Smart pH Sensor for Real-Time Water Quality Monitoring

A Project submitted by

Hizola, Justine
Libao, David Anthony
Marasigan, Jullianne Yen
Mutuc, John Carlo
Ramos, Anton Gerard

Bachelor of Science in Computer Engineering

A Project submitted To

Dr. Rufo Marasigan Jr.

Submitted to the Computer Engineering Department
Technological Institute of the Philippines—Manila

December 13, 2024

ACKNOWLEDGMENTS

Completion of this research is made possible by the guidance, support, and encouragement of many individuals. First and foremost, we acknowledge the blessings of God, whose wisdom, strength, and perseverance have provided the foundation for this achievement. We are therefore very grateful for His divine guidance throughout this journey.

We would like to extend our deepest gratitude to our academic advisor, Dr. Rufo I. Marasigan Jr., for the constant support, expert mentoring, and insightful feedback. His guidance has played a pivotal role in setting the direction and quality of this research. We would greatly appreciate his commitment, thoughtful advice, and invaluable time he devoted to this work.

We also appreciate our families' constant support, which, above all, has inspired us. Their patience, understanding, and encouragement led us to keep going at all costs, even through the worst times.

To all our friends, colleagues, and persons who gave their advice or assistance or even just the kind word that came their way, we are in debt. Each contribution played a vital role in ensuring that this work was successful.

This is not one of our own achievements alone but with everybody who kept us going during this process. Thank you for being with us on this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 EXECUTIVE SUMMARY

1. Introduction to the Project

This is an IoT-based monitoring system intended to ensure the optimal living conditions of aquatic organisms through the maintenance of water quality in aquaculture systems. It focuses on real-time monitoring of pH levels in water, an important parameter in the solubility of nutrients, toxicity levels, and the overall water quality that are necessary for the growth and survival of aquatic species. This is based on a system using an ESP8266 microcontroller for a low-cost and scalable approach in the acquisition of real-time data for monitoring purposes. With this, control over the aquatic environment improves, bringing down mortality rates and sustainable growth of organisms in the aquaculture setups. The project hence solves the requirement for reliable and efficient methods of maintaining water quality for sustainability and health in aquatic life.

2. Core Objectives

The main goal of this project is to design an IoT-based user-friendly monitoring and maintenance system for aquaculture water quality, in particular pH levels. The system will be helpful to the users in their decisions, which would be implemented to support the growth, health, and survival of aquatic organisms. Stable conditions of water in the environment of aquaculture would ensure a reduction in the mortality rate and, in turn, increase the general efficiency of aquaculture.

3. Features and Functionality

The system features a real-time data collection function through the use of a PH4502C pH sensor and ESP8266 microcontroller. This is the primary hardware that deals with the

processing and forwarding of water quality data. A wireless data transmission eliminates connectivity issues with a cloud-based platform, thus it is easy to access collected information. A user-friendly web application visualizes both live and historical pH data, giving insights through graphical representations. It has threshold-based alerts so users can respond to critical pH level changes in real-time. The system is also designed to be scalable; it can accommodate other sensors like temperature and dissolved oxygen, which can expand the monitoring capabilities of the system.

4. Benefits and Significance

This project presents an affordable, practical aquaculture water quality management solution. It uses automation for monitoring and visualizing pH and minimizes human interference to ensure an immediate response to changes in water quality. The system is easy to carry along and portable and can therefore be used for small-scale commercial aquaculture operations towards sustainable practice. This invention supports aquatic ecosystems stability and growth as well as progress in accessibility to IoT-based environmental monitoring technology.

2 INTRODUCTION

The study will show how the researchers approach the current challenges in aquaculture. Aquaculture has been one of the most contributors in the Philippine Economy. The researchers will introduce you to what the study is all about, what are the objectives of this study, finding and determining the significance of the study, and to set scope for this study, and also the limitations.

2.1 Background of the Study

According to the Food and Agriculture Organization (2020) the Philippines ranked 11th in the world when it comes to Aquaculture production with 826.01 thousand MT that is approximately 1.01% share globally, this was mainly fish, mollusks, crustaceans.

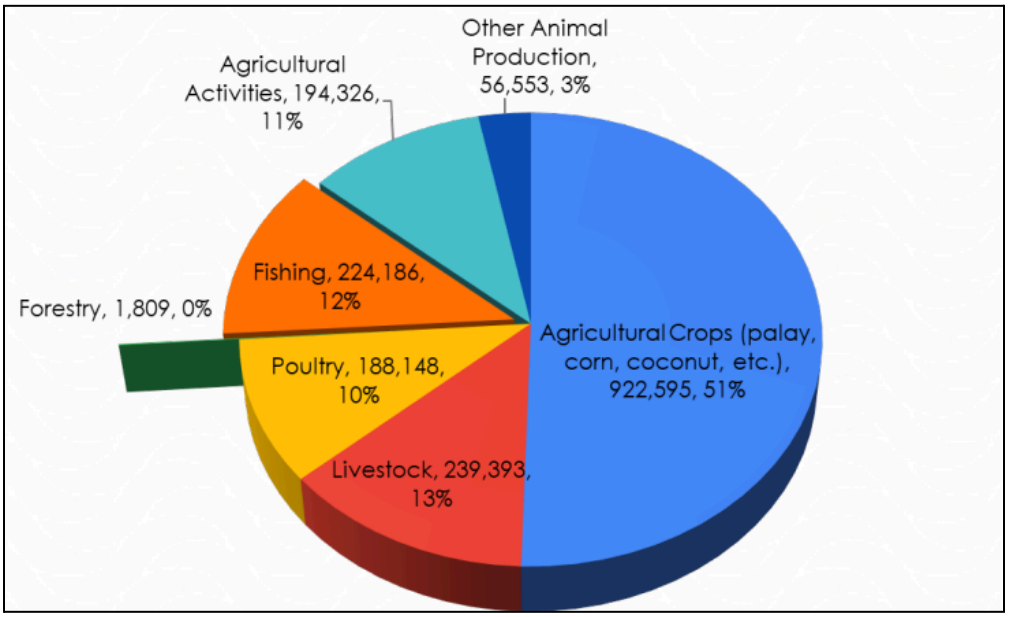


Figure 2.1 Aquaculture production

Figure 1.0: Contribution to GVA in Agriculture(this image is adopted from the Philippines Fisheries Profile 2020)

The Food And Agriculture Organization also stated that the Philippines is 4th in the world when it comes to producing aquatic plants. The Fishing Industry alone in the Philippines contributes 1.5 % share in Gross Domestic Product(GDP). Fish Farming also significantly contributes to GVA producing 224,186 which is 12% of the total GVA contribution of the Philippines.

2.2 Objectives of the Study

The objective of this project is to design an automated pH level monitoring system that can assist the fisherman. Specifically, the project aims to design a system that:

- A system that can accurately measure the pH level in fish cages.
- A system that can measure the ph Level of five different aqua organisms.
- To test the overall functionality of the system. Specifically, to test the accuracy of the pH level sensor based on the current inhabitant in the water.

2.3 Significance of the Study

The significance of this study is based on how the researchers enhance the current solution of the fish farmers when it comes to monitoring the water quality in their fish cages, and fish farms. To introduce to the Bulacan fish farmers a system that can automatically detect the

current pH level of their fish ponds or fish farms without them going near the farm. This will serve as the foundation for future research especially for integrating Internet-Of-Things based systems.

2.4 Scope and Delimitations

The Smart pH Sensor for Real-Time Water Quality Monitoring functions through the combined use of sensors and microcontroller to monitor the pH level of the water. The project cover the following:

- To identify if the water is too acidic or too alkaline.
- To send notifications about the current pH level of the water from a website.
- Used microcontroller that has WiFi as an integrated component.
- Testing will be conducted on a multiple sample of water that comes with the pH level sensor module.

Despite the aforementioned features, the project can only accomplish the following for the reason of the following constraints: Time, finances, and other factors.

- The system will only be implemented in Catanghalan, Obando Bulacan
- The system is not waterproof. Therefore, it's impossible to monitor the water pH level when it's raining but the system offers a case that will cover the important parts of the device to compromise.

- The device will only focus on the pH level monitoring, it will not have any kind of suggestion based on the current pH level of the water.
- The notification system will not be able to notify the farmer outside the web app.

3 SYSTEM ARCHITECTURE

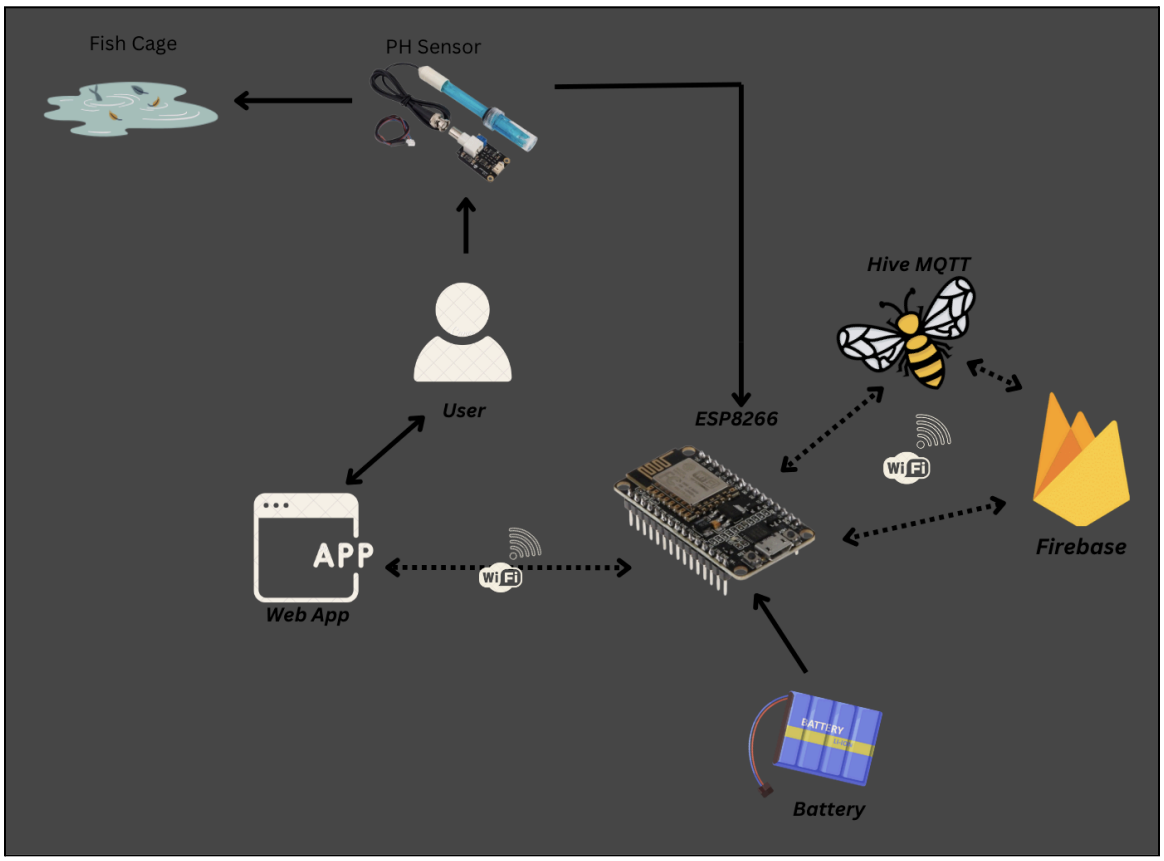


Figure 3.1 System Architecture

The system architecture defines how the different parts should be designed and should interact with each other, integrating the ESP8266 microcontroller, sensors, backend APIs, and web application to enable real-time monitoring and data visualization. It depicts how streams from sensors reach the microcontroller, through processing in the backend to a web interface through Wi-Fi and APIs. A supporting diagram visualizes these communication pathways, showing the structure and functionality of the system in achieving its objectives.

3.1 High-Level Diagram

3.2 Actual Picture of the Prototype

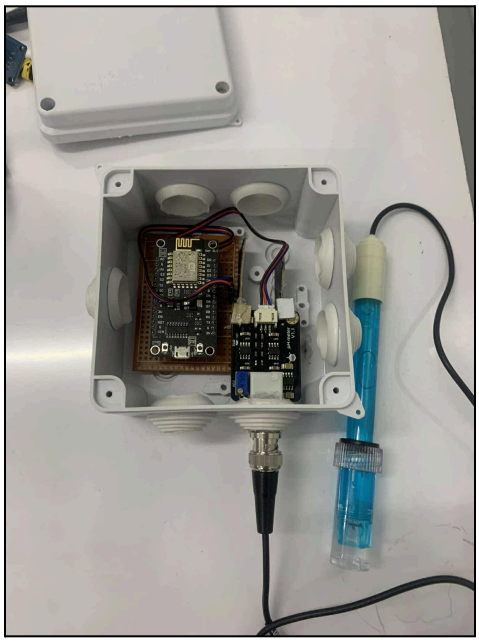


Figure 3.2 Actual Picture of the Prototype

3.3 Prototype Dimensions and Measurements

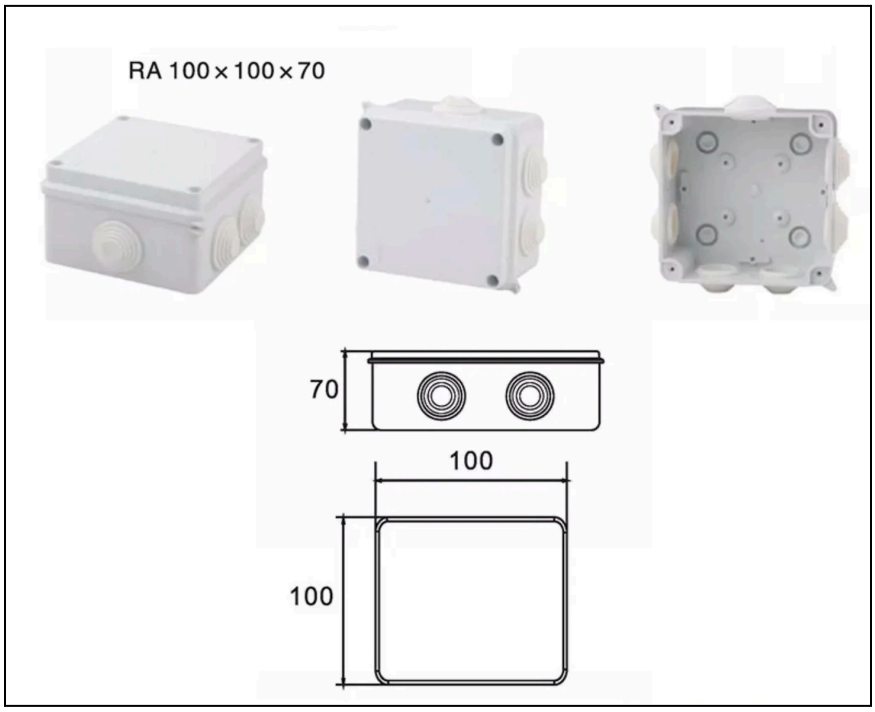


Figure 3.3 Prototype Dimensions and Measurements

3.4 System Specifications

Specification	Details
System Name	IoT-Based pH level Water Smart pH Sensor for Real-Time Water Quality Monitoring
Purpose	Real-time pH level monitoring
Data Collection Frequency	1 sample per 5 seconds
Data Transmission Protocol	MQTT
Operating Voltage	3.3V for microcontroller, 5V for peripherals
System Latency	Average response time of 500ms
Network Requirements	2.4 GHz Wi-Fi
Storage Capacity	less than 1 GB database for historical data storage

User Interface	Web application with real-time data visualization
Scalability	Supports up to 1 0 devices simultaneously

Table 3.1 Specification of the system

The system will be named IoT-Based pH level Water Smart pH sensor for Real-time Water Quality Monitoring. The system will be able to send a new reading of data every 5 seconds. This is to ensure that no cloggings will happen to the database and to ensure the stability of the readings, Since the system is IoT-Based WiFi is essential in using the system functionality.

4 HARDWARE DESIGN IMPLEMENTATIONS

This section refers to the selection, configuration, and integration of physical components like sensors, microcontrollers, and communication modules for the attainment of objectives by the system. It relates to circuit design, the generation of schematics, and dealing with real problems in the actual operation of the system, like power management and signal interference.

4.1 Hardware Components and Bill of Materials

4.1.1 Components

4.1.1.1 ESP8266 Node MCU

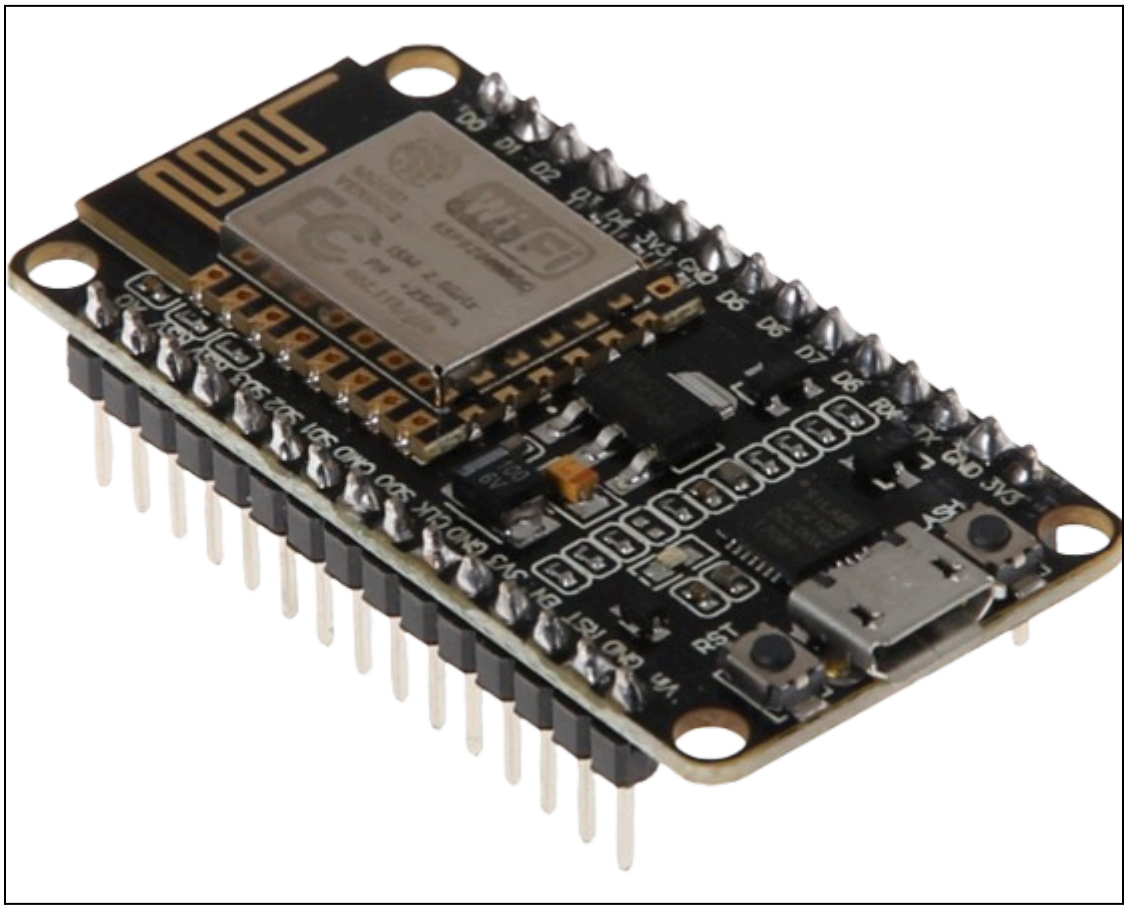


Figure 4.1.1 ESP8266 Node MCU

Santos, R. (2024) the ESP8266 NodeMCU is a low-cost development board with Wi-Fi enabled and powered by the ESP8266-12E chip, having a 32-bit L106 microprocessor running at 80–160 MHz. It supports up to 4 MB of flash memory and has 80 KB of user-data RAM. The NodeMCU also supports IEEE 802.11 b/g/n Wi-Fi with WPA/WPA2 authentication. With 17 GPIO pins, it supports UART, SPI, I²C, I²S, and a 10-bit ADC for versatile peripheral interfacing. The board can be used as a standalone microcontroller or as a UART-to-Wi-Fi adapter, which makes it suitable for IoT applications like home automation, sensor monitoring, data logging, and

real-time control via web servers or online services. Compatibility with Arduino and MicroPython programming makes it accessible to developers.

4.1.1.2 pH Sensor

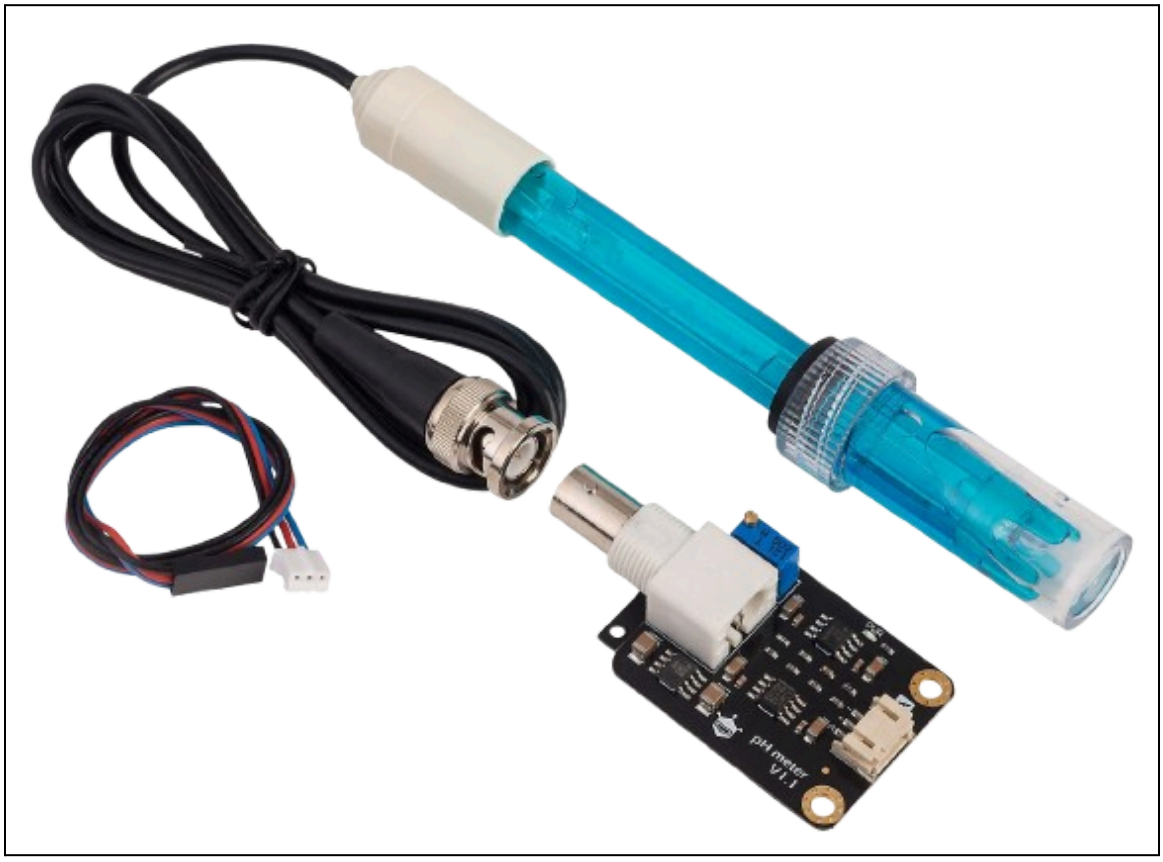


Figure 4.1.2 pH Sensor

Prateek. (2023) the pH sensor is an analog electrochemical device. It measures the acidity or alkalinity of a solution. The voltage range is from 4.2V to 5V. It works on the basis of a glass electrode reacting with hydrogen ions in the solution, producing a voltage proportional to the

concentration of hydrogen ions. This voltage is read by a reference electrode and translated into a pH value that is usually between 0 and 14. The sensor comes with an LED power indicator, a BNC connector, and a PH2.0 sensor interface to easily connect it to controllers like Arduino. The pH sensor has a wide operational range, which ranges from 0 to 50°C with an accuracy of 0.01 pH. The response time is 1 minute and is designed for real-time pH monitoring. Its technical specifications include an input supply of 5V with the measuring range from 0 to 14pH along with a cable of length 75 cm and comes with a very broad compatibility with different applications.

4.1.1.3 Junction box

Figure 4.1.3 Junction box

Sutton, B. (2024) a junction box, also referred to as an electrical box, JBox, or terminal box, is a protective enclosure that houses and organizes electrical wiring connections. It serves multiple essential purposes: it protects people from coming into contact with live wires, shields wires from dust, moisture, and damage (like chewing by rodents), and prevents the spread of fire when properly enclosed. Junction boxes are usually fabricated from materials such as metal, PVC (plastic), ABS (plastic), or fiberglass (reinforced plastic) with waterproof types for outdoor applications. The junction box keeps all electrical wiring contained safely that makes installation and maintenance simpler, hence providing organized secure wiring configurations in residential, commercial, and industrial establishments.

4.1.2 Bill of Materials

Components	Description	Quantity	Price
ESP8266 Node MCU	It is a Wi-Fi-enabled microcontroller for IoT applications	1	₱160
pH Sensor	Measures the acidity or alkalinity of a water.	1	₱1800
Junction box	It is a protective enclosure for electrical wiring connections	1	₱65
Total Cost of Components			₱2,025

Table 4.1 Computation of the Total Cost of Materials

4.2 Circuit Design

4.2.1 Block Diagram

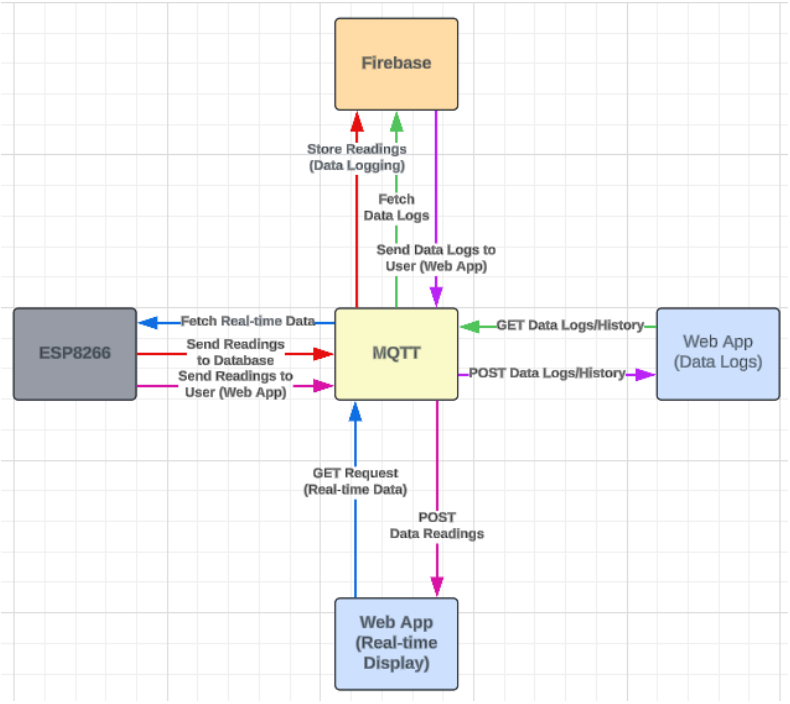


Figure 4.2.1 Block Diagram of the whole system

4.2.2 Schematic Diagram

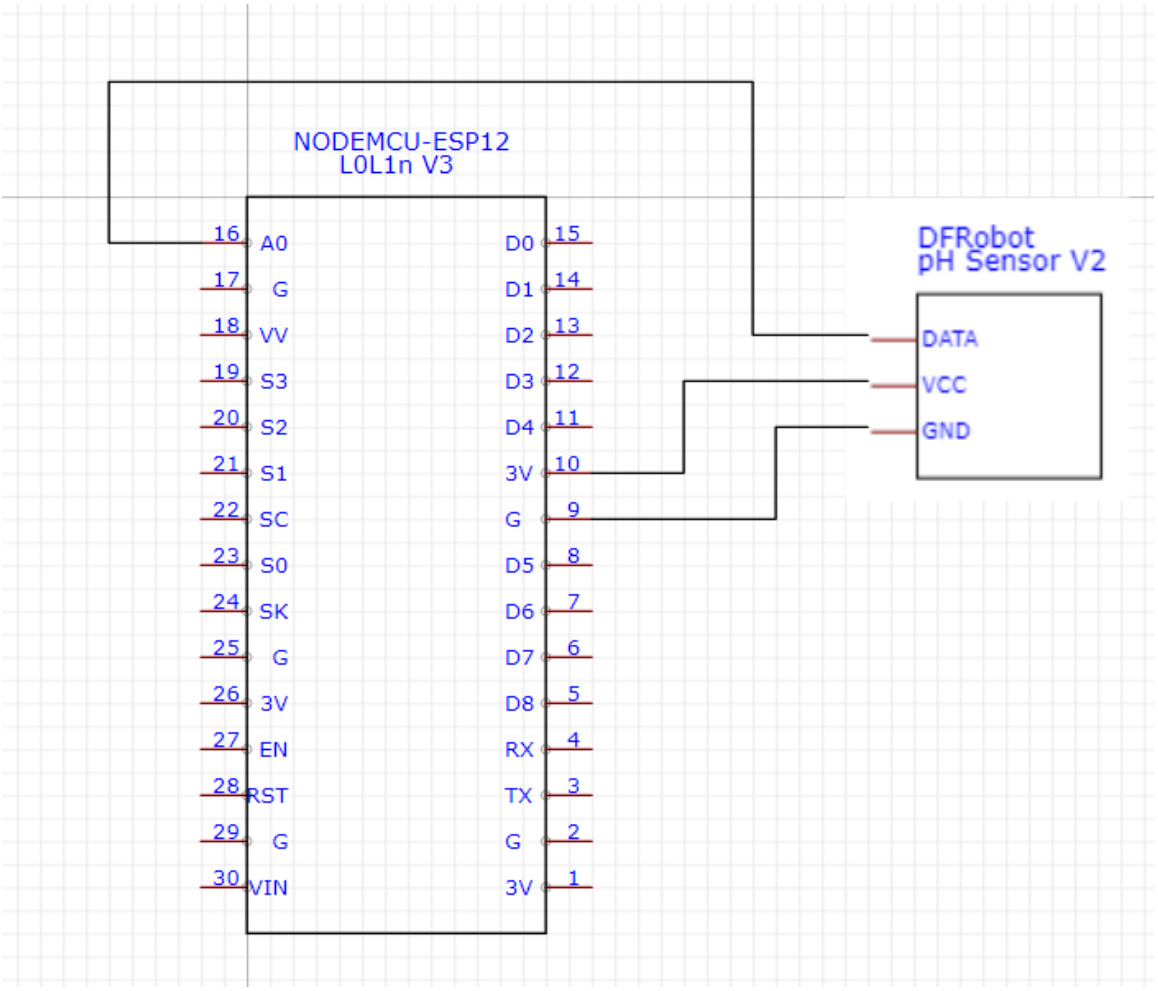


Figure 4.2.2 Block Diagram of the whole system

4.2.3 PCB Layout

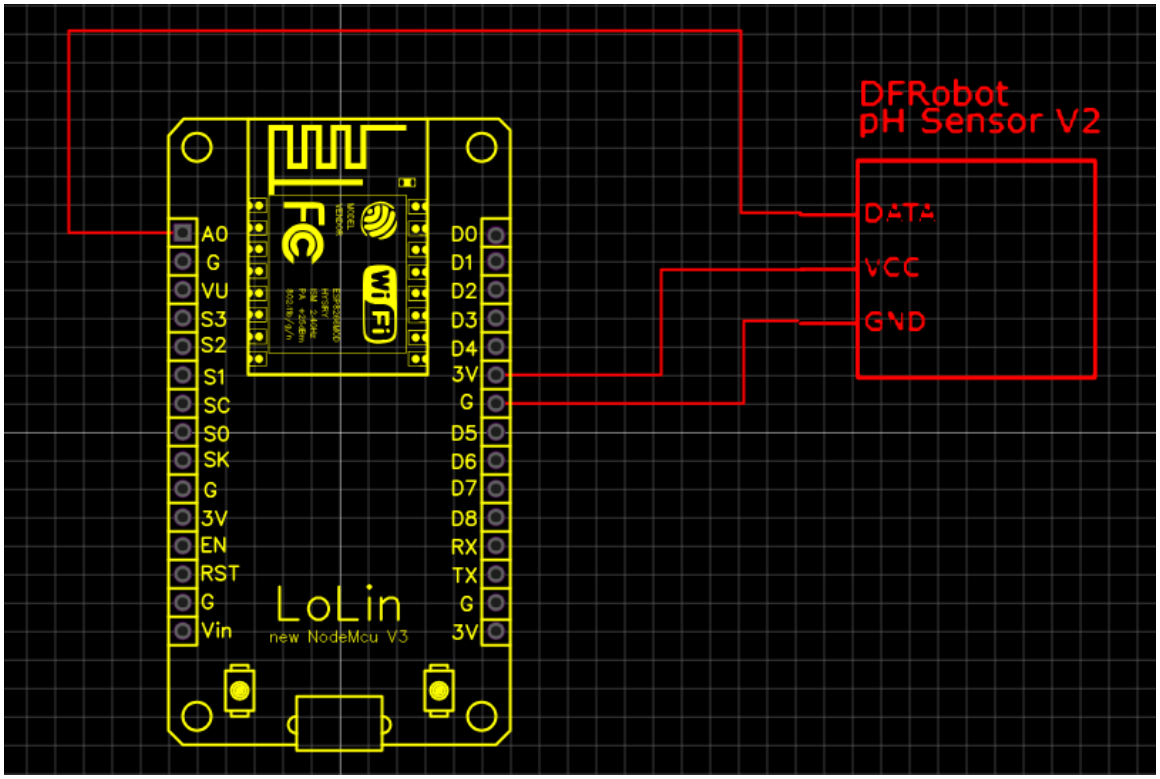


Figure 4.2.3 Block PCB of the whole system

4.3 Configuration

4.3.1 Physical Setup

All the hardware components, ESP8266 NodeMCU, and the pH sensor are placed within a junction box to ensure they are safe from the environment. The protoboard mounts the NodeMCU microcontroller and the module for the pH sensor strongly so that the connections are kept stable. The design does not take much space and yet facilitates easy accessibility for maintenance.

4.3.2 Wiring and Connections Tabulations

The Electrical connections are established as follows:

Components	Node MCU PIn	Function	Voltage/Signal
pH Sensor Module	A0	Analog GPIO	
Power	3v3	3v3 Power	3.3V
ground	GND	Ground	

Table 4.3 Wiring and Connections Tabulations

4.3.3 Software Settings

The software defines and initializes the hardware as:

- **GPIO Configuration:** Configure it as an input pin to detect high or low values from the pH sensor.
- **Wi-Fi Module Initialization:** Configured to connect to a local network to enable real-time data visualization on the web.
- **Libraries Used:** The ESP8266WiFi and ESP8266WebServer libraries handle Wi-Fi communication and host the server.

4.3.4 Calibrations

To ensure accurate readings:

- **pH Sensor Calibration**: The pH sensor is calibrated using buffer solutions (e.g., pH 4, 7, and 10) to correct any offsets.

- **Software Tuning**: Conversion logic from analog to pH values is fine-tuned by verifying against standard measurements.

5 SOFTWARE DESIGN IMPLEMENTATIONS

How the researchers design and implement software is integral for smooth and clean functional software and how the software part will be integrated into the hardware. The following is the modular structure of our software:

- **Data Acquisition**: The pH level sensor will be connected and interface with the ESP8266 to collect water pH level data.
- **Data Processing**: The data is collected and processed for clean and easy fetching of data when they are needed.
- **Communication:** MQTT is used for transmitting data from the sensor to the backend and the frontend when the user needs to see real-time data.
- **User Interface:** Vue JS is used for developing the web app and controlling the communications between API's.

Tools and Frameworks

- **Microcontroller Programming:** Arduino IDE for manipulating ESP8266.
- **Backend:** Firebase for storing data and for using API.
- **Frontend:** For creating interactable web applications, Vue.Js is used.
- **Communication Protocol:** For efficient message handling, we used MQTT

4.4 Embedded System Programming

ESP8266 is the main microcontroller used for interfacing with the pH level sensor, to collect data, and communicate with the backend. The following is the programming logic:

- **Sensor Data Acquisition:** ESP8266 gets the analog signal from the pH sensor, and converts its value into digital. The microcontroller is also responsible for validating the readings.
- **Communication:** MQTT is used for data transfer to firebase and the frontend ensures the reliability of the transferring data.

4.5 Backend API

Using Firebase for the backend API and MQTT's API, it is the middleman between the microcontroller and the web application. The key features are as follows:

- **Data Processing:** Using the two APIs, we were able to create real-time fetching of data directly from the ESP8266 to the web application while saving it to the database
- **API Endpoints:** The endpoints serve as commands to facilitate data retrieval and delivery.
- **Scalability:** Firebase is inherently scalable while ensuring the performance in different loads.

4.6 Web Application

The development of the web application is powered by Vue.JS, which allows the user to monitor the pH level of the water and manage which aquatic organisms are present in the fish cage.

4.6.1 Functional View

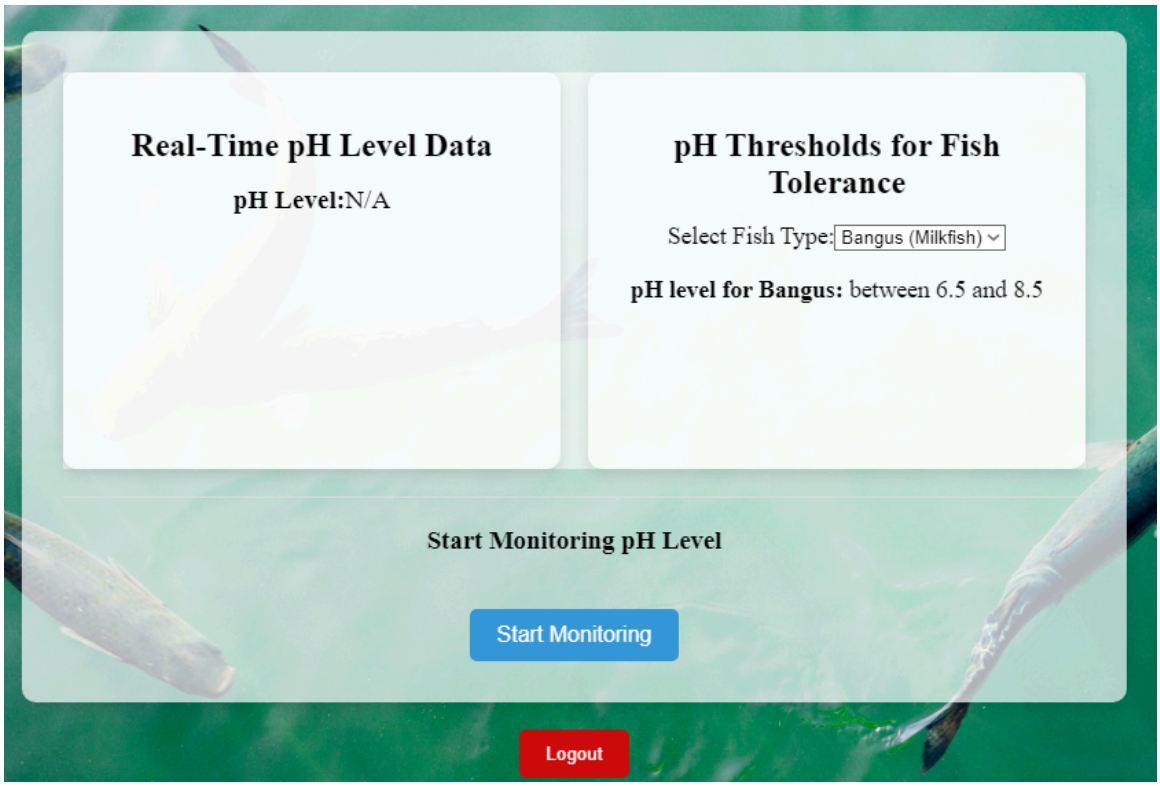


Figure 4.6.1 Web Application Interface

- **Real-Time Monitoring:** in Figure 4.4, you will be able to see the whole interface of the web application where you can see the real-time value of the pH level.

- **Data Visualization:** Displaying the real-time value of pH level with the normal display of numbers.
- **Control Panel:** Figure 4.4 shows that the user can control when he wants to start monitoring and also change the aquatic organism that will be monitored.

4.6.2 Database Design

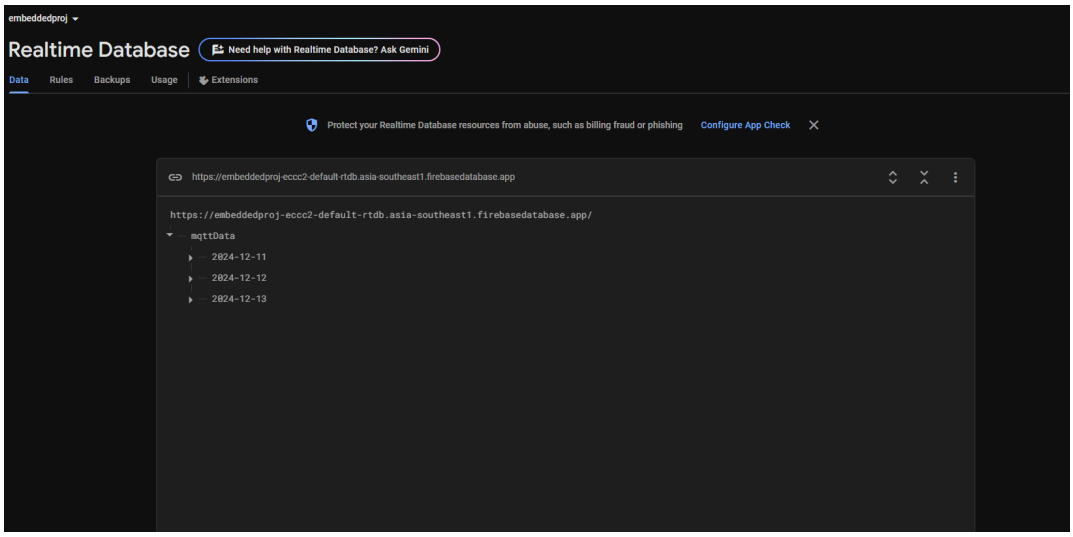


Figure 4.6.2 Firebase structure

Figure 4.5 shows the structure of the system database using Firebase, the data ved in a manner where it's easy to track when the storing of data happens because of this planned structure it is also easy to fetch the data according to the date they are stored.

4.6.3 Dataflow Diagram Level 1

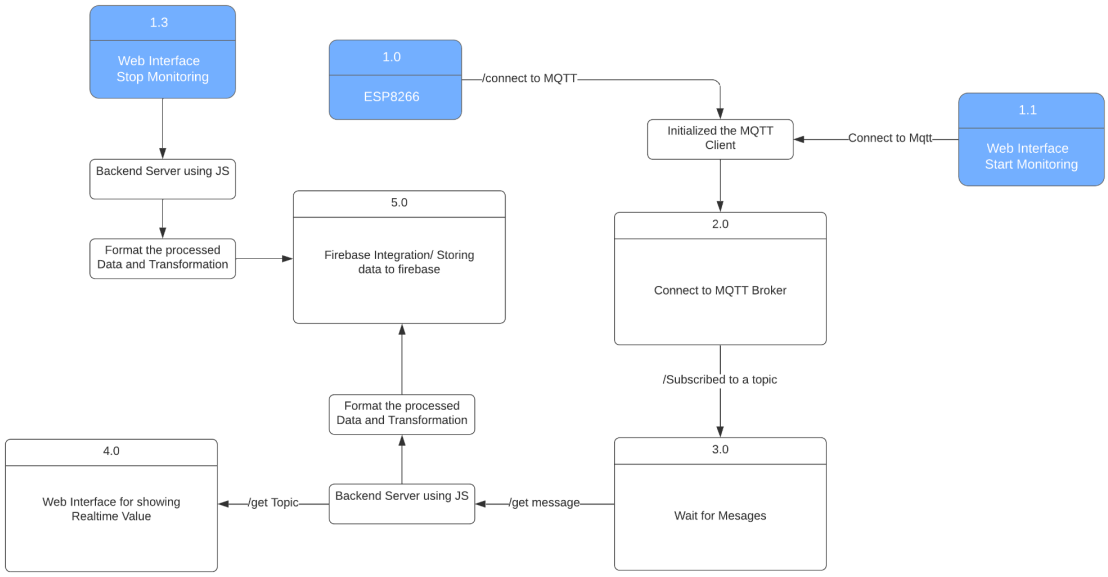


Figure 4.6.3 Dataflow Diagram Level 1

Figure 4.6 shows the DFD level 1 of how the microcontroller communicates with MQTT to web application and Microcontroller to MQTT to Firebase. With this DFD level 1, it helps the researchers to create a more elaborate design for the communication between microcontrollers to MQTT and Firebase.

4.6.4 User Interface and Features

Figure 4.4, shows the overall user interface of the web application, it is designed to be simple to make it more user-friendly a simple button with text indicating what it does helps the user to understand the web application quickly, we also put a simple dropdown component, in this component user will able to see the different aquatic organism that is available for monitoring

once the user chooses an aquatic organism the threshold then changes according to the chosen fish.

6 NETWORK AND COMMUNICATIONS

Network Protocols

- **WIFI:** it enables the microcontroller(ESP8266) to transmit data to MQTT.
- **MQTT:** it efficiently manages the exchange of messages between the microcontroller and the web application backend, and the microcontroller for storing data to Firebase.

Error Handling

- **Synchronization:** Putting timestamps and dates when storing data to ensure the accuracy of data sequencing.

Challenges

The main challenge we encountered when developing and testing the project is the instability of the internet makes the passing of data slower, due to that we've experienced slow latency when getting from the microcontroller to displaying it on the web application.

7 DEPLOYMENT

Hardware Setup

- Connecting the pH level sensor with the microcontroller(ESP8266).
- Calibration of pH level sensor to ensure accuracy and verify its functionality.
- Creation of housing to serve as a casing of the whole component.

Software Configuration

- The embedded code will be uploaded to ESP8266 using Arduino IDE.
- Setup the MQTT and Firebase endpoints.
- Deploying the web application using Vercel as a web hosting

Testing and Adjustment

- Optimized the configuration of MQTT to have a better message sending.
- Using POSTMAN to debug API integration problems.

8 TESTING AND DEBUGGING

Testing is crucial when developing an IoT-based application to ensure the system is fully functioning and has the best performance possible. The following testing was done:

- **Hardware Testing:** Each sensor was tested individually and calibrated to ensure that the data readings were accurate, we also tested the connectivity of the microcontroller which is crucial for the whole development process, using normal script to monitor the stability of the connection.
- **Software Testing:** The code used in ESP8266 was tested for any bugs that may occur and the correctness of the code's logic, it ensures that they return a correct value. API we used POSTMAN to ensure that the data or payloads are sent correctly based on their destination.

4.7 Debugging techniques and resolved issues.

- **Logging and Monitoring:** Throughout the development, the researchers used extensive logging to track what was happening during the runtime of the code; the same method was used on the hardware side to ensure the right value was passed.
- **Debugging Tools:** For embedded code, we used the serial monitor, while for the web application console it is enough to track what is happening within the code.
- **Code Optimization and Hardware Calibration**: Some performance bottlenecks were identified throughout the development process the reason was because of poor structure of the source code and multiple calls for a function that is not necessary.

9 RESULTS AND ANALYSIS

Summary of Collected Data

During the testing, data was collected to assess the overall performance of the system. The following metrics are:

- **Accuracy** - The accuracy of the pH sensor is most important and according to USDP (2024) use of fresh buffer solution is best for each point of calibration, so always keep the pH glass or bulb wet.

- **Response Time -** 200 milliseconds was the average response time of our system which is ideal considering there are multiple communication happening at once.

Data Analysis

- **Patterns** - Through the testing, the system was able to deliver consistent results and accurate results. There are some minor changes in data readings but it's because of environmental factors.

- **Anomalies -** At first pH sensor was not reading the water quality properly and it took time to read the current water quality till the researchers discovered that the pH sensor component had a cover that prevented the sensor from reading properly.

10 CONCLUSIONS, RECOMMENDATIONS AND CHALLENGES AND SOLUTIONS

10.1 Conclusions

Throughout the development of this project, we learned how to implement many things, starting from taking a strategic approach to developing an IoT-based program with multiple APIs communicating with each other. Because of this, we were able to develop a system that ensures the accuracy and readability of data and achieves the ideal response time for an automated monitoring system. Despite achieving these, the most important thing is that we researchers were able to learn how to analyze the code and how to debug when we encountered problems.

10.2 Recommendations

- Implement a calibration that can automatically recalibrate itself to address the drift that happens in the sensor because of prolonged use.
- To develop a waterproof enclosure for environmental protection.
- To add more sensors for cross-validation of water quality.
- To reduce redundancy in the program and multiple fallback protocols doing these may improve the stability of the network when using the device

10.3 Challenges and Solutions

The main challenge that the researchers encountered was implementing the MQTT to Firebase connection, initially, we thought that the use of another API was needed for the communication of MQTT and Firebase to work but later realized and remembered that there is a Firebase SDK that can be installed to our vue.js program, synchronization of the data is also proving to be a challenged at first the data that was saved in firebase was not synchronized with the current date when it was saved later realized that the wrong time zone was used. These problems were mentioned and solutions were given to help us to develop our critical thinking skills, and problem-solving that will help us in the future.

11 REFERENCES

Santos, R., & Santos, R. (2024, August 5). Getting Started with ESP8266 NodeMCU Development Board| Random Nerd Tutorials. Random Nerd Tutorials. https://randomnerdtutorials.com/getting-started-with-esp8266-wifi-transceiver-review/

Prateek, & Prateek. (2023, July 12). PH sensor. JustDoElectronics. https://justdoelectronics.com/ph-sensor/

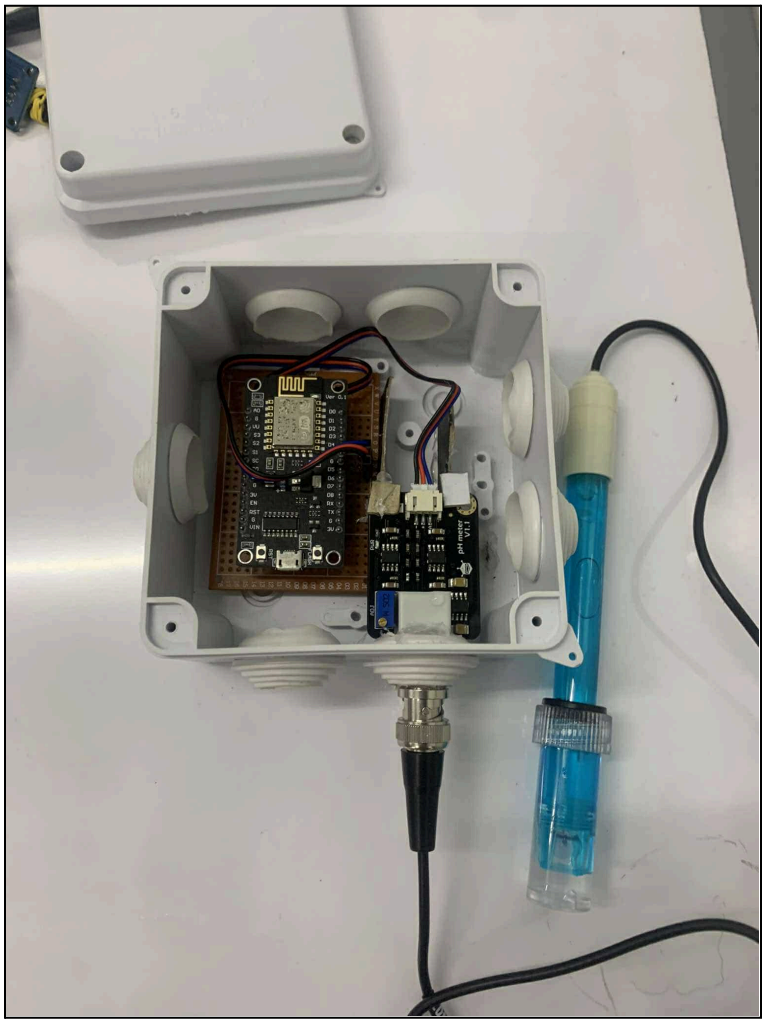
Sutton, B. (2024, February 25). What is a Junction Box (Electrical Box)? Electrical Knowledge. https://www.electricalknowledge.com/electricians-tools/junction-box/

Neufeld, J. L. (2020). The State of Food and Agriculture 2020. In *FAO eBooks*. https://doi.org/10.4060/cb1447en

Philippine Bureau of Fisheries and Aquatic Resources. (2020). *Fisheries profile: Final report*. https://www.bfar.da.gov.ph

APPENDICES

Appendix A: Picture of the Actual Testing



Appendix B: Test Results

pH_Level_Monitoring_3Days_Test_Results

Dec-11-test-results | dec-12-test-results | dec-13-test-results

|sensors__ph	sensors__ph__date	sensors__ph__value	timestamp	sensors__ph_topic
1733940215036	11/12/2024	5.63	1733940215036	sensors/ph
1733940220033	11/12/2024	5.62	1733940220033	sensors/ph
1733940225034	11/12/2024	5.62	1733940225034	sensors/ph
1733940230240	11/12/2024	5.62	1733940230240	sensors/ph
1733940235239	11/12/2024	5.62	1733940235239	sensors/ph
1733940240240	11/12/2024	5.63	1733940240240	sensors/ph
1733940245241	11/12/2024	5.62	1733940245241	sensors/ph
1733940250244	11/12/2024	5.63	1733940250244	sensors/ph
1733940255240	11/12/2024	5.63	1733940255240	sensors/ph
1733940281007	11/12/2024	5.62	1733940281007	sensors/ph
1733940286006	11/12/2024	5.63	1733940286006	sensors/ph

|	|__date	|__pH_value	|__timestamp	|__topic
1733988712435	2024-12-12	5.60	1733988712435	sensors/ph
1733988712438	2024-12-12	5.60	1733988712438	sensors/ph
1733988712439	2024-12-12	5.60	1733988712439	sensors/ph
1733988717395	2024-12-12	5.60	1733988717395	sensors/ph
1733988717396	2024-12-12	5.60	1733988717396	sensors/ph
1733988722402	2024-12-12	5.59	1733988722402	sensors/ph
1733988722403	2024-12-12	5.59	1733988722403	sensors/ph
1733988722404	2024-12-12	5.59	1733988722404	sensors/ph
1733988727398	2024-12-12	5.60	1733988727398	sensors/ph
1733988727400	2024-12-12	5.60	1733988727400	sensors/ph
1733988732410	2024-12-12	5.60	1733988732410	sensors/ph
1733988732411	2024-12-12	5.60	1733988732411	sensors/ph
1733988732412	2024-12-12	5.60	1733988732412	sensors/ph

|	|__date	|__pH_value	|__timestamp	|__topic
1734064501493	2024-12-13	5.60	1734064501493	sensors/ph
1734064506589	2024-12-13	5.60	1734064506589	sensors/ph
1734064511455	2024-12-13	5.60	1734064511455	sensors/ph
1734064511547	2024-12-13	5.60	1734064511547	sensors/ph
1734064516500	2024-12-13	5.60	1734064516500	sensors/ph
1734064516690	2024-12-13	5.60	1734064516690	sensors/ph
1734064521492	2024-12-13	5.60	1734064521492	sensors/ph
1734064521512	2024-12-13	5.60	1734064521512	sensors/ph
1734064526499	2024-12-13	5.60	1734064526499	sensors/ph
1734064526527	2024-12-13	5.60	1734064526527	sensors/ph
1734064531421	2024-12-13	5.60	1734064531421	sensors/ph
1734064531447	2024-12-13	5.60	1734064531447	sensors/ph
1734064536444	2024-12-13	5.60	1734064536444	sensors/ph
1734064546449	2024-12-13	5.60	1734064546449	sensors/ph

Appendix C: Source Code

Login Page.Vue:

```
<template>
  <div class="login-container">
    <h2 class="login-title">Aquatic pH Tracker</h2>

    <form @submit.prevent="login" class="login-form">
      <div class="form-group">
        <label for="username">Username or Email</label>
        <input
          type="text"
          id="username"
          v-model="username"
          placeholder="Enter your username or email"
          required
        />
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input
          type="password"
          id="password"
          v-model="password"
          placeholder="Enter your password"
          required
        />
      </div>
                                    <button      type="submit"
class="login-button">Login</button>
    </form>


         <p  v-if="loginError"  class="error-message">Invalid
username/email or password. Please try again.</p>
  </div>
</template>

<script>
```

```
export default {
  name: 'LoginPage',
  data() {
    return {
      username: '',
      password: '',
      loginError: false,
    };
  },
  methods: {
    login() {
       // Simulate a simple validation (you can replace this
with actual logic)
      const validUsername = 'Admin';
      const validPassword = 'password123';

      // Check if entered credentials match the valid ones
      if (this.username === validUsername && this.password ===
validPassword) {
        console.log('Login successful');
        this.loginError = false;
        this.$emit('loginSuccess');
      } else {
        console.log('Invalid credentials');
        this.loginError = true;
      }
    },
  },
};
</script>

<style scoped>

:root {
  --primary-color: #4CAF50;
  --secondary-color: #0a3d4d;
  --background-color: #5c9e94;
```

```
  --text-color: #000000;
  --button-hover-color: #388e3c;
  --button-color: #ffffff;
  --button-text-color: #000000;
  --light-blue: #a1d6e9;
  --seafoam-color: #99d1d1;
  --wave-color: rgba(255, 255, 255, 0.6);
}


html, body {
  margin: 0;
  padding: 0;
  height: 100%;
  width: 100%;
}

body {
  font-family: 'Arial', sans-serif;
  background-color: var(--background-color);
  background-image: linear-gradient(135deg, var(--light-blue),
var(--seafoam-color));
  display: flex;
  justify-content: center;
  align-items: center;
  color: black;
  background-size: 400% 400%;
  animation: gradientShift 10s ease infinite;
}



.login-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
```

```
  width: 100%;
  max-width: 400px;
  padding: 40px 20px;
  background-color: rgba(255, 255, 255, 0.9);
  border-radius: 12px;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
  border: 2px solid rgba(0, 0, 0, 0.1);
}

.login-title {
  font-size: 26px;
  font-weight: bold;
  margin-bottom: 20px;
  color: black;
     font-family:  'Segoe  UI',  Tahoma,  Geneva,  Verdana,
sans-serif;
}


.login-form {
  display: flex;
  flex-direction: column;
  width: 100%;
}

.form-group {
  margin-bottom: 16px;
}


label {
  font-size: 14px;
  font-weight: 600;
  margin-bottom: 8px;
  color: black;
  text-shadow: 1px 1px 2px rgba(255, 255, 255, 0.8);
}
```

```css
input {
  padding: 12px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 6px;
  background-color: rgba(255, 255, 255, 0.7);
  color: #333;
  width: 100%;
  box-sizing: border-box;
  transition: border-color 0.3s ease;
}

input:focus {
  border-color: var(--primary-color);
  outline: none;
  box-shadow: 0 0 5px rgba(72, 158, 136, 0.6);
}

button {
  padding: 12px;
  background-color: green;
  color: var(--button-text-color);
  font-size: 16px;
  font-weight: 600;
  border: 1px solid #098d2a;
  border-radius: 6px;
  cursor: pointer;
  transition: background-color 0.3s ease, transform 0.2s ease;
  width: 100%;
  text-align: center;
  letter-spacing: 0.5px;
}
```

```
button:hover {

  background-color: darkgreen;
  transform: translateY(-2px);
}

button:focus {
  outline: none;
  box-shadow: 0 0 0 2px rgba(72, 158, 136, 0.6);
}


.error-message {
  color: red;
  font-size: 14px;
  margin-top: 10px;
}
</style>
```

SensorStore.vue:

```
<template>
  <div class="container">

    <div class="box-container">

<section class="box realtime-box">
  <h2 class="box-title">Real-Time pH Level Data</h2>
  <div class="box-content">
    <div class="data-item">
      <strong>pH Level:</strong>
      <span v-if="isMonitoring">{{ latestpHLevel }}</span>
      <span v-else>N/A</span>
    </div>


    <canvas id="phGauge" ref="phGaugeCanvas"></canvas>
```

```
    <div class="alert-section">
      <div
        v-for="alert in alerts"
        :key="alert.message"
        :style="{ backgroundColor: alert.color, color: 'white'
}"
        class="alert"
      >
        <strong>{{ alert.message }}</strong>
      </div>
    </div>
  </div>
</section>




      <section class="box threshold-box">
               <h2  class="box-title">pH  Thresholds  for  Fish
Tolerance</h2>
        <div class="box-content">

          <div class="fish-selection">
            <label for="fishType">Select Fish Type:</label>
                                <select   v-model="selectedFish"
@change="checkPHThreshold">
                                  <option  value="bangus">Bangus
(Milkfish)</option>
              <option value="tilapia">Tilapia</option>
              <option value="mudcrab">Mudcrab</option>
              <option value="shrimp">Shrimp</option>
              <option value="galunggong">Galunggong</option>
            </select>
          </div>


          <div v-if="selectedFish === 'bangus'">
```

```
            <p><strong>pH level for Bangus:</strong> between
6.5 and 8.5</p>
          </div>
          <div v-if="selectedFish === 'tilapia'">
             <p><strong>pH level for Tilapia:</strong> between
6.0 and 8.5</p>
          </div>
          <div v-if="selectedFish === 'mudcrab'">
             <p><strong>pH level for mudcrab:</strong> between
6.5 and 8.0</p>
          </div>
          <div v-if="selectedFish === 'shrimp'">
              <p><strong>pH level for shrimp:</strong> between
7.5 and 8.5</p>
          </div>
          <div v-if="selectedFish === 'galunggong'">
                 <p><strong>pH level for galunggong:</strong>
between 6.5 and 8.5</p>
          </div>
        </div>
      </section>

      <!-- Historical Data Box -->
                    <section   class="box   historical-box"
v-if="historicalData.length > 0">
            <h2 class="box-title">Historical pH Data for {{
selectedDate }}</h2>
        <div class="box-content">
          <table class="data-table">
            <thead>
              <tr>
                <th>Time</th>
                <th>pH Level</th>
              </tr>
            </thead>
            <tbody>
```

```
                <tr v-for="(entry, index) in historicalData"
:key="index">
                <td>{{ entry.time }}</td>
                <td>{{ entry.phLevel }}</td>
              </tr>
            </tbody>
          </table>
        </div>
      </section>
    </div>


    <div class="divider"></div>


    <section class="monitoring-section">
      <h3>Start Monitoring pH Level</h3>
            <button  @click="toggleMonitoring"  class="button
primary-button">
  {{ isMonitoring ? 'Stop Monitoring' : 'Start Monitoring' }}

      </button>
    </section>
  </div>
</template>




<script>
import    {    connectMQTT,    subscribeToTopic    }    from
"@/services/mqttService";
import { useSensorStore } from "@/stores/sensorStore";
import { get, getDatabase, ref } from "firebase/database";


export default {
```

```
  data() {
    return {
      selectedDate: "",
      historicalData: [],
      alerts: [],
      selectedFish: "bangus",
      isMonitoring: false,
    };
  },
  computed: {
  latestpHLevel() {

    if (!this.isMonitoring) {
      return "N/A";
    }
    const sensorStore = useSensorStore();
    return sensorStore.getLatestpHLevel || "N/A";
  },
},

  methods: {



    toggleMonitoring() {
    this.isMonitoring = !this.isMonitoring;

    if (this.isMonitoring) {

      this.startMonitoring();
    } else {

      this.stopMonitoring();
    }
  },
```

```
    startMonitoring() {
  console.log("Monitoring started");

                        const            brokerUrl            =
"wss://ed9b94fcc7ad47119453174c68d2ae31.s1.eu.hivemq.cloud:888
4/mqtt";
  const options = {
    username: "Admin",
    password: "Admin123",
  };


  connectMQTT(brokerUrl, options);

  const sensorStore = useSensorStore();

  const topics = ["sensors/ph"];
  const callbacks = {
    "sensors/ph": (message) => {
      const pHData = parseFloat(message);
      if (!isNaN(pHData)) {
        sensorStore.handleIncomingData({
          topic: "sensors/ph",
          message: pHData,
        });
      }
    },
  };


  this.subscription = subscribeToTopic(topics, callbacks);


  this.checkPHThreshold();
},
```

```
    stopMonitoring() {
  console.log("Monitoring stopped");


            if       (this.subscription       &&       typeof
this.subscription.unsubscribe === "function") {
    this.subscription.unsubscribe();
    console.log("Unsubscribed from the topic");
  } else {
       console.warn("No  active  subscription  to  unsubscribe
from");
  }

  // Access the MQTT client from the store and unsubscribe
  const mqttClient = useSensorStore().mqttClient;
  if (mqttClient) {

    mqttClient.unsubscribe("sensors/ph", (err) => {
          if  (err)  console.error("Error  unsubscribing  from
sensors/ph:", err);
      else console.log("Unsubscribed from sensors/ph");
    });


    mqttClient.end(false, () => {
      console.log("MQTT client disconnected");
    });
  } else {
    console.warn("No MQTT client available");
  }
```

```
  this.alerts = [];
},



    // Fetch historical data for the selected date
    fetchDataByDate() {
      console.log("Selected date:", this.selectedDate);

      if (!this.selectedDate) {
        console.error("No date selected");
        return;
      }

      const db = getDatabase();
                              const     dbRef     =     ref(db,
`sensorData/${this.selectedDate}`);

      get(dbRef)
        .then((snapshot) => {
          if (snapshot.exists()) {
            const data = snapshot.val();
            console.log("Fetched data:", data);


            this.historicalData = Object.entries(data).map(([,
entry]) => ({
              time: entry.time,
              phLevel: entry.phLevel,
            }));
                                  console.log("Formatted   data:",
this.historicalData);
          } else {
             console.warn("No data available for the selected
date");
            this.historicalData = [];
          }
```

```
        })
        .catch((error) => {
          console.error("Error fetching data:", error);
          this.historicalData = [];
        });
    },



    // Existing method to check pH thresholds
    checkPHThreshold() {
    const bangusMin = 6.5;
    const bangusMax = 8.5;
    const tilapiaMin = 6.0;
    const tilapiaMax = 8.5;
    const mudcrabMin = 6.5;
    const mudcrabMax = 8.0;
    const shrimpMin = 7.5;
    const shrimpMax = 8.5;
    const galunggongMin = 6.5;
    const galunggongMax = 8.5;

    let newAlerts = [];

    // Check pH thresholds based on the selected fish
    if (this.selectedFish === "bangus") {
      if (this.latestpHLevel < bangusMin) {
        newAlerts.push({
             message: "pH level is too low for Bangus (below
6.5).",
          color: "red",
        });
      } else if (this.latestpHLevel > bangusMax) {
        newAlerts.push({
            message: "pH level is too high for Bangus (above
8.5).",
          color: "red",
```

```
        });
      }
    } else if (this.selectedFish === "tilapia") {
      if (this.latestpHLevel < tilapiaMin) {
        newAlerts.push({
            message: "pH level is too low for Tilapia (below
6.0).",
          color: "red",
        });
      } else if (this.latestpHLevel > tilapiaMax) {
        newAlerts.push({
            message: "pH level is too high for Tilapia (above
8.5).",
          color: "red",
        });
      }
    } else if (this.selectedFish === "mudcrab") {
      if (this.latestpHLevel < mudcrabMin) {
        newAlerts.push({
            message: "pH level is too low for Mudcrab (below
6.5).",
          color: "red",
        });
      } else if (this.latestpHLevel > mudcrabMax) {
        newAlerts.push({
            message: "pH level is too high for Mudcrab (above
8.0).",
          color: "red",
        });
      }
    } else if (this.selectedFish === "shrimp") {
      if (this.latestpHLevel < shrimpMin) {
        newAlerts.push({
             message: "pH level is too low for Shrimp (below
7.5).",
          color: "red",
        });
```

```
      } else if (this.latestpHLevel > shrimpMax) {
        newAlerts.push({
            message: "pH level is too high for Shrimp (above
8.5).",
          color: "red",
        });
      }
    } else if (this.selectedFish === "galunggong") {
      if (this.latestpHLevel < galunggongMin) {
        newAlerts.push({
           message: "pH level is too low for Galunggong (below
6.5).",
          color: "red",
        });
      } else if (this.latestpHLevel > galunggongMax) {
        newAlerts.push({
          message: "pH level is too high for Galunggong (above
8.5).",
          color: "red",
        });
      }
    }


    this.alerts = newAlerts;
  },
  },
};
</script>



<style scoped>

body {
  font-family: 'Roboto', sans-serif;
  background-color: #f4f7fb;
```

```
  color: black;
  margin: 0;
  padding: 0;
}

.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 30px;
  background: rgba(255, 255, 255, 0.7);
  border-radius: 10px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
}


.box-container {
  display: flex;
  justify-content: space-between;
  gap: 20px;
  flex-wrap: wrap;
  background: rgba(255, 255, 255, 0.6);
}


.box {
  background-color: rgba(255, 255, 255, 0.7);
  border-radius: 10px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
  padding: 20px;
  flex: 1 1 30%;
  min-width: 280px;
}


.box-title {
  font-size: 1.5rem;
  color: black;
```

```
  margin-bottom: 15px;
  font-weight: bold;
}

.box-content {
  font-size: 1.1rem;
  color: black;
}


.data-item {
  font-size: 1.2rem;
  margin: 10px 0;
  color: black;
}

.alert-section .alert {
  padding: 15px;
  margin-top: 15px;
  font-size: 1.1rem;
  font-weight: bold;
  border-radius: 5px;
  color: rgb(0, 0, 0);
}


.button {
  padding: 10px 20px;
  font-size: 1rem;
  border-radius: 5px;
  cursor: pointer;
}

.primary-button {
  background-color: #3498db;
  color: white;
  border: none;
```

```
  transition: background-color 0.3s ease;
}

.primary-button:hover {
  background-color: #2980b9;
}

.monitoring-section h3 {
  color: black;
}


.data-table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 20px;
}

.data-table th, .data-table td {
  border: 1px solid #833434;
  padding: 12px;
  text-align: left;
  color: black;
}

.data-table th {
  background-color: #f4f4f4;
  font-weight: bold;
}

.data-table tbody tr:nth-child(even) {
  background-color: #f9f9f9;
}

.data-table tbody tr:hover {
  background-color: #f1f1f1;
}
```

```css
.fetch-data-section {
  text-align: center;
  margin-top: 30px;
}

.fetch-data-section input {
  padding: 10px;
  font-size: 1rem;
  border-radius: 5px;
  border: 1px solid #ccc;
}

.fetch-data-section button {
  margin-left: 10px;
}


.divider {
  margin: 20px 0;
  height: 1px;
  background-color: #ddd;
}


@media (max-width: 768px) {
  .container {
    padding: 20px;
  }

  .box-container {
    flex-direction: column;
  }

  .data-table {
    font-size: 0.9rem;
```

```
  }
}

</style>
```

SensorStore.js:

```
import { defineStore } from "pinia";

export const useSensorStore = defineStore("sensorStore", {
  state: () => ({
    pHLevel: null,
    pHThresholds: {
      min: 4,
      max: 9,
    },
    alerts: [],
    latestPHData: [],
  }),
  actions: {
    handleIncomingData({ topic, message }) {
      if (topic === "sensors/ph") {
        this.pHLevel = message;
        this.addPHData(message);
        this.checkpHAlerts(message);
      }
    },

    addPHData(pH) {
      // Add new data to the beginning of the array
      this.latestPHData.unshift({
        time: new Date().toLocaleTimeString(),
        phLevel: pH,
      });

      // Keep only the last 10 entries
      if (this.latestPHData.length > 10) {
```

```
        this.latestPHData.pop();
      }
    },
    checkpHAlerts(pH) {
      if (pH < this.pHThresholds.min) {
        this.addAlert("pH level is too low!", "red");
      } else if (pH > this.pHThresholds.max) {
        this.addAlert("pH level is too high!", "red");
      } else {

        this.removeAlert("pH level is too low!");
        this.removeAlert("pH level is too high!");
      }
    },
    addAlert(message, color) {
      // Prevent adding duplicate alerts
           if  (!this.alerts.some(alert  =>  alert.message  ===
message)) {
        this.alerts.push({ message, color });
      }
    },
    removeAlert(alertMessage) {
       this.alerts = this.alerts.filter(alert => alert.message
!== alertMessage);
    },
    setpHThresholds(min, max) {
      this.pHThresholds = { min, max };
    }
  },
  getters: {
    getLatestpHLevel: (state) => state.pHLevel,
    getAlerts: (state) => state.alerts,
    getLatestPHData: (state) => state.latestPHData,
  }
});
```

router.js

```
import { createRouter, createWebHistory } from 'vue-router';
import LoginPage from '../components/LoginPage.vue';
import SensorStore from '../components/SensorStore.vue';

const routes = [
  {
    path: '/',
    name: 'Login',
    component: LoginPage,
  },
  {
    path: '/sensor-store',
    name: 'SensorStore',
    component: SensorStore,
  },
];

const router = createRouter({
  history: createWebHistory(),
  routes,
});

export default router;
```

mqttService.js:

```
import { database } from "@/firebaseConfig";
import { ref, set } from "firebase/database";
import mqtt from "mqtt";

let client = null;
let isSubscribed = false;

export const connectMQTT = (brokerUrl, options) => {
  client = mqtt.connect(brokerUrl, options);
```

```
  client.on("connect", () => {
    console.log("Connected to MQTT broker");
  });

  client.on("error", (err) => {
    console.error("MQTT connection error:", err);
  });
};

export const subscribeToTopic = (topics, callbacks) => {
  if (client && !isSubscribed) {
    topics.forEach((topic) => {
      client.subscribe(topic, (err) => {
        if (err) {
                console.error(`Failed to subscribe to topic:
${topic}`);
        } else {
          console.log(`Subscribed to ${topic}`);
        }
      });
    });

    client.on("message", (topic, message) => {
      const messageStr = message.toString();
           console.log("Received message:", messageStr, "on
topic:", topic);

      if (callbacks[topic]) {
        callbacks[topic](messageStr);
      }

      // Send data to Firebase
      sendDataToFirebase(topic, messageStr);
    });

    isSubscribed = true;
  }
```

```
};

const sendDataToFirebase = (topic, message) => {

  const now = new Date();


  const timestamp = now.getTime();


   const date = now.toISOString().split('T')[0];   // UTC date
(YYYY-MM-DD)


            const      firebaseRef      =      ref(database,
`mqttData/${date}/${topic}/${timestamp}`);

  set(firebaseRef, {
    topic,
    message,
    date,
    timestamp,
  })
    .then(() => {
      console.log("Data saved to Firebase successfully");
    })
    .catch((error) => {
      console.error("Error saving data to Firebase:", error);
    });
};


export const stopSendingData = () => {
  if (client && isSubscribed) {

    client.unsubscribe("sensors/ph", (err) => {
      if (err) {
```

```
        console.error("Failed to unsubscribe:", err);
      } else {
        console.log("Unsubscribed from all topics");
      }
    });


    client.removeAllListeners("message");

    isSubscribed = false;
  }


  if (client) {
    client.end(() => {
      console.log("MQTT client disconnected");
    });
  }
};
```

firebaseConfig.js

```
import { initializeApp } from "firebase/app";
import { getDatabase } from "firebase/database";

const firebaseConfig = {
  apiKey: "AIzaSyBt55S8j6SJH98vXqPARvAfIO_VlU8dW9M",
  authDomain: "embeddedproj-eccc2.firebaseapp.com",
                                                databaseURL:
"https://embeddedproj-eccc2-default-rtdb.asia-southeast1.fireb
asedatabase.app",
  projectId: "embeddedproj-eccc2",
  storageBucket: "embeddedproj-eccc2.firebasestorage.app",
  messagingSenderId: "746785243044",
  appId: "1:746785243044:web:de1afe8fba410092cecb31",
  measurementId: "G-H47M17P4GY"
};
```

```
// Initialize Firebase
const app = initializeApp(firebaseConfig);
const database = getDatabase(app);

export { database };
```

main.js

```
// src/main.js
import { createPinia } from 'pinia';
import { createApp } from 'vue';
import App from './App.vue';
import router from './router/router';


const app = createApp(App);
const pinia = createPinia();

app.use(pinia);
app.use(router);
app.mount('#app');
```

App.vue:

```
<template>
  <div id="app">

                              <LoginPage       v-if="!isLoggedIn"
@loginSuccess="handleLoginSuccess" />


    <SensorStore v-else />
```

```
      <button v-if="isLoggedIn" @click="logout" class="button
primary-button">Logout</button>
  </div>
</template>

<script>
import { stopSendingData } from "@/services/mqttService";
import LoginPage from './components/LoginPage.vue';
import SensorStore from './components/SensorStore.vue';

export default {
  name: 'App',
  components: {
    LoginPage,
    SensorStore,
  },
  data() {
    return {
      isLoggedIn: false,
    };
  },
  methods: {
    handleLoginSuccess() {

      this.isLoggedIn = true;
    },
    logout() {
    // Call to stop sending data
    stopSendingData();
    this.isLoggedIn = false;

    console.log("User logged out");
  },
  },
};
</script>
```

```
<style>

html, body {
  margin: 0;
  padding: 0;
  height: 100%;
}

#app {
  text-align: center;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  min-height: 100vh;


  background-image: url('@/assets/aquatic_bg.jpg');
  background-size: cover;
  background-position: center;
  background-repeat: no-repeat;
  color: white;
}


button {
  margin-top: 20px;
  padding: 10px 20px;
  background-color: #d10d0d;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

button:hover {
```

```
  background-color: #b10404;
}
</style>
```

ESP8266 Arduino Ide Code:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// Wi-Fi Credentials
const char* ssid = "0R@c13";
const char* password = "K33p0nkeep1ng0n";

// HiveMQTT Credentials
const char* mqtt_server =
"ed9b94fcc7ad47119453174c68d2ae31.s1.eu.hivemq.cloud";
const int mqtt_port = 8883; // Use 1883 for non-secure, or
8883 for SSL
const char* mqtt_user = "Admin";
const char* mqtt_password = "Admin123";

// MQTT Client
WiFiClientSecure espClient; // For SSL connections
PubSubClient client(espClient);

// pH Sensor Setup
#define PH_SENSOR_PIN A0     // Analog pin where the pH sensor
is connected

// MQTT Topic
const char* ph_topic = "sensors/ph";

// Timing variables
unsigned long lastPublishTime = 0;

// Function to connect to Wi-Fi
void setupWiFi() {
```

```
  delay(10);
  Serial.print("Connecting to Wi-Fi");
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("\nConnected to Wi-Fi");
}

// Function to reconnect to MQTT
void reconnect() {
  unsigned long startAttemptTime = millis();
   while (!client.connected() && millis() - startAttemptTime <
10000) {
    yield(); // Allows background tasks to run
    Serial.print("Attempting MQTT connection...");
            if   (client.connect("ESP8266Client",   mqtt_user,
mqtt_password)) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}


// Function to read pH value from the sensor
float readPH() {
  int sensorValue = analogRead(PH_SENSOR_PIN);
  float voltage = sensorValue * (3.3 / 1023.0);
   float phValue = 7 + (voltage - 2.5);  // Adjust based on
calibration
```

```
  return phValue;
}

// Function to publish pH data
void publishData(float phValue) {
  char phStr[8];
  dtostrf(phValue, 6, 2, phStr);
  client.publish(ph_topic, phStr);
  Serial.print("pH published: ");
  Serial.println(phStr);
}

void setup() {
  Serial.begin(115200);
  setupWiFi();
  espClient.setInsecure(); // For SSL without certificates
  client.setServer(mqtt_server, mqtt_port);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  if (millis() - lastPublishTime >= 5000) {
    lastPublishTime = millis();
    float phValue = readPH();
    publishData(phValue);
  }
}
```

Appendix D: GitHub link of the Front End, Backend and the ESP8266 code

https://github.com/jhizola18/Aquatic_pH_Level_Monitoring.git

PROJECT RUBRIC

Criteria	4 - Exceptional	3 - Proficient	2 - Developing	1-Needs Improvement	Score
Relevance and Innovation	Highly innovative application of IoT in aquaculture, solving a significant problem with a clear and meaningful solution.	Effective IoT application addressing a relevant aquaculture problem, showing moderate creativity.	Limited IoT application, addressing a generic or poorly defined problem.	Minimal or unclear relevance to aquaculture, with no meaningful use of IoT.	
Technical Implementation	Fully functional system with seamless integration of hardware, software, and communication, demonstrating reliability and excellent performance.	Functional system with minor issues, good integration, and satisfactory performance in testing.	Partially functional system with gaps in integration or performance.	System is largely non-functional, with major technical issues or missing components.	
Documentation and Presentation	Clear, comprehensive, and professional documentation, with well-organized sections and visuals. Polished presentation via video and GitHub.	Clear and complete documentation with minor issues in organization or visuals. Presentation is adequate and informative.	Documentation lacks clarity or is incomplete, with minimal use of visuals. Presentation lacks detail or clarity.	Poorly organized or missing documentation, with minimal or uninformative presentation.	
Testing and Results	Comprehensive testing with detailed analysis, supported by meaningful charts and graphs, showing reliability and effectiveness.	Adequate testing demonstrating reliability, with some analysis and supporting visuals.	Limited or incomplete testing, with minimal analysis or visuals.	Minimal or absent testing, with no meaningful analysis or evidence of results.	

Develop appropriate experimentation	The students are able to develop a lab experiment/activity appropriate to the chosen topic and aligned to the engineering principles learned in the previous experiments. The developed lab experiment/activity has a complete presentation of block/schematic diagram, provided the use of modern tools and techniques, and integrated principles/discussions that were based on proven studies.	The students are able to develop components of a laboratory experiment appropriate to the chosen topic and aligned to the engineering principles learned in the previous experiments	The students are able to develop component of a laboratory experiment but has no presentations of analysis of data yet has presented conclusion or recommendation	The students are unable to develop a basic component of a laboratory experiment.	
Conduct appropriate experimentation	The students are able to conduct appropriate laboratory experiment/activity with sufficient results and able draw a valid conclusion	The students are able to conduct appropriate laboratory experiment/activity with sufficient results and able draw a valid conclusion	The students conduct some laboratory experiments/activities but did not arrive at the correct results	The students are unable to conduct a laboratory experiment/activity.	
Protocol to conduct an experiment	Develop a protocol to conduct an experiment and members follow good and safe laboratory practice at all times in the conduct of experiments.	Develop a protocol to conduct an experiment exceeding the requirements	The students are able to determine the objectives of the experiment or test to be performed but fail to identify test to performed	The students failed to determine the objectives of the experiment or test to be performed.	

Ability to analyze and interpret data	The students use multiple data analysis techniques appropriate for data collected, informative with respect to the experimentation/activity being conducted. Data analysis is reported with comprehensive interpretation.	The students use adequate data analysis techniques appropriate for data collected, informative with respect to the experimentation/activity being conducted. Data analysis is reported with sufficient interpretation.	The students provide limited analysis of data with no interpretation.	The students are unable to provide analysis and interpretation of data.	
Use of engineering judgment to draw conclusions	The student was able to use engineering judgement* more than sufficient to draw correct conclusions and was able to provide new insights.	The student was able to use engineering judgement more than sufficient to draw correct conclusions.	The student was able to use engineering judgement but insufficient to draw correct conclusions.	The student failed to use engineering judgement to draw conclusions.	
					Total Score
					Percentage rating = (Total Score/36) x 100

Evaluated by:

_____ _____

Printed Name and Signature of Faculty Member Date