BOOTCAMP PROJECT REPORT

Project Title	:	**Autonomous Monitoring Control System in Agriculture and Fisheries Sector**
Job Desk	:	**Integrate the water cooling system (water cooler & water heater) into the existing system.**
Bootcamp Client	:	**Badan Riset dan Inovasi Nasional (BRIN)**
Concentration	:	**Internet of Things (IoT)**

By:

Alysa Milano (001202300089)

INFORMATICS STUDY PROGRAM
FACULTY OF COMPUTER SCIENCE
PRESIDENT UNIVERSITY

APPROVAL PAGE

INTERNSHIP REPORT

Autonomous Monitoring Control System in Agriculture and Fisheries Sector

By:
Alysa Milano
Student ID: 001202300089
Approved on: 27 May 2025

Approved by:

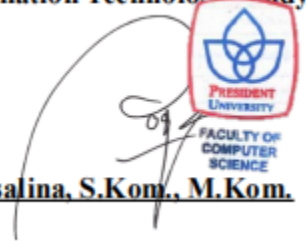
Institution Supervisor 1	Institution Supervisor 2
Azrizal Akbar, S.T.	Nashrullah Taufik, M.Sc

Internship Academic Lecturer

Williem, M.Kom

Acknowledged by,

Head of Information Technology Study Program



Rosalina, S.Kom., M.Kom.

STATEMENT LETTER OF INTERNSHIP REPORT WRITING

Autonomous Monitoring and Control System for Water Temperature in Aquaculture Using ESP32 and DS18B20 with ESP-NOW Communication

I hereby declare that this internship report is a genuine work that I independently prepared, wrote, and completed under the guidance and supervision of assigned mentors at Badan Riset dan Inovasi Nasional (BRIN). The report describes the design, development, and testing of a real-time temperature control system integrated into an existing Autonomous Monitoring and Control System (AMCS) using the ESP32 microcontroller, DS18B20 sensor, relay actuators, RTC module, and ESP-NOW protocol for data transmission.

All the research, analysis, technical development, and findings presented in this report were carried out during my internship at BRIN. Any use of third-party literature, code, or data is properly cited in accordance with academic and professional integrity standards.

This report has not been submitted previously for any academic course, thesis, final project, or publication, except where clearly mentioned and referenced. I understand and agree that this document may be used by President University or BRIN for plagiarism detection and academic evaluation purposes.

Furthermore, I acknowledge that any publication, distribution, or reproduction of this report or its contents for external or commercial purposes must receive prior written approval, particularly in relation to intellectual property rights, from Badan Riset dan Inovasi Nasional (BRIN).

Cikarang, 27 May 2025
Sincerely,

Alysa Milano
Student ID: 001202300089

ACKNOWLEDGMENT

First of all, I would like to express my deepest gratitude to God Almighty for giving me strength, health, and peace of mind to complete my internship and write this report. This report entitled "Autonomous Monitoring and Control System for Water Temperature in Aquaculture Using ESP32 and DS18B20 with ESP-NOW Communication" is the result of various learnings, trials, and reflections during the author's internship at the National Research and Innovation Agency (BRIN).

I would like to express my deepest gratitude to Mr. Azrizal Akbar, S.T. and Mr. Nashrullah Taufik, M.Sc. for their valuable guidance and support throughout the project. Their feedback and technical direction helped me improve my understanding and implementation of IoT-based systems in real-world settings. I would also like to thank Mr. Reza Septiawan, Head of the Safe Electronic Design Research Group, for the opportunity provided to participate in this program and for the research environment provided to us during the internship.

My deepest appreciation also goes to Mr. Williem, M.Kom and Mrs. Rosalina.S.Kom.,M.Kom, my academic advisor at President University, for his continuous encouragement, feedback, and assistance throughout the report writing process. To all parties involved in the internship program, supervisors and fellow interns. Thank you for creating a collaborative environment that allowed me to grow both technically and personally. I truly learned a lot through teamwork, discussions, and hands-on experiments.

This internship has been a valuable experience that not only strengthened my skills in embedded systems and automation, but also taught me how to adapt, solve problems independently, and take over responsibility for a project from start to finish.

I realize that this report may still have areas for improvement, and I welcome constructive suggestions. Hopefully this report can be a small contribution to the ongoing efforts in developing smart and sustainable aquaculture systems.

Cikarang, 27 May 2025
Sincerely,

Alysa Milano
Student ID: 001202300089

Table of Contents

List Of Tables

Table of Figures

ABSTRACT

This project focuses on the design and implementation of an autonomous temperature control system for aquaculture, developed as part of a larger Autonomous Monitoring and Control System (AMCS). The system uses an ESP32 microcontroller to monitor water temperature in real time via a DS18B20 sensor, and controls actuators including a water heater and cooler based on threshold values set by the user. A DS3231 real-time clock (RTC) module ensures time-based synchronization and logging.

Users can configure minimum and maximum temperature limits, as well as hysteresis values, via a serial interface. These settings are stored in EEPROM to ensure persistence after power loss. The system implements hysteresis control logic to prevent rapid switching of relays and enhance hardware stability. All sensor readings and actuator statuses are transmitted via ESP-NOW to a receiving node for integration with other modules, managed by a project collaborator.

The prototype was tested in both home and laboratory environments, showing stable performance in detecting temperature changes and activating relays accordingly. The communication between devices via ESP-NOW was reliable within a 100–200 meter range. This project demonstrates that a low-cost, flexible IoT-based control system can be successfully implemented in aquaculture settings and provides a solid foundation for future development and scaling.

INTRODUCTION

1.1 Background

For efficient fish production and high productivity in today's aquaculture industry, it is essential that the best environmental conditions prevail. The role of temperature plays a key part in the aquatic environment as it has a direct effect on fish metabolism, immunity, and overall health. For tilapia, for example, maintaining a constant water temperature between 26°C and 30°C is imperative for growth and survival.

The integration of smart systems with aquaculture systems has enabled more efficient and automated control and monitoring systems. One such system is the Autonomous Monitoring and Control System (AMCS), which is based on a synergy of communication modules, microcontrollers, and sensors to automatically regulate environmental conditions based on real-time data.

The project aims to integrate a water heating and cooling system with an existing AMCS platform utilizing ESP32 as the central controller. The system employs a DS18B20 temperature sensor to read the real-time water temperature and a DS3231 RTC module to provide accurate timestamps for logging and control logic. Based on the temperature reading, the system will switch on a water heater whenever the temperature goes below the minimum set point (e.g., 26°C) and the water cooler and the circulation pump whenever the temperature goes above the maximum set point (e.g., 30°C).

Apart from this, this system also has the provision for user-configured parameters through serial interface, such that parameters such as minimum and maximum temperature and actuator values of hysteresis, helpful in preventing rapid on-off switching of actuators. Real-time temperature values and actuator states are also transmitted through ESP-NOW to be incorporated in other modules governed by a partner colleague.

This development of temperature control integration is one of a larger research study under the Badan Riset dan Inovasi Nasional (BRIN) aimed at encouraging automation and digital surveillance in the fisheries sector. This report summarizes the analysis, design, implementation, and testing of the temperature control system and records the issues that have been encountered and the strategies adopted during the internship duration.

1.2 Problem Statement

- How is an automatic temperature control system designed to provide optimal water temperature between 26°C and 30°C for aquaculture?
- How is real-time data from the temperature sensor (DS18B20) processed and used to trigger water heater and cooler functions efficiently and reliably?
- How is hysteresis logic applied to prevent actuator instability due to temperature fluctuations near the threshold values?

- How can the system offer the capacity for the user to alter minimum temperature, maximum temperature, and hysteresis values dynamically through a user interface such as the serial monitor?
- How can real-time temperature data and system status be sent to other modules through ESP-NOW for real-time integration?
- How can all system operations be synchronized with real-time clock data from the RTC (DS3231) module for accurate time-based logging and control?

1.3 Objectives

The overall objective of this project is to design and implement a reliable, standalone temperature control system for inclusion in an existing Autonomous Monitoring and Control System (AMCS) for aquaculture use. The system is expected to enhance the overall effectiveness of aquaculture management by keeping the water temperature within the optimal range required for fish health and production. The following are the detailed objectives of this project:

1. To integrate a water heating and cooling system into the existing AMCS platform using the ESP32 microcontroller.
2. To utilize the DS18B20 digital temperature sensor for real-time and accurate water temperature monitoring.
3. To give conditional logic to turn on the water heater when the temperature drops below the minimum level and the water cooler and the circulation pump when the temperature increases beyond the upper limit.
4. To introduce hysteresis logic to the control system so that the actuators are not switched suddenly near threshold levels, thereby improving system stability and component lifespan.
5. In order to enable users to set the minimum and maximum temperature limits and hysteresis value dynamically through the serial monitor interface.
6. In order to use the DS3231 Real-Time Clock (RTC) module to gain accurate time-based logging as well as control synchronization.
7. In order to transmit real-time temperature readings and actuator status through ESP-NOW for seamless integration with external modules and dashboards managed by collaborators.
8. To test and verify the system through laboratory simulations and field trials to validate its functionality, accuracy, and reliability across different environmental conditions.

1.4 Scope and Limitations

1.4.1 Scope:

This project involves designing and implementing a temperature control system in an already working AMCS (Autonomous Monitoring and Control System) through IoT devices. The key features implemented in the system are:

- Real-time temperature monitoring using the DS18B20 sensor.
- Auto-control of the water heater, water cooler, and circulation pump using temperature values.

- Adjustable minimum/max temperature and hysteresis settings through the serial monitor.
- Timestamp logging using the DS3231 RTC module.
- Data transfer using ESP-NOW to transmit sensor data and actuator state to external ESP32 modules.
- System testing and validation using lab simulations and limited real-world testing.

1.4.2 Limitations:

- Temperature is the only parameter that is regulated; other parameters like pH, DO, and TDS are controlled separately.
- There is minimal integration with external modules like cloud dashboards or other ESP32 units via local ESP-NOW communication depending on the efforts of the other members of the team.
- Implementation is presently single-node focused and yet to be scaled for multi-tank or large-scale operations.
- Power supply and internet availability were taken to be stable for testing and can impact the performance of the system in field deployment.
- The configuration interface is restricted to serial input only; no web or graphical interface is implemented at this level.
- Hysteresis and threshold values are set manually and no machine learning or adaptive logic is employed for environment prediction.

1.5 Structure of the Report

This project report documents the development of an IoT-based temperature control system for aquaculture. The first chapter introduces the background of the project, identifies the core problem, states the objectives, defines the scope and limitations, and outlines the overall structure of the report. The second chapter describes the methodology used in developing the system, including the selected tools, hardware components, and the overall workflow. The third chapter presents the key results, such as the integration of temperature sensors, actuators, RTC module, and ESP32 microcontroller, along with examples of output data and relay behavior. The fourth chapter provides a discussion of the results, evaluating the system's performance, challenges encountered, and potential areas for improvement. The fifth chapter concludes the report by summarizing the outcomes and offering suggestions for future development. Finally, the references and appendices section includes all cited sources and supporting materials such as source code, configuration settings, and system diagrams.

1.6 Relevance to Real-World Problem

Temperature control via automation with the aid of real-time sensor data is extremely beneficial in aquaculture setups. It conserves labor and time by reducing the effort required for monitoring manually, allowing farm operators to focus on other critical tasks. Controlled environmental conditions courtesy of temperature control automation enhance fish welfare and boost farm productivity as a whole. The use of low-cost components, such as the ESP32, keeps this automation cost-effective for deployment by small-scale farms. The system is also flexible with threshold adjustability, and its ESP-NOW ability of communication makes it functional even in conditions where there is no stable form of internet connectivity, rendering it very versatile under a variety of farm conditions.

1.7 Schedule & Timeline

The project was conducted for six weeks, from April 15 to May 20, 2025, at Badan Riset dan Inovasi Nasional (BRIN). The duration was allocated into several phases to ensure a systematic flow of system analysis, design, development, and testing.

Table 1.0 Timeline

Week	Date	Task Description
Week 1	April 15 – 19, 2025	Analysis of the existing system, literature review on sensor parameters, preparation of proposal and block diagram.
Week 2	April 21 – 25, 2025	System design for temperature control, flowchart creation, testing ESP32, DS18B20, RTC, and relay module.
Week 3	April 28 – May 2, 2025	Real-world testing of the temperature system using water samples, connection to MQTT and database, integration of RTC and relay with DS18B20.
Week 4	May 5 – 9, 2025	Implementation of dynamic temperature setting via serial monitor, EEPROM integration, data parsing, and system logic refinement.
Week 5	May 12 – 16, 2025	System testing, integration with ESP-NOW, message formatting, hysteresis logic implementation and optimization.
Week 6	May 19 – 20, 2025	Final evaluation, reporting, and documentation of project outcomes.

Final Deliverables on May 20, 2025:

- Temperature control system fully integrated into existing AMCS.
- Efficient communication between ESP32 units using ESP-NOW.
- Full documentation in the form of flowcharts, system diagrams, test results, and source code

Literature Review and Methodology

2.1 Literature Review

2.1.1 Brief History of the Institution

Badan Riset dan Inovasi Nasional (BRIN), also known as Indonesia's National Research and Innovation Agency, was established to foster research, innovation and technology development across the country. Its main goal is to improve the quality of Indonesia's research and encourage technological innovation that contributes to national development.

BRIN was formed through the merger of various government research institutes and innovation agencies, aiming to create a unified and streamlined organization to better address Indonesia's scientific and technological challenges. The agency oversees various sectors, including agriculture, health, defense and environmental sustainability. One of its main missions is to promote the practical, real-world application of research results, thereby accelerating the development of industries that benefit society.

The institute has been actively involved in various national and international collaborations, working with universities, research centers, and private industries to advance Indonesia's innovation capabilities. LIPI plays an important role in supporting the country's transition to a knowledge-based economy by bridging the gap between academic research and industrial applications.

In the context of aquaculture and fisheries, BRIN has pioneered projects that incorporate Internet of Things (IoT) technology to improve the efficiency and sustainability of fish farming. The Autonomous Monitoring and Control System (AMCS) for aquaculture, developed as part of the institute's initiative, is one such project that aims to modernize the fisheries sector by integrating advanced technologies.

2.1.2 The Use of IoT Technology for Temperature Control in Aquaculture

The integration of Internet of Things (IoT) technology in aquaculture, especially for controlling water temperature, is crucial for fish health and growth. For species such as tilapia, maintaining temperatures between 26°C and 30°C is essential for optimal growth and immune function. Deviations from this range can stunt growth and increase susceptibility to disease (Mardiana, 2020).

Recent studies show how IoT solutions, such as temperature sensors and ESP-NOW communication, enable real-time monitoring and control. These systems improve accuracy in maintaining temperature, reduce human error, and lower labor costs. Low-cost solutions, such as those discussed by Nugroho and Setyawan (2021), make it possible for small-scale farms to adopt automated systems.

Automated systems help stabilize water temperature, increasing productivity and efficiency, as shown by Surya and Wijaya (2021). However, challenges such as unreliable internet connectivity in remote areas make ESP-NOW a very important technology, as it allows devices to communicate without the need for an internet connection, making it ideal for off-grid farming locations (Prasetya, 2021).

Overall, the use of IoT in aquaculture improves sustainability and operational efficiency by automating environmental controls, paving the way for more effective farm management.

2.2 Methodology

2.2.1 Project Approach

The development of the project follows an applied engineering and iterative development approach, incorporating elements of system integration, sensor-based automation, and embedded IoT programming. The objective is to develop a practical and low-cost solution that can be adopted in real-life aquaculture environments.

The project began with the study of the existing Autonomous Monitoring and Control System (AMCS) used in fisheries, identifying the available modules and points of integration. This was followed by a study of the optimal temperature range for tilapia and the relevant sensor technology, namely the DS18B20 temperature sensor and the DS3231 RTC module.

After the research phase, the development was carried out incrementally, starting with the testing of the individual components (ESP32, temperature sensor, RTC, and relay modules). Having confirmed that the components behaved as anticipated, the system was integrated and tested progressively in laboratory settings with rule-based control logic using real-time data.

Software code was developed in Arduino IDE, incorporating logic for activating actuators based on threshold levels and dynamic parameter setting through the serial interface. Temperature settings were persistently stored in EEPROM, and ESP-NOW was employed for wireless peer-to-peer communication with other ESP32 modules.

The approach places emphasis on reusability, flexibility, and real-world applicability, with the ultimate goal of creating a system that is not only workable, but also scalable and adaptable to diverse farm setups.

2.2 Development Tools

2.2.1 Hardware Components

Table 2.0 Hardware Components

Component	Description	Function / Use in System
ESP32	Microcontroller with Wi-Fi and Bluetooth capabilities	Acts as the brain of the system, interpreting sensor readings, executing control logic, and communication via ESP-NOW.
DS18B20	Waterproof digital temperature sensor	Tracks real-time water temperature inside the fish tank.

DS3231 RTC	Real-Time Clock module with built-in battery backup	Ensures accurate timekeeping for timestamping data and synchronization of control.
4-Channel Relay	Electromechanical relay module with 4 output channels	Controls high-voltage actuators such as the water heater and cooler from logic inputs.
Water Heater	Heating element for aquaculture tanks	Warms up the water if it goes below the required minimum threshold.
Water Cooler	Cooling device or fan system connected via relay	Ensures water temperature reduction when exceeding the maximum temperature.
Circulation Pump	Water pump to maintain water movement	Facilitates even distribution of hot or cold water throughout the tank.
Power Supply Unit	5V / 12V regulated power source	Powers the ESP32, relay module, sensors, and actuators.
Jumper Wires & Breadboard / PCB	Wiring and prototyping tools	Used when wiring components together in prototyping and connecting circuits.

2.2.1 Software Components

Table 2.1 Software Components

Software / Library	Description	Function / Use in System
Arduino IDE	Open-source development environment for microcontroller programming	Used to write, compile, and upload code to the ESP32 board.
WiFi.h	Built-in library for ESP32 Wi-Fi connectivity	Connects to the NTP server to do the first-time sync of time.
NTPClient.h	Library to get time from NTP servers	Downloads the real-time clock data from the internet to set the RTC module.
RTClib.h	Library for communicating with the DS3231 RTC	Reads and sets time on the DS3231 module.

Wire.h	I2C communication protocol library	Supports communication with the RTC module.
EEPROM.h	Library for reading/writing to EEPROM memory	Stores temperature limits and hysteresis settings permanently.
OneWire.h & DallasTemperature.h	Libraries to interface with DS18B20 sensor	Reads temperature data properly from the DS18B20.
esp_now.h	Library for ESP-NOW communication	Transfers data wirelessly between ESP32 devices without using the internet.

2.3 System Design Diagrams

2.3.1. System Architecture Diagrams

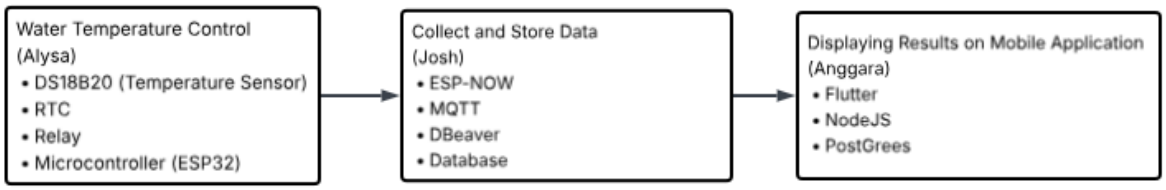


Figure 2.0 System Architecture Diagram

This is a flow diagram of the water temperature control system with three main steps. The first step is where the water temperature is read using a DS18B20 sensor, where data is processed by an ESP32 (microcontroller) which controls relays and uses an RTC to maintain time. In the second step, it receives and stores data using ESP-NOW and MQTT for communication and stored in the database by DBaver application. In the last step, received data is shown on the mobile application created using Flutter, whose backend is built using NodeJS and PostgreSQL database for data storage and data handling.

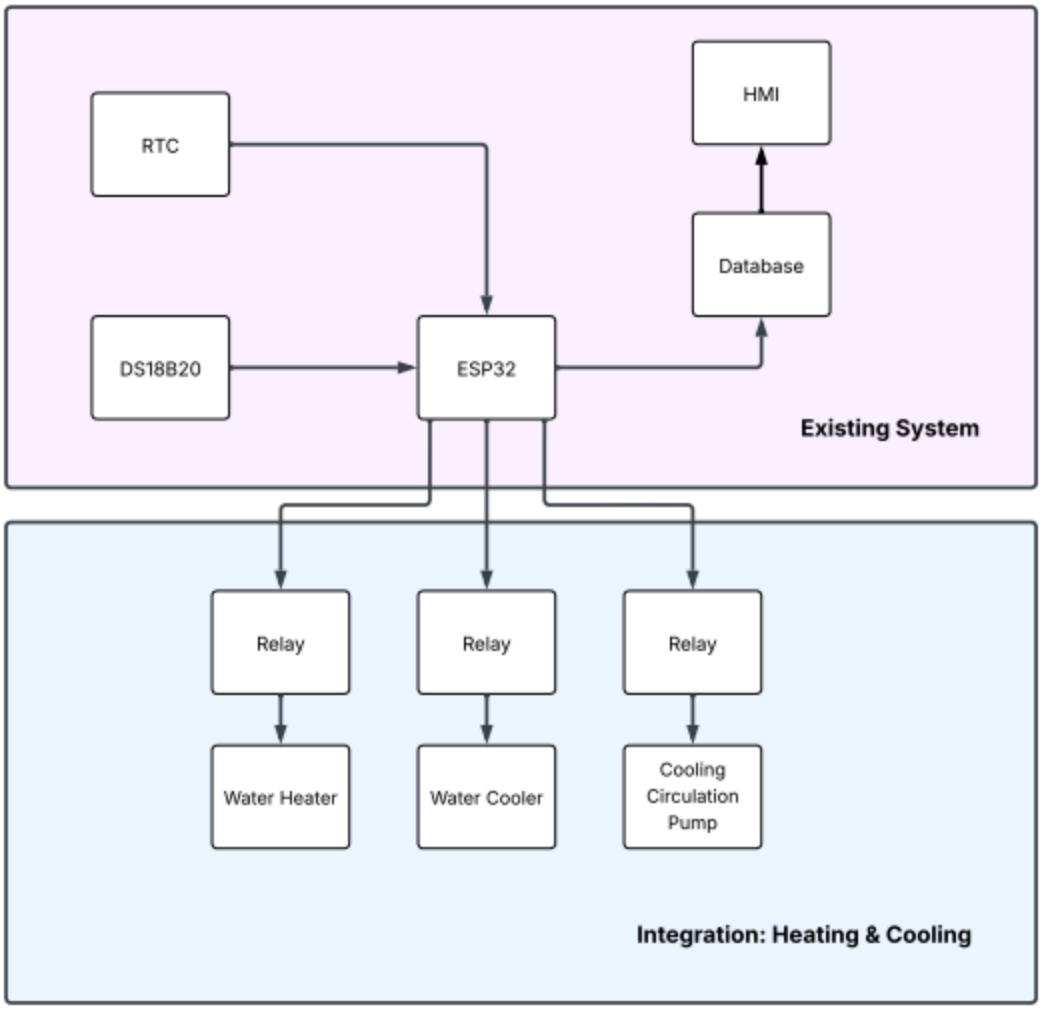


Figure 2.1 Block Diagram

This diagram illustrates a water temperature control system with heating and cooling. The ESP32 talks to an RTC for timing purposes and a DS18B20 temperature sensor for measuring water temperature. The ESP32 transmits the data, and my colleague does the HMI and database work. The ESP32 controls three relays to control the water heater, cooler, and cooling circulation pump to maintain the water temperature constant.

2.3.2 System Flowchart

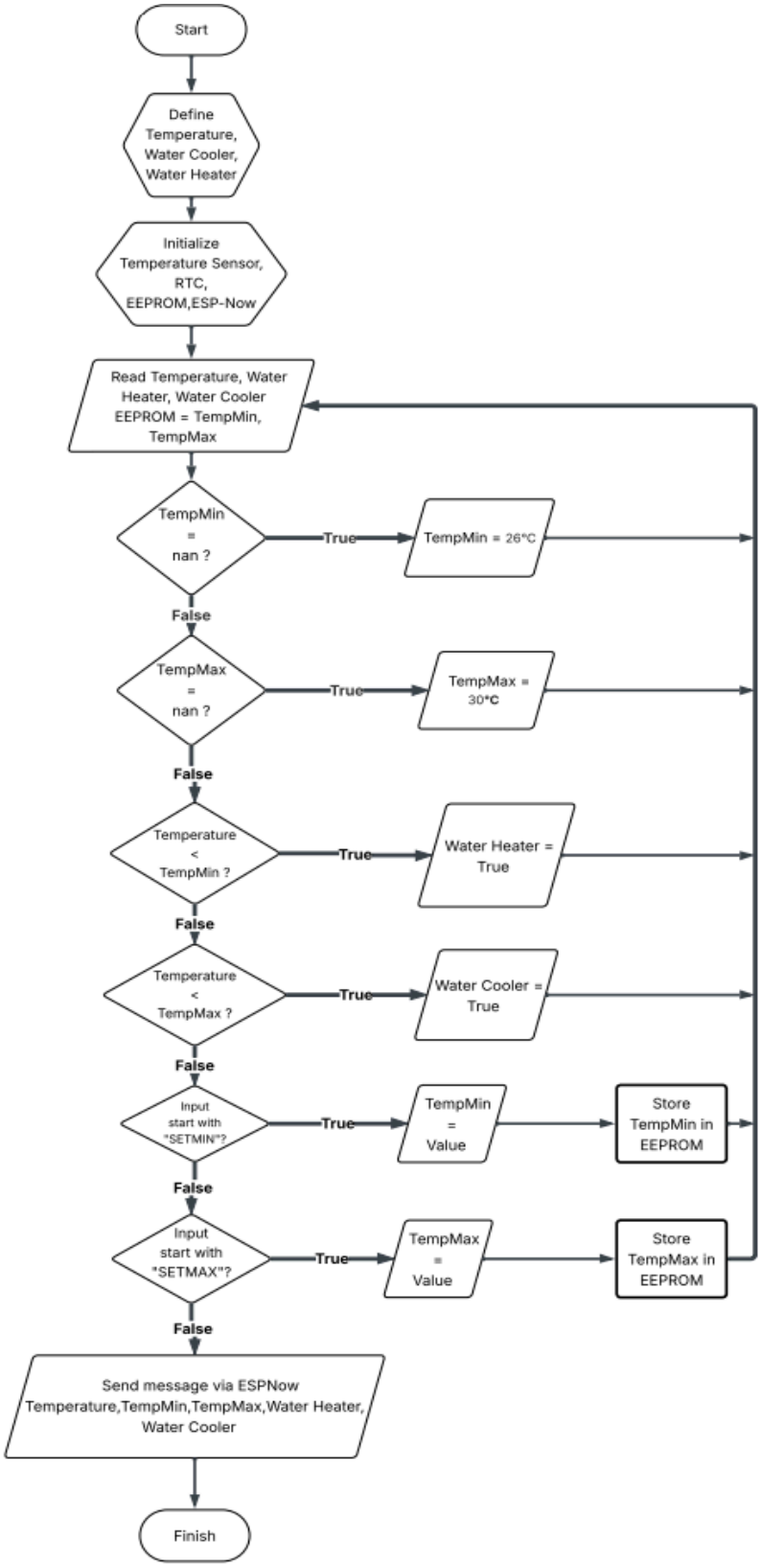


Figure 2.2 System Flowchart

The system begins by creating basic variables such as temperature, relay states (heater and cooler), and threshold values (TempMin, TempMax). It then continues by initializing basic hardware components such as the DS18B20 temperature sensor, RTC DS3231 for real-time observation, EEPROM for configuration storage, and ESP-NOW for wireless connectivity.

ESP32 reads the present temperature and loads threshold values from EEPROM. In case they are undefined or corrupted (NaN), they are reset to defaults 26°C for TempMin and 30°C for TempMax. The system then is subject to control logic: if the temperature drops below TempMin, the water heater is turned on; if rises above TempMax, the water cooler is turned on.

The system is also reactive to user inputs through the Serial Monitor. User inputs like SETMIN and SETMAX allow the user to modify limits on temperature in real-time, and these are stored back to EEPROM for preservation. Finally, the present status of the system temperature, threshold limits, and the status of the relay is transmitted through ESP-NOW to a receiver module for smooth integration with the overall monitoring system. The process is looped continuously, providing for automatic, adaptive temperature control.

2.3.3. Control Logic

Threshold Parameters (User-Adjustable):

- suhuMin: Minimum setpoint temperature (e.g., 26.0°C)
- suhuMax: Maximum setpoint temperature (e.g., 30.0°C)
- suhuMinHysteresis: Hysteresis bandwidth below suhuMin (e.g., 0.3°C)
- suhuMaxHysteresis: Hysteresis bandwidth above suhuMax (e.g., 0.3°C)

Logic Overview:

Table 2.2 Logic Overview

Condition	Action
temperature < (suhuMin - suhuMinHysteresis)	Turn Water Heater ON
temperature > (suhuMax + suhuMaxHysteresis)	Turn Water Cooler & Pump ON
temperature in between	Turn both relays OFF

```
void kontrolSuhu(float suhu) {
  bool aktifHeater = suhu < (suhuMin - suhuMinHysteresis);
  bool aktifCooler = suhu > (suhuMax + suhuMaxHysteresis);

  waterHeaterOn = aktifHeater;
  waterCoolerOn = aktifCooler;

  digitalWrite(RELAY_WATER_HEATER, waterHeaterOn ? HIGH : LOW);
  digitalWrite(RELAY_WATER_COOLER, waterCoolerOn ? HIGH : LOW);
}
```

Figure 2.3 Relay Control Logic for Heater and Cooler

Hysteresis is used to prevent "relay flickering" a condition where temperature hovers around threshold, cycling relays on and off at high frequency. By including a buffer margin (e.g., ±0.3°C), the system ensures relays switch state only when temperature significantly varies from setpoints.

These values are saved to EEPROM for persistence:

```
EEPROM.get(0, suhuMin);
EEPROM.get(10, suhuMax);
EEPROM.get(20, suhuMinHysteresis);
EEPROM.get(30, suhuMaxHysteresis);
```

Figure 2.4 EEPROM Data Persistence for System Configuration

The logic is run inside the main loop() along with sensor reads and timestamping:

```
sensorSuhu.requestTemperatures();
float suhu = sensorSuhu.getTempCByIndex(0);
```

Figure 2.5 Main Loop with Sensor Readings and Timestamps

After running the control logic, the latest data like temperature, relay status, and RTC time is wrapped and published through ESP-NOW to a receiver node for remote integration and monitoring.

2.4 Workflow and Collaboration

The project was carried out by a coordinated approach in which one member focused on the integration and design of the temperature control system while the other focused on integrating the data using ESP-NOW to the AMCS dashboard. The key components, including the DS18B20 temperature sensor, DS3231 RTC, and 4-channel relay module, were installed and tested. Control logic was coded to turn on a water heater if temperature is less than 26°C and a cooler and pump if greater than 30°C, with parameters user-adjustable and stored in EEPROM. Serial outputs were time-stamped for real-time monitoring, and laboratory tests verified system operation. Communication was sent using ESP-NOW to

an ESP32 receiver node, with good communication demonstrated over 100-200 meter ranges, and incorporated into the AMCS interface.

Workflow Summary :

Table 2.3 Workflow

Week	Dates	Tasks
Week 1	15–18 April	Literature review, block diagramming, and proposal documentation
Week 2	21–25 April	System flowcharts validated and component testing
Week 3	28 April–2 May	Temperature testing, EEPROM incorporation, and output formatting
Week 4	5–9 May	Serial command interface implementation, relay logic debugging, and hysteresis setup
Week 5	12–16 May	Last-stage integration using ESP-NOW, wireless range testing, and control parameter calibration

Supporting Tools:

Table 2.4 Supporting Tools

Tool	Function
GitHub	Hosted the complete codebase and integration logic: 🔗 https://github.com/isaajaya/FISHERY-BRIN
Arduino IDE & Serial Monitor	System programming, debugging, and real-time monitoring
ESP-NOW	Wireless transmission of sensor data without using Wi-Fi
Google Docs & Canva	Creating presentation slides, diagrams, and compiling weekly reports

This workflow proved effective for a two-person development team working within a limited timeframe. Despite not using formal project management tools, consistent documentation, testing, and coordination ensured that the project ran smoothly and all functional objectives were achieved on time.

RESULTS

3.1 Progress

3.1.1 First Week Progress

a. Tuesday, 15th April 2025

- Prepared and submitted internship proposal including project goals and portfolio.

  https://docs.google.com/document/d/11FVuGhNrCLzN_qaThY5EGVXL_eHcO8tr-JgAkfPm2W0/edit?usp=sharing
  https://www.canva.com/design/DAGkiE6egEU/bBua4Dg4Y5qBAbyKQcPspg/edit?utm_content=DAGkiE6egEU&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

b. Wednesday, 16th April 2025

- Presented self-study results and discussed activity plans with supervisor.
- Outlined tasks such as sensor research and component planning.

Figure 3.0 presented ppt

c. Thursday, 17th April 2025

- Presented PPT on tilapia and sensors.
- Revised explanation and structure of presentation based on feedback.

- Got some components for testing (Relay, Solenoid Valve, Pilot Lamp, Mosfet, Sytepper Motor

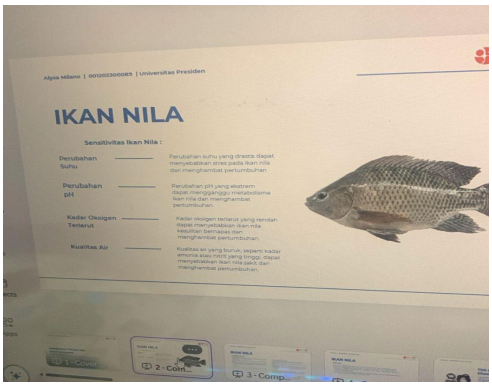


Figure 3.1 PPT tilapia and sensors

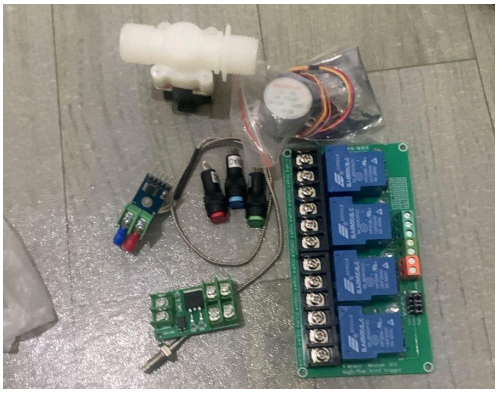


Figure 3.2 Components for testing

d. Friday, 18th April 2025

- Began compiling a PPT about system components.

  https://www.canva.com/design/DAGlK0md5i0/mhRXt5_vOugjvFMPhSwg-g/edit?utm_content=DAGlK0md5i0&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

3.1.2. Second Week Progress

a. Tuesday, 22nd April 2025

- Designed an initial block diagram for component interaction.

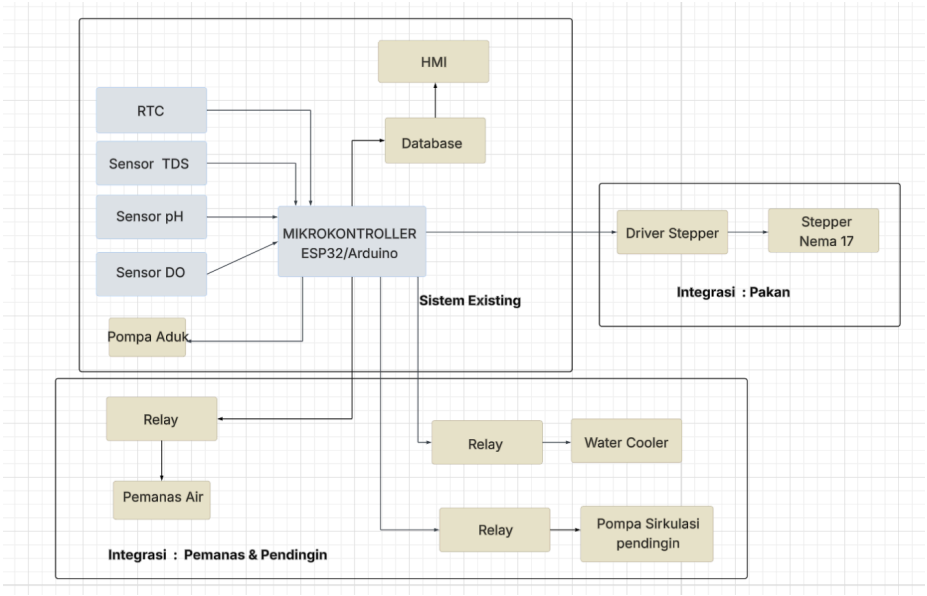


Figure 3.3 Diagram Block Components Interactions

b. Wednesday, 23rd April 2025

- Created system flowchart and finalized previous documentation.

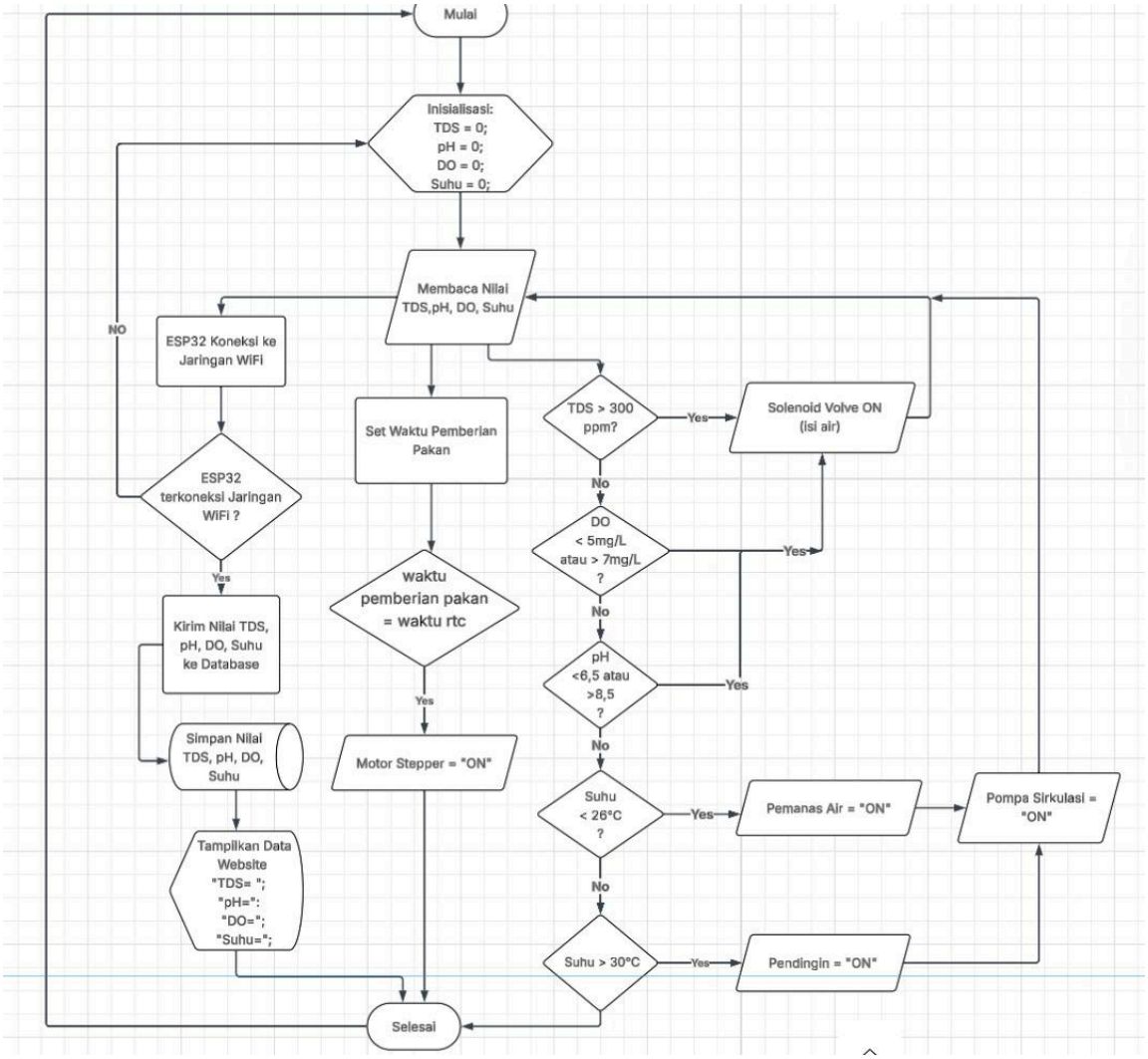


Figure 3.4 Flowchart Components Interactions

c. Thursday, 24th April 2025

- Successfully tested ESP32, DS18B20, RTC, and relay module.

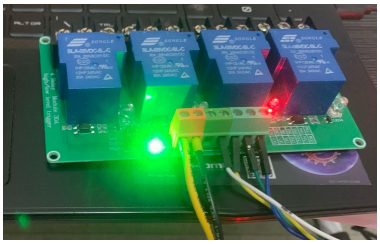


Figure 3.5 Relay Tested

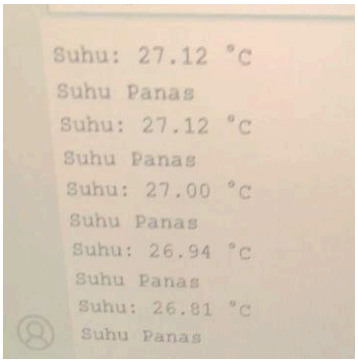


Figure 3.6 DS18B20 Tested

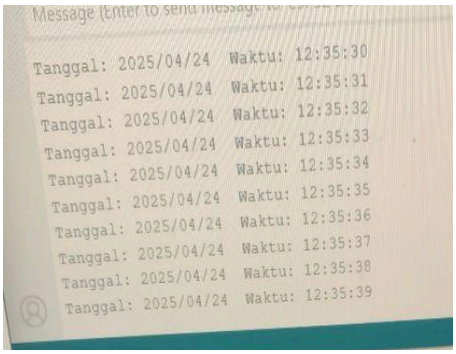


Figure 3.7 RTC Tested

e. Friday, 25th April 2025

- Finalized MQTT and database integration for sensor data.

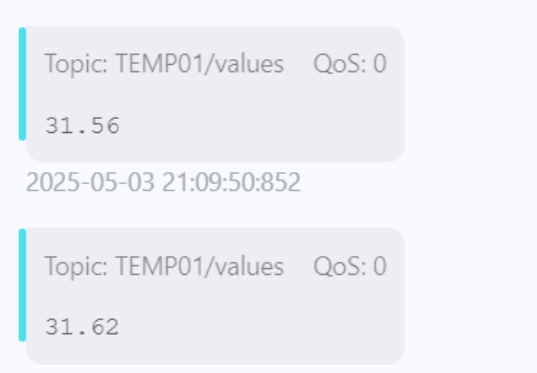


Figure 3.8 MQTT Interface

			no	timestamp	sensor_id	sensor_values	farm_id
Ubah	Salin	Hapus	1	2025-04-28 12:32:45	TEM01	29.37	61
Ubah	Salin	Hapus	2	2025-04-28 12:32:49	TEM01	29.37	61
Ubah	Salin	Hapus	3	2025-04-28 12:32:59	TEM01	29.37	61
Ubah	Salin	Hapus	4	2025-04-28 12:33:03	TEM01	29.37	61
Ubah	Salin	Hapus	5	2025-04-28 12:33:07	TEM01	29.37	61
Ubah	Salin	Hapus	6	2025-04-28 12:33:12	TEM01	29.37	61
Ubah	Salin	Hapus	7	2025-04-28 12:33:16	TEM01	29.37	61
Ubah	Salin	Hapus	8	2025-04-28 12:33:20	TEM01	29.37	61
Ubah	Salin	Hapus	9	2025-04-28 12:33:24	TEM01	29.37	61
Ubah	Salin	Hapus	10	2025-04-28 12:33:34	TEM01	29.37	61
Ubah	Salin	Hapus	11	2025-04-28 12:33:38	TEM01	29.37	61

Figure 3.9 Database Interface

3.1.3. Third Week Progress

a. Monday, 28th April 2025

- Conducted real-world test using water temperature changes.
- Exported results to DBeaver.

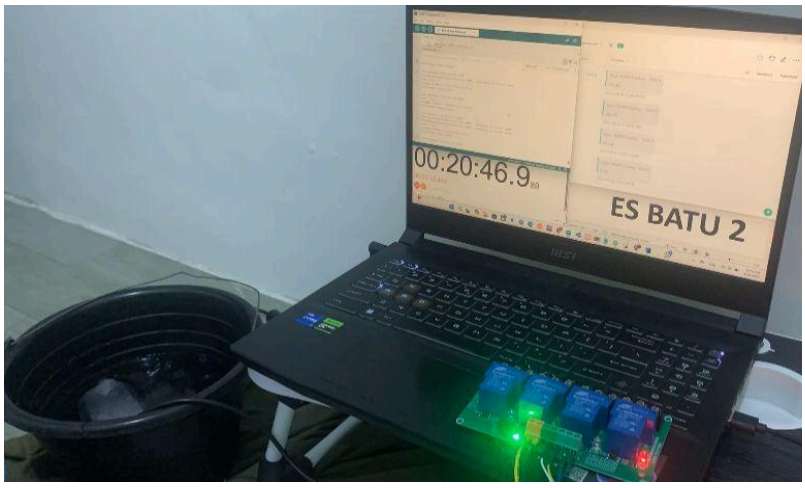


Figure 3.1.0 Tested at-Home

	no	timestamp	sensor_id	sensor_values	farm_id
1	1	2025-04-28 12:32:45	TEM01	29.37	61
2	2	2025-04-28 12:32:49	TEM01	29.37	61
3	3	2025-04-28 12:32:59	TEM01	29.37	61
4	4	2025-04-28 12:33:03	TEM01	29.37	61
5	5	2025-04-28 12:33:07	TEM01	29.37	61
6	6	2025-04-28 12:33:12	TEM01	29.37	61
7	7	2025-04-28 12:33:16	TEM01	29.37	61
8	8	2025-04-28 12:33:20	TEM01	29.37	61
9	9	2025-04-28 12:33:24	TEM01	29.37	61
10	10	2025-04-28 12:33:34	TEM01	29.37	61
11	11	2025-04-28 12:33:38	TEM01	29.37	61
12	12	2025-04-28 12:33:43	TEM01	29.37	61

Figure 3.1.1 DBeaver Result

b. Tuesday, 29th April 2025

- Successfully to integrate temp sensor + RTC + relay.

```
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-04-30 04.20.13, 32.63, 0, 1)
(2025-04-30 04.20.17, 32.69, 0, 1)
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-04-30 04.20.21, 32.75, 0, 1)
(2025-04-30 04.20.25, 32.75, 0, 1)
```

Figure 3.1.2 Successfully integrate DS18B20+RTC+Relay

c. Wednesday, 30th April 2025

- Visited aquaculture field system.

Figure 3.1.3 Agriculture Field System

Figure 3.1.4 Aquaculture Field System

3.1.4. Fourth Week Progress

a. Saturday, 3rd May 2025

- Rewrote code for field deployment and began configuring via serial monitor.

```
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:00, 30.00, 0, 0)
(2025-05-03 22:27:03, 30.06, 0, 0)
Input diterima: SETMIN 22.0
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:09, 30.06, 0, 1)
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:41, 30.19, 0, 1)
Input diterima: SETMAX 28.0
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:45, 30.19, 0, 1)
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:50, 30.19, 0, 1)
(2025-05-03 22:27:54, 30.19, 0, 1)
```

Figure 3.1.5 Set Temperature

b. Sunday, 4th May 2025

- Learned EEPROM usage and resolved issue with relay not triggering.

```
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-04 00:54:02, 34.31, 1, 0)
(2025-05-04 00:54:05, 34.25, 1, 0)
Input diterima: SETMIN 26.0
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-04 00:54:10, 34.13, 1, 0)
```

```
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-04 00:54:38, 33.69, 1, 0)
Input diterima: SETMAX 30.0
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-04 00:54:42, 33.63, 1, 0)
```

Figure 3.1.6 EEPROM Result

d. Tuesday, 6th May 2025

- Added min/max temp labels and lab-tested the system.

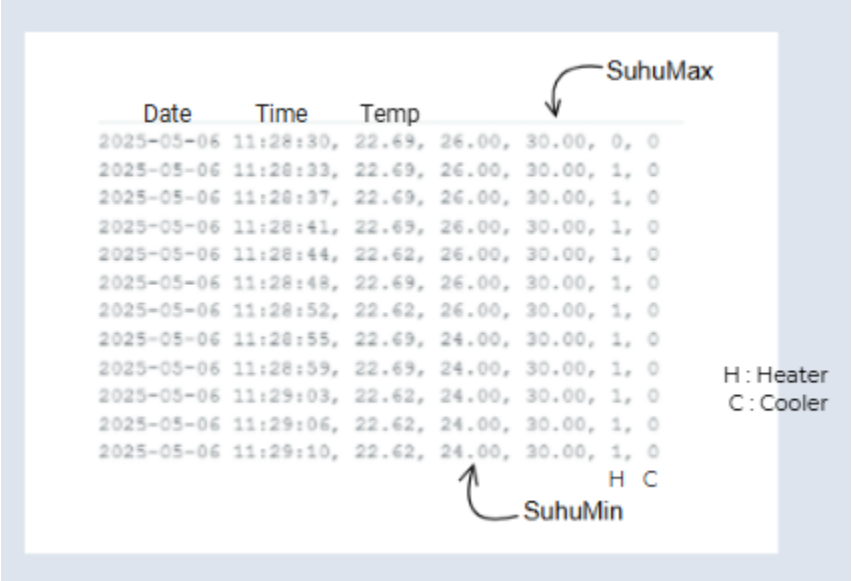


Figure 3.1.7 Temperature log with min/max labels and system control

Figure 3.1.8 Sensor in the Water System

Figure 3.1.9 Testing setup

```
2025-05-06 14:23:30, 31.00, 31.00, 30.00, 0, 1
2025-05-06 14:23:34, 30.94, 29.00, 30.00, 0, 1
```

Figure 3.2.0 Data output with temperature readings and system status.

Issue(s) and Solution(s):

- **Issue:** System instability near setpoint temperatures.

```
2025-05-06 14:21:05, 30.94, 31.00, 25.00, 1, 0
2025-05-06 14:21:08, 31.00, 31.00, 25.00, 1, 0
2025-05-06 14:21:12, 31.00, 31.00, 25.00, 0, 1
2025-05-06 14:21:16, 31.00, 31.00, 25.00, 0, 1
2025-05-06 14:21:19, 30.94, 31.00, 25.00, 0, 1
2025-05-06 14:21:23, 31.00, 31.00, 25.00, 1, 0
2025-05-06 14:21:27, 30.94, 31.00, 25.00, 0, 1
2025-05-06 14:21:30, 31.00, 31.00, 25.00, 1, 0
2025-05-06 14:21:34, 31.00, 31.00, 25.00, 0, 1
2025-05-06 14:21:37, 30.94, 31.00, 25.00, 0, 1
2025-05-06 14:21:41, 31.00, 31.00, 25.00, 1, 0
2025-05-06 14:21:45, 31.00, 31.00, 25.00, 0, 1
2025-05-06 14:21:48, 31.00, 31.00, 25.00, 0, 1
2025-05-06 14:21:52, 31.00, 31.00, 30.00, 0, 1
2025-05-06 14:21:56, 30.94, 31.00, 30.00, 0, 1
2025-05-06 14:21:59, 31.00, 31.00, 30.00, 1, 0
2025-05-06 14:22:03, 31.00, 31.00, 30.00, 0, 1
2025-05-06 14:22:07, 31.00, 31.00, 30.00, 0, 1
2025-05-06 14:22:10, 31.00, 31.00, 30.00, 0, 1
2025-05-06 14:22:14, 31.00, 31.00, 30.00, 0, 1
```

Figure 3.2.1 Data showing instability near setpoint temperatures.

- **Solution:** Identified lack of hysteresis and planned control logic revision.

3.1.5. Fifth Week Progress

a. Sunday, 11th May 2025

- Added hysteresis logic for relay control near temperature thresholds**.**

```
float suhuMin = 26.0;
float suhuMax = 30.0;
float suhuMinHysteresis = 0.5;
float suhuMaxHysteresis = 0.5;
```

Figure 3.2.2 Code for adding hysteresis logic

b. Monday, 12th May 2025

- Implemented override logic for tempMin and tempMax.

```
if (nilai > suhuMin && nilai < 100) {
  suhuMax = nilai;
  EEPROM.put(10, suhuMax);
  EEPROM.commit();
}
```

```
if (nilai > 0 && nilai < suhuMax) {
  suhuMin = nilai;
  EEPROM.put(0, suhuMin);
  EEPROM.commit();
```

Figure 3.2.3 Code for implementing override logic for tempMin and tempMax

c. Tuesday, 13th May 2025

- Labeled messages (A, B, SP) in serial output; used RTC timestamp.

```
A;12;13/5/2025 20:01:07;30.62;29.12;76.33;6.78;1010.56;7.89;1
B;12;13/5/2025 20:01:08;1;1;0;1;0
SP;12;13/5/2025 20:01:08;8;6;1200;800;30.00;29.00;85;60
```

Figure 3.2.4 Labeled messages (A, B, SP) with RTC timestamp in serial output

```
Perikanan
Pesan A
A;12;5/5/2025 20:33:45;27.45;29.12;76.33;6.78;1010.56;7.89;1
[A];[farmID];[timestamp];[waterTemp];[airTemp];[airHum];[PHValue];[tdsValue];[oxygenValue];[upLevel]

Pesan B
B;12;5/5/2025 20:33:45;1;1;0;1;0
[B];[farmID];[timestamp];[valve];[upLevel];[feederactive];[activeCooler];[activeHeater]

Pesan SP
SP;12;5/5/2025 20:33:45;8;6;1200;800;32;24;85;60
[SP];[farmID];[timestamp];[thPHUp];[thPHDown];[thTDSUp];[thTDSDown];[thWTempUp];[thWTempDown];[thHumUp];[thHumDown]
```

Figure 3.2.5Serial output showing labeled messages with sensor data and timestamps

d. Wednesday, 14th May 2025

- Integrated system with Josh via ESP-NOW.

```
A;66;14/5/2025 13:26:27;21.69;29.12;76.33;6.78;1010.56;7.89;1
B;66;14/5/2025 13:26:27;1;1;0;0;1
SP;66;14/5/2025 13:26:27;8;6;1200;800;30.00;26.00;85;60
```

Figure 3.2.6 Data sent via ESP-NOW, integrated with Josh

Figure 3.2.7 Serial monitor output showing the integration of data with Josh via ESP-NOW

e. Thursday, 15th May 2025

- Presented hysteresis improvement and added serial-based hysteresis input.

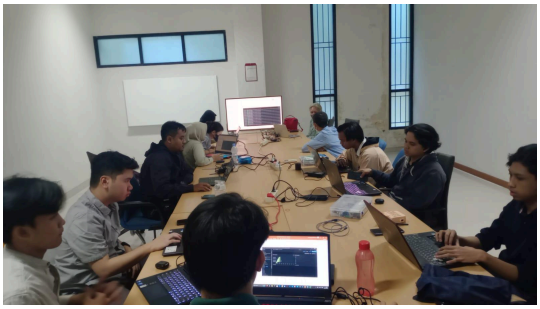


Figure 3.2.8 Team meeting where hysteresis improvements and serial-based input were presented

3.1.6. Sixth Week Progress

a. Monday, 19th May 2025

- Designed 3D casing for main components (ESP32, DS18B20, RTC, relay) to enhance durability.

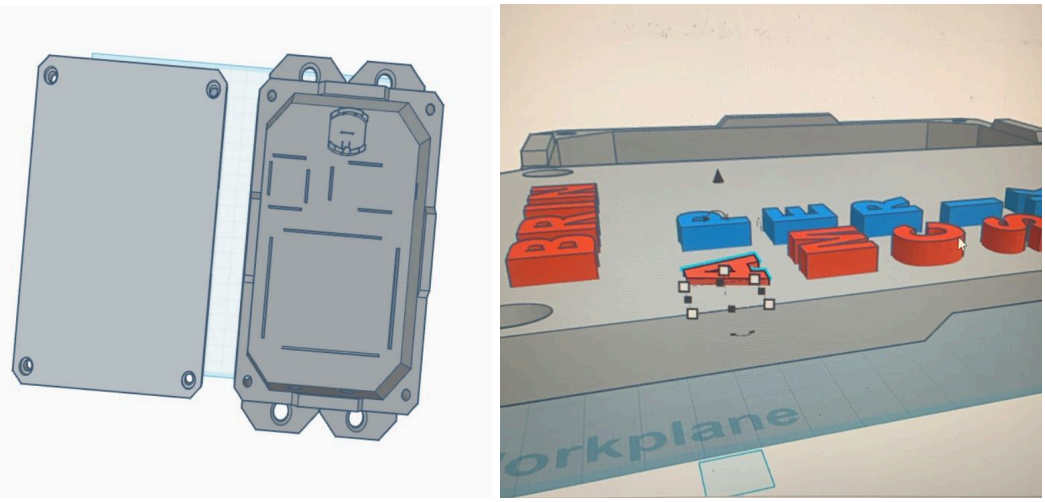


Figure 3.2.9 3D model of the printed enclosure for the system

b. Tuesday, 20th May 2025

- Printed the 3D enclosure and finalized system testing with Josh.

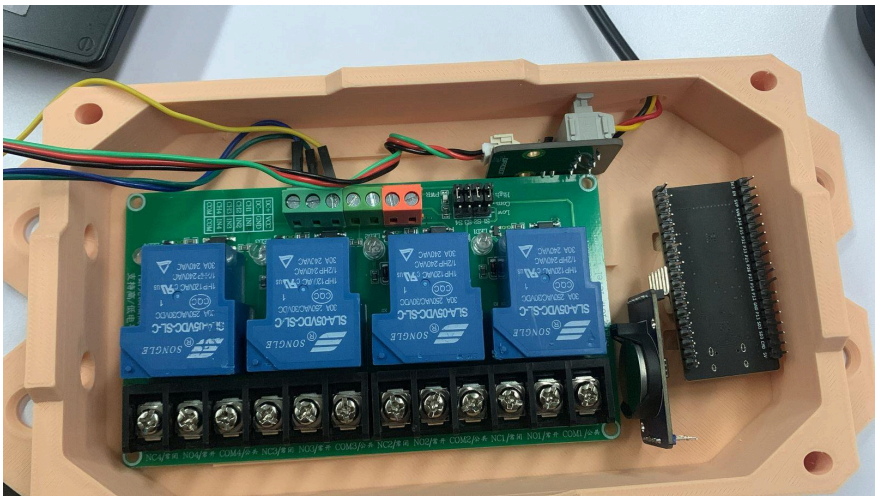


Figure 3.3.0 Finalized 3D-printed enclosure with the relay board installed inside

```
12:52:45.682 -> 📩 MQTT → [sigma/66/sensor]:
12:52:45.682 -> {"TIPEDATA":"A","FARMID":"66","timeNow":"19/5/2025 12:52:43","waterTemp":24.31,"airTemp":29.12,"airHum":76.33,"oxygen"
12:52:57.342 ->
12:52:57.342 -> 📩 ESP-NOW Received → 66;B;19/5/2025 12:52:54;1;1;0;0;1
12:52:57.386 -> 📩 MQTT → [sigma/66/actuator]:
12:52:57.386 -> {"TIPEDATA":"B","FARMID":"66","timeNow":"19/5/2025 12:52:54","valve":1,"upLevel":1,"feeder":0,"cooler":0,"heater":1}
12:52:57.912 ->
12:52:57.912 -> 📩 ESP-NOW Received → 66;SP;19/5/2025 12:52:54;8;6;1200;800;30.00;26.00;85;60
12:52:57.912 -> 📩 MQTT → [sigma/66/setpoint]:
12:52:57.912 -> {"TIPEDATA":"SP","FARMID":"66","timeNow":"19/5/2025 12:52:54","thWaterTempUp":8.00,"thWaterTempDown":6.00}
12:53:19.309 ->
12:53:19.309 -> 📩 ESP-NOW Received → 66;A;19/5/2025 12:53:17;24.12;29.12;76.33;6.78;1010.56;7.89;0
12:53:19.360 -> 📩 MQTT → [sigma/66/sensor]:
12:53:19.360 -> {"TIPEDATA":"A","FARMID":"66","timeNow":"19/5/2025 12:53:17","waterTemp":24.12,"airTemp":29.12,"airHum":76.33,"oxygen"
```

Figure 3.3.1 Serial monitor output showing system data after final testing with Josh

3.2 Result

3.2.1 Hardware Integration and System Capabilities

- Temperature Measurement

The DS18B20 digital temperature sensor was integrated successfully and provided consistent real-time water environment temperature values. The sensor values were displayed via the serial monitor and complemented by real-time timestamps from the DS3231 RTC module.

```
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 21:19:53, 31.87, 0, 1)
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 21:19:58, 31.87, 0, 1)
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 21:20:02, 31.81, 0, 1)
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 21:20:07, 31.87, 0, 1)
```

Figure 3.3.2 Serial monitor output showing temperature data with real-time timestamps

- Automated Temperature Control via Relays

A 4-channel relay module was used to control external actuators:

- **Relay 1:** Water Heater (activated when temperature < setmin)

```
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:00, 30.00, 0, 0)
(2025-05-03 22:27:03, 30.06, 0, 0)
Input diterima: SETMIN 22.0
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:09, 30.06, 0, 1)
```

Figure 3.3.3 Relay activation for water heater below minimum temperature

```
if (nilai > 0 && nilai < suhuMax) {
  suhuMin = nilai;
  EEPROM.put(0, suhuMin);
  EEPROM.commit();
```

Figure 3.3.4 Code for setting and saving minimum temperature (suhuMin)

- **Relay 2:** Water Cooler and Circulation Pump (activated when temperature > setmax)

```
if (nilai > suhuMin && nilai < 100) {
  suhuMax = nilai;
  EEPROM.put(10, suhuMax);
  EEPROM.commit();
}
```

Figure 3.3.5 Code for setting and saving maximum temperature (suhuMax)

```
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:41, 30.19, 0, 1)
Input diterima: SETMAX 28.0
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:45, 30.19, 0, 1)
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:50, 30.19, 0, 1)
(2025-05-03 22:27:54, 30.19, 0, 1)
```

Figure 3.3.6 MQTT connection and SETMAX value received

- Configurable Parameters

  Through serial input commands, users could dynamically configure:

  - SETMIN <value> — Minimum temperature threshold
  - SETMAX <value> — Maximum temperature threshold
  - SETHYMIN <value> — Minimum Hysteresis range
  - SETHYMAX <value> —- Maximum Hysteresis range

```
float suhuMin = 26.0;
float suhuMax = 30.0;
float suhuMinHysteresis = 0.3;
float suhuMaxHysteresis = 0.3;
```

Figure 3.3.7 Code for defining minimum and maximum temperature and hysteresis values

```
if (input.startsWith("SETMIN")) {
```

Figure 3.3.8 Code for setting the minimum temperature (suhuMin)

```
} else if (input.startsWith("SETMAX")) {
```

Figure 3.3.9 Code for setting the maximum temperature (suhuMax)

```
} else if (input.startsWith("SETHYMIN")) {
```

Figure 3.4.0 Code for setting the minimum hysteresis value (suhuMinHysteresis)

```
} else if (input.startsWith("SETHYMAX")) {
```

Figure 3.4.1 Serial output showing the relay activation for suhuMax settings

```
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:41, 30.19, 0, 1)
Input diterima: SETMAX 28.0
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:45, 30.19, 0, 1)
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:50, 30.19, 0, 1)
(2025-05-03 22:27:54, 30.19, 0, 1)

Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:00, 30.00, 0, 0)
(2025-05-03 22:27:03, 30.06, 0, 0)
Input diterima: SETMIN 22.0
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
Menghubungkan ke broker MQTT...Terhubung ke broker MQTT
(2025-05-03 22:27:09, 30.06, 0, 1)
```

Figure 3.4.2 Serial output showing the relay activation for suhuMin settings

These values were stored persistently in EEPROM, ensuring the settings remained valid after power loss or reset.

- Relay Stability with Hysteresis Logic

  A hysteresis margin was implemented to prevent relay chattering when the temperature fluctuated around threshold values. This addition significantly improved system stability and reduced mechanical wear on the relays.
- Real-Time Serial Output

  The system output included current water temperature, heater/cooler status, threshold values, and timestamps.

```
A;12;13/5/2025 20:01:07;30.62;29.12;76.33;6.78;1010.56;7.89;1
B;12;13/5/2025 20:01:08;1;1;0;1;0
SP;12;13/5/2025 20:01:08;8;6;1200;800;30.00;29.00;85;60
```

Figure 3.2.4 Labeled messages (A, B, SP) with RTC timestamp in serial output

```
Perikanan
Pesan A
A;12;5/5/2025 20:33:45;27.45;29.12;76.33;6.78;1010.56;7.89;1
[A];[farmID];[timestamp];[waterTemp];[airTemp];[airHum];[PHValue];[tdsValue];[oxygenValue];[upLevel]

Pesan B
B;12;5/5/2025 20:33:45;1;1;0;1;0
[B];[farmID];[timestamp];[valve];[upLevel];[feederactive];[activeCooler];[activeHeater]

Pesan SP
SP;12;5/5/2025 20:33:45;8;6;1200;800;32;24;85;60
[SP];[farmID];[timestamp];[thPHUp];[thPHDown];[thTDSUp];[thTDSDown];[thWTempUp];[thWTempDown];[thHumUp];[thHumDown]
```

Figure 3.2.5 Serial output showing labeled messages with sensor data and timestamps

- Wireless Data Transmission via ESP-NOW

Using ESP-NOW, the relay status and measured temperature were successfully sent to a paired ESP32 node that was kept up to date by a teammate. Within a 100–200 m range, data was reliably received, indicating strong peer-to-peer communication independent of Wi-Fi infrastructure.

```
A;66;14/5/2025 13:26:27;21.69;29.12;76.33;6.78;1010.56;7.89;1
B;66;14/5/2025 13:26:27;1;1;0;0;1
SP;66;14/5/2025 13:26:27;8;6;1200;800;30.00;26.00;85;60
```

Figure 3.2.6 Data sent via ESP-NOW, integrated with Josh

```
12:52:45.682 -> MQTT → [sigma/66/sensor]:
12:52:45.682 -> {"TIPEDATA":"A","FARMID":"66","timeNow":"19/5/2025 12:52:43","waterTemp":24.31,"airTemp":29.12,"airHum":76.33,"oxygen"
12:52:57.342 ->
12:52:57.342 -> ESP-NOW Received → 66;B;19/5/2025 12:52:54;1;1;0;0;1
12:52:57.386 -> MQTT → [sigma/66/actuator]:
12:52:57.386 -> {"TIPEDATA":"B","FARMID":"66","timeNow":"19/5/2025 12:52:54","valve":1,"upLevel":1,"feeder":0,"cooler":0,"heater":1}
12:52:57.912 ->
12:52:57.912 -> ESP-NOW Received → 66;SP;19/5/2025 12:52:54;8;6;1200;800;30.00;26.00;85;60
12:52:57.912 -> MQTT → [sigma/66/setpoint]:
12:52:57.912 -> {"TIPEDATA":"SP","FARMID":"66","timeNow":"19/5/2025 12:52:54","thWaterTempUp":8.00,"thWaterTempDown":6.00}
12:53:19.309 ->
12:53:19.309 -> ESP-NOW Received → 66;A;19/5/2025 12:53:17;24.12;29.12;76.33;6.78;1010.56;7.89;0
12:53:19.360 -> MQTT → [sigma/66/sensor]:
12:53:19.360 -> {"TIPEDATA":"A","FARMID":"66","timeNow":"19/5/2025 12:53:17","waterTemp":24.12,"airTemp":29.12,"airHum":76.33,"oxygen"
```

Figure 3.3.1 Serial monitor output showing system data after final testing with Josh

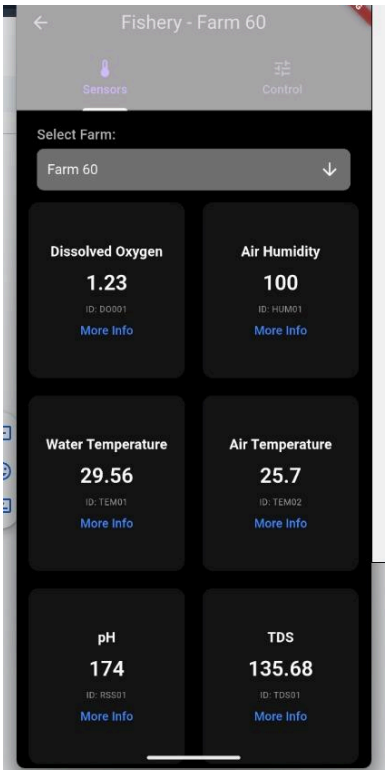


Figure 3.4.3 Mobile app interface showing sensor data

3.2.2 Laboratory and At-Home Testing

Testing was conducted in two stages: at home and in the laboratory, covering functional and integration validation.

Tabel 3.0 Components

Name of Components	Components
ESP32	 Figure 3.4.4 ESP32
RTC DS3231	 Figure 3.4.5 RTC DS3231
DS18B20	 Figure 3.4.6 DS18B20

Relay	 Figure 3.4.7 Relay 4 Channel
Water Cooler	 Figure 3.4.8 Water Cooler
Pump Circulation	 Figure 3.4.9 Pump Circulation
Water Heater	 Figure 3.5.0 Water Heater

Ice Cube	 Figure 3.5.1 Ice Cube
Water Heating Pot	 Figure 3.5.2 Water Heating Pot
Power Supply 30V 10A	 Figure 3.5.3 Power Supply 30V 10A

3.2.2.1 At-Home Testing (April 28 – May 2, 2025)

Early system tests were performed at home using simple materials such as buckets filled with hot, cold (ice), and room-temperature water. These tests verified that the DS18B20 sensor

could accurately detect changes in temperature. The system output was displayed in the Serial Monitor, and basic logic for heater and cooler activation was confirmed. During this phase, database formatting and serial output structure were also refined (e.g., datetime, temp, relay1, relay2).

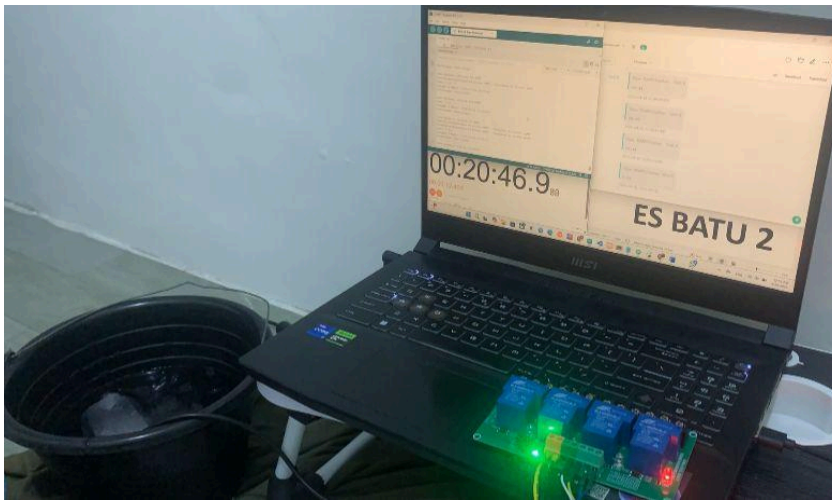


Figure 3.1.0 Tested at-Home

	no	timestamp	sensor_id	sensor_values	farm_id
1	1	2025-04-28 12:32:45	TEM01	29.37	61
2	2	2025-04-28 12:32:49	TEM01	29.37	61
3	3	2025-04-28 12:32:59	TEM01	29.37	61
4	4	2025-04-28 12:33:03	TEM01	29.37	61
5	5	2025-04-28 12:33:07	TEM01	29.37	61
6	6	2025-04-28 12:33:12	TEM01	29.37	61
7	7	2025-04-28 12:33:16	TEM01	29.37	61
8	8	2025-04-28 12:33:20	TEM01	29.37	61
9	9	2025-04-28 12:33:24	TEM01	29.37	61
10	10	2025-04-28 12:33:34	TEM01	29.37	61
11	11	2025-04-28 12:33:38	TEM01	29.37	61
12	12	2025-04-28 12:33:43	TEM01	29.37	61

Figure 3.1.1 DBeaver Result

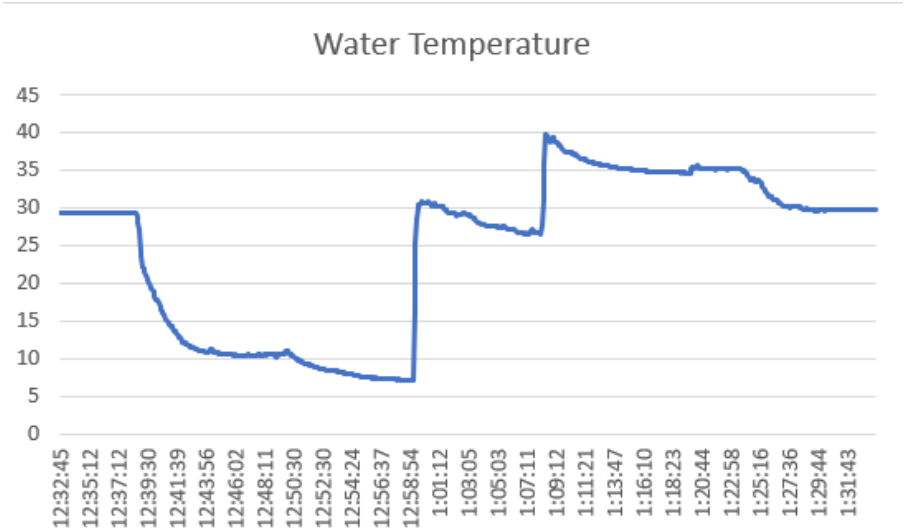


Figure 3.5.4 Graph showing water temperature over time

3.2.2.2 Laboratory Testing (May 3 – May 14, 2025)

In the lab environment, the system was fully assembled and tested with all components integrated. Serial command features (SETMIN, SETMAX, SETHYMAX, SETHYMIN) were validated, and EEPROM functionality was confirmed. On May 6, hysteresis logic was introduced and helped reduce unnecessary relay switching near threshold points. The hysteresis value was later tuned from 0.5°C to 0.3°C on May 11 for more responsive control. On May 14, ESP-NOW data transmission was successfully tested with a peer ESP32 device, demonstrating reliable communication at distances up to 200 meters.And also I design a molded case. This case enables safe outdoor use by protecting components from dust, water splashes, and accidental discharges.

Figure 3.1.8 Sensor in the Water System

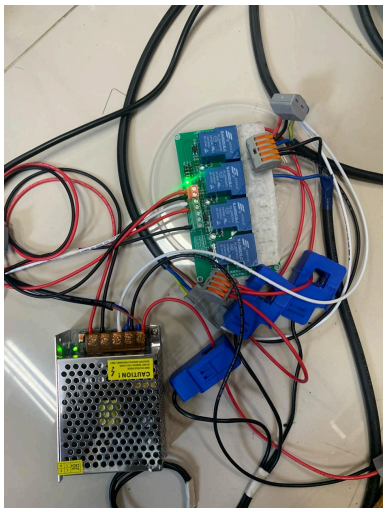


Figure 3.5.5 Sensor integrated

Figure 3.1.9 Testing setup

```
2025-05-06 14:30:02, 30.94, 29.00, 30.00, 0, 1
2025-05-06 14:30:06, 30.94, 29.00, 30.00, 0, 1
2025-05-06 14:30:09, 30.94, 29.00, 30.00, 0, 1
2025-05-06 14:30:13, 30.94, 29.00, 30.00, 0, 1
2025-05-06 14:30:17, 30.88, 29.00, 30.00, 0, 1
2025-05-06 14:30:20, 30.88, 29.00, 30.00, 0, 1
2025-05-06 14:30:24, 30.88, 29.00, 30.00, 0, 1
2025-05-06 14:30:27, 30.88, 29.00, 30.00, 0, 1
```

Figure 3.5.6 Serial output showing water temperature and system status

```
2025-05-06 14:30:02, 30.94, 29.00, 30.00, 0, 1
2025-05-06 14:30:06, 30.94, 29.00, 30.00, 0, 1
2025-05-06 14:30:09, 30.94, 29.00, 30.00, 0, 1
2025-05-06 14:30:13, 30.94, 29.00, 30.00, 0, 1
2025-05-06 14:30:17, 30.88, 29.00, 30.00, 0, 1
2025-05-06 14:30:20, 30.88, 29.00, 30.00, 0, 1
2025-05-06 14:30:24, 30.88, 29.00, 30.00, 0, 1
2025-05-06 14:30:27, 30.88, 29.00, 30.00, 0, 1
```

Figure 3.5.7 Continuation of serial output showing real-time data from the system

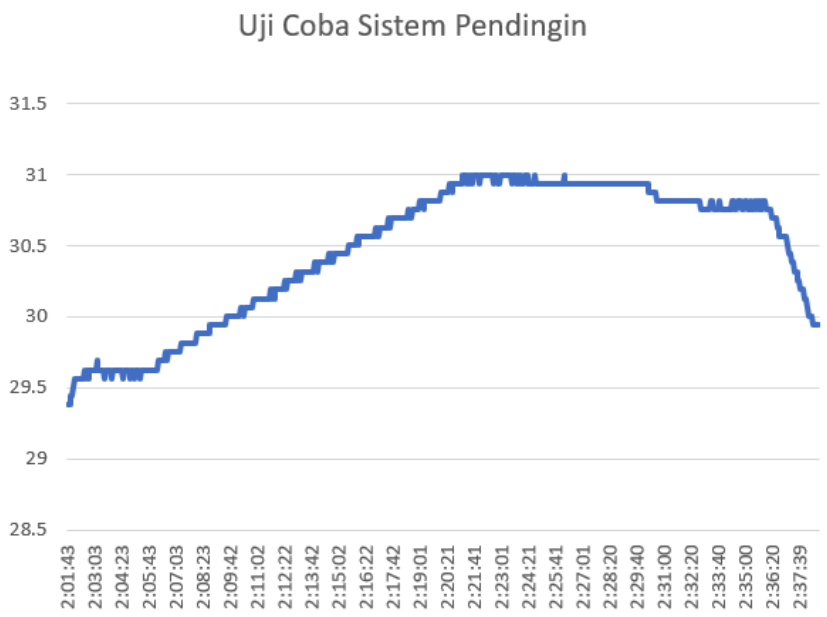


Figure 3.5.8 Graph showing water temperature over time at Lab

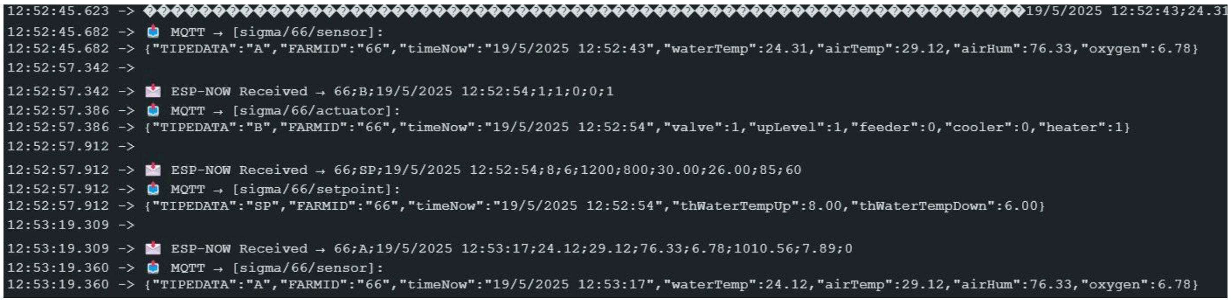
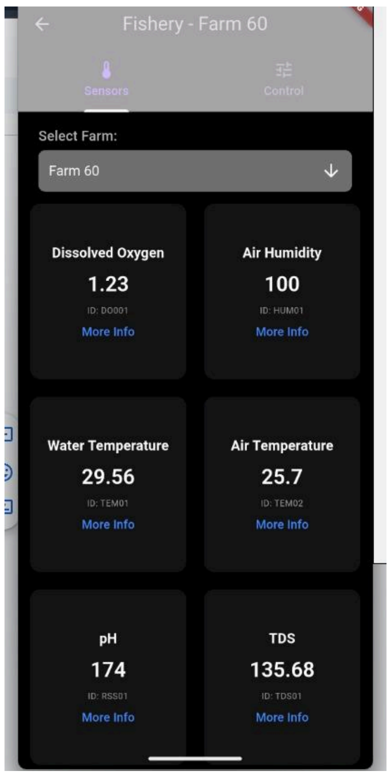


Figure 3.3.1 Serial monitor output showing system data after final testing with Josh



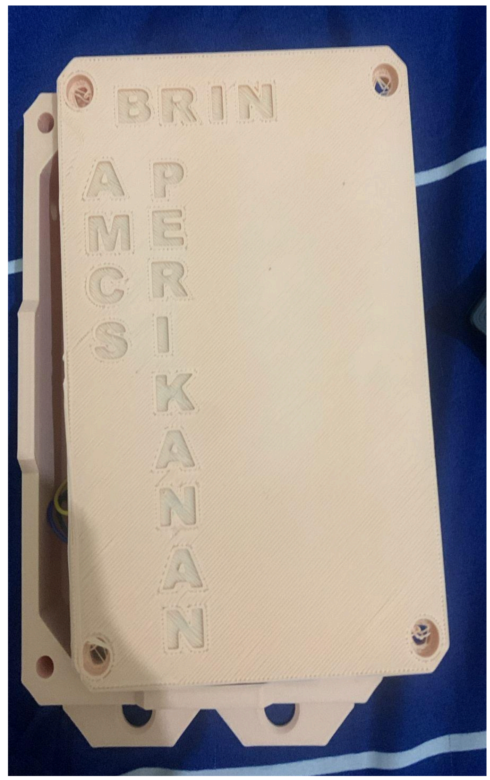


Figure 3.4.3 Mobile app interface showing sensor data Figure 3.5.9 3DPrinting Result

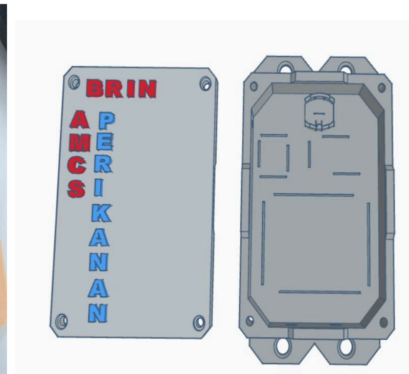


Figure 3.6.0 Finalized 3D-printed enclosure with the relay board installed inside Figure 3.6.7 3D model of the printed enclosure for the system

3.3 User Test or Feedback Summary

During the internship and bootcamp time, the system was tested through hands-on simulations with hot water, cold water, and room temperature water. Internal testing on a weekly basis with coworkers and supervisors was conducted with internal assessments. It was observed through comments that the DS18B20 temperature sensor provided stable and accurate outputs, and the automatic heater and cooler operation based on threshold settings worked as expected. The ability to define minimum and maximum temperature limits, and hysteresis values via Serial Monitor was felt necessary for calibration and adjustment. Use of hysteresis logic also successfully avoided rapid relay switching. Additionally, the use of ESP-NOW for wireless data transfer between ESP32 boards was successfully verified at a distance of 100–200 meters. Implied that in real-world usage, there ought to be some sort of display or easy-to-use interface replacing the current serial-based monitoring, reviewers praised enthusiastically the prototype's basic performance and flexibility when it comes to configuring

DISCUSSION

The automatic temperature control system successfully integrates DS18B20 sensor, DS3231 RTC module, and 4-channel relay with ESP32 microcontroller. The system can set the water temperature in real-time, store the configuration in EEPROM, and send data via ESP-NOW. Serial monitor commands (SETMIN, SETMAX, SETHYMIN, SETHYMAX) facilitated adjustments during testing.

Challenges included relay glitches due to EEPROM, unstable ESP32 connections, and initial difficulties with MQTT and ESP-NOW communication. Troubleshooting and trial-and-error were required to determine the optimal hysteresis value so that the relays would not switch frequently.

It was originally planned to use MQTT for real-time cloud dashboard integration, but due to connectivity limitations, this feature was postponed. Instead, configuration and control communication is done via a serial monitor that sends messages directly to the ESP32, then forwarded using ESP-NOW. This approach allows stable peer-to-peer data communication without relying on the internet network.

The temperature control and relay functions performed as planned. ESP-NOW proved to be stable as a short-range communication, facilitating real-time data transmission between devices within the local scope. Looking ahead, visual interfaces such as a web dashboard or LCD screen, additional sensors (pH, TDS, dissolved oxygen), as well as a waterproof casing are important to improve usability and reliability in the field. Synchronization of embedded programming, wireless networking, and hardware timing are key challenges. Safety and ethics are also crucial, as the system controls devices that affect living beings. The addition of error detection, fail-safe, and secure protocols is necessary especially for remote or internet-based control.

CONCLUSION

5.1 Conclusion

The project successfully developed an IoT-based system for monitoring and controlling water temperature in aquaculture environments. The system used an ESP32 microcontroller, a DS18B20 temperature sensor, an RTC DS3231 module, and a 4-channel relay to control heating and cooling systems. It responded automatically to temperature variations within set thresholds, configurable via a serial interface.

Testing was conducted at home and in the lab. The home tests verified core functionalities, such as sensor accuracy and relay response, while lab tests confirmed the system's stability, especially after implementing hysteresis to reduce relay switching. Configuration settings were stored in EEPROM, ensuring persistence after a restart. The system successfully used the ESP-NOW protocol for peer-to-peer wireless data transmission, with a range of up to 200 meters. A custom 3D-printed enclosure improved portability and durability.

In conclusion, the project demonstrated that a cost-effective and adaptable IoT setup can be built for real-time temperature management in aquaculture settings. It lays a strong foundation for future development and potential real-world applications.

5.2 Suggestions and improvement

To make the system more practical and scalable for real-world applications, several improvements are recommended. First, incorporating a user interface, such as an LCD screen or a web/mobile dashboard, would make it more accessible to non-technical users, allowing them to monitor and adjust settings easily. Second, integrating additional environmental sensors, such as pH, TDS, and dissolved oxygen sensors, would provide a more comprehensive understanding of water quality, crucial for fish health. Third, enhancing the network capability by using MQTT with secure communication protocols like TLS would enable remote monitoring and control, improving the system's security. Fourth, a weatherproof and waterproof housing design is essential for durability, especially when deployed outdoors. Lastly, implementing error handling and safety alerts would ensure reliability by providing automatic shutoffs and real-time alerts for sensor failures or hazardous temperature levels. These modifications would elevate the system from a prototype to a robust, user-friendly product, ready for large-scale deployment in aquaculture and integration with smart farming systems.

REFERENCES

[1] S. Mardiana, "Pengaruh suhu air terhadap pertumbuhan dan kelangsungan hidup ikan nila (*Oreochromis niloticus*)," *Jurnal Akuakultur Indonesia*, vol. 19, no. 1, pp. 45–52, 2020.

[2] A. Nugroho and D. Setyawan, "Rancang Bangun Sistem Monitoring Suhu Air Kolam Ikan Berbasis IoT Menggunakan NodeMCU ESP8266 dan Sensor DS18B20," *Jurnal Teknik Elektro dan Komputer (JTEK)*, vol. 10, no. 2, pp. 89–96, 2021.

[3] R. Surya and D. Wijaya, "Implementasi Sistem Otomatisasi Kontrol Suhu pada Budidaya Ikan Nila dengan Arduino UNO dan Sensor DS18B20," *Jurnal Teknologi dan Sistem Komputer*, vol. 9, no. 1, pp. 58–65, 2021.

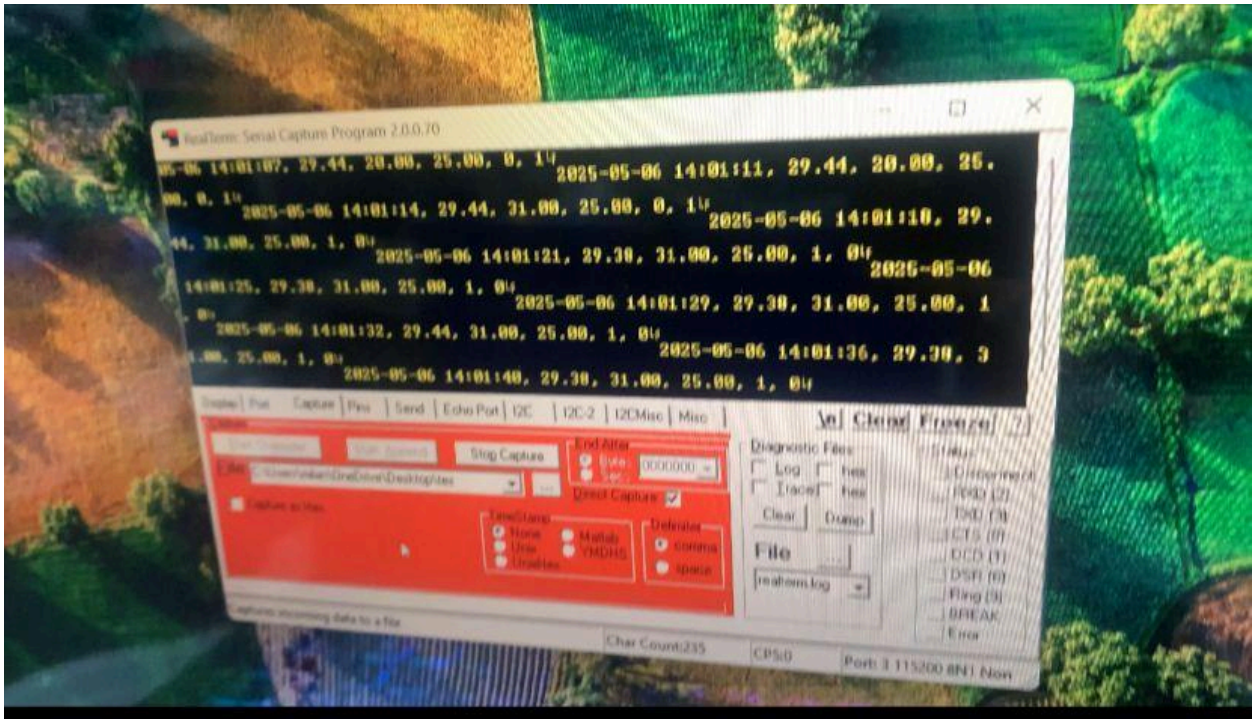
[4] H. Prasetya, "Analisis Komunikasi ESP-NOW untuk Implementasi Sistem IoT Tanpa Internet di Daerah Terpencil," *Jurnal Sistem Komputer dan Informatika (JSON)*, vol. 10, no. 1, pp. 30–35, 2021.

[5] N. P. Sari and E. Cahyono, "Pengaruh Kualitas Air terhadap Pertumbuhan dan Kelangsungan Hidup Ikan Nila (*Oreochromis niloticus*)," *Jurnal Perikanan dan Kelautan*, vol. 10, no. 2, pp. 93–99, 2019.

[6] W. Bolton, *Programmable Logic Controllers*, 6th ed. Oxford, UK: Elsevier, 2015.

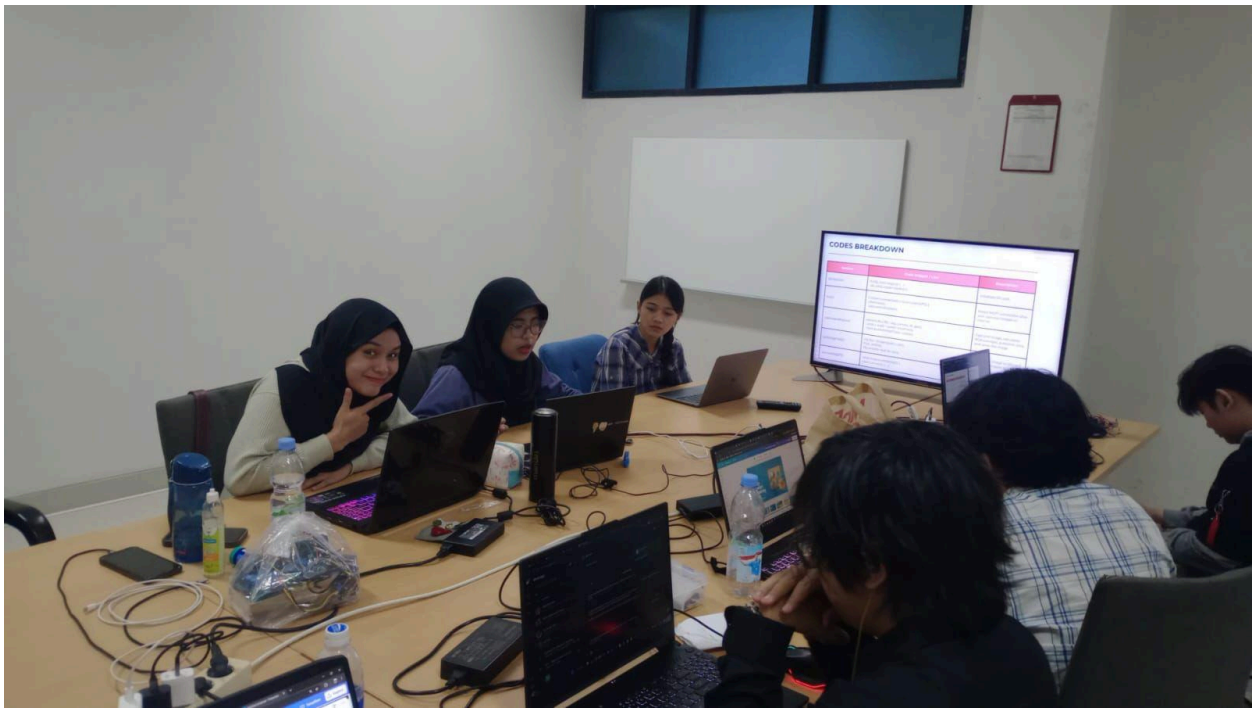
[7] S. Madakam, R. Ramaswamy, and S. Tripathi, "Internet of Things (IoT): A Literature Review," *Journal of Computer and Communications*, vol. 3, no. 5, pp. 164–173, 2015.

APPENDICES


RealTerm: Serial Capture Program 2.0.0.70
2025-05-06 14:01:11, 29.44, 28.00, 25.
2025-05-06 14:01:14, 29.44, 31.00, 25.00, 0, 1
2025-05-06 14:01:10, 29.
2025-05-06 14:01:21, 29.30, 31.00, 25.00, 1, 0
2025-05-06
2025-05-06 14:01:25, 29.30, 31.00, 25.00, 1, 0
2025-05-06 14:01:29, 29.30, 31.00, 25.00, 1
2025-05-06 14:01:32, 29.44, 31.00, 25.00, 1, 0
2025-05-06 14:01:36, 29.30, 3
2025-05-06 14:01:40, 29.30, 31.00, 25.00, 1, 0
Stop Capture
Clear Frozen
Diagnostic Files
Log
Trace
Clear
Dump
File
realterm.log
Char Count:235
CPS:0
Port: 3 115200 8N1 Non


KATANA I5 B13V


纯钛蒸发器


CODES BREAKDOWN


GOLDEN CODE
PRESIDENT UNIVERSITY X UNITAR
UNITAR
The Future We Build: Innovate Without Limits
14-16 May 2025
Fablab, President University


GOLDEN CODE
HACKATHON 2025
PRESIDENT UNIVERSITY X UNITAR