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

Aerolyze

(Smart Hazardous Gas Leakage Detection System for Home Applications)

GROUP 10

Aulia Anugrah Aziz	**2206059364**
Aliyah Rizky Al-Afifah Polanda	**2206024682**
Christopher Satya Fredella Balakosa	**2206059755**
Mario Matthews Gunawan	**2206810452**

PREFACE

In today's digital era, the Internet of Things (IoT) has rapidly evolved and increasingly exists in our surroundings. IoT technologies are being applied to improve safety, convenience, and efficiency in various aspects of daily life. One pressing issue that IoT can address is household safety particularly in detecting and preventing hazardous gas leaks that can pose significant risks to residents. Therefore, our group plans to overcome this problem by creating a project, "Aerolyze: Dangerous Gas Leak Detection System for Homes."

This project was developed as part of the Internet of Things Laboratory Final Project, reflecting our commitment to applying IoT principles to real-world challenges. Aerolyze is designed to provide a proactive solution for detecting dangerous gases like LPG and carbon monoxide, offering timely alerts to enhance safety and prevent potential disasters. The project not only demonstrates the technical capabilities of IoT systems but also highlights their practical applications in creating a safer living environment.

Through this project, we gained valuable experience in teamwork, problem-solving, and applying IoT-based systems to real-world challenges. By completing this project, we were able to deepen our understanding of the 10 modules studied during the lab sessions and explore the practical applications of IoT technology. We hope that Aerolyze serves as a meaningful initiative in advancing the field of IoT and inspires further innovations to improve household safety and security.

Depok, December 06, 2024

Group 10

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION....4
1.1 PROBLEM STATEMENT....4
1.2 PROPOSED SOLUTION....4
1.3 ACCEPTANCE CRITERIA....5
1.4 ROLES AND RESPONSIBILITIES....5
1.5 TIMELINE AND MILESTONE....6
CHAPTER 2
IMPLEMENTATION....7
2.1 HARDWARE DESIGN AND SCHEMATIC....7
2.2 SOFTWARE DEVELOPMENT....8
2.2.1 Leaf Program....8
2.2.2. Root Program....10
2.3 HARDWARE AND SOFTWARE INTEGRATION....14
CHAPTER 3
TESTING AND EVALUATION....16
3.1 TESTING....16
3.2 RESULT....16
3.3 EVALUATION....19
CHAPTER 4
CONCLUSION....21
REFERENCES....22
APPENDICES....23

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

In household safety measures nowadays, detecting hazardous gas leaks often relies on manual observation or standalone alarms which may result in delayed responses and slow handling, potentially leading to fatal consequences. Such delays can lead to severe consequences, including fires, explosions, or carbon monoxide poisoning that pose significant risks to the safety of residents.

Hazardous gas leaks, such as LPG or carbon monoxide (CO) from house fires can occur unexpectedly and spread rapidly, endangering lives and property. Without an effective and real-time detection system, these situations often go unnoticed until they escalate into critical emergencies. This lack of proactive monitoring increases the potential for harm and highlights the need for a more reliable solution.

To address these challenges, Aerolyze introduces a smart gas leakage detection system that provides continuous real-time monitoring, immediate alerts, and automated responses. By leveraging IoT technologies, Aerolyze ensures prompt detection and notification, enabling swift action to minimize risks and enhance household safety effectively.

1.2 PROPOSED SOLUTION

Aerolyze is an innovative device designed to detect hazardous gases in households and provide real-time notifications to users for enhanced safety. Our project utilizes the MQ-2 sensor which can detect LPG and toxic carbon monoxide (CO) gases that pose serious risks to health and property if left undetected. Once a gas leak is detected, the device promptly sends a notification to the user via WhatsApp.

In addition to digital notifications, Aerolyze also features an integrated alarm system that uses both a buzzer and a red LED to deliver immediate auditory and visual warnings within the household. This dual-alert system ensures that users are promptly made aware of potential threats, allowing them to respond swiftly and effectively to minimize the risks associated with gas leaks. By combining advanced sensor technology, visual and auditory

alerts, and IoT-based real-time communication, Aerolyze provides a reliable and efficient solution to enhance household safety.

1.3 ACCEPTANCE CRITERIA

The acceptance criteria of this project are as follows:

1. Detect LPG and toxic carbon monoxide (CO) gases.
   Aerolyze utilizes the MQ-2 sensor to detect hazardous gases such as LPG and toxic carbon monoxide (CO), ensuring accurate and continuous monitoring of household air quality.
2. Give real-time alerts using buzzer and red LED.
   The device activates an integrated buzzer and red LED to deliver immediate auditory and visual warnings when dangerous gas levels are detected, ensuring residents are promptly alerted to potential threats.
3. Give notification to user by Whatsapp.
   Aerolyze sends real-time notifications to the user's WhatsApp, providing instant updates and enabling swift action to address the detected hazard.

1.4 ROLES AND RESPONSIBILITIES

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

Roles	Responsibilities	Person
Project Leader	Implement the code for monitoring gas thresholds and to create the report	Christopher Satya Fredella Balakosa
Staff	Make the program for the leaf node, provide the necessary hardware, and create the report	Aulia Anugrah Aziz

Staff	Make the program for the root node and create the report	Aliyah Rizky Al-Afifah Polanda
Staff	Create the report, make readme, and help the project leader.	Mario Matthews Gunawan

Table 1. Roles and Responsibilities

1.5 TIMELINE AND MILESTONE

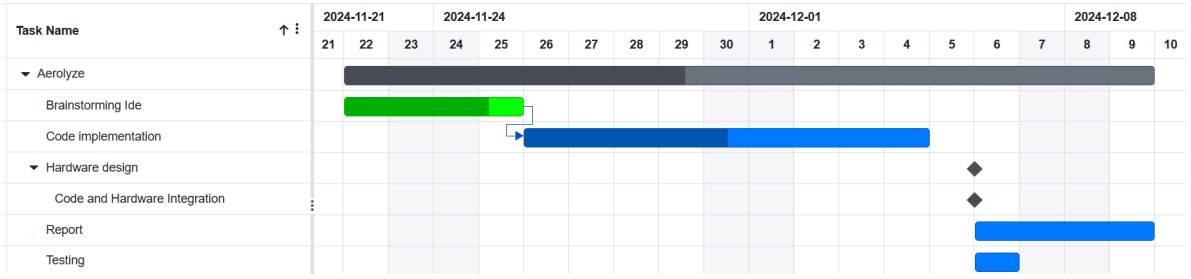


Fig. 1 Gantt Chart

CHAPTER 2

IMPLEMENTATION

2.1 HARDWARE DESIGN AND SCHEMATIC

Hardware implementation on this system includes a number of available components and is centered on the MQ-2 gas sensor. The components in the MQ-2 gas sensor will be designed in conjunction with other components. This is done to combine available features to be able to send messages to WhatsApp using API and activate buzzer and Red LED if gas concentration exceeds the threshold. The system design aims to improve household safety by offering real-time hazardous gas monitoring and alerting capabilities.

The MQ-2 gas sensor plays a key role in detecting LPG and Carbon Monoxide (CO), which are common hazardous gases. This sensor will be connected to the Master which serves as the main processing unit. The master will connect to the WiFi network and process the sensor data to determine the gas concentration levels. If the concentration exceeds the pre-defined safety threshold, then the system triggers multiple alert mechanisms to ensure user awareness and swift action.

The integrated buzzer and Red LED will be responsible for immediate auditory and visual warnings. When activated, these alerts will notify household members locally of potential gas leaks. Additionally, the master sends real-time notification through the user's WhatsApp via IoT platform and ensures that users are informed when they are not home.

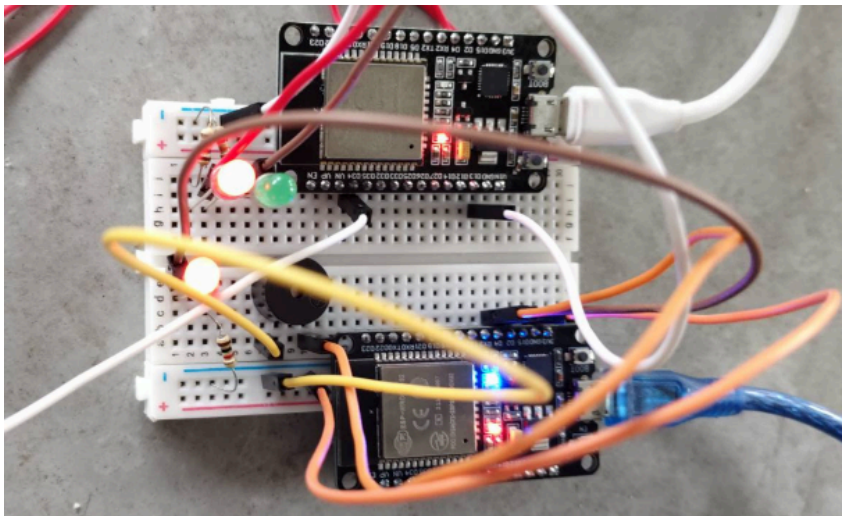


Fig. 2 System Schematic

2.2 SOFTWARE DEVELOPMENT

The software for this project is developed in C++ and implements several IoT modules. The following libraries are used:

```
#include <painlessMesh.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <UrlEncode.h>
#include <time.h>
```

- painlessMesh.h: This library allows the creation of a mesh network with low power, low cost, and efficient communication between devices.
- WiFi.h: This library provides functionality to connect the device (ESP32) to a Wi-Fi network. It enables the device to send and receive data over the internet or local network.
- HTTPClient.h: This library is used to send HTTP requests and receive responses. It allows the device to interact with web APIs or servers via HTTP/HTTPS protocols.
- UrlEncode.h: This library provides functions to URL-encode strings. URL encoding is necessary for transmitting special characters in web URLs, ensuring the data is properly formatted for HTTP requests.
- time.h: This standard library is used for handling time-related functions.

The program is divided into two main parts: Leaf Program and Root Program.

2.2.1 Leaf Program

The leaf program is responsible for receiving data from the MQ2 sensor and sending the sensor values to the root node using the mesh topology. It also indicates the gas level by turning on the green light, during safe conditions, and by turning on the red light, during dangerous conditions.

Here is the explanation of the library, functions and tasks used in the program:

a. The program uses the painlessMesh library to handle mesh networking and creates an instance of the library in the beginning. It then determines the MESH_SSID, MESH_PASSWORD, and MESH_PORT to match the root node for possible communication.

b. The program uses a queue called MQ2Queue for transferring gas sensor data between tasks.

c. setup()

This function initializes the serial communication at a 115200 baud rate and configures mesh debug message types for the mesh network. It then initializes the mesh network with the SSID, password and port and determines the receive callback function using mesh.onReceive. This function also configures the MQ2 sensor as input and configures the LED pins as outputs. This function also creates a FreeRTOS queue with a size of 5 items, each of type GasData and checks if the queue creation was successful. Two tasks are created using xTaskCreate for MQ2task and MeshTask.

d. loop()

This function continuously updates the mesh network to manage connections and data.

e. void MQ2task(void* pvParameters)

This function handles reading the MQ2 sensor data while also toggling the LEDs. At every iteration, it will begin by creating a variable to store sensor data and read the MQ2 value

```
GasData gasData;
    gasData.gasvalue = analogRead(MQ2);
             Serial.printf("MQ2   Sensor   Value:   %.2f\n",
gasData.gasvalue);
```

It will also check the gas value. If the gas value is greater than 1200, it will turn off the green LED and turn on the red LED, else it will turn on the green LED and turn off the red LED. Once the gas data is read, it sends the gas sensor data to the queue. Logs whether the data was successfully enqueued.

```
if  (xQueueSend(MQ2Queue,  &gasData,  pdMS_TO_TICKS(10))  !=
pdPASS) {
      Serial.println("Failed to send MQ2 data to queue");
    } else {
                Serial.println("MQ2  data  sent  to  queue
successfully!");
    }
```

f. void receivedCallback(uint32_t from, String& msg)

this function handles incoming messages from other mesh nodes.

g. void MeshTask(void* pvParameters)

This function handles broadcasting gas sensor data over the mesh network. At every iteration, it declares a variable to store data retrieved from the queue. It will block until the data is available. Once it receives the data, the program will broadcast the gas sensor data to all mesh nodes.

```
if (xQueueReceive(MQ2Queue, &receivedData, portMAX_DELAY) ==
pdTRUE) {
        String msg = String(receivedData.gasvalue);
        mesh.sendBroadcast(msg);
                Serial.printf("Broadcast  message:  %s\n",
msg.c_str());
    }
```

2.2.2. Root Program

The root program is responsible for managing the entire system's operations, including receiving gas sensor data from leaf nodes, handling alarm notifications, activating and deactivating alarms based on gas levels, and sending messages through WhatsApp.

Here is the explanation of the functions and tasks used in the program:

a. setup().

This function initializes components for the program to function. It begins by setting up serial communication at a baud rate of 115200, then configures the pins for the LED and buzzer as output. This function registers a callback to handle Wi-Fi connection events and starts the process of connecting to the specified Wi-Fi network. The mesh network is also initialized, allowing the ESP32 to communicate with other nodes. A timer is created with a specified timeout period, and a task is created for handling dangerous gas detection.

b. loop().

This function continuously updates the mesh network to handle communication between nodes and process incoming messages.

c. WiFiStationConnected(WiFiEvent_t event, WiFiEventInfo_t info).

This function is a WiFi event handler that is triggered when the device successfully connects to a WiFi network. The output from this function is a confirmation message that is displayed to the serial monitor.

d. receivedCallback(uint32_t from, String& msg).

This function handles incoming messages from other nodes in the mesh network and priority inversion.

e. taskHandleGas(void *pvParameters).

This task handles the detection of hazardous gas levels. Here is the workflow of the task:

- Continuously check the gas value.
- If the gas value is higher than the danger threshold (1200) and the previous condition were normal, it triggers the alarm by calling activateAlarm():

```
if (gasValue > 1200) {
  if (previousGasState == 0) {
    activateAlarm();
    ...
  }
}
```

sends a WhatsApp notification:

```
sendMessage("Gas berbahaya terdeteksi!");
```

- If the gas value is normal, it checks if the timeout timer is active. If it's not, it starts the timer to deactivate the alarm after 5 minutes:

```
else {
  if (previousGasState == 1) {
```

```
    if (!xTimerIsTimerActive(xTimeoutTimer)) {
       xTimerStart(xTimeoutTimer, 0);
       Serial.println("\nMemulai timer untuk mematikan
alarm!");
    }
    ...
  }
}
```

- Send a notification indicating that the gas is no longer dangerous:

```
sendMessage("Gas berbahaya tidak terdeteksi!");
```

f. activateAlarm().

This function activates the alarm system. It turns on the LED and buzzer to signal that hazardous gas has been detected.

g. sendMessage(String message).

This function sends a WhatsApp notification using the CallMeBot API. Here is the workflow of the function:

- Construct the URL with the message and required credentials (mobile number and API key):

```
String url =
"https://api.callmebot.com/whatsapp.php?phone=" +
MobileNumber + "&text=" + urlEncode(message) +
"&apikey=" + APIKey;
```

- Use the HTTPClient library to send a POST request to the API:

```
HTTPClient http;
http.begin(url, true);
int httpResponseCode = http.POST(url);
```

- If the response code is 200, it confirms that the notification was successfully sent:

```
if (httpResponseCode == 200) {
```

```
  Serial.println("Berhasil mengirimkan notifikasi");
}
```

If there is an error, it prints the error response code:

```
else {
  Serial.println("Terjadi kesalahan saat mengirimkan notifikasi");
   Serial.print("HTTP response code: ");
   Serial.println(httpResponseCode);
}
```

h. vTimeoutCallback(TimerHandle_t xTimer)

This is a callback function for the timeout timer. When the timeout occurs, it checks if the taskHandleSafe is already created. If not, it creates the taskHandleSafe task to deactivate the alarm. This ensures that the alarm is turned off after the gas is no longer detected.

i. taskHandleSafe(void *pvParameters)

This task is responsible for deactivating the alarm. Here is the workflow of the task:

- Turn off the LED and buzzer when the hazardous gas no longer detected:

```
digitalWrite(LED_PIN, LOW);
digitalWrite(BUZZER_PIN, LOW);
```

- Send a WhatsApp notification, saying that the hazardous gas no longer detected:

```
sendMessage("Gas berbahaya sudah tidak terdeteksi!");
```

- This task is deleted once the alarm is turned off:

```
xTaskHandleSafe = NULL;
vTaskDelete(NULL);
```

2.2.3 Flowchart

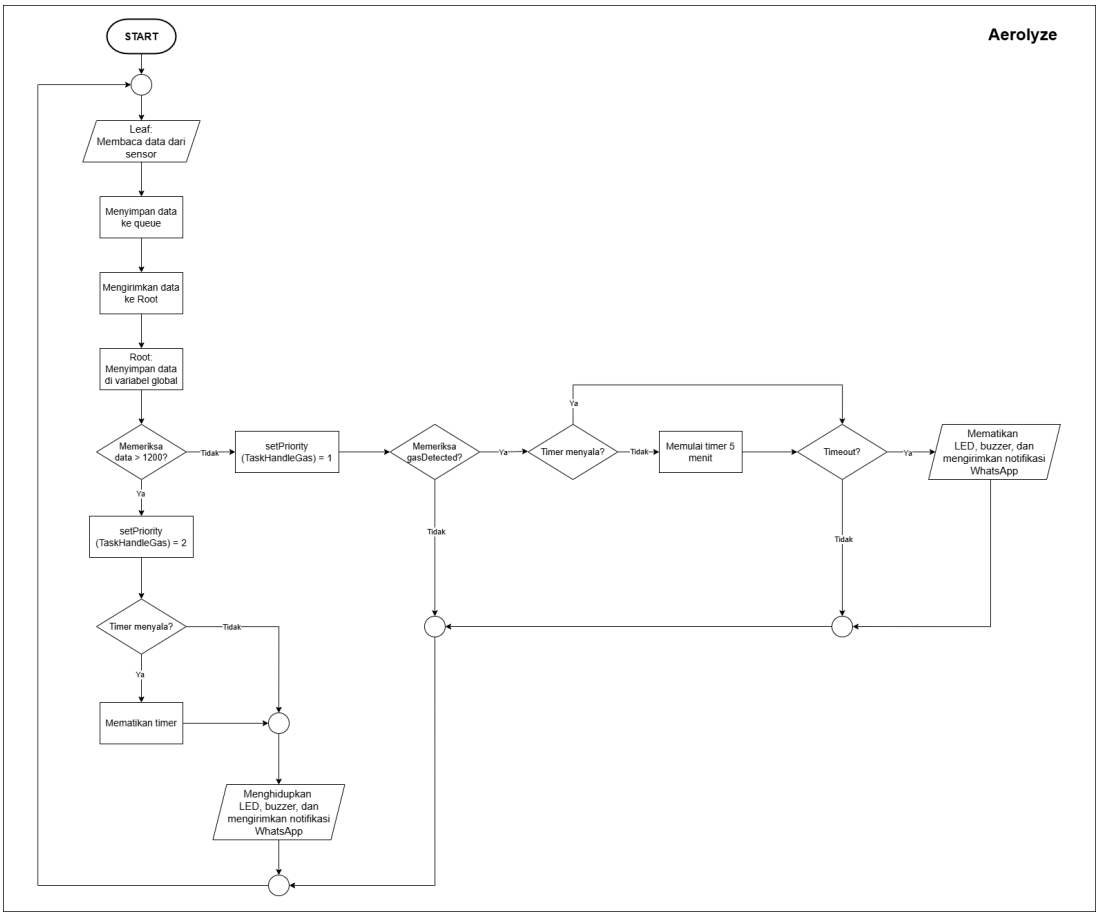


Fig. 3 Flowchart

2.3 HARDWARE AND SOFTWARE INTEGRATION

The integration of hardware and software in this system ensures seamless communication between the master node and the slave node in the mesh topology to send data of the gas sensor and alert the users of dangerous gas leaks. The components are interconnected to enable accurate gas monitoring, real-time data processing, and the notification of gas alerts.

The hardware is centered around the MQ-2 gas sensor, which is responsible for detecting hazardous gases like LPG and carbon monoxide (CO). It is connected to an ESP32 that functions as the master. The key components in our system are:

1. **MQ-2 Gas Sensor**

The sensor detects the presence of gas. The analog output of the sensor is connected to the analog pin of the microcontroller for continuous data sampling

2. **ESP32**

   The ESP32 serves as the master and the slave node. Both of them have a Wifi module which enables wireless communication using painless mesh. This also makes the notification alert possible through WhatsApp as the master node establishes a connection with the internet.

3. **Buzzer and LEDs**

   These components act as alerts to notify the user when the gas concentration exceeds the threshold.

4. **Power Supply**

   The master node is connected to a laptop for the user to be able to read the serial monitor. Meanwhile the slave node is connected to a power bank to ensure the transfer of the sensor data to the master node.

The software in this project acts as the bridge between the hardware, the mesh network, and the user notification system. It allows the gas sensor, microcontroller, and alert mechanisms to work together seamlessly and efficiently. The program is divided into two main programs, the leaf node and the root node. These nodes work together to detect hazardous gasses, send alerts, and manage the alarm system. The mesh topology in this system is supported by the painlessMesh library. After the two nodes have successfully established a connection, testing is conducted to ensure that the two nodes can communicate and are able to work according to each of their responsibilities.

CHAPTER 3

TESTING AND EVALUATION

3.1 TESTING

The purpose of testing was to ensure the functionality and reliability of the IoT system in detecting hazardous gas and sending appropriate notifications. The testing process aimed to validate the system's individual components (unit testing) and its combined operation (integration testing).

Unit Testing

The unit testing focused on verifying the functionality of the system's individual components. The leaf node was tested to ensure the sensor could accurately detect and read hazardous gas levels when exposed to smoke. Meanwhile, the root node was tested to confirm it could process the data from the sensor and send notifications to WhatsApp without errors. Each component was tested separately to isolate potential issues and validate that they performed as expected in controlled conditions.

Integration Testing

Integration testing was performed by combining the leaf and root nodes into a single system and running the program end-to-end. The system was exposed to smoke to simulate hazardous gas, and the combined components successfully detected the gas and sent WhatsApp notifications.

All tests were carried out using the Arduino IDE and the actual hardware setup. To simulate hazardous gas, smoke generated by burning paper was used. This setup allowed for a realistic environment to validate the sensor's responsiveness and the system's ability to handle real-time data and notifications effectively.

3.2 RESULT

The testing confirmed the functionality of both the leaf and root nodes, validating their roles in the IoT system.

Leaf Node

- The leaf node successfully initiated the mesh network.

```
setLogLevel: ERROR | STARTUP | CONNECTION |
STARTUP: init(): 1
STARTUP: AP tcp server established on port 5555
```

- The leaf node starts scanning for connections to other nodes in the same network.

```
CONNECTION: eventScanDoneHandler: ARDUINO_EVENT_WIFI_SCAN_DONE
CONNECTION: scanComplete(): Scan finished
CONNECTION: scanComplete():-- > Cleared old APs.
CONNECTION: scanComplete(): num = 3
CONNECTION:     Found 0 nodes
CONNECTION: connectToAP(): No unknown nodes found scan rate set to normal
```

- When detecting another node, the leaf node established a stable connection to form a mesh network.

```
CONNECTION: New AP connection incoming
CONNECTION: painlessmesh::Connection: New connection established.
CONNECTION: newConnectionTask():
CONNECTION: newConnectionTask(): adding 3378394273 now= 76302849
```

Here, you can see the number of nodes connected to the leaf node.

```
CONNECTION: eventScanDoneHandler: ARDUINO_EVENT_WIFI_SCAN_DONE
CONNECTION: scanComplete(): Scan finished
CONNECTION: scanComplete():-- > Cleared old APs.
CONNECTION: scanComplete(): num = 3
CONNECTION:     found : Mesh_Kelompok_10, -41dBm
CONNECTION:     Found 1 nodes
CONNECTION: connectToAP(): No unknown nodes found scan rate set to normal
```

- The leaf node accurately read the sensor data from the MQ2 sensor and stored the readings in a queue for further processing.

```
MQ2 Sensor Value: 952.00
MQ2 data sent to queue successfully!
```

```
MQ2 Sensor Value: 1456.00
MQ2 data sent to queue successfully!
```

- The node then broadcasted the stored data from the queue to other connected nodes in the mesh network.

```
Broadcast message: 952.00
```

```
Broadcast message: 1456.00
```

Root Node

- The root node connected successfully to the Wi-Fi network, allowing the system to communicate with external devices and services.

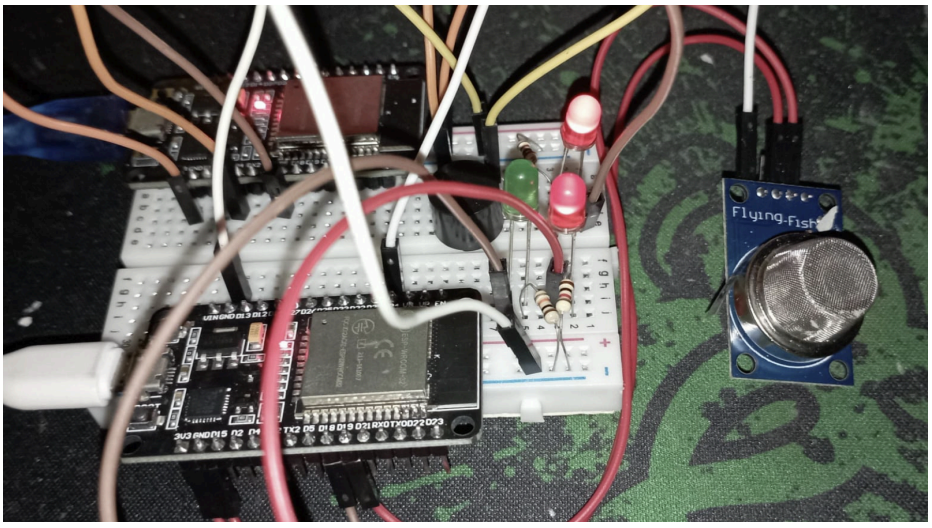
```
Menghubungkan ke WiFi dengan SSID: Redmi 10 2022
Berhasil terhubung dengan Redmi 10 2022
```

- When the root node receives data from the leaf node, and the value exceeds 1200, the root node will display a message indicating this value.

```
Pesan diterima dari Node ID 3206858717: 1456.00
```

The value of 1200 is the threshold. If the value exceeds this limit, it will be considered that hazardous gas has been detected.

- The system activated an alarm, including an LED light and buzzer, to alert users to the dangerous gas.

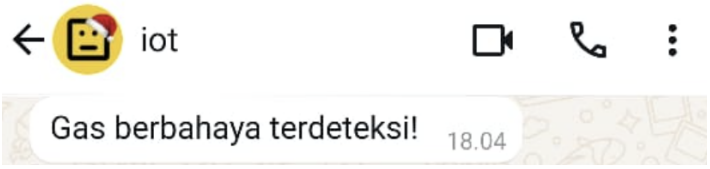



```
ALARM: Gas berbahaya terdeteksi!
```

- At the same time, the root node sent a WhatsApp notification to inform the user about the gas detection.

```
Berhasil mengirimkan notifikasi
```

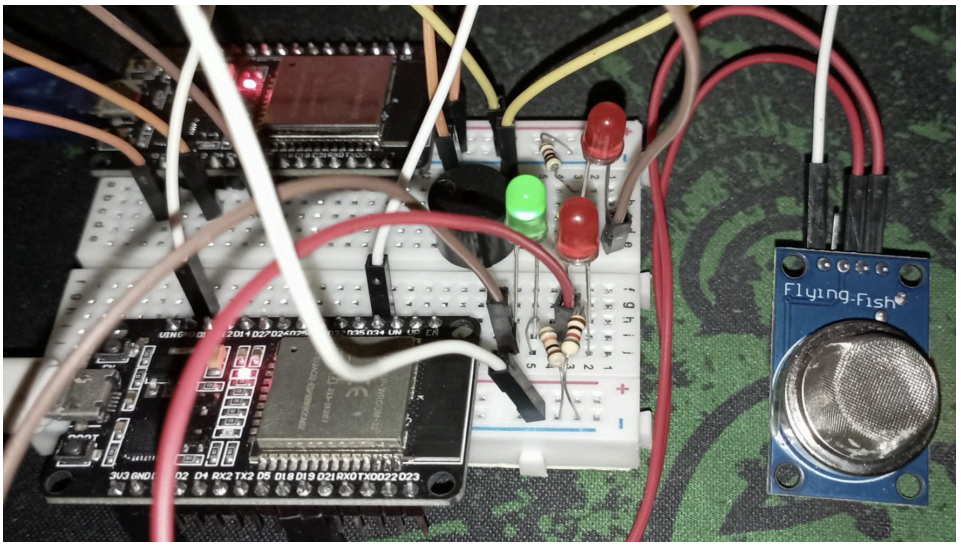
Here is the warning message received by the user via WhatsApp:



- If the gas was no longer detected, the root node started a 5-minute timer to automatically turn off the alarm after that time.

```
Memulai timer untuk mematikan alarm!
```

- Once the timer ran out, the alarm was turned off, and a notification was sent to WhatsApp to confirm that the gas was no longer present.




```
Gas berbahaya tidak terdeteksi. Mematikan alarm!
Berhasil mengirimkan notifikasi
```

Here is the message received by the user via WhatsApp:

- If the gas was detected again before the timer expired, the root node reset the timer and continued to monitor the gas levels.

```
Gas berbahaya kembali terdeteksi!
Mematikan timer
```

In conclusion, the system performed as expected during testing. The leaf and root nodes effectively communicated within the mesh network, with the leaf node accurately detecting hazardous gas levels and the root node processing the data and triggering appropriate actions, such as activating alarms and sending notifications to WhatsApp. Overall, the system demonstrated its functionality in detecting dangerous gas concentrations and alerting the user in real-time, confirming that the components worked as intended.

3.3 EVALUATION

Based on the testing and results, the system successfully met the functional requirements, with the leaf node accurately detecting hazardous gas, while the root node activating the alarm and sending notifications to WhatsApp. The integration testing confirmed that the components worked seamlessly together, with real-time alerts being triggered upon smoke exposure. However, there is room for improvement in the system's response time, particularly in varying environmental conditions. Occasionally, the notification system failed to send alerts due to the API server being busy, which impacted reliability. Further optimization of the notification handling could enhance the system's performance in more complex real-world scenarios.

CHAPTER 4

CONCLUSION

The Aerolyze is an advanced and innovative solution to enhance household safety by detecting hazardous gas leaks using the MQ-2 sensor and IoT technology. The system monitors gases such as LPG and carbon monoxide, which can pose significant risks to health and safety. The project utilizes the ESP32 microcontroller, a mesh network for efficient communication, and a dual-alert mechanism with a buzzer and LED to alert users when dangerous gas levels are detected. Additionally, Aerolyze sends real-time notifications via WhatsApp, ensuring users stay informed even when they are not at home.

The software design is divided into the Leaf Program and Root Program, which handle data collection, processing, and communication. The Leaf Program is responsible for reading the gas sensor data and sending it to the root node, while the Root Program manages the alarm system and notifications. The modular structure of the code, using FreeRTOS, allows for efficient parallel task processing, ensuring smooth operation and scalability. The integration of the mesh network and IoT platform enables seamless communication and real-time notifications, enhancing the overall system's effectiveness.

In our testing, Aerolyze performed well and successfully addressed the issue of detecting hazardous gas leaks in households by combining advanced sensor technology, a reliable notification system, and IoT capabilities. While the system meets functional requirements, improvements in notification reliability and response time would enhance its performance for a more reliable and efficient solution in diverse environments.

REFERENCES

[1] “How does MQ-2 flammable gas and smoke sensor work with Arduino?,” Arduino MQ-2 Gas Sensor Tutorial - How Does a Gas Sensor Work and How to Interface it with Arduino? [Online]. Available: https://circuitdigest.com/microcontroller-projects/interfacing-mq2-gas-sensor-with-arduino. [Accessed: 09-Dec-2024]

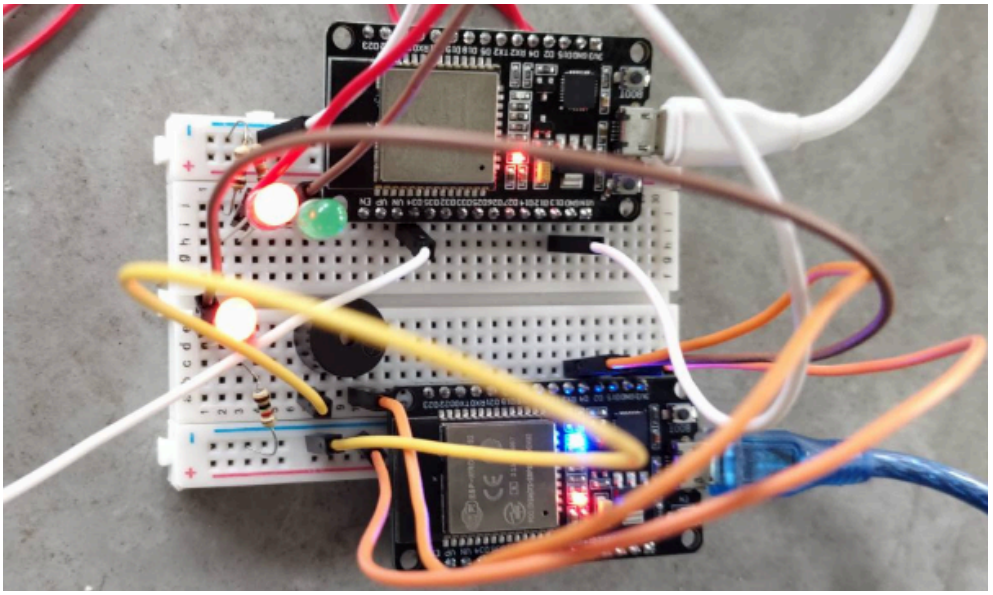
[2] Jimswen and mKwene, “Guide for MQ-2 gas sensor with Arduino,” Random Nerd Tutorials, 02-Apr-2019. [Online]. Available: https://randomnerdtutorials.com/guide-for-mq-2-gas-smoke-sensor-with-arduino/. [Accessed: 09-Dec-2024]

[3] Mehmet and Kobi, “ESP32: Send messages to WhatsApp,” Random Nerd Tutorials, 15-Mar-2024. [Online]. Available: https://randomnerdtutorials.com/esp32-send-messages-whatsapp/. [Accessed: 09-Dec-2024]

[4] “ESP32 - piezo buzzer,” ESP32 Tutorial. [Online]. Available: https://esp32io.com/tutorials/esp32-piezo-buzzer. [Accessed: 09-Dec-2024]

APPENDICES

Appendix A: Project Schematic



Appendix B: Documentation