****

**STTH3113 SENSOR-BASED SYSTEM**

**SEMESTER 2024/2025 (A242)**

**GROUP A**

**FINAL ASSIGNMENT**

**PREPARED BY:**

|  |  |
| --- | --- |
| **NAME** | **MATRIC NO** |
| AHMAD AZEEM BIN ANUAR | 288777 |

**LECTURER:**

**EN AHMAD HANIS BIN MOHD SHABLI**

**DUE DATE:**

**19/06/2025**

**SMART FLOOD DETECTION AND ALERT SYSTEM USING ESP32**

**PROBLEM STATEMENTS:**

Floods are one of the most devastating natural disasters, often resulting in significant damage to infrastructure, the environment, and human lives. In many rural or low-income urban areas, the absence of early warning systems contributes to a delayed response, which can amplify the impact of flooding.

Traditional flood detection systems are either expensive, difficult to deploy, or not scalable for local use. Therefore, there is a strong need for a low-cost, real-time flood monitoring system that can provide early warnings and help mitigate risks.

**OBJECTIVES:**

This project aims to develop an intelligent, IoT-enabled flood detection and monitoring system using the ESP32 microcontroller and multiple environmental sensors. The system is designed to:

* Monitor environmental parameters (humidity and temperature) and water level conditions continuously.
* Visually display the data on an OLED display for local monitoring.
* Transmit live sensor readings to a cloud database via Wi-Fi.
* Provide real-time visualization of sensor data through a responsive web dashboard.

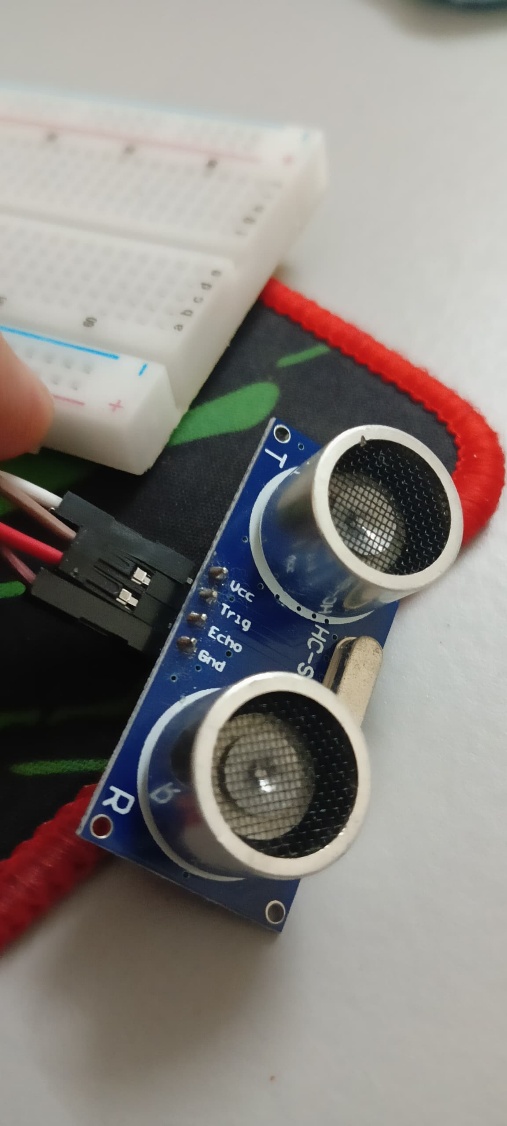
Use sensor data thresholds to categorize flood risk into Normal, Caution, and Critical zones using colour coding.

The ultimate goal is to provide a low-cost, scalable, and easy-to-deploy solution for flood-prone households, schools, and small communities.

**SELECTED SENSORS:**

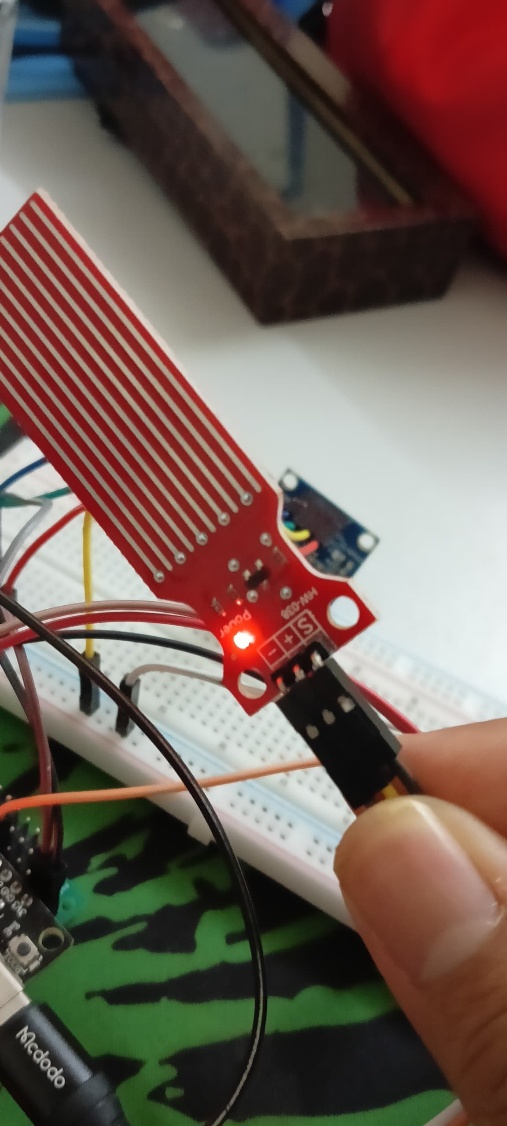
1. ***Ultrasonic Sensor (HC-SR04)***

The ultrasonic sensor continuously measures the distance between the sensor and the water surface. This data is crucial in detecting changes in water level—especially rapid rises that indicate potential flooding. The non-contact nature of this sensor makes it ideal for wet and dirty environments.



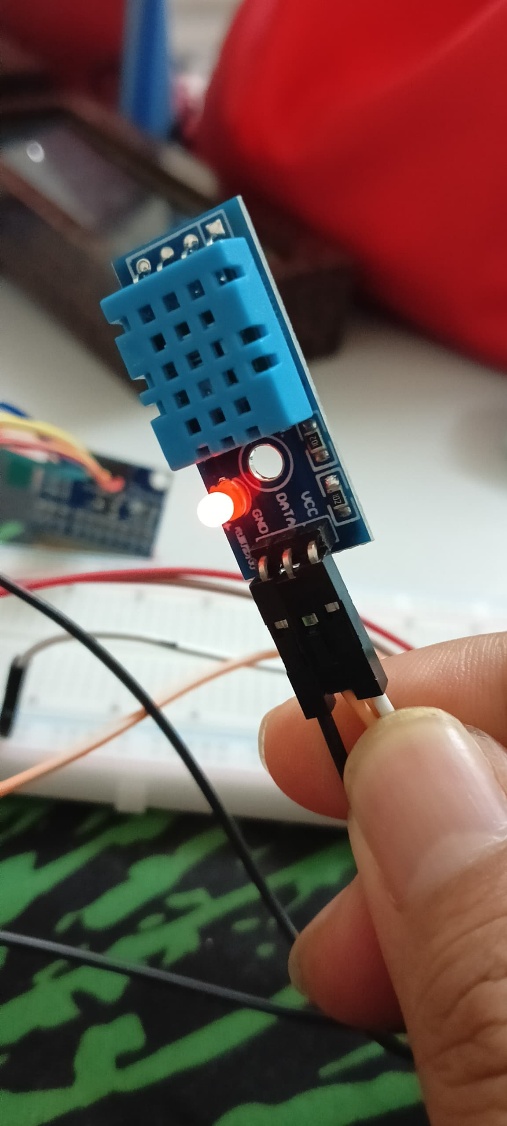
1. ***Water Sensor (Analog)***

This sensor detects the presence of water or moisture and measures the intensity (wetness level). It is particularly useful in confirming if water has reached a surface level (e.g., floor or threshold), helping identify whether flooding has physically entered the premises.



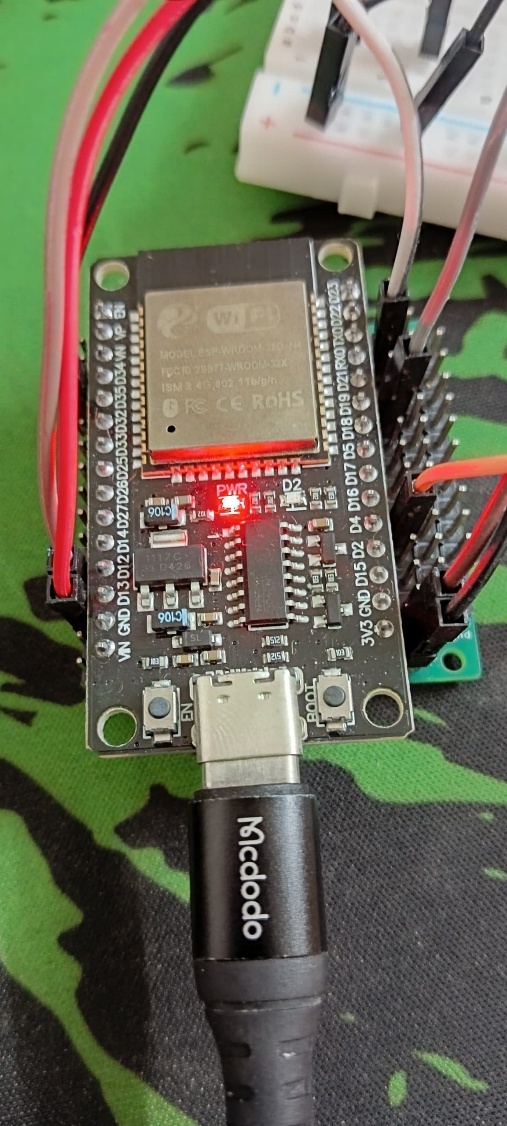
1. ***DHT11 Sensor***

This temperature and humidity sensor is used to monitor atmospheric conditions, which are often precursors to flooding. For example, rising humidity levels and sudden temperature drops may suggest heavy rain or storm conditions.



1. ***ESP32 Microcontroller***

The ESP32 acts as the brain of the system. It reads sensor data, displays real-time readings on an OLED screen, and sends the data to a cloud database via Wi-Fi. Its dual-core processing and integrated wireless capability make it highly suitable for IoT-based systems.

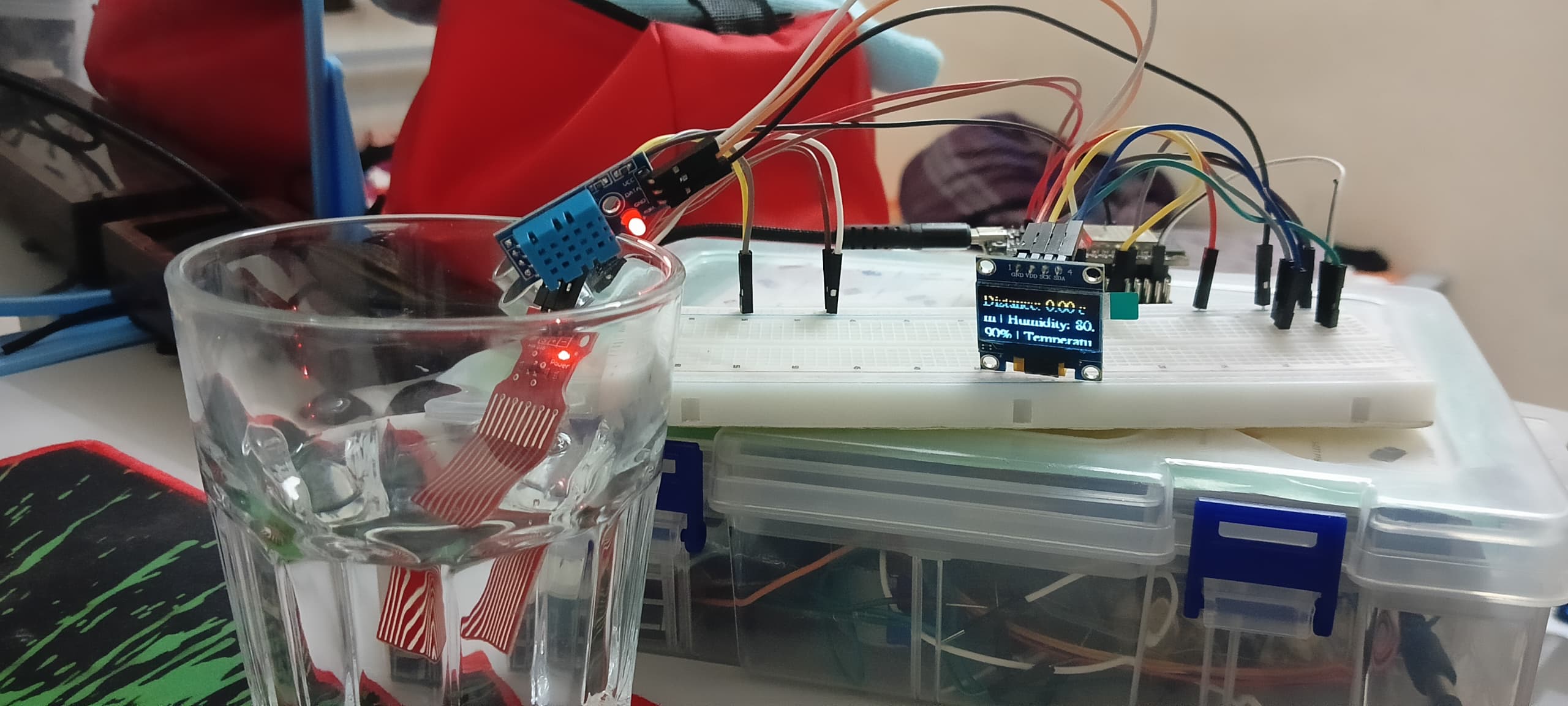


**SYSTEM SCHEMATIC:**

The following schematic outlines the connection of all hardware components to the ESP32:

|  |  |
| --- | --- |
| **Component** | **ESP32 Pin** |
| DHT11 | GPIO 16 |
| Ultrasonic Sensor | Trig: GPIO 12  Echo: GPIO 13 |
| Water Sensor | Analog GPIO 34 |
| OLED Display | SDA: GPIO 21  SCL: GPIO 22 |
| Relay Module | GPIO 17 |

Table 1 - Components and Pin Connections



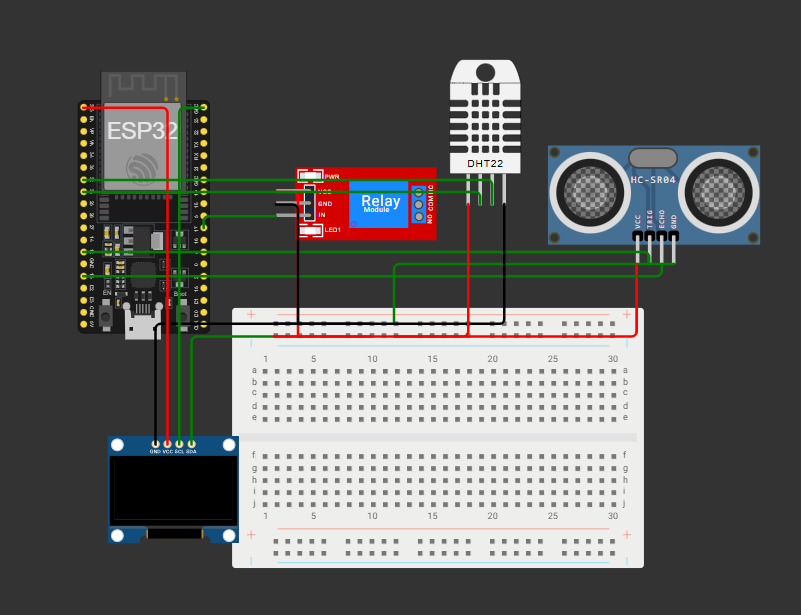
****

Figure 1 - Schematic of the system.

**CODE DEMONSTRATION:**

1. ***Arduino Code***

The main program is written using the Arduino IDE and performs the following tasks:

* Initializes the OLED screen and Wi-Fi connection.
* Continuously reads values from the ultrasonic sensor (distance), water sensor (wetness level), and DHT11 (temperature and humidity).
* Displays sensor readings on the OLED.
* Sends sensor data as a POST request to a secure HTTPS PHP endpoint using the WiFiClientSecure and HTTPClient libraries.
* Calculates the wetness percentage from the analog water sensor value.
* Structures data in a readable format for both visual and backend processing.

The code is well-commented, modular, and follows good programming practices such as input validation and error handling (e.g., checking for failed sensor reads).

#include <WiFi.h>

#include <WiFiClientSecure.h>

#include <HTTPClient.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#include <Fonts/FreeSerif9pt7b.h>

#include <DHT.h>

// OLED config

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1);

// DHT config

#define DHTPIN 16

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

// Ultrasonic config

const int trigPin = 12;

const int echoPin = 13;

const int relay = 17;

// Water sensor config

#define WATER\_SENSOR\_PIN 34

// WiFi and server

const char\* ssid = "Tersangat la laju";

const char\* password = "12345678";

const char\* serverName = "https://mexaze.com.my/post-esp-data.php";

String apiKeyValue = "d3d2b9b8-19df-4f23-a898-e30787d81cbb";

String sensorName = "ESP32";

String sensorLocation = "Home";

void setup() {

  pinMode(trigPin, OUTPUT);

  pinMode(echoPin, INPUT);

  pinMode(relay, OUTPUT);

  digitalWrite(relay, LOW); // Make sure it's off initially

  Serial.begin(115200);

  dht.begin();

  WiFi.begin(ssid, password);

  Serial.println("Connecting to WiFi...");

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("\nConnected to WiFi. IP: " + WiFi.localIP().toString());

  if (!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) {

    Serial.println("SSD1306 allocation failed");

    while (true);

  }

  display.clearDisplay();

  display.display();

  display.setFont(&FreeSerif9pt7b);

  display.setTextColor(WHITE);

  delay(2000);

}

void loop() {

  // Ultrasonic

  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);

  float duration\_us = pulseIn(echoPin, HIGH);

  float distance\_cm = 0.017 \* duration\_us;

  // DHT11

  float humidity = dht.readHumidity();

  float temperature = dht.readTemperature();

  // Water Sensor

  int waterValue = analogRead(WATER\_SENSOR\_PIN);

  bool waterDetected = waterValue < 2000;

  float waterPercent = 100.0 - (waterValue / 4095.0) \* 100.0;

  String combinedText = "Distance: " + String(distance\_cm, 2) + " cm | Humidity: " + String(humidity, 2) + "% | Temperature: " + String(temperature, 2) + "°C | Water: %.1f%%: " + String(waterPercent, 2);

  int16\_t x1, y1;

  uint16\_t textWidth, textHeight;

  display.getTextBounds(combinedText, 0, 0, &x1, &y1, &textWidth, &textHeight);

  // Send to server

  if (WiFi.status() == WL\_CONNECTED) {

    WiFiClientSecure \*client = new WiFiClientSecure;

    client->setInsecure();

    HTTPClient https;

    https.setTimeout(4500);

    https.begin(\*client, serverName);

    https.addHeader("Content-Type", "application/x-www-form-urlencoded");

    String httpRequestData = "api\_key=" + apiKeyValue +

                             "&sensor=" + sensorName +

                             "&location=" + sensorLocation +

                             "&humvalue=" + String(humidity, 2) +

                             "&temvalue=" + String(temperature, 2) +

                             "&disvalue=" + String(distance\_cm, 2) +

                             "&watervalue=" + String(waterPercent, 2);

    Serial.print("httpRequestData: ");

    Serial.println(httpRequestData);

    int httpResponseCode = https.POST(httpRequestData);

    if (httpResponseCode > 0) {

      Serial.print("HTTP Response code: ");

      Serial.println(httpResponseCode);

    }

    else {

      Serial.print("Error code: ");

      Serial.println(httpResponseCode);

    }

    https.end();

    delete client;

  }

  else {

    Serial.println("WiFi Disconnected");

  }

  // Display to OLED

  if (isnan(humidity) || isnan(temperature) || isnan(distance\_cm) || isnan(waterPercent)){

    display.clearDisplay();

    display.setFont(&FreeSerif9pt7b);

    display.setCursor(0, 20);

    display.println("Sensor error!");

    display.display();

    Serial.println("Sensor error!");

  }

  else {

    display.clearDisplay();

    display.setFont(&FreeSerif9pt7b);

    display.setCursor(0, 20);

    display.print(combinedText);

    display.display();

    // Serial debug

    Serial.println("---- Sensor Readings ----");

    Serial.printf("Temp: %.1f °C\n", temperature);

    Serial.printf("Humidity: %.1f %%\n", humidity);

    Serial.printf("Distance: %.1f cm\n", distance\_cm);

    Serial.printf("Water Sensor: %d (%.1f%% wet)\n", waterValue, waterPercent);

    Serial.println("-------------------------\n");

  }

  delay(10000); // 10s delay between readings

}

1. ***PHP Code***

A responsive web application was developed using HTML, CSS (Bootstrap), and JavaScript (Highcharts) to visualize real-time sensor data retrieved from the MySQL database. The following are the key features that available:

* Live Status Dashboard: Shows the latest readings for temperature, humidity, distance (water level), and water wetness with color-coded bars:

🟢 Green – Safe

🟡 Yellow – Caution

🔴 Red – Critical

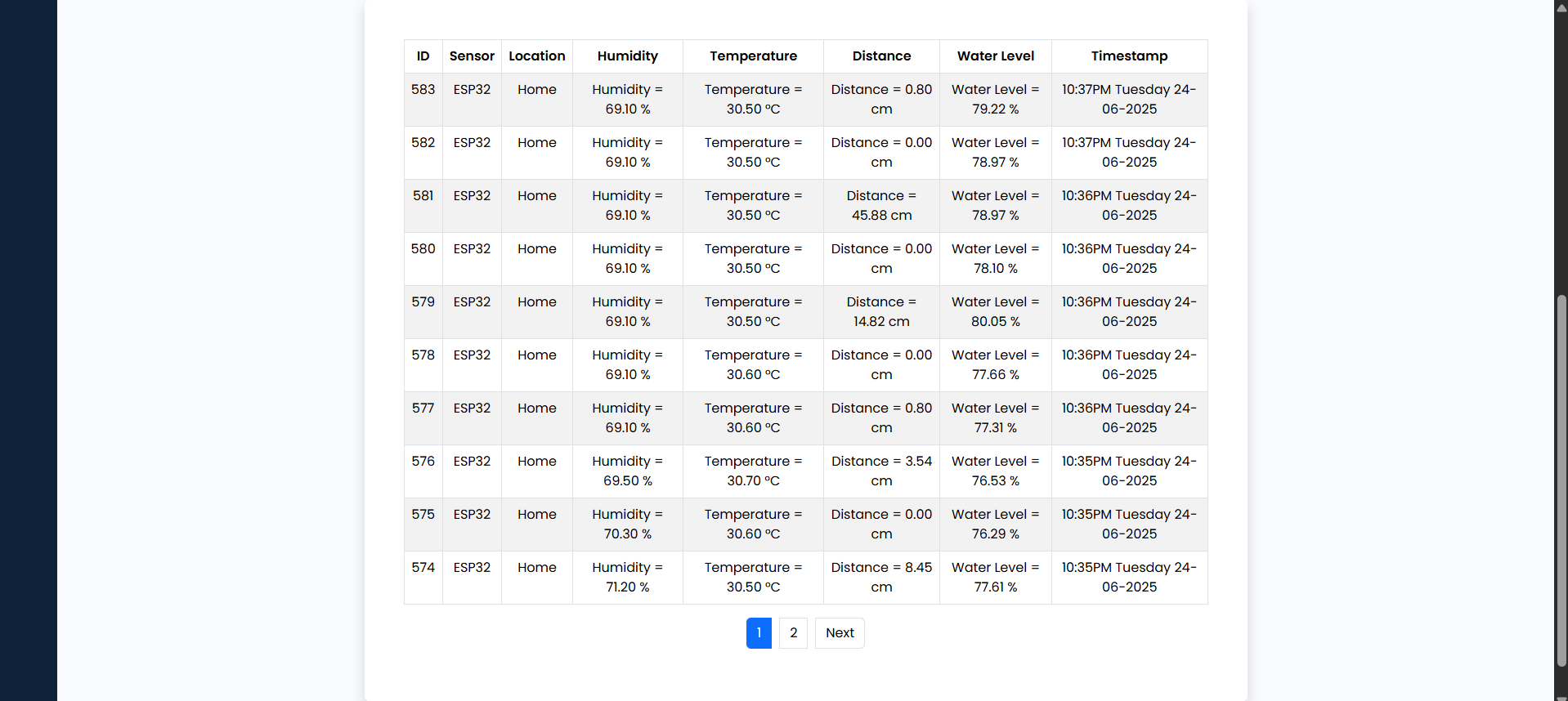
* Historical Data Table: Paginated, sortable table displaying sensor logs with timestamps.
* Navigation Sidebar: Allows users to switch between different sensor visualizations.
* Real-Time Graphs: Line charts display historical trends for each sensor metric, helping analyze flood patterns over time.

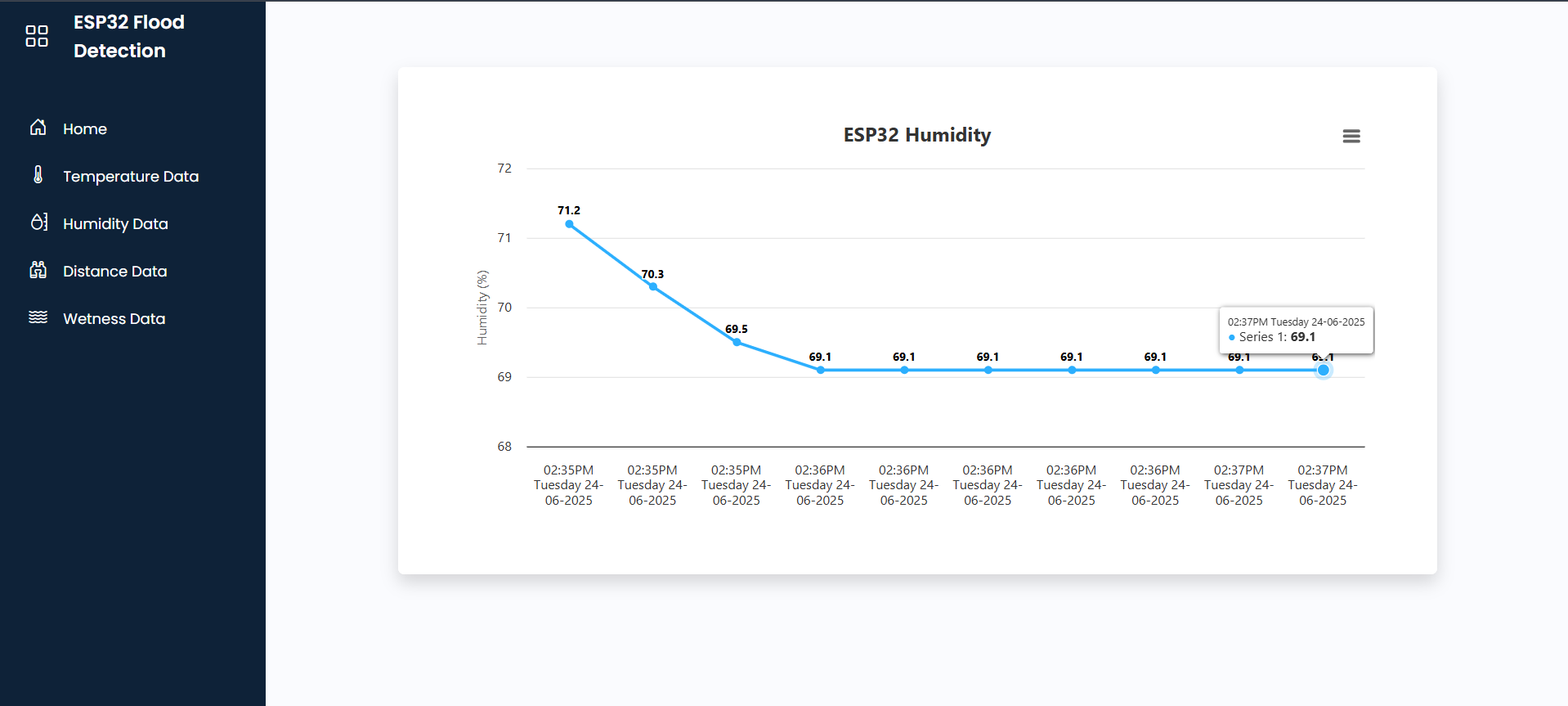
The web demonstration can be access from this website link:

<https://mexaze.com.my>

Interface:









1. index.php

<!DOCTYPE html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>ESP Flood Detection</title>

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.13.1/font/bootstrap-icons.min.css">

<script src="https://code.highcharts.com/highcharts.js"></script>

<script src="https://code.highcharts.com/modules/solid-gauge.js"></script>

<script src="https://code.highcharts.com/modules/exporting.js"></script>

<style>

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap');

\*,

::after,

::before {

box-sizing: border-box;

padding: 0;

margin: 0;

}

a {

text-decoration: none;

}

li {

list-style: none;

}

body {

font-family: 'Poppins', sans-serif;

height: 500px;

margin: 0 auto;

}

h2 {

font-family: Arial;

font-size: 2.5rem;

text-align: center;

}

.container {

display: flex;

}

.wrapper {

display: flex;

}

.main {

min-height: 100vh;

width: 100%;

overflow: hidden;

transition: all 0.35s ease-in-out;

background-color: #fafbfe;

}

#sidebar {

width: 70px;

min-width: 70px;

z-index: 1000;

transition: all .25s ease-in-out;

display: flex;

flex-direction: column;

background-color: #0e2238;

}

#sidebar.expand {

width: 260px;

min-width: 260px;

}

#toggler {

display: none;

}

.toggle-btn {

font-size: 1.5rem;

cursor: pointer;

color: #FFF;

padding: 1rem 1.5rem;

width: max-content;

}

#toggle-btn {

background-color: transparent;

cursor: pointer;

border: 0;

padding: 1rem 1.5rem;

}

#toggle-btn i {

font-size: 1.5rem;

color: #fff;

}

.sidebar-logo {

margin: auto 0;

}

.sidebar-logo a {

color: #fff;

font-size: 1.15rem;

font-weight: 600;

}

#sidebar:not(.expand) .sidebar-logo,

#sidebar:not(.expand) a.sidebar-link span {

display: none;

}

.sidebar-nav {

padding: 2rem 0;

flex: 1 1 auto;

}

a.sidebar-link {

padding: .625rem 1.625rem;

color: #FFF;

display: block;

font-size: 0.9rem;

white-space: nowrap;

border-left: 3px solid transparent;

}

.sidebar-link i {

font-size: 1.1rem;

margin-right: .75rem;

}

a.sidebar-link:hover {

background-color: rgba(255,255,255,.075);

border-left: 3px solid #3b7ddd;

}

.sidebar-item {

position: relative;

}

#sidebar:not(.expand) .sidebar-item .sidebar-dropdown{

position: absolute;

top: 0;

left: 70px;

background-color: 0e2238;

padding: 0;

min-width: 15rem;

display: none;

}

#sidebar:not(.expand) .sidebar-item:hover .has-dropdown+.sidebar-dropdown{

display: block;

max-height: 15rem;

width: 100%;

opacity: 1;

}

#sidebar.expand .sidebar-link[data-bs-toggle="collapse"]::after{

border: solid;

border-width: 0 .075rem .075rem 0;

content: "";

display: inline-block;

padding: 2px;

position: absolute;

right: 1.5rem;

top: 1.4rem;

transform: rotate(-135deg);

transition: all .2s ease-out;

}

#sidebar.expand .sidebar-link[data-bs-toggle="collapse"].collapsed::after{

transform: rotation(45deg);

transition: all .2s ease-out;

}

#sidebar input[type="checkbox"]:checked~.sidebar-nav {

width: 260px;

min-width: 260px;

}

#sidebar input[type="checkbox"]:not(:checked)~\* .sidebar-link span {

display: none;

}

th {

cursor: pointer;

}

.pagination {

display: flex;

justify-content: center;

}

.pagination li {

list-style-type: none;

margin: 0 5px;

}

.pagination a {

color: #000;

text-decoration: none;

}

.modal-xl {

max-width: 95% !important;

}

#current-status-chart {

margin-top: 2rem;

}

</style>

</head>

<body>

<?php

$servername = "zen";

$dbname = "mexazeco\_esp\_data";

$username = "mexazeco\_Mexaze";

$password = "@Z33m2011";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Get the total number of records

$total\_sql = "SELECT COUNT(\*) FROM SensorData";

$total\_result = $conn->query($total\_sql);

$total\_row = $total\_result->fetch\_row();

$total\_records = $total\_row[0];

// Define how many results you want per page

$results\_per\_page = 10;

// Determine the total number of pages available

$total\_pages = ceil($total\_records / $results\_per\_page);

// Find out the current page number

if (isset($\_GET['page']) && is\_numeric($\_GET['page'])) {

$page = $\_GET['page'];

} else {

$page = 1;

}

// Calculate the starting limit number

$start\_limit = ($page - 1) \* $results\_per\_page;

$sql = "SELECT id, sensor, location, humvalue, temvalue, disvalue, watervalue, reading\_time FROM SensorData ORDER BY id DESC LIMIT " . $start\_limit . ", " . $results\_per\_page;

$result = $conn->query($sql);

$sensor\_data = []; // Initialize the array

while ($data = $result->fetch\_assoc()){

// Adjust reading\_time and format it

$data['reading\_time'] = date("h:iA l d-m-Y", strtotime($data['reading\_time'] ));

$sensor\_data[] = $data;

}

$readings\_time = array\_column($sensor\_data, 'reading\_time');

$humvalue = json\_encode(array\_reverse(array\_column($sensor\_data, 'humvalue')), JSON\_NUMERIC\_CHECK);

$temvalue = json\_encode(array\_reverse(array\_column($sensor\_data, 'temvalue')), JSON\_NUMERIC\_CHECK);

$disvalue = json\_encode(array\_reverse(array\_column($sensor\_data, 'disvalue')), JSON\_NUMERIC\_CHECK);

$reading\_time = json\_encode(array\_reverse($readings\_time)); // No need for JSON\_NUMERIC\_CHECK here

$watervalue = json\_encode(array\_reverse(array\_column($sensor\_data, 'watervalue')), JSON\_NUMERIC\_CHECK);

?>

<div class="wrapper">

<aside id=sidebar>

<div class="d-flex">

<button id="toggle-btn">

<i class="bi bi-grid"></i>

</button>

<div class="sidebar-logo">

<a href="#">ESP32 Flood Detection</a>

</div>

</div>

<ul class="sidebar-nav">

<li class="sidebar-item">

<a href="#" class="sidebar-link" data-target="home-container">

<i class="bi bi-house-door"></i>

<span>Home</span>

</a>

</li>

<li class="sidebar-item">

<a href="#" class="sidebar-link" data-target="temperature-container">

<i class="bi bi-thermometer-half"></i>

<span>Temperature Data</span>

</a>

</li>

<li class="sidebar-item">

<a href="#" class="sidebar-link" data-target="humidity-container">

<i class="bi bi-moisture"></i>

<span>Humidity Data</span>

</a>

</li>

<li class="sidebar-item">

<a href="#" class="sidebar-link" data-target="distance-container">

<i class="bi bi-binoculars"></i>

<span>Distance Data</span>

</a>

</li>

<li class="sidebar-item">

<a href="#" class="sidebar-link" data-target="wetness-container">

<i class="bi bi-water"></i>

<span>Wetness Data</span>

</a>

</li>

</ul>

</aside>

<div class="main p-3">

<!-- Home Page -->

<div id="home-container" class="content-container">

<div class="text-center">

<div class="col-lg-10 mx-auto">

<div class="card rounded shadow border-0">

<div class="card-body p-5 bg-white rounded">

<div class="table-responsive">

<div class="card mt-4">

<div class="card-body">

<div id="current-status-chart" style="height: 400px;"></div>

</div>

</div>

</div>

</div>

</div>

</div>

<div class="container py-5">

<div class="col-lg-10 mx-auto">

<div class="card rounded shadow border-0">

<div class="card-body p-5 bg-white rounded">

<div class="table-responsive">

<table id="example" style="width:100%" class="table table-striped table-bordered table-hover">

<thead>

<tr>

<th onclick="sortTable(0)">ID</th>

<th onclick="sortTable(1)">Sensor</th>

<th onclick="sortTable(2)">Location</th>

<th onclick="sortTable(3)">Humidity</th>

<th onclick="sortTable(4)">Temperature</th>

<th onclick="sortTable(5)">Distance</th>

<th onclick="sortTable(6)">Water Level</th>

<th onclick="sortTable(7)">Timestamp</th>

</tr>

</thead>

<tbody>

<?php

if ($result = $conn->query($sql)) {

while ($row = $result->fetch\_assoc()) {

$row\_id = htmlspecialchars($row["id"]);

$row\_sensor = htmlspecialchars($row["sensor"]);

$row\_location = htmlspecialchars($row["location"]);

$row\_humvalue = htmlspecialchars(number\_format($row["humvalue"], 2));

$row\_temvalue = htmlspecialchars(number\_format($row["temvalue"], 2));

$row\_disvalue = htmlspecialchars(number\_format($row["disvalue"], 2));

$row\_watervalue = htmlspecialchars(number\_format($row["watervalue"], 2));

$row\_reading\_time = date("h:iA l d-m-Y", strtotime($row["reading\_time"] . " +8 hours"));

echo "<tr>

<td>{$row\_id}</td>

<td>{$row\_sensor}</td>

<td>{$row\_location}</td>

<td>Humidity = {$row\_humvalue} %</td>

<td>Temperature = {$row\_temvalue} °C</td>

<td>Distance = {$row\_disvalue} cm</td>

<td>Water Level = {$row\_watervalue} %</td>

<td>{$row\_reading\_time}</td>

</tr>";

}

$result->free();

}

$conn->close();

?>

</tbody>

</table>

</div>

<nav aria-label="Page navigation example">

<ul class="pagination justify-content-center">

<?php

// Previous Button

if ($page > 1) {

echo '<li class="page-item"><a class="page-link" href="index.php?page=' . ($page - 1) . '">Previous</a></li>';

}

// Display the links to the pages

for ($p = 1; $p <= $total\_pages; $p++) {

if ($p == $page) {

echo '<li class="page-item active"><a class="page-link" href="#">' . $p . '</a></li>';

} else {

echo '<li class="page-item"><a class="page-link" href="index.php?page=' . $p . '">' . $p . '</a></li>';

}

}

// Next Button

if ($page < $total\_pages) {

echo '<li class="page-item"><a class="page-link" href="index.php?page=' . ($page + 1) . '">Next</a></li>';

}

?>

</ul>

</nav>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- Home Page -->

<!-- Temperature Page -->

<div id="temperature-container" class="content-container d-none">

<div class="container py-5">

<div class="col-lg-10 mx-auto">

<div class="card rounded shadow border-0">

<div class="card-body p-5 bg-white rounded">

<div id="chart-temperature" class="container"></div>

</div>

</div>

</div>

</div>

</div>

<!-- Temperature Page -->

<!-- Humidity Page -->

<div id="humidity-container" class="content-container d-none">

<div class="container py-5">

<div class="col-lg-10 mx-auto">

<div class="card rounded shadow border-0">

<div class="card-body p-5 bg-white rounded">

<div id="chart-humidity" class="container"></div>

</div>

</div>

</div>

</div>

</div>

<!-- Humidity Page -->

<!-- Distance Page -->

<div id="distance-container" class="content-container d-none">

<div class="container py-5">

<div class="col-lg-10 mx-auto">

<div class="card rounded shadow border-0">

<div class="card-body p-5 bg-white rounded">

<div id="chart-pressure" class="container"></div>

</div>

</div>

</div>

</div>

</div>

<!-- Distance Page -->

<!-- Wetness Page -->

<div id="wetness-container" class="content-container d-none">

<div class="container py-5">

<div class="col-lg-10 mx-auto">

<div class="card rounded shadow border-0">

<div class="card-body p-5 bg-white rounded">

<div id="chart-wetness" class="container"></div>

</div>

</div>

</div>

</div>

</div>

<!-- Wetness Page -->

</div>

</div>

<!-- JavaScript for Table Sorting -->

<script>

function sortTable(n) {

var table, rows, switching, i, x, y, shouldSwitch, dir, switchcount = 0;

table = document.getElementById("example");

switching = true;

dir = "asc"; // Set the sorting direction to ascending

while (switching) {

switching = false;

rows = table.rows;

for (i = 1; i < (rows.length - 1); i++) {

shouldSwitch = false;

x = rows[i].getElementsByTagName("TD")[n];

y = rows[i + 1].getElementsByTagName("TD")[n];

// Compare based on direction

if (dir == "asc") {

if (x.innerText.toLowerCase() > y.innerText.toLowerCase()) {

shouldSwitch = true;

break;

}

} else if (dir == "desc") {

if (x.innerText.toLowerCase() < y.innerText.toLowerCase()) {

shouldSwitch = true;

break;

}

}

}

if (shouldSwitch) {

rows[i].parentNode.insertBefore(rows[i + 1], rows[i]);

switching = true;

switchcount++;

} else {

// If no switching has been done AND direction is "asc", switch to "desc"

if (switchcount == 0 && dir == "asc") {

dir = "desc";

switching = true;

}

}

}

}

</script>

<!-- Chart Container -->

<script>

var humvalue = <?php echo $humvalue; ?>;

var temvalue = <?php echo $temvalue; ?>;

var disvalue = <?php echo $disvalue; ?>;

var watervalue = <?php echo $watervalue; ?>;

var reading\_time = <?php echo $reading\_time; ?>;

var chartT = new Highcharts.Chart({

chart:{ renderTo : 'chart-temperature' },

title: { text: 'ESP32 Temperature' },

series: [{

showInLegend: false,

data: temvalue // Corrected variable name

}],

plotOptions: {

line: { animation: false,

dataLabels: { enabled: true }

},

series: { color: '#059e8a' }

},

xAxis: {

type: 'datetime',

categories: reading\_time

},

yAxis: {

title: { text: 'Temperature (Celsius)' }

//title: { text: 'Temperature (Fahrenheit)' }

},

credits: { enabled: false }

});

var chartH = new Highcharts.Chart({

chart:{ renderTo:'chart-humidity' },

title: { text: 'ESP32 Humidity' },

series: [{

showInLegend: false,

data: humvalue // Corrected variable name

}],

plotOptions: {

line: { animation: false,

dataLabels: { enabled: true }

}

},

xAxis: {

type: 'datetime',

//dateTimeLabelFormats: { second: '%H:%M:%S' },

categories: reading\_time

},

yAxis: {

title: { text: 'Humidity (%)' }

},

credits: { enabled: false }

});

var chartW = new Highcharts.Chart({

chart:{ renderTo:'chart-wetness' },

title: { text: 'ESP32 Wetness' },

series: [{

showInLegend: false,

data: watervalue // Corrected variable name

}],

plotOptions: {

line: { animation: false,

dataLabels: { enabled: true }

}

},

xAxis: {

type: 'datetime',

//dateTimeLabelFormats: { second: '%H:%M:%S' },

categories: reading\_time

},

yAxis: {

title: { text: 'Wetness (%)' }

},

credits: { enabled: false }

});

var chartP = new Highcharts.Chart({

chart:{ renderTo:'chart-pressure' },

title: { text: 'ESP32 Distance' },

series: [{

showInLegend: false,

data: disvalue // Corrected variable name

}],

plotOptions: {

line: { animation: false,

dataLabels: { enabled: true }

},

series: { color: '#18009c' }

},

xAxis: {

type: 'datetime',

categories: reading\_time

},

yAxis: {

title: { text: 'Distance (cM)' }

},

credits: { enabled: false }

});

</script>

<!-- Bootstrap JS Bundle (includes Popper) -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YpC+nAAM+1JlVkP3y1VZ4zqT3BfvzQ14uGHljPPV6vOMV3B9OQxWv0FdADtI/5hI" crossorigin="anonymous"></script>

<script>

const hamburger = document.querySelector("#toggle-btn");

hamburger.addEventListener("click", function () {

document.querySelector("#sidebar").classList.toggle("expand");

});

</script>

<!-- Current Status charge -->

<script>

const latestTemp = temvalue[temvalue.length - 1];

const latestHum = humvalue[humvalue.length - 1];

const latestDist = disvalue[disvalue.length - 1];

const latestWater = watervalue[watervalue.length - 1];

const chartStatus = Highcharts.chart('current-status-chart', {

chart: {

type: 'column'

},

title: {

text: 'Live Flood Detection Status'

},

xAxis: {

categories: ['Temperature (Celcius)', 'Humidity (%)', 'Distance (cm)', 'Water Level (%)'],

crosshair: true

},

yAxis: {

min: 0,

title: {

text: 'Sensor Value'

}

},

tooltip: {

shared: true

},

plotOptions: {

column: {

dataLabels: {

enabled: true

},

colorByPoint: true

}

},

series: [{

name: 'Latest Reading',

data: [

{

y: latestTemp,

color: getColor('temperature', latestTemp)

},

{

y: latestHum,

color: getColor('humidity', latestHum)

},

{

y: latestDist,

color: getColor('distance', latestDist)

},

{

y: latestWater,

color: getColor('water', latestWater)

}

]

}]

});

function getColor(sensor, value) {

switch (sensor) {

case 'temperature':

if (value < 30) return 'green';

if (value < 35) return 'yellow';

return 'red';

case 'humidity':

if (value < 60) return 'green';

if (value < 80) return 'yellow';

return 'red';

case 'distance': // assume < 10 cm is critical

if (value > 20) return 'green';

if (value > 10) return 'yellow';

return 'red';

case 'water':

if (value < 30) return 'green';

if (value < 60) return 'yellow';

return 'red';

}

}

</script>

<!-- Container change -->

<script>

document.querySelectorAll('.sidebar-link').forEach(link => {

link.addEventListener('click', function (e) {

e.preventDefault();

const targetId = this.getAttribute('data-target');

const containers = document.querySelectorAll('.content-container');

containers.forEach(container => {

if (container.id === targetId) {

container.classList.remove('d-none');

} else {

container.classList.add('d-none');

}

});

});

});

</script>

</body>

</html>

1. post-esp-data.php

<?php

$servername = "zen";

// REPLACE with your Database name

$dbname = "mexazeco\_esp\_data";

// REPLACE with Database user

$username = "mexazeco\_Mexaze";

// REPLACE with Database user password

$password = "@Z33m2011";

$api\_key\_value = "d3d2b9b8-19df-4f23-a898-e30787d81cbb";

$api\_key= $sensor = $location = $humvalue = $temvalue = $disvalue = $watervalue = "";

if ($\_SERVER["REQUEST\_METHOD"] == "POST")

{

$api\_key = test\_input($\_POST["api\_key"]);

if($api\_key == $api\_key\_value)

{

$sensor = test\_input($\_POST["sensor"]);

$location = test\_input($\_POST["location"]);

$humvalue = test\_input($\_POST["humvalue"]);

$temvalue = test\_input($\_POST["temvalue"]);

$watervalue = test\_input($\_POST["watervalue"]);

$disvalue = test\_input($\_POST["disvalue"]);

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error)

{

die("Connection failed: " . $conn->connect\_error);

}

$sql = "INSERT INTO SensorData (sensor, location, humvalue, temvalue, disvalue, watervalue)

VALUES ('" . $sensor . "', '" . $location . "', '" . $humvalue . "', '" . $temvalue . "', '" . $disvalue . "', '" . $watervalue . "')";

if ($conn->query($sql) === TRUE)

{

echo "New record created successfully";

}

else

{

echo "Error: " . $sql . "<br>" . $conn->error;

}

$conn->close();

}

else

{

echo "Wrong API Key provided.";

}

}

else

{

echo "No data posted with HTTP POST.";

}

function test\_input($data)

{

$data = trim($data);

$data = stripslashes($data);

$data = htmlspecialchars($data);

return $data;

}

?>

**CONCLUSIONS:**

The proposed Smart Flood Detection System achieves the objective of delivering a robust, scalable, and affordable early warning system for flood monitoring. Leveraging the ESP32's networking capability and reliable sensors, this project successfully demonstrates how IoT can be applied to real-world disaster management. The combination of real-time data logging, remote monitoring, and visual alerts ensures enhanced preparedness against flooding events.