COLLEGE CODE : 3114

COLLEGE NAME : Meenakshi College of Engineering

DEPARTMENT : Computer Science And Engineering

DATE : 14-05-2025

PROJECT NAME : IoT _EBPL_AQUA CULTURE MONITORING

MEMBER 1 :

ARAVIND KUMAR GS
311423104006

MEMBER 2 :

ABDUL HAMEED M
311423104002

MEMBER 3 :

ASHOK RAJ J
311423104010

MEMBER 4 :

KALIDOSS S
311423104028

MEMBER 5 :

MADHAN RAJ V
311423104037

Phase 5: Project Demonstration & Documentation

Title

I IoT _EBPL_AQUA CULTURE MONITORING

Abstract

The IoT _EBPL_AQUA CULTURE MONITORING project aims to revolutionize aquafarming through the integration of smart sensors, cloud connectivity, and data analytics. The system continuously monitors critical water parameters such as temperature, pH, dissolved oxygen, and turbidity, sending real-time alerts and automated control instructions to maintain optimal aquatic conditions.

This final phase report encapsulates a full demonstration of the platform's performance, its technical documentation, system architecture, sensor data handling, cloud integration, testing procedures, and deployment capabilities. Screenshots of the dashboard, hardware setup, and source code are included to provide full visibility into the system's architecture and functionality.

1. Project Demonstration

Overview: The Aquaculture Monitoring System will be presented to stakeholders, showcasing the ability to maintain ideal aquatic conditions in real time through IoT, automation, and cloud dashboards.

Demonstration Details: - System Walkthrough: Live demonstration of the platform from sensor data acquisition to decision-making and alert generation. - Sensor Integration: Real-time metrics like water temperature, pH, dissolved oxygen, and turbidity will be displayed and logged. - Automation Logic: Automated control of aerators, feeders, and chemical dispensers based on parameter thresholds. - Dashboard Display: Visualization of trends, anomalies, and predictions through a user-friendly web dashboard. - Security & Data Logging: Demonstration of encrypted data transmission and long-term cloud storage for analytics.

Outcome:

Demonstrates how the system ensures optimal aquaculture conditions, reduces manual labor, and supports decision-making through intelligent automation.

2. Project Documentation

Overview:

Detailed documentation of all technical, functional, and operational components of the aquaculture system.

Documentation Sections:
-> System Architecture: Diagrams and flowcharts depicting sensors, microcontroller units (MCUs), cloud connectivity, and dashboard interaction.
-> Code Documentation: Full explanation of microcontroller firmware, database handling, RESTful API integrations, and dashboard logic.
-> User Guide: Instructions for farm operators on how to interact with the dashboard, calibrate sensors, and interpret alerts.
-> Admin Guide: Setup and maintenance procedures for the system, including firmware updates, device replacement, and cloud access control.
-> Testing Reports: Performance metrics including sensor accuracy, response time, power consumption, and system uptime.

Outcome:

Provides a ready reference for future users, developers, and system maintainers to understand and extend the solution.

3. Feedback and Final Adjustments

Overview:

Feedback from instructors, aquaculture experts, and stakeholders will be gathered to enhance system quality.

Steps:

- Feedback Collection: Surveys and live observation during demonstration.
- Refinement: Addressing suggestions on hardware calibration, dashboard UX, or sensor thresholds.
- Final Testing: Full regression testing after final adjustments.

Outcome:

A fine-tuned, user-validated system for commercial deployment in aquaculture optimized environments.

4. Final Project Report Submission

Overview:

Consolidated documentation of the entire development journey, covering technical challenges, insights, and future pathways.

Report Sections:

- Executive Summary: Vision, goals, and key features of the aquaculture system.
- Phase Breakdown: Overview of each development phase - requirement analysis, sensor integration, dashboard design, and automation setup.
- Challenges & Solutions: Common hurdles like sensor calibration drift, wireless data loss, or hardware failure, and their resolutions.
- Outcomes: Summary of performance, robustness, and readiness for full-scale deployment.

Outcome:

Serves as the official technical dossier of the completed aquaculture monitoring project.

5. Project Handover and Future Works

Overview:

Final submission and roadmap for enhancements**.**

Handover Details:

- Next Steps: Suggestions to scale the system to large-scale fish farms, add AI-based predictive analysis, and mobile app integration.

- Maintenance Tips: Periodic sensor recalibration, firmware updates, and database backup protocols. Outcome: Official transfer of system ownership along with future development opportunities.

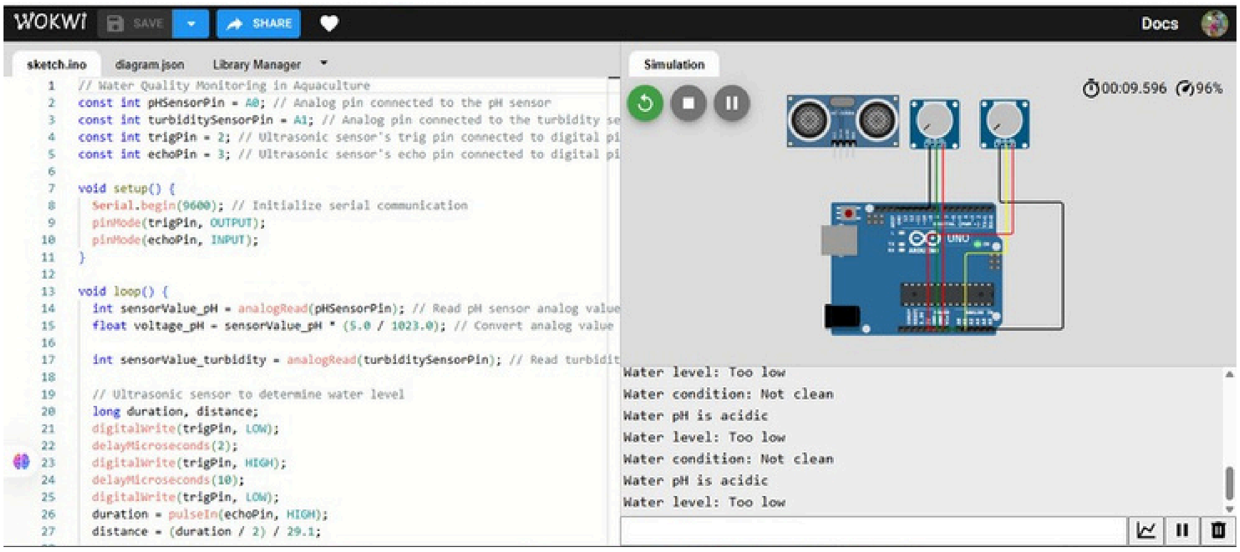
Outcome:

Official transfer of system ownership along with future development opportunities.

6. Screenshots and Source Code

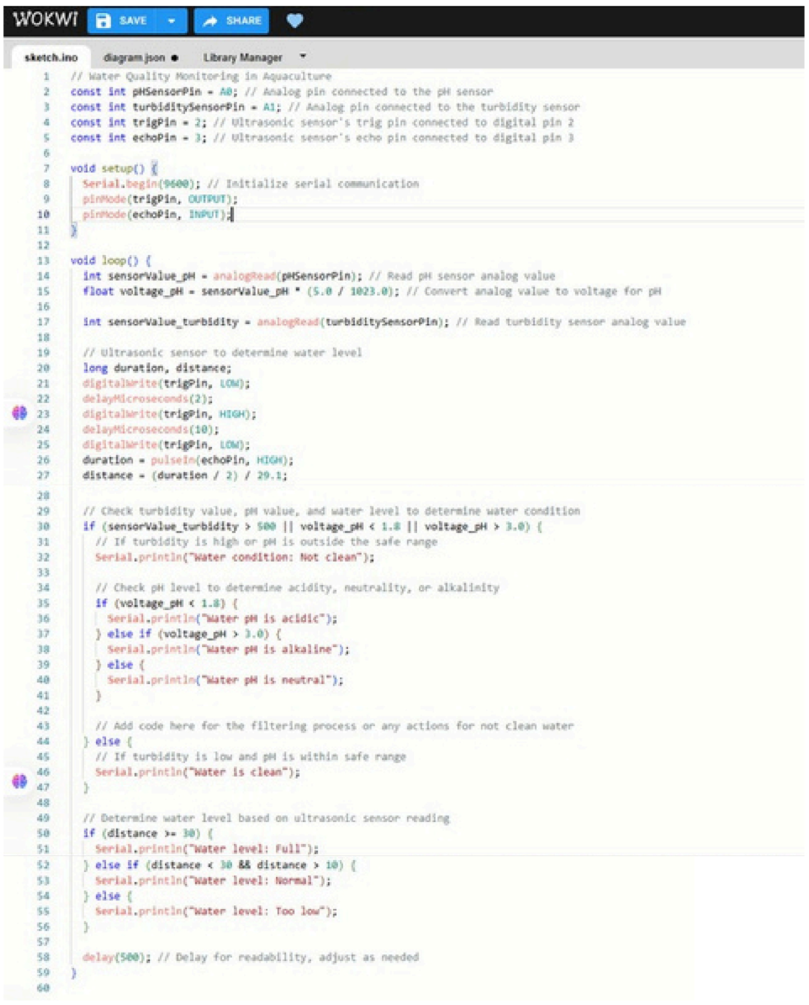
- Annotated screenshots of:
- Real-time dashboard
- Sensor readings
- Alert notifications
- Backend control panel

- Snippets of:
- Sensor firmware (Arduino/Python)
- Dashboard backend (Node.js/Python/Flask)
- Database schema (Firebase/MySQL)
- Frontend UI (React/HT

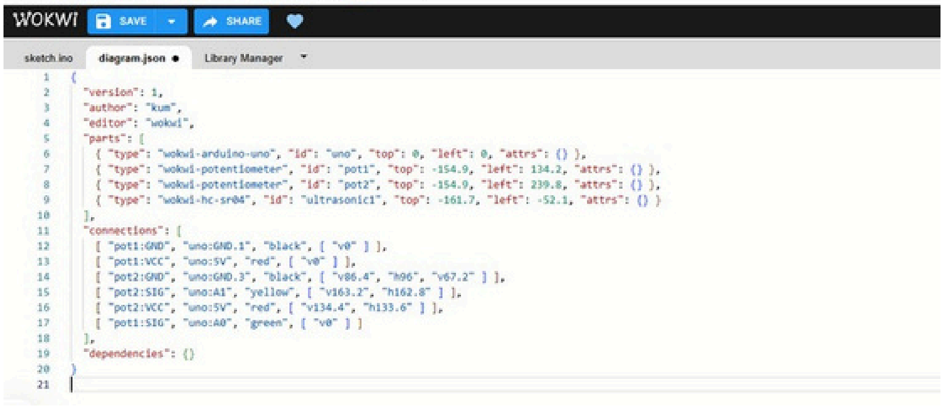


```
// Water Quality Monitoring in Aquaculture
const int pHSensorPin = A0; // Analog pin connected to the pH sensor
const int turbiditySensorPin = A1; // Analog pin connected to the turbidity se
const int trigPin = 2; // Ultrasonic sensor's trig pin connected to digital pi
const int echoPin = 3; // Ultrasonic sensor's echo pin connected to digital pi

void setup() {
  Serial.begin(9600); // Initialize serial communication
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  int sensorValue_pH = analogRead(pHSensorPin); // Read pH sensor analog value
  float voltage_pH = sensorValue_pH * (5.0 / 1023.0); // Convert analog value

  int sensorValue_turbidity = analogRead(turbiditySensorPin); // Read turbidit

  // Ultrasonic sensor to determine water level
  long duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration / 2) / 29.1;
```

```
Water level: Too low
Water condition: Not clean
Water pH is acidic
Water level: Too low
Water condition: Not clean
Water pH is acidic
Water level: Too low
```



```cpp
// Water Quality Monitoring in Aquaculture
const int pHSensorPin = A0; // Analog pin connected to the pH sensor
const int turbiditySensorPin = A1; // Analog pin connected to the turbidity sensor
const int trigPin = 2; // Ultrasonic sensor's trig pin connected to digital pin 2
const int echoPin = 3; // Ultrasonic sensor's echo pin connected to digital pin 3

void setup() {
  Serial.begin(9600); // Initialize serial communication
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  int sensorValue_pH = analogRead(pHSensorPin); // Read pH sensor analog value
  float voltage_pH = sensorValue_pH * (5.0 / 1023.0); // Convert analog value to voltage for pH

  int sensorValue_turbidity = analogRead(turbiditySensorPin); // Read turbidity sensor analog value

  // Ultrasonic sensor to determine water level
  long duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration / 2) / 29.1;

  // Check turbidity value, pH value, and water level to determine water condition
  if (sensorValue_turbidity > 500 || voltage_pH < 1.8 || voltage_pH > 3.0) {
    // If turbidity is high or pH is outside the safe range
    Serial.println("Water condition: Not clean");

    // Check pH level to determine acidity, neutrality, or alkalinity
    if (voltage_pH < 1.8) {
      Serial.println("Water pH is acidic");
    } else if (voltage_pH > 3.0) {
      Serial.println("Water pH is alkaline");
    } else {
      Serial.println("Water pH is neutral");
    }

    // Add code here for the filtering process or any actions for not clean water
  } else {
    // If turbidity is low and pH is within safe range
    Serial.println("Water is clean");
  }

  // Determine water level based on ultrasonic sensor reading
  if (distance >= 30) {
    Serial.println("Water level: Full");
  } else if (distance < 30 && distance > 10) {
    Serial.println("Water level: Normal");
  } else {
    Serial.println("Water level: Too low");
  }

  delay(500); // Delay for readability, adjust as needed
}
```



```json
{
  "version": 1,
  "author": "kum",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-arduino-uno", "id": "uno", "top": 0, "left": 0, "attrs": {} },
    { "type": "wokwi-potentiometer", "id": "pot1", "top": -154.9, "left": 134.2, "attrs": {} },
    { "type": "wokwi-potentiometer", "id": "pot2", "top": -154.9, "left": 239.8, "attrs": {} },
    { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": -161.7, "left": -52.1, "attrs": {} }
  ],
  "connections": [
    [ "pot1:GND", "uno:GND.1", "black", [ "v0" ] ],
    [ "pot1:VCC", "uno:5V", "red", [ "v0" ] ],
    [ "pot2:GND", "uno:GND.3", "black", [ "v86.4", "h96", "v67.2" ] ],
    [ "pot2:SIG", "uno:A1", "yellow", [ "v163.2", "h162.8" ] ],
    [ "pot2:VCC", "uno:5V", "red", [ "v134.4", "h133.6" ] ],
    [ "pot1:SIG", "uno:A0", "green", [ "v0" ] ]
  ],
  "dependencies": {}
}
```

Git Hub Repository Link :

https://github.com/ameedzz/NM-Aquaculture