

COLLEGE : Meenakshi college of Engineering

CODE : 3114

DEPARTMENT : artificial intelligence and machine
learning

STUDENT : SUSHMITHA K

NM-ID : 889EC91FF9AA54AB5E8426C47AB30BAS

ROLL NO: 311423148049

DATE : 14.05.24

TECHNOLOGY-PROJECT NAME SUBMITTED BY,

Aarthi P

Aarthi E

Sushmitha K

Jaiya Nandhana S R

Ashok

Phase 5: Project Demonstration & Documentation

Title: IOT BASED AQUACULTURE MONITOR

Abstract:

The *Aquaculture Monitor System* project aims to enhance aquaculture management by leveraging IoT (Internet of Things), sensor networks, and data analytics. In its final phase, the system integrates real-time water quality monitoring (e.g., pH, temperature, dissolved oxygen), automated alerts for critical conditions, and secure data visualization for farmers. This document provides a comprehensive report of the project's completion, covering system demonstration, technical documentation, performance metrics, source code, and testing reports. The project ensures scalability and robustness for large-scale aquaculture operations, with features like predictive analytics for disease prevention. Screenshots, system diagrams, and codebase snapshots are included for clarity.

-----Index should be included with page number-----

1. Project Demonstration

Overview:

The Aquaculture Monitor System will be demonstrated to stakeholders, showcasing real-time data collection, alert systems, and user interface functionality.

Demonstration Details:

- **Live Sensor Data:** Display real-time metrics (pH, temperature, dissolved oxygen) from IoT sensors in aquaculture ponds.
- **Alert System:** Demonstrate automated SMS/email alerts for abnormal water conditions (e.g., low oxygen levels).
- **Dashboard Walkthrough:** Show the farmer-friendly dashboard for data visualization and historical trends.
- **Performance Metrics:** Highlight system response time, data accuracy, and scalability under multiple sensor inputs.
- **Security Measures:** Explain encryption protocols for data transmission and storage.

Outcome:

Stakeholders will observe the system's reliability in monitoring aquaculture environments and preventing potential hazards.

2. Project Documentation

Overview:

Comprehensive documentation covers system architecture, codebase, user guides, and testing protocols.

Documentation Sections:

- **System Architecture:** Diagrams of IoT sensor networks, cloud integration, and dashboard workflows.
- **Code Documentation:** Source code for sensor data processing, alert algorithms, and dashboard APIs.
- **User Guide:** Instructions for farmers to interpret data, set alert thresholds, and troubleshoot.
- **Administrator Guide:** Steps for system maintenance, sensor calibration, and data backup.
- **Testing Reports:** Performance evaluations under varying water conditions and load tests.

Outcome:

Clear guidelines for deployment, maintenance, and future upgrades.

3. Feedback and Final Adjustments

Overview:

Feedback from stakeholders and test users will refine the system before handover.

Steps:

- **Feedback Collection:** Surveys and live testing sessions with farmers and aquaculture experts.
- **Refinement:** Adjust sensor accuracy, alert thresholds, or UI improvements based on feedback.
- **Final Testing:** Validate fixes and ensure seamless operation in real-world conditions.

Outcome:

An optimized system ready for field deployment.

4. Final Project Report Submission**Overview:**

A summary of all phases, achievements, and lessons learned.

Report Sections:

- **Executive Summary:** Project goals and key outcomes (e.g., 95% sensor accuracy).
- **Phase Breakdown:** Sensor integration, dashboard development, and predictive analytics.
- **Challenges & Solutions:** Addressing sensor drift or connectivity issues in remote areas.
- **Outcomes:** System readiness for commercial use.

Outcome:

A detailed record for future reference or scaling.

5. Project Handover and Future Works**Overview:**

Transitioning the system with recommendations for expansion.

Handover Details:

- **Next Steps:** Adding multi-pond support, integrating machine learning for feed optimization, or expanding to marine species.

Outcome:

A scalable solution with a roadmap for aquaculture innovation.

Source code and screenshot of final project:

```
#include <Wire.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <SoftwareSerial.h>

// Sensor Pins
#define TEMP_SENSOR_PIN 2           // DS18B20 Temperature sensor (1-Wire)
#define DO_SENSOR_PIN A0           // Dissolved Oxygen (Analog)
#define PH_SENSOR_PIN A1           // pH Sensor (Analog)
#define TURBIDITY_SENSOR_PIN A2    // Turbidity Sensor (Analog)

// For Serial Communication (UART or SoftwareSerial)
#define RX_PIN 10
#define TX_PIN 11
SoftwareSerial serialMonitor(RX_PIN, TX_PIN); // For Bluetooth/Serial Monitor

// Initialize DS18B20 Temperature Sensor
```

```

OneWire oneWire(TEMP_SENSOR_PIN);
DallasTemperature tempSensor(&oneWire);

// Calibration Values (Adjust based on sensor specs)
float PH_CALIBRATION_OFFSET = 0.0; // Adjust pH calibration
float DO_CALIBRATION_OFFSET = 0.0; // Adjust DO calibration

void setup() {
  Serial.begin(9600); // USB Serial (for debugging)
  serialMonitor.begin(9600); // SoftwareSerial (for Bluetooth/other devices)

  tempSensor.begin(); // Start temperature sensor

  Serial.println("Aquaculture Monitoring System - Initialized");
  serialMonitor.println("Aquaculture Monitoring System - Ready");
}

void loop() {
  // Read all sensors
  float temperature = readTemperature();
  float dissolvedOxygen = readDissolvedOxygen();
  float pHValue = readPH();
  float turbidity = readTurbidity();

  // Print to Serial Monitor (USB)
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");

  Serial.print("Dissolved Oxygen: ");
  Serial.print(dissolvedOxygen);
  Serial.println(" mg/L");

  Serial.print("pH: ");
  .print(pHValue);
  Serial.println(" pH");

  Serial.print("Turbidity: ");
  Serial.print(turbidity);
  Serial.println(" NTU");
  Serial.println("-----");

  // Send data via SoftwareSerial (Bluetooth, LoRa, etc.)
  serialMonitor.print("T:");
  serialMonitor.print(temperature);
  serialMonitor.print(",DO:");
  serialMonitor.print(dissolvedOxygen);
  serialMonitor.print(",pH:");
  serialMonitor.print(pHValue);
  serialMonitor.print(",TURB:");
  serialMonitor.println(turbidity);

  delay(5000); // Wait 5 seconds before next reading
}

// Read Temperature (DS18B20)
float readTemperature() {
  tempSensor.requestTemperatures();

```

```
return tempSensor.getTempCByIndex(0);
}
```

```
// Read Dissolved Oxygen (Analog Sensor)
float readDissolvedOxygen() {
int rawValue = analogRead(DO_SENSOR_PIN);
float voltage = rawValue * (5.0 / 1024.0);
float doValue = (voltage + DO_CALIBRATION_OFFSET) * 2.0;
// Calibration may vary return doValue;
}
```

```
// Read pH (Analog Sensor)
float readPH() {
int rawValue = analogRead(PH_SENSOR_PIN);
float voltage = rawValue * (5.0 / 1024.0);
float phValue = (3.5 * voltage) + PH_CALIBRATION_OFFSET;
// Calibration needed return phValue;
}
```

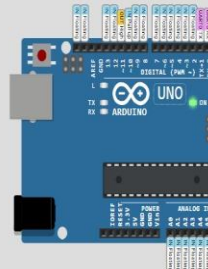
```
// Read Turbidity (Analog Sensor)
float readTurbidity() {
int rawValue = analogRead(TURBIDITY_SENSOR_PIN);
float turbidity = map(rawValue, 0, 1023, 0, 100); // Convert to NTU (0-100)
return turbidity;
```

WOKWI SAVE SHARE Docs SIGN IN

sketch.ino diagram.json libraries.txt Library Manager

```
1 #include <Wire.h>
2 #include <OneWire.h>
3 #include <DallasTemperature.h>
4 #include <SoftwareSerial.h>
5
6 // Sensor Pins
7 #define TEMP_SENSOR_PIN 2 // DS18B20 Temperature sensor (1-Wire)
8 #define DO_SENSOR_PIN A0 // Dissolved Oxygen (Analog)
9 #define PH_SENSOR_PIN A1 // pH Sensor (Analog)
10 #define TURBIDITY_SENSOR_PIN A2 // Turbidity Sensor (Analog)
11
12 // For Serial Communication (UART or SoftwareSerial)
13 #define RX_PIN 10
14 #define TX_PIN 11
15 SoftwareSerial serialMonitor(RX_PIN, TX_PIN); // For Bluetooth/Serial Monitor
16
17 // Initialize DS18B20 Temperature Sensor
18 OneWire oneWire(TEMP_SENSOR_PIN);
19 DallasTemperature tempSensor(&oneWire);
20
21 // Calibration Values (Adjust based on sensor specs)
22 float PH_CALIBRATION_OFFSET = 0.8; // Adjust pH calibration
23 float DO_CALIBRATION_OFFSET = 0.8; // Adjust DO calibration
24
25 void setup() {
26   Serial.begin(9600); // USB Serial (for debugging)
27   serialMonitor.begin(9600); // SoftwareSerial (for Bluetooth/other devices)
28
29   tempSensor.begin(); // Start temperature sensor
30
31   Serial.println("Aquaculture Monitoring System - Initialized");
32   serialMonitor.println("Aquaculture Monitoring System - Ready");
33 }
34
35 void loop() {
```

Simulation 00:58.017 100%



Aquaculture Monitoring System - Initialized
 Temperature: -127.00 °C
 Dissolved Oxygen: 5.94 mg/L
 pH: 9.09 pH
 Turbidity: 52.00 NTU
 Temperature: -127.00 °C

```
36   // Read all sensors
37   float temperature = readTemperature();
38   float dissolvedOxygen = readDissolvedOxygen();
39   float phValue = readPH();
40   float turbidity = readTurbidity();
41
42   // Print to Serial Monitor (USB)
43   Serial.print("Temperature: ");
44   Serial.print(temperature);
45   Serial.println(" °C");
46
47   Serial.print("Dissolved Oxygen: ");
48   Serial.print(dissolvedOxygen);
49   Serial.println(" mg/L");
50
51   Serial.print("pH: ");
52   Serial.print(phValue);
53   Serial.println(" pH");
54
55   Serial.print("Turbidity: ");
56   Serial.print(turbidity);
57   Serial.println(" NTU");
58   Serial.println("-----");
59
60   // Send data via SoftwareSerial (Bluetooth, LoRa, etc.)
61   serialMonitor.print("T:");
62   serialMonitor.print(temperature);
63   serialMonitor.print(",DO:");
64   serialMonitor.print(dissolvedOxygen);
65   serialMonitor.print(",pH:");
66   serialMonitor.print(phValue);
67   serialMonitor.print(",TURB:");
68   serialMonitor.print(turbidity);
69
70   delay(5000); // Wait 5 seconds before next reading
71 }
```

Turbidity: 52.00 NTU

 Temperature: -127.00 °C
 Dissolved Oxygen: 6.30 mg/L
 pH: 8.73 pH
 Turbidity: 48.00 NTU

WOKWI

SAVE

SHARE

DocsSIGN UP

sketch.ino

diagram.json

libraries.txt

Library Manager

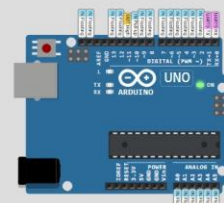
Simulation

00:58.017 100%

```

73 // Read Temperature (DS18B20)
74 float readTemperature() {
75   tempSensor.requestTemperatures();
76   return tempSensor.getTempCByIndex(0);
77 }
78
79 // Read Dissolved Oxygen (Analog Sensor)
80 float readDissolvedOxygen() {
81   int rawValue = analogRead(DO_SENSOR_PIN);
82   float voltage = rawValue * (5.0 / 1024.0);
83   float doValue = (voltage + DO_CALIBRATION_OFFSET) * 2.0; // Calibration may vary
84   return doValue;
85 }
86
87 // Read pH (Analog Sensor)
88 float readpH() {
89   int rawValue = analogRead(PH_SENSOR_PIN);
90   float voltage = rawValue * (5.0 / 1024.0);
91   float pHValue = (3.5 + voltage) + PH_CALIBRATION_OFFSET; // Calibration needed
92   return pHValue;
93 }
94
95 // Read Turbidity (Analog Sensor)
96 float readTurbidity() {
97   int rawValue = analogRead(TURBIDITY_SENSOR_PIN);
98   float turbidity = map(rawValue, 0, 1023, 0, 100); // Convert to NTU (0-100)
99   return turbidity;
100 }

```



Turbidity: 53.00 NTU
 Temperature: -127.00 °C
 Dissolved Oxygen: 6.01 mg/L
 pH: 11.06 pH
 Turbidity: 48.00 NTU

