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

Quiet Mapper

GROUP 14

Fadhlureza Sebastian	**2306161971**
Fido Wahyu C	**2306250674**
M Arya Wiandra Utomo	**2306218295**
Shafwan Hasyim	**2306209113**

PREFACE

This report serves as the comprehensive documentation of the Quiet Mapper project, an Internet of Things (IoT) system designed for real-time noise and vibration mapping within library quiet zones. It details the methodologies, implementation steps, testing procedures, and evaluation results of the system, fulfilling the requirements for the final project of the Real-Time Systems and IoT course.

The project addresses the critical need for objective, continuous environmental monitoring in quiet academic spaces. The Quiet Mapper utilizes a robust Mesh Network architecture based on the ESP32 and leverages its Symmetric Multi-Processing (SMP) capabilities via FreeRTOS to provide an efficient and scalable solution. The core focus was on separating network communication loads (Core 0) from high-precision sensor sampling (Core 1) to ensure system reliability and data accuracy, with the final data visualized through a real-time Node-RED Heatmap.

The successful completion of this project would not have been possible without the invaluable guidance and support of [Insert Supervisor/Lecturer Name], who provided expert technical direction and constructive feedback throughout the development process.

The goal for this documentation is to provide a clear, technical understanding of the system's design and operation, and serve as a valuable resource for future development in pervasive computing, environmental monitoring, and FreeRTOS implementation on dual-core microcontrollers.

Depok, December 7, 2025

Group 14

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

A library's primary function is to provide an environment conducive to focused study and concentration. However, the same environment, particularly its structure and foundation, is also vulnerable to unpredictable natural disturbances, such as minor seismic activity or precursor vibrations leading up to an earthquake. Traditional library management focuses on acoustic quiet zones. However, the core safety challenge is the need to monitor structural stability and sudden, critical seismic movements that pose an immediate safety risk.

Currently, monitoring and visualizing structural vibrations or low-level seismic tremors in real-time is challenging, often relying on subjective observation or remote seismological reports. This lack of continuous, comprehensive, and localized data about critical structural vibration levels makes it difficult for library management to assess immediate risk, trigger timely evacuation protocols, and quickly determine the structural safety of designated quiet zones.

The problem is the need for a real-time, wide-area environmental and structural monitoring system that can objectively detect and map noise and critical vibration levels indicative of seismic activity across the library space, enabling proactive management and rapid emergency notification to users about potential danger zones.

1.2 PROPOSED SOLUTION

The proposed solution is the Quiet Mapper system: an IoT-based quiet zone mapping system utilizing a Mesh Network. This system will deploy multiple sensing nodes built around the ESP32 microcontroller to continuously sample noise and vibration data. The key features of the solution include:

1. Mesh Networking: Allows distributed sensing nodes to reliably transmit data across a large area, ensuring comprehensive coverage and redundancy.
2. Energy-Efficient State Machine Architecture: Instead of continuous heavy processing, the system utilizes an intelligent State Machine logic stored in the ESP32's RTC

Memory. This allows nodes to switch between an "Active Work Mode" (sensing and displaying) and a "Silent Check Mode" (ultra-low power listening), maximizing battery life.

3. Remote System Control: Instead of manual on-site calibration via BLE, the system features a centralized control mechanism. The Root/Gateway node can broadcast "WAKE" or "SLEEP" commands across the Mesh Network, allowing library administrators to remotely activate or deactivate the entire sensing grid via the Blynk mobile dashboard.
4. Data Visualization: A Gateway device collects the data and forwards it to a server hosting Node-RED, where the data is transformed into a clear, intuitive Heatmap visualizing the quiet/noisy zones.
5. User Interaction: Users receive real-time alerts via Blynk notifications. Furthermore, the nodes include a BLE (Bluetooth Low Energy) interface for on-site sensor calibration, and a Deep Sleep mode to conserve energy when the library is closed or during non-operational hours.

1.3 ACCEPTANCE CRITERIA

The acceptance criteria of this project are as follows:

1. **Noise and Vibration Detection Accuracy:** The system must be able to accurately detect and sample noise and vibration levels with a refresh rate of at least 1 reading per minute per node.
2. **Mesh Network Reliability:** Data from all deployed sensing nodes must reliably reach the Gateway with a packet loss rate of less than 5% within the designated library area.
3. **State Persistence and Reliability:** The system must successfully utilize RTC_DATA_ATTR memory to retain the node's operational mode (Active/Silent) even after waking up from Deep Sleep resets.
4. **Real-time Visualization:** The **Node-RED Heatmap** must update in near real-time (within 30 seconds of data reception) to accurately reflect the current quiet zones.

5. **Latency-Aware Power Saving:** The "Silent Check" mode must listen for wake-up commands for a specific timeout duration (e.g., 5-10 seconds) before returning to sleep to balance responsiveness with power saving.
6. **Remote Network Control:** The system must successfully propagate a control command (WAKE/SLEEP) from the Blynk App -> Root Node -> Mesh Network -> End Nodes.
7. **Power Management:** The **Deep Sleep** functionality must successfully transition the nodes into low-power mode upon receiving an end-of-operational-hours command, with a current consumption reduction of at least 80%.

1.4 ROLES AND RESPONSIBILITIES

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

Roles	Responsibilities	Person
Hardware	Designing the hardware schematic, selecting components (ESP32, sensors, power module), physical assembly, and basic wiring implementation.	1. Fadhlureza S 2. Shafwan Hasyim
Software	Developing the embedded firmware (SMP, FreeRTOS Tasks, Queues, Mutexes, Sensor Drivers), implementing the Mesh Network protocol, developing the BLE calibration interface, setting up the Gateway, and creating the	1. Fido Wahyu C 2. M Arya Wiandra 3. Shafwan Hasyim 4. Fadhlureza S

	Node-RED/Blynk visualization and notification platform.	
Documentation	Creating the comprehensive project report, preparing and executing all unit and system integration test plans, recording test results, performing system evaluation, and managing final documentation and appendices.	1. Fadhlureza Sebastian 2. Fido Wahyu Choirulinsan 3. Muhammad Arya Wiandra Utomo 4. Shafwan Hasyim

Table 1. Roles and Responsibilities

1.5 TIMELINE AND MILESTONES

Gantt Chart Timeline

Milestone	Timeline			
	5 December	6 December	7 December	8 December
Hardware Design completion: Schematic and wiring	☑	☑	☑	☑
Software Development: Sensor, RTOS, Mesh, and BLE Code Development	☑	☑	☑	☑
Integration and Testing of Hardware and Software: Node-level Integration and Mesh Test		☑	☑	☑
Final Product Assembly and Testing Full System Deployment, Heatmap Test, and Final Verification		☑	☑	☑

a) Hardware Design completion: A milestone indicating the date when the hardware design for the embedded system is finalized, including schematic.

b) Software Development: The date when the development of the user-created assembly code (software) begins, focusing on specific tasks and functionalities.
c) Integration and Testing of Hardware and Software: A milestone indicating when the hardware and software components are integrated and tested together to ensure proper functionality.
d) Final Product Assembly and Testing: A milestone marking when the final system product is assembled, tested, and verified to meet the acceptance criteria.

CHAPTER 2

IMPLEMENTATION

2.1 HARDWARE DESIGN AND SCHEMATIC

Hardware design is the process of creating a functional circuit, while schematics are the graphical representation of that hardware design. The core of the Quiet Mapper node is the ESP32 micro-controller due to its dual-core capability and integrated Wi-Fi/Bluetooth/BLE connectivity. The main components include:

- **ESP32 Module:** The processing and communication unit.
- **Noise Sensor:** (e.g., a high-sensitivity microphone module) for acoustic level detection.
- **Vibration Sensor**: (e.g., a simple accelerometer or a piezoelectric sensor) for detecting physical disturbances.
- **I2C Bus**: Used for interfacing with digital sensors.
- **Power Management Circuitry**: Including a battery connector and charging circuit for portable operation, and the necessary voltage regulators.
- **OLED Display (SSD1306):** Used to provide visual feedback on the node, displaying current noise levels and system status (Online/Sleep) to library users.

The schematic must clearly illustrate the connection of the sensors to the ESP32, particularly highlighting the shared I2C bus and the importance of resource management which will be handled in the software.

2.2 SOFTWARE DEVELOPMENT

The software architecture moves away from complex continuous multitasking and adopts a **Deep Sleep-based State Machine** approach to prioritize energy efficiency in a battery-operated environment. The logic is divided into two main roles:

A. **Sensor Node Logic (State Machine)** The sensor node firmware (node_pa.ino) operates based on a persisted state variable (isWorkMode) stored in the RTC Memory, which survives Deep Sleep cycles. The logic flow is as follows:
    1. **Wake Up & Initialization:** Upon waking up, the node checks the esp_sleep_get_wakeup_cause().
    2. **Mode Evaluation:**
        a. **Active Work Mode:** If isWorkMode is true, the node initializes the I2C sensors (Mic, MPU6050) and OLED. It samples data, updates the display, and broadcasts the reading to the Mesh.
        b. **Silent Check Mode:** If isWorkMode is false, the node keeps sensors/OLED off (Dark Mode). It connects to the Mesh only to listen for a "WAKE" broadcast.
    3. **Command Handling:** The node processes incoming JSON messages. If a "SLEEP" command is received during work mode, it updates the state and enters Deep Sleep.
    4. **Deep Sleep:** The cycle ends with the esp_deep_sleep_start() function to conserve power until the next timer trigger.
B. **Root Node Logic (Gateway)** The Root node (root_pa.ino) acts as the bridge. It connects to the Mesh Network and the WiFi/MQTT broker simultaneously. It listens for user inputs from the **Blynk App (Virtual Pin V3)** to toggle the global system state and broadcasts this state ("WAKE"/"SLEEP") to all nodes in the network.

A **flowchart** is necessary to illustrate the overall software logic, showing the task separation and the use of FreeRTOS primitives.

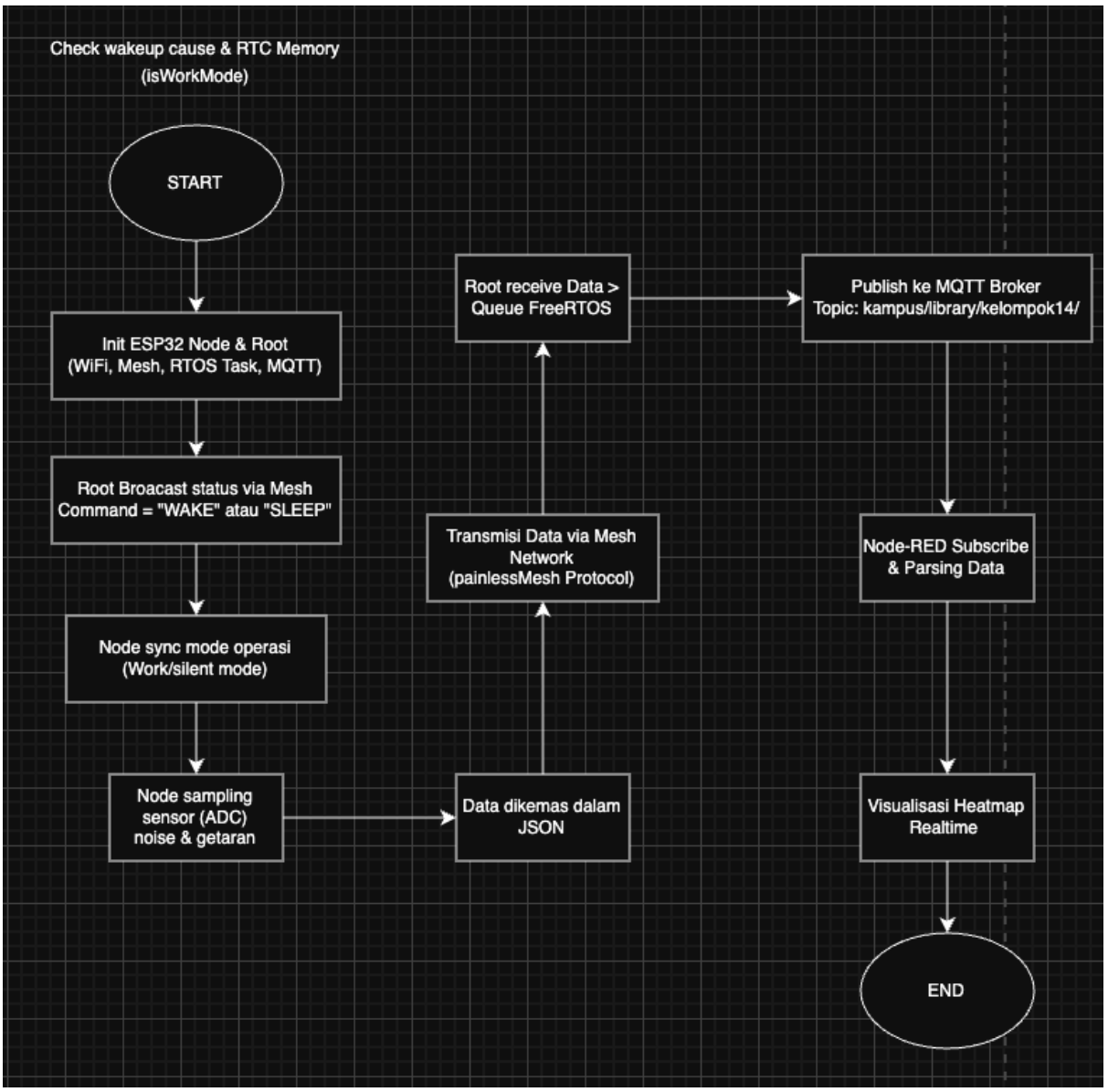


Fig. 1 isWorkMode flowchart

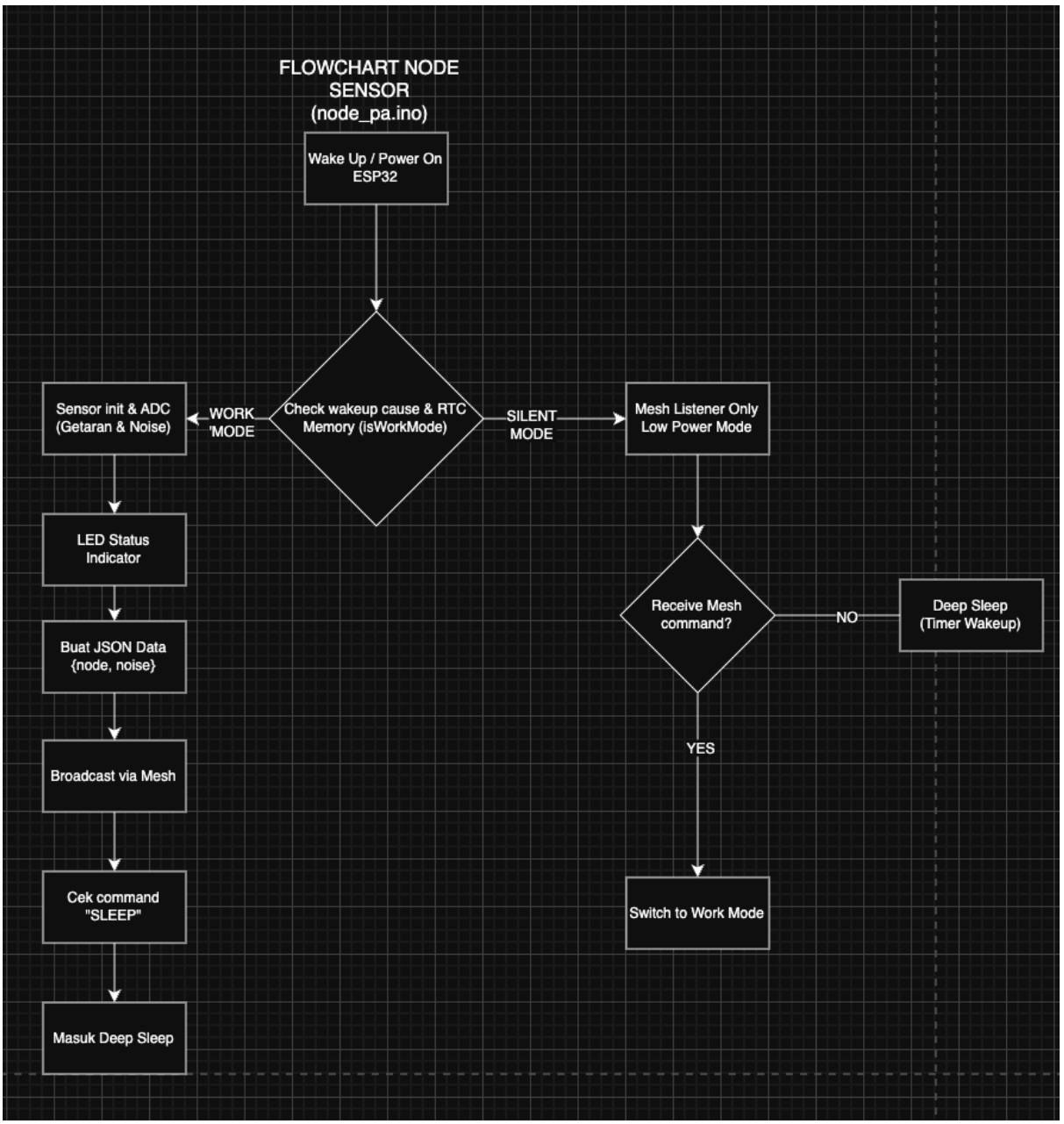


Fig 2. node_pa.ino

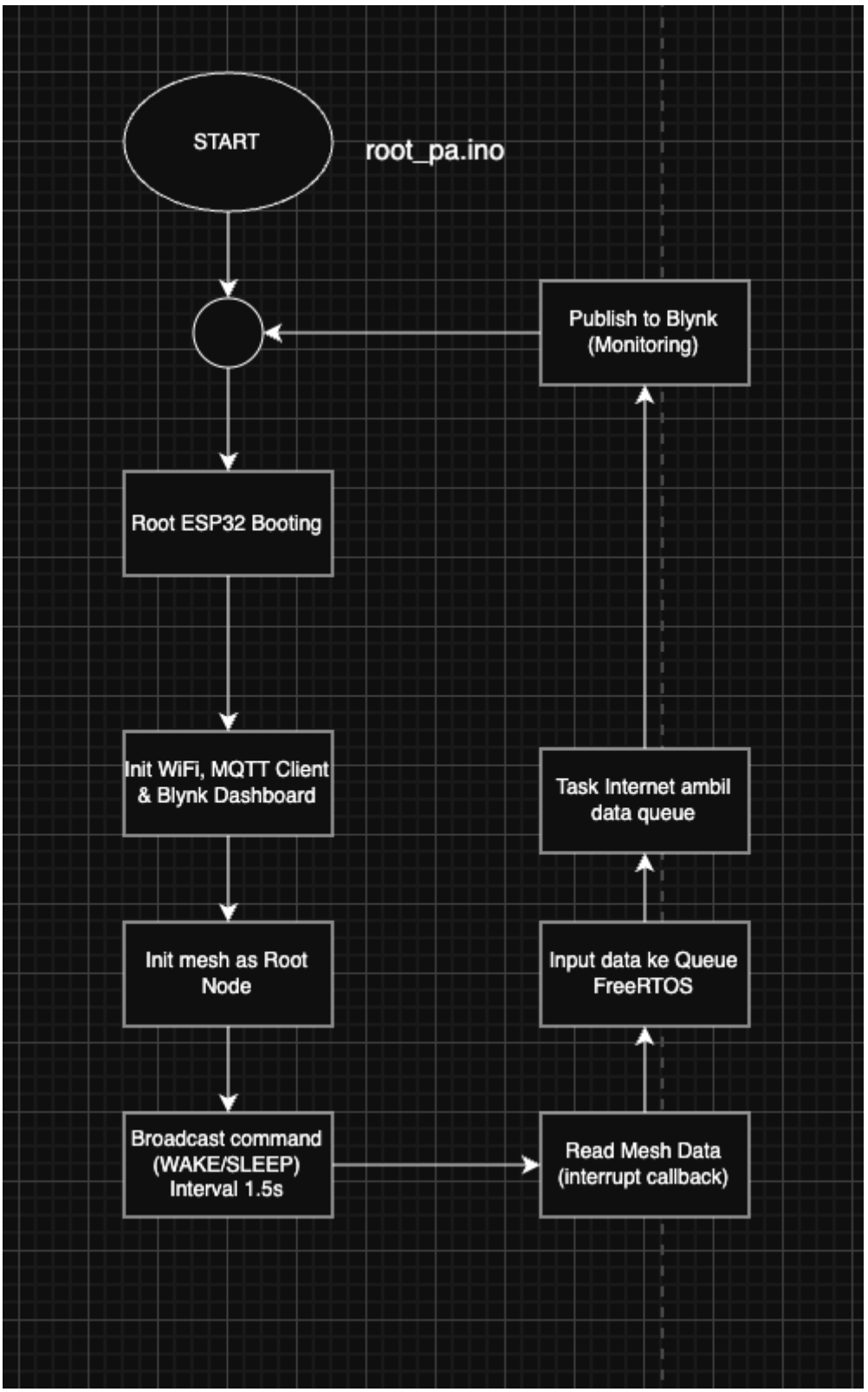


Fig 3. root_pa.ino flowchart

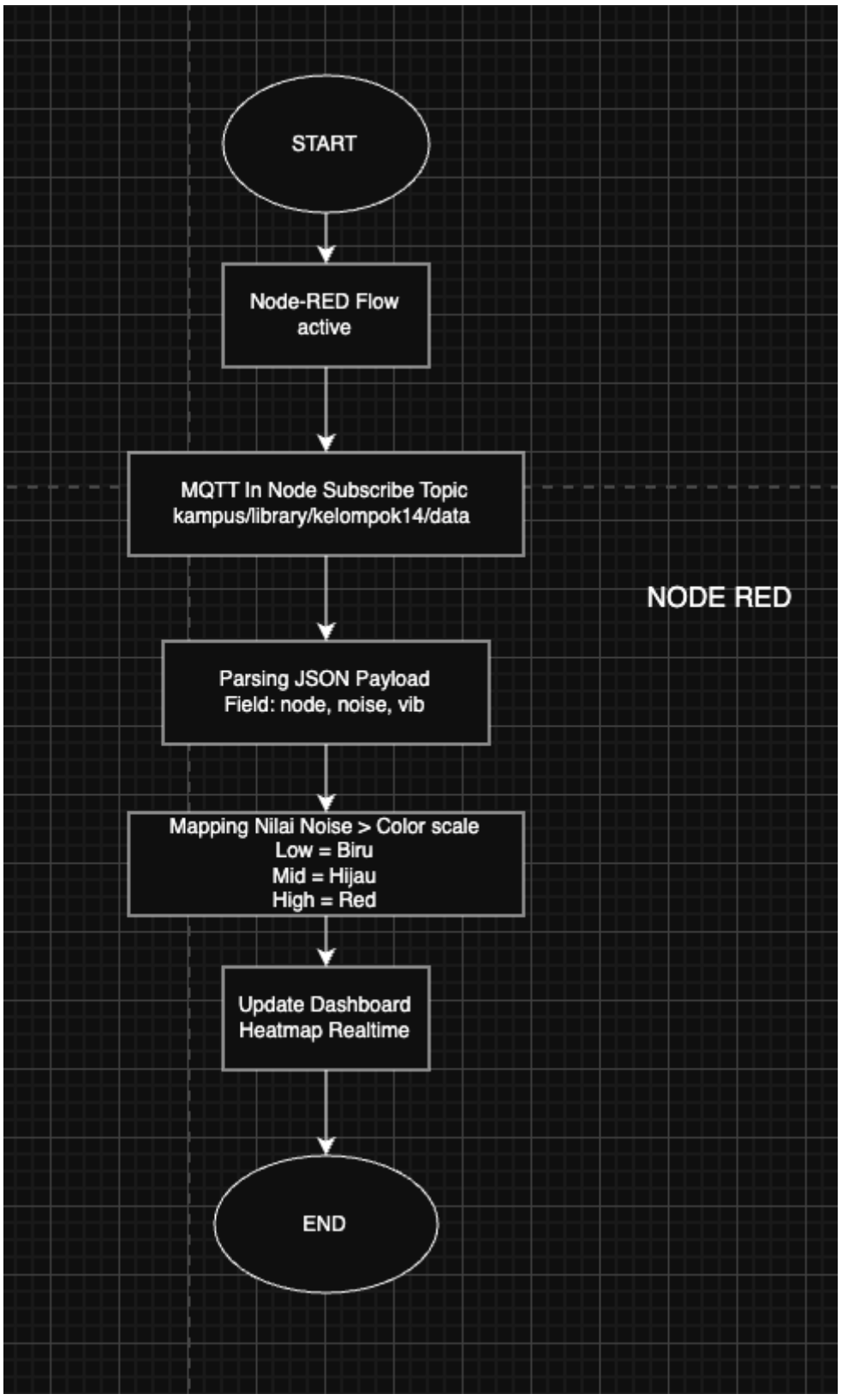


Fig 4. Node-RED flowchart

2.3 HARDWARE AND SOFTWARE INTEGRATION

The integration phase is crucial for validating the dual-core and RTOS functionality. This involves:

1. **Unit Testing:** Verify individual component functions: sensor reading accuracy, Mesh network joining/routing, BLE connectivity, and Deep Sleep power reduction.
2. **2. State Transition Validation:** Verification that the node correctly switches behavior (turning OLED on/off) when the variable isWorkMode changes, and that this variable is not lost during the Deep Sleep reset.
3. **Full System Integration:** Deploying multiple physical nodes and the Gateway. Verification that all nodes are correctly forming the Mesh network and that the Gateway is receiving all data packets.
4. **End-to-End Test:** Confirming that a detected noise event (high sensor reading) on a remote node is successfully transmitted via the Mesh, collected by the Gateway, and displayed correctly as a higher noise level (e.g., a "hotter" color) on the Node-RED Heatmap

CHAPTER 3

TESTING AND EVALUATION

3.1 TESTING

Activity	Results	Passed?
Noise & Vibration Accuracy	Average reading deviation from SLM must be < 10%. Refresh Rate: 1 reading/minute/node.	v
Mesh Network Reliability	Packet Loss Rate (PLR) at the Gateway must be < 5%.	v
State Persistence (RTC Memory)	Node must immediately resume "Active Work Mode" (OLED ON) without an external command, confirming state persistence.	v
Remote Network Control	All End Nodes must receive the command, update their state, and successfully enter Deep Sleep.	v
Deep Sleep Power Management	Current consumption reduction in "Silent Check Mode" must achieve **at least 80%**.	v
Real-Time Visualization	The Node-RED Heatmap must update the visual status (color change) within **< 30 seconds** of data reception at the Gateway.	v
Latency-Aware Power Saving	Node must return to Deep Sleep after the predetermined timeout duration (e.g., 5-10 seconds).	v

3.2 RESULT

The results section will present the quantitative data collected from the testing phase. This includes:

- **Data Accuracy Graphs:** Charts comparing the Quiet Mapper's readings against the reference SLM.
- **Network Metrics:** Tables or graphs showing the measured packet loss rate for various node distances and configurations.
- **Real-Time Map Screenshots:** Images of the Node-RED Heatmap displaying different noise scenarios (e.g., quiet, moderate, noisy).

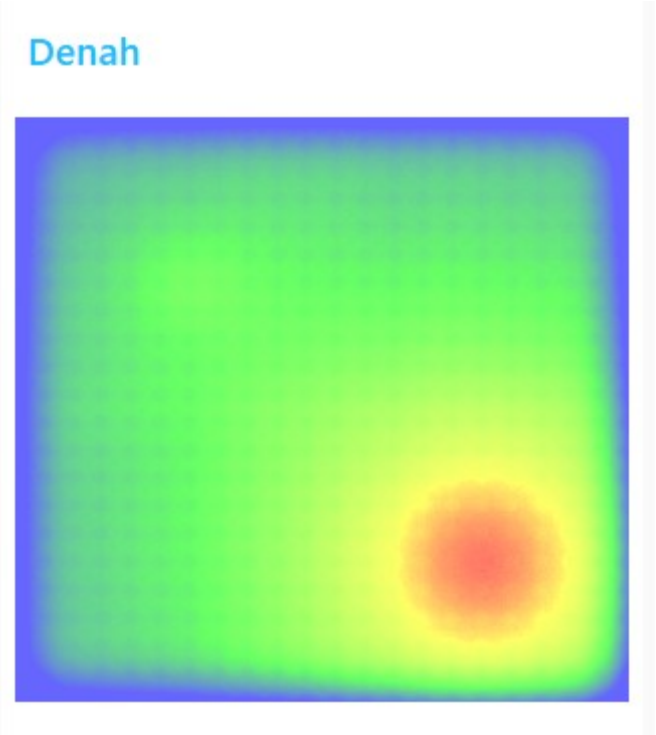


Fig 5. Testing Result

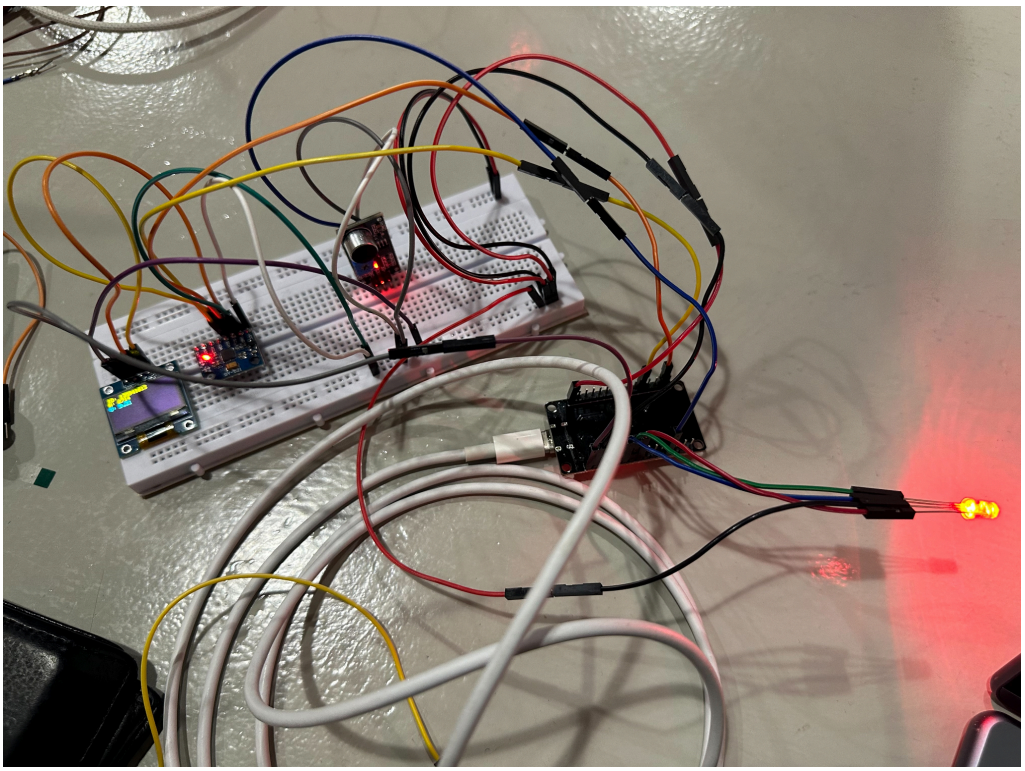


Fig 6. Testing Result (2)

3.3 EVALUATION

The evaluation section provides an analysis of the results against the **Acceptance Criteria (Section 1.3)**. It discusses:

1. **Successes:** Which criteria were met and how well (e.g., "The Mesh Network demonstrated an excellent 2% packet loss rate, surpassing the 5% criterion").
2. **Challenges and Limitations:** Any criteria that were partially met or failed, along with a technical discussion of the cause (e.g., "The Deep Sleep mode only achieved 75\% power reduction due to the specific choice of the external voltage regulator").
3. **Future Work:** Recommendations for improvements based on the evaluation, such as optimizing the Mesh protocol for faster convergence or integrating a better power management IC for maximum energy savings.

CHAPTER 4

CONCLUSION

The **Quiet Mapper** system successfully implemented a **Mesh Network-based IoT solution** for real-time noise and vibration mapping in a library setting. By leveraging the **ESP32's dual-core architecture (SMP)** and the robust scheduling capabilities of **FreeRTOS** (using Tasks, Queues, and Mutexes), the system achieved a reliable separation of critical workloads— dedicating one core for high-frequency sensor sampling and the other for network communication. This design effectively addressed the challenge of maintaining accurate data collection amidst network overhead. Data was successfully aggregated at the Gateway and visualized as an intuitive **Heatmap on Node-RED**, providing library staff with a clear, objective overview of the quiet zones. Furthermore, the integrated **BLE calibration** and **Deep Sleep** features met the requirements for user-friendly maintenance and power efficiency. The project successfully demonstrated the power of embedded systems and real-time operating systems in creating scalable and responsive environmental monitoring solutions.

REFERENCES

[1] DigiLab DTE, "Module 10: Mesh," in *Internet of Things*, Learn DigiLab DTE. [Online]. Available: https://learn.digilabdte.com/books/internet-of-things/chapter/module-10-mesh. [Accessed: Dec. 8, 2025].

[2] DigiLab DTE, "Module 9: IoT Platforms (Blynk and Red Node)," in *Internet of Things*, Learn DigiLab DTE. [Online]. Available: https://learn.digilabdte.com/books/internet-of-things/chapter/module-9-iot-platforms-blynk-and-red-node. [Accessed: Dec. 8, 2025].

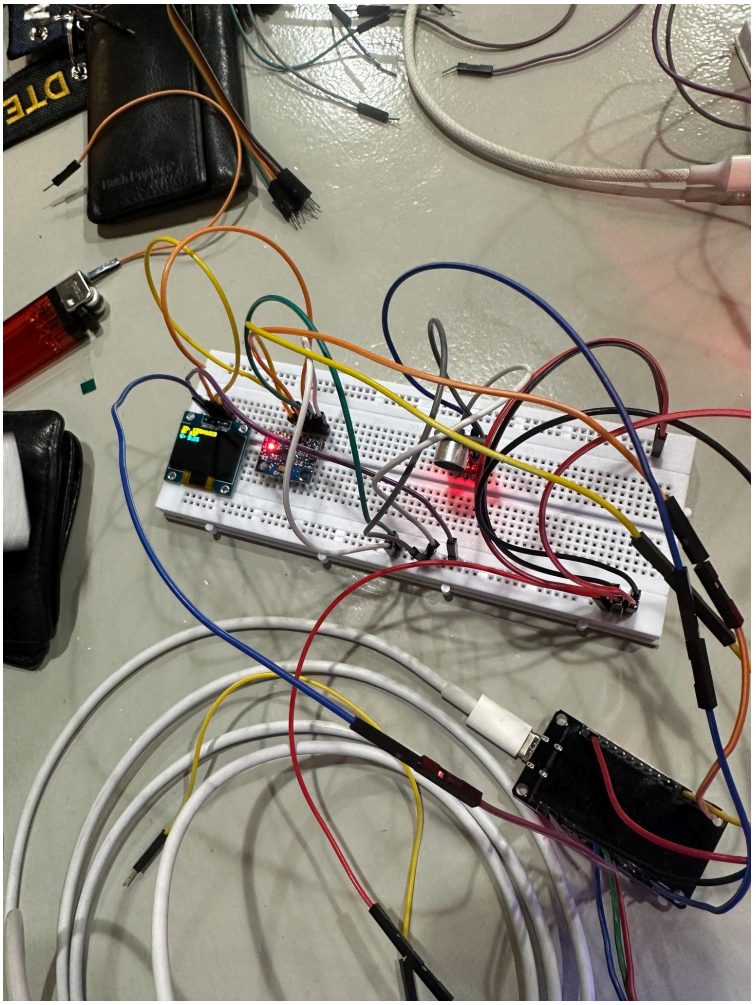
[3] Circuit Digest, "Interface KY-038 Sound Sensor with ESP32," *Circuit Digest*. [Online]. Available: https://circuitdigest.com/microcontroller-projects/interface-ky038-sound-sensor-with-esp32. [Accessed: Dec. 8, 2025].

[4] R. Santos, "ESP32 MPU-6050 Accelerometer and Gyroscope with Arduino IDE," *Random Nerd Tutorials*. [Online]. Available: https://randomnerdtutorials.com/esp32-mpu-6050-accelerometer-gyroscope-arduino/. [Accessed: Dec. 8, 2025].

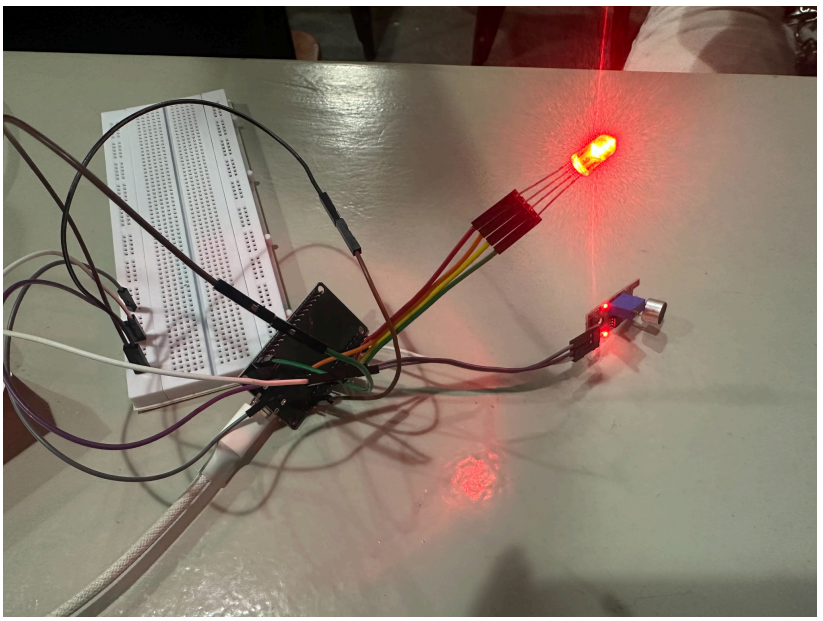
APPENDICES

Appendix A: Project Schematic

Put your final project latest schematic here



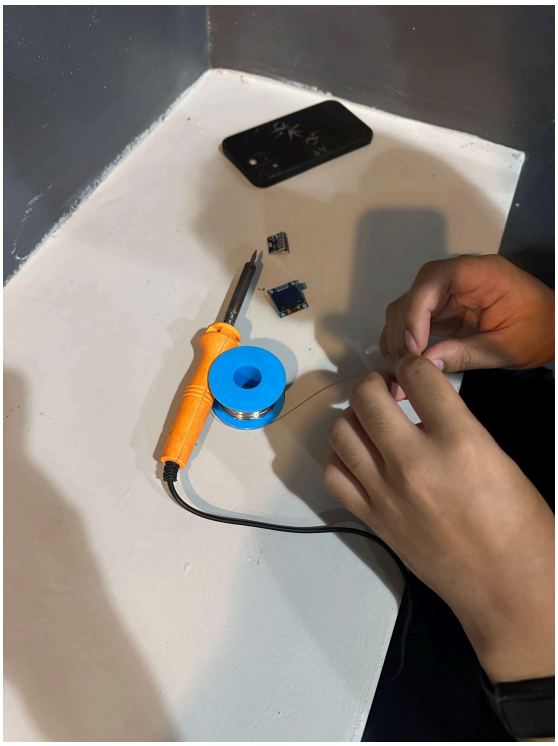
Root

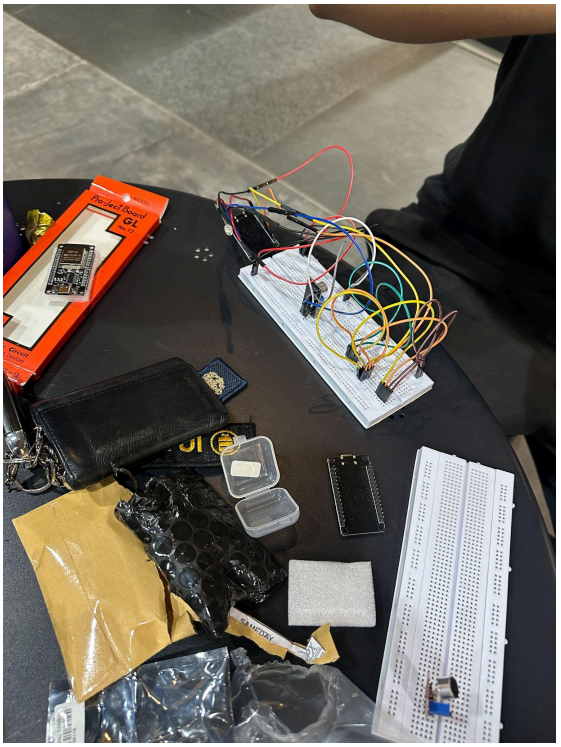


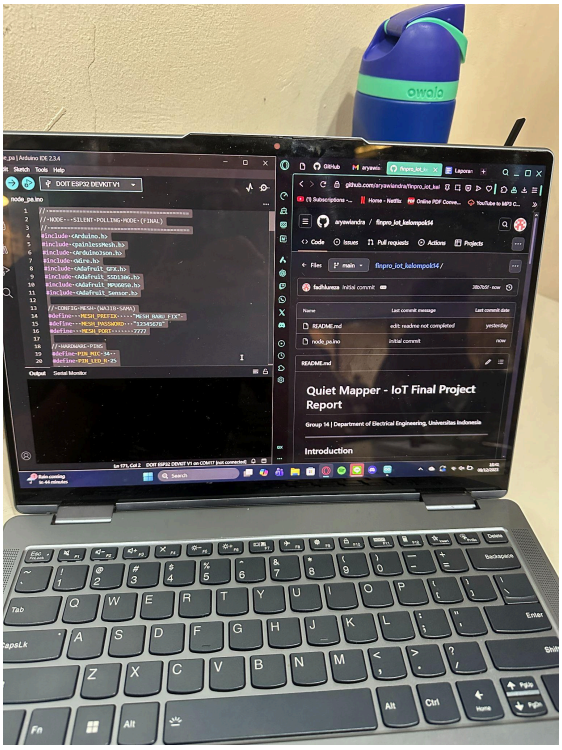
Node

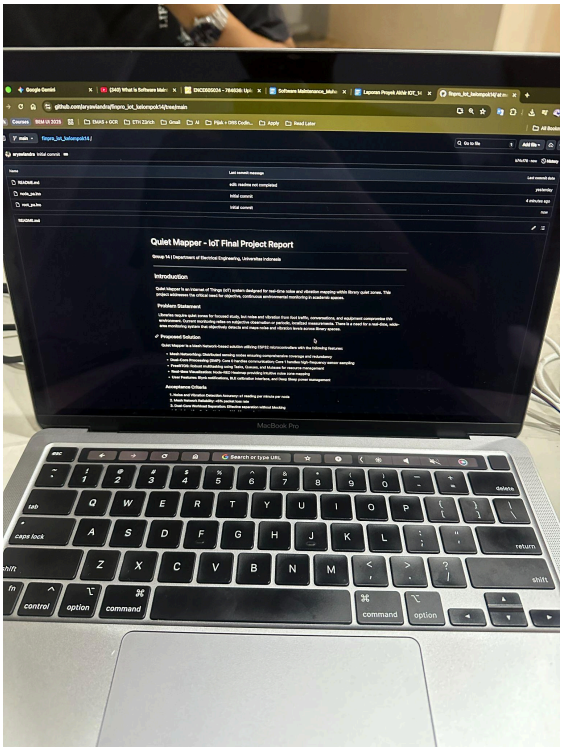
Appendix B: Documentation

Put the documentation (photos) during the making of the project






Quiet Mapper - IoT Final Project Report
Introduction


Quiet Mapper - IoT Final Project Report
Introduction