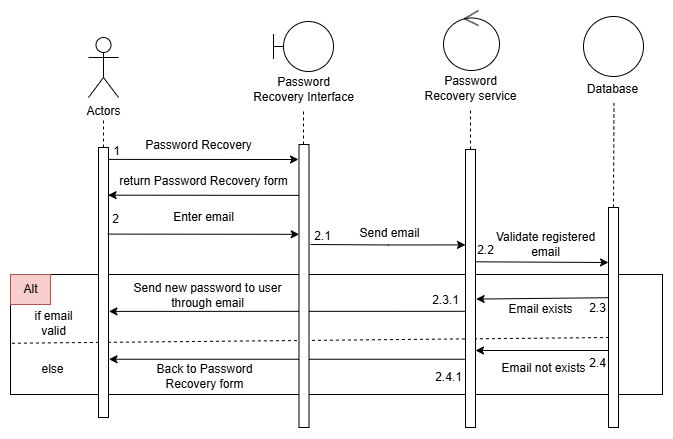
******

Figure : 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***

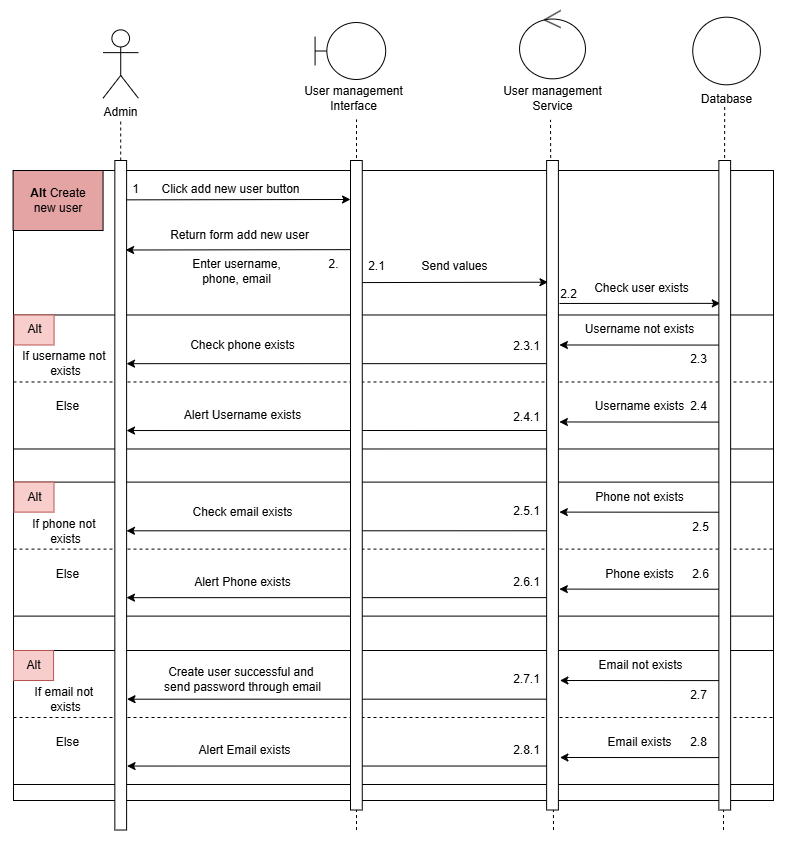
******

Figure : 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***

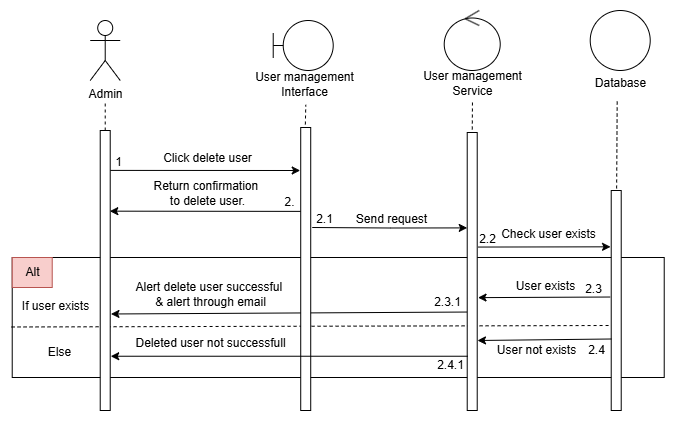
******

Figure : 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.

**Export Logs**

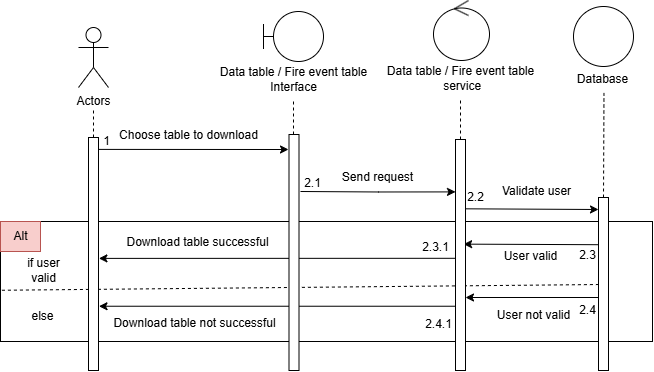
****

Figure : 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**

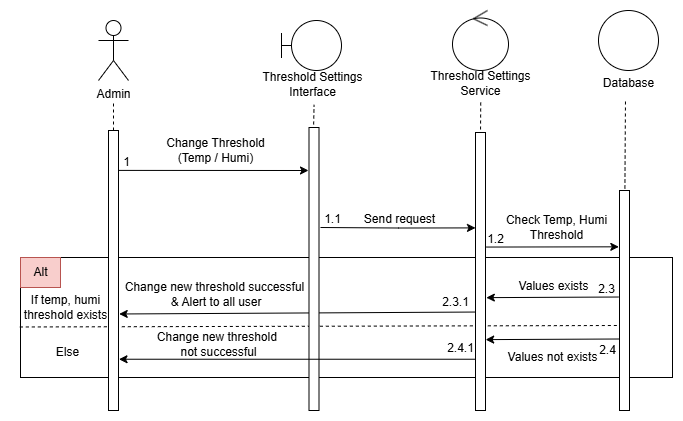
******

Figure : 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**

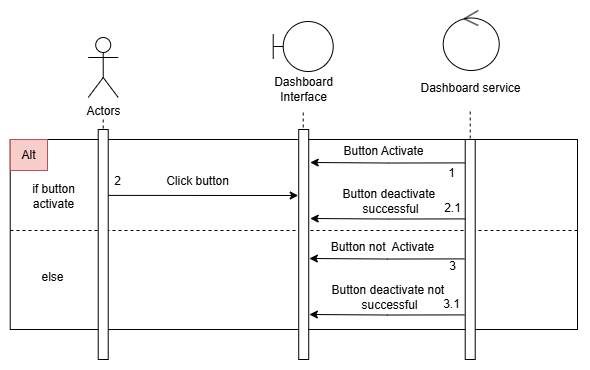
******

Figure : 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.

### 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.

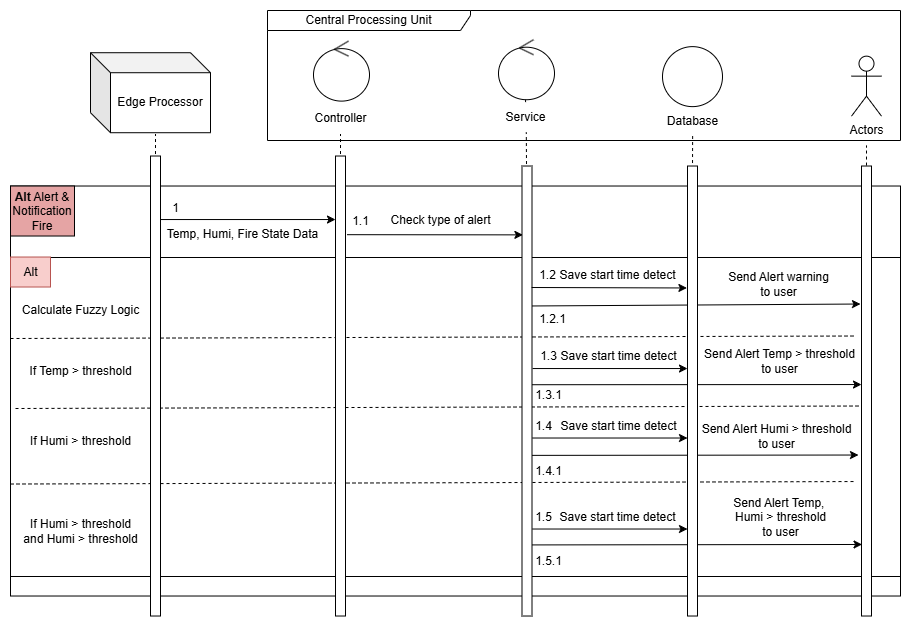
******

Figure : Alert Sequence Diagram.

1. ***Temperature & Humidity Threshold Alerts***

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.

1. ***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.

1. ***Fuzzy Logic***

Fuzzy Logic [18] [19]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.

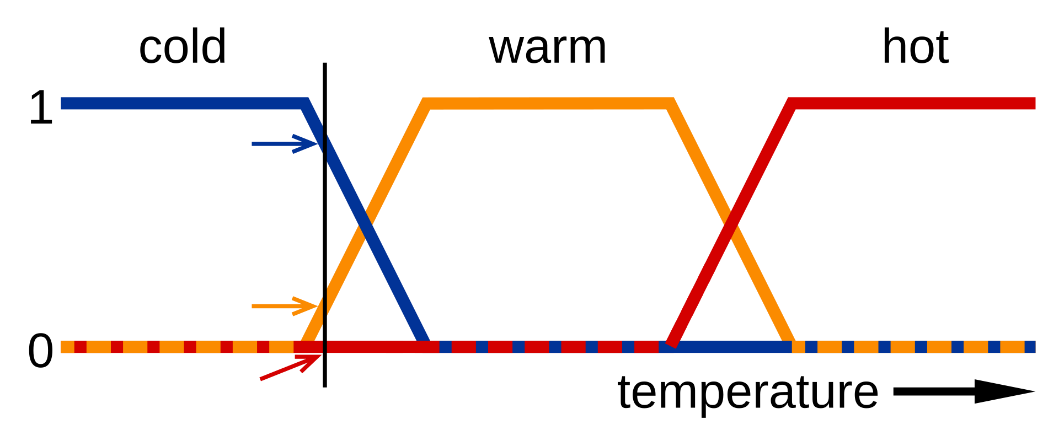


Figure : Fuzzy Membership Functions for Water State (Cold, Warm, Hot).

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:**

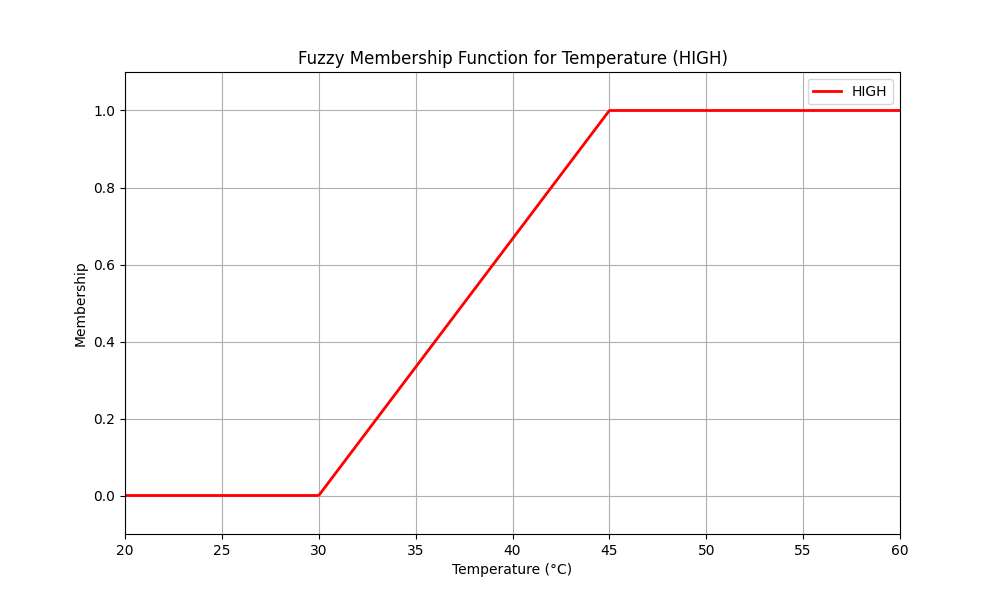


Figure : Membership of Temperature.

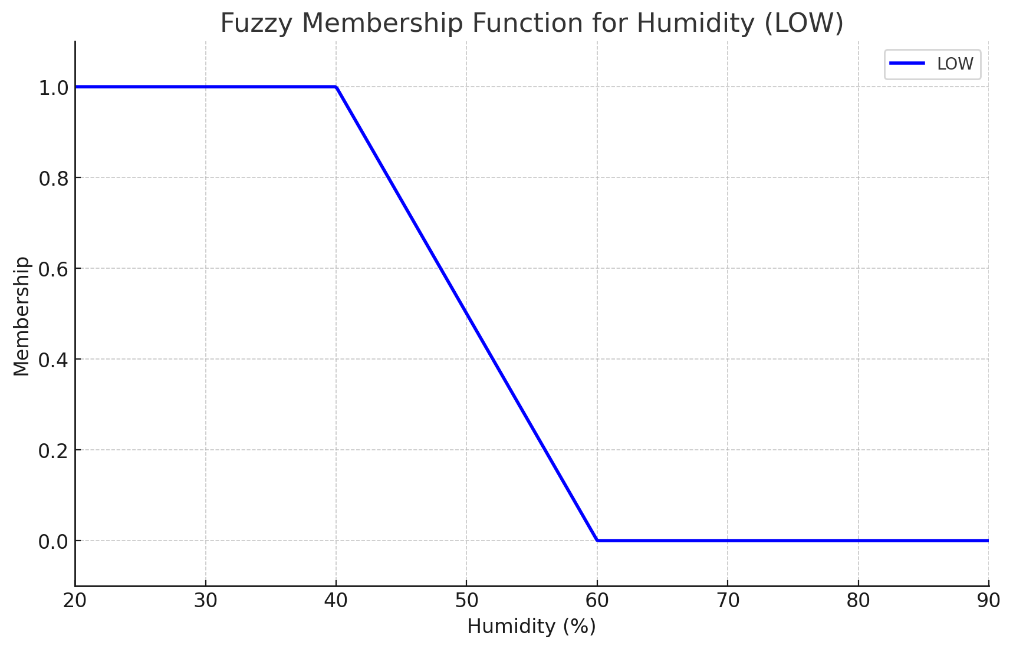


Figure : Membership of Humidity

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 : 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.

## 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.

# 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 functions that 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.

## Implementation

On the edge processor side, sensor data was successfully collected as described in the previous chapters. The data was processed, encoded, and packaged at the node, then transmitted wirelessly without errors. On the CPU side, the system successfully received the transmitted data, decoded it, and extracted the core information, including temperature, humidity, and fire state, accurately and consistently. Following the data processing pipeline, the software functions specified in the *Software Development at Central Processing Unit* section of Chapter 2 were fully implemented. All functionalities operated smoothly, and user contact with the touchscreen interface (detailed in the hardware section) was smooth and responsive. Key features, including Login, Sign Up, Password Recovery, View Logs, Export Logs, and User Management, were implemented exactly as designed for both administrators and regular users. The complete system demonstrated effective integration of hardware, firmware, and software components, confirming that the IoT-based fire monitoring and alarm system would function as planned in real-world scenarios.

## Reliability and Error Rates

Temperature and humidity data have been collected in real time from the environment using the DHT11 sensor. The data sampling technique was highly consistent, with almost no mistakes or data loss detected throughout each collecting cycle. When the sensor encountered environmental errors (for example, temperature rises), it responded correctly, displaying changes in value that were equal to the degree of variation in the environment.

The KY-026 flame sensor also worked well, continually detecting heat from fire sources with no obvious delay. It responded quickly and accurately when a flame or high-heat source was detected, completing its critical detection function.

When the sensor data moved through the data packaging and encoding phases, each message was packaged correctly without any errors. The wireless communication from the communication node to the receiver has an extremely low packet loss rate. This is mainly due to the employment of channel encoding methods, which enhanced the reliability of data by allowing bit-level recovery while transmitting through radio frequency. As a result, the overall message loss rate was almost zero, ensuring that all packets were safely delivered to the Central Processing Unit and correctly decoded for later processing.

After decoding, the user interface displayed real-time temperature, humidity, and fire state information. The data was displayed continuously and completely, with no missing or delayed readings. Furthermore, when high temperatures were detected, the "Deactivate Alert" button was immediately activated and turned on, allowing the system administrator to quickly recognize and respond to fire situations, either by deactivating alerts after the potential danger had been resolved or by handling any system-related issues.

These results verify the system's real-time responsiveness, data dependability, and resistance to transmission mistakes, confirming it's suitable for real-world implementation in fire monitoring applications. In addition, with the integration of a VPN (TailScale), the system also allowed remote users to access and interact with the platform without interruption, and the communication delay remained nearly zero.

## Conclusion

Finally, following the complete deployment of the fire monitoring and alarm system, all system components performed as planned. From sensor nodes to microcontrollers, each module effectively executed its function, including data collecting, packet encoding, transmission, and decoding, with data transferred wirelessly to the CPU completely and accurately.

The CPU effectively decoded and analyzed the incoming data packets with minimal delay. Throughout the process, the end-to-end delay was extremely low, having no meaningful effect on the system's performance or responsiveness.

Furthermore, the data was displayed in real-time through an online interface. All web-based features also responded fast and reliably, consistent with the expected behavior of an IoT system. Critical alert capabilities have been successfully implemented, enabling the early detection and communication of fire threats in alignment with the system's monitoring and warning objectives.

Overall, the system demonstrated excellent integration of hardware, firmware, and software components, as well as high dependability, real-time responsiveness, and efficient use for fire detection and alerting in real-world settings.

# Conclusions and Perspective

## Conclusion

In recent years, there has been significant research and development 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 central processing unit — the system's “brain” — 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 needs reliable and timely information is very critical.

## Future work

Fire monitoring and alarm systems are not new technologies; however, each newly developed idea for such systems serves as a foundation for further innovation and enhancement in the future. Therefore, it would be inaccurate to consider the current system of thesis implementation as fully optimized. There are still notable limitations and several areas that offer opportunities for development. In future iterations, this thesis envisions the integration of a wider variety of sensors within each sensor node, including gas detectors, smoke sensors, vibration sensors, and comprehensive environmental modules. Additionally, rather than relying on a single sensor node as in the current prototype, the system will be scaled to include multiple distributed nodes, covering a larger area. This will provide more varied data and enable more efficient monitoring and management of larger environments if applied in the future.

About wireless communication, enhancing data packaging and encoding algorithms will also be a key objective. This will make wireless communication stronger and dependable, particularly in complicated or high-interference contexts. The system now uses VPN technology (TailScale) for remote access. While successful, it lacks the scalability and dependability of a dedicated, always-on server. Future enhancements will include an operational server to improve accessibility, stability, and overall system performance. Furthermore, in an age of rapidly developing artificial intelligence, the system intends to include AI-based fire detection and prediction technologies. These models can significantly increase accuracy and response time. The addition of 360-degree thermal imaging cameras capable of operating in full darkness would further improve the system's performance, particularly at night, when fire events are more prevalent and difficult to detect. Although the present system is written in Python, future versions may switch to lower-level languages like as C or C++ to improve performance in embedded situations and ensure minimal resource use. Furthermore, the system may employ application development tools to generate an independent, executable desktop program, making it easier for end users to install, run, and interact with the system.

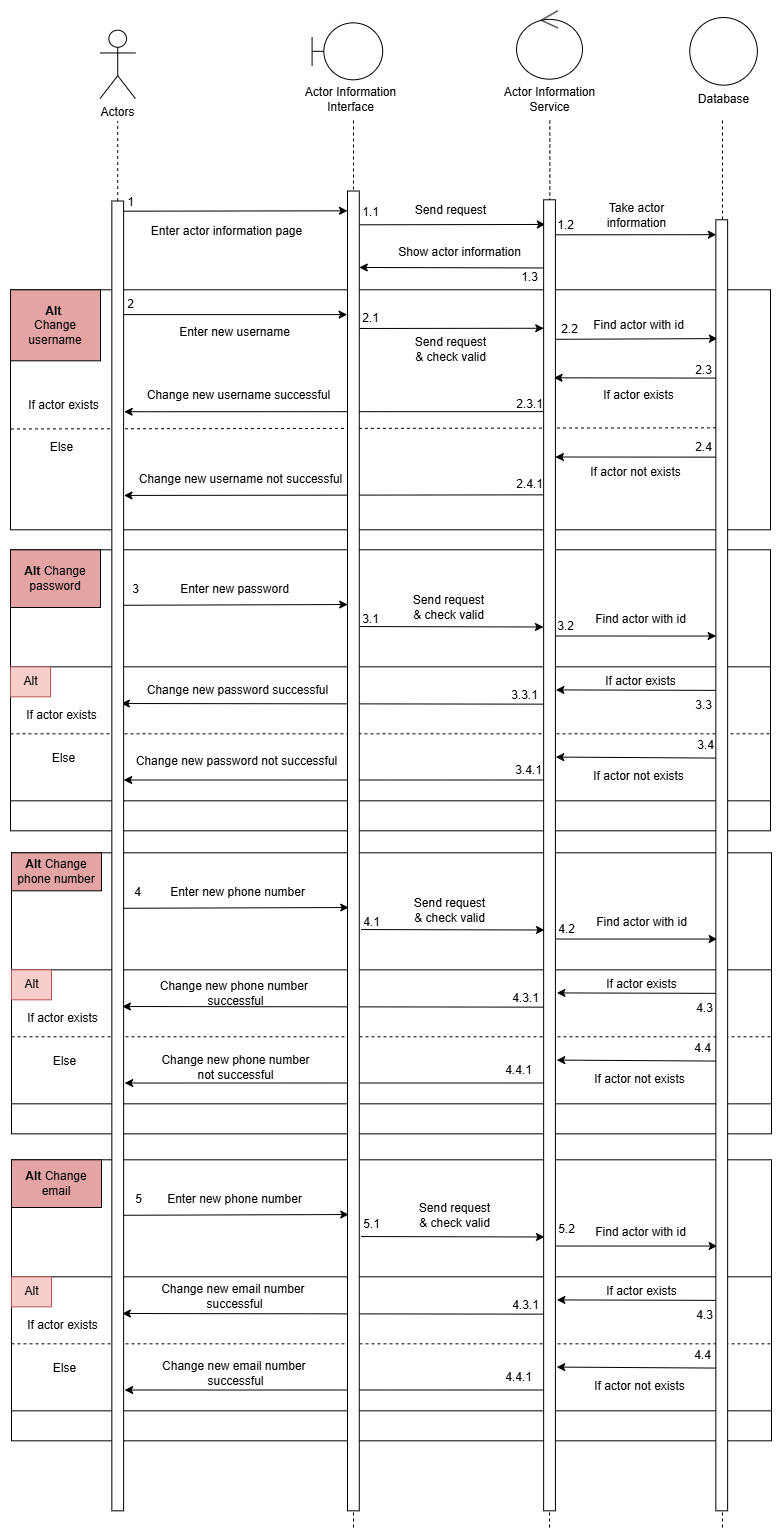
# References

|  |  |
| --- | --- |
| [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] | "COMPOINDIA," [Online]. Available: https://compoindia.com/product/flame-sensor-module/. |
| [11] | "Circuit Rocks," [Online]. Available: https://circuit.rocks/products/sensors-temperature-and-humidity-dht11-sensor-html. |
| [12] | "ePanorama.net," [Online]. Available: https://www.epanorama.net/blog/2022/06/18/introduction-to-the-stm32-blue-pill-stm32duino-and-other-stm32-boards/. |
| [13] | "IC Master," [Online]. Available: https://icmasteronline.com/product/433mhz-hc12-wireless-serial-port/. |
| [14] | "HSHOP," [Online]. Available: https://hshop.vn/anten-433mhz-35dbi-sma-duc. |
| [15] | "The Register," [Online]. Available: https://www.theregister.com/2023/09/28/raspberry\_pi\_5\_revealed/?td=amp-keepreading. |
| [16] | "Amazon," [Online]. Available: https://www.amazon.sg/ASUS-ZenScreen-MB16AC-Portable-Monitor/dp/B0BZR6ZNB5?th=1. |
| [17] | "CSDN," [Online]. Available: https://blog.csdn.net/weixin\_43321489/article/details/108062380. |
| [18] | "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Fuzzy\_logic. |
| [19] | 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:

[Appendix 1: Edit User Information Diagram. 43](#_Toc201700795)

******

Appendix : Edit User Information Diagram.