**PHASE -5**

**Documentation for Environmental Monitoring**

|  |  |
| --- | --- |
| **Date** | 31/10/2023 |
| **Team ID** | Proj\_223986\_Team |
| **Project Name** | Environmental Monitoring in Parks |
| **Students with ID** | **Vimal K**  Nirmalkumar K  Parthiban G  Shreepal S  Sabari Nathan C |

**TITLE: Environmental Monitoring**

Certainly! Let's break down the steps to implement the IoT-based environmental monitoring system for public parks using ESP32 and other sensors, following the design thinking approach outlined in your document.

**1. Introduction**

The objective of this project is to implement an IoT-based environmental monitoring system for public parks using temperature and humidity sensors. The system leverages ThingSpeak as the cloud platform to collect, analyze, and visualize real-time environmental data. The project aims to enhance environmental awareness, support responsible park management, and provide valuable insights to park visitors.

**2. Project Components**

* Hardware Components
* IoT devices (ESP32in this example)
* Temperature and humidity sensors (DHT22)
* Jumper wires

**2.2 Software Components**

Python script for data collection and transmission

ThingSpeak platform for data storage and visualization

**3. Methodology**

3.1 Hardware Setup

Connect temperature and humidity sensors to the ESP32.

Ensure a stable power supply for continuous monitoring.

3.2 ThingSpeak Configuration

Create a ThingSpeak account.

Set up a new channel for environmental data with two fields for temperature and humidity.

Obtain the Write API Key for data transmission.

3.3 Python Script Development

Install necessary libraries using pip3 install Adafruit\_DHT requests.

Develop a Python script (environmental\_monitoring.py) to read sensor data and send it to ThingSpeak.

Integrate error handling and continuous monitoring.

**4.Monitor the terminal for temperature and humidity readings. Code used**

// channel id : 2307358

//channel api key : 1U2N21SZEGP74GFZ

#include <WiFi.h>

#include "DHTesp.h"

#include "ThingSpeak.h"

const int DHT\_PIN = 15;

const int LED\_PIN = 13;

const char\* WIFI\_NAME = "Wokwi-GUEST";

const char\* WIFI\_PASSWORD = "";

const int myChannelNumber =2307358 ;

const char\* myApiKey = "1U2N21SZEGP74GFZ";

const char\* server = "api.thingspeak.com";

DHTesp dhtSensor;

WiFiClient client;

void setup() {

**Serial**.begin(115200);

  dhtSensor.setup(DHT\_PIN, DHTesp::DHT22);

  pinMode(LED\_PIN, OUTPUT);

  WiFi.begin(WIFI\_NAME, WIFI\_PASSWORD);

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

    delay(1000);

**Serial**.println("Wifi not connected");

  }

**Serial**.println("Wifi connected !");

**Serial**.println("Local IP: " + String(WiFi.localIP()));

  WiFi.mode(WIFI\_STA);

  ThingSpeak.begin(client);

}

void loop() {

  TempAndHumidity  data = dhtSensor.getTempAndHumidity();

  ThingSpeak.setField(1,data.temperature);

  ThingSpeak.setField(2,data.humidity);

  if (data.temperature > 35 || data.temperature < 12 || data.humidity > 70 || data.humidity < 40) {

    digitalWrite(LED\_PIN, HIGH);

  }else{

    digitalWrite(LED\_PIN, LOW);

  }

  int x = ThingSpeak.writeFields(myChannelNumber,myApiKey);

**Serial**.println("Temp: " + String(data.temperature, 2) + "°C");

**Serial**.println("Humidity: " + String(data.humidity, 1) + "%");

  if(x == 200){

**Serial**.println("Data pushed successfull");

  }else{

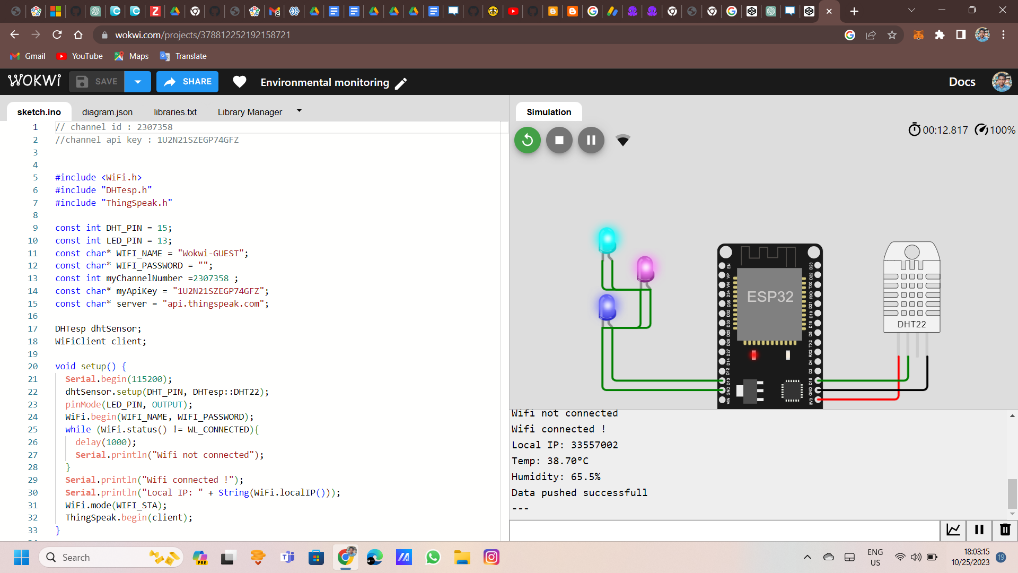
**Serial**.println("Push error" + String(x));

  }

**Serial**.println("---");

  delay(10000);

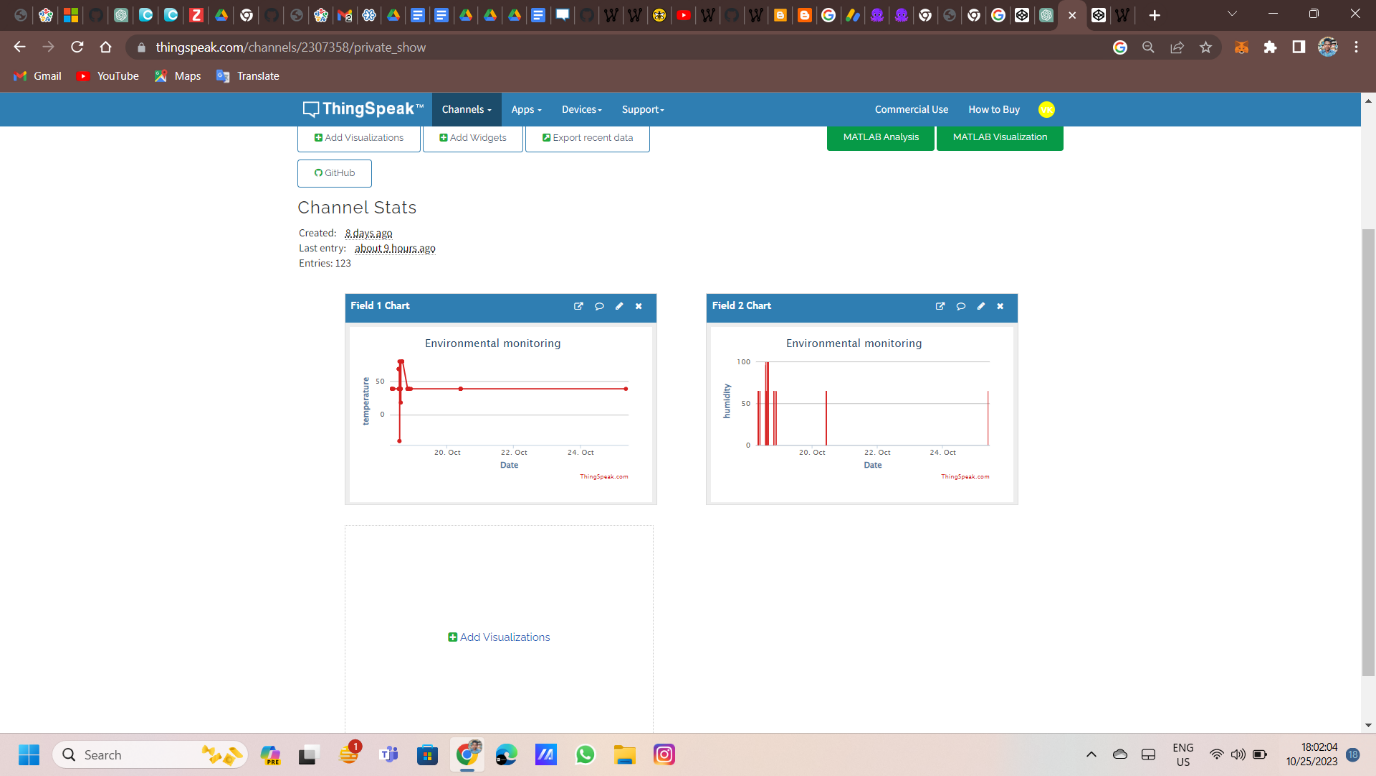
}



Verify data transmission to ThingSpeak by checking the channel.

**5. Results and Observations**

Real-time temperature and humidity data are successfully transmitted to ThingSpeak.

The ThingSpeak platform provides visualizations and historical data for analysis.

**6. Challenges and Solutions**

6.1 Challenges

Connectivity issues: Ensuring a stable Wi-Fi connection for continuous data transmission.

6.2 Solutions

Implementing error handling in the Python script.

Optimizing Wi-Fi setup for improved stability.

**HTML: For structuring the webpage.**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Environmental Monitoring Dashboard</title>

<link rel="stylesheet" href="styles.css">

<script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

<div class="container">

<h1>Environmental Monitoring Dashboard</h1>

<div id="data-container">

<canvas id="environment-chart"></canvas>

</div>

<div id="analysis-container">

<h2>Data Analysis Tool</h2>

<label for="startDate">Start Date:</label>

<input type="date" id="startDate">

<label for="endDate">End Date:</label>

<input type="date" id="endDate">

<button onclick="compareData()">Compare</button>

<div id="comparison-chart-container">

<canvas id="comparison-chart"></canvas>

</div>

</div>

<div id="history-container">

<h2>Historical Data</h2>

<ul id="history-list"></ul>

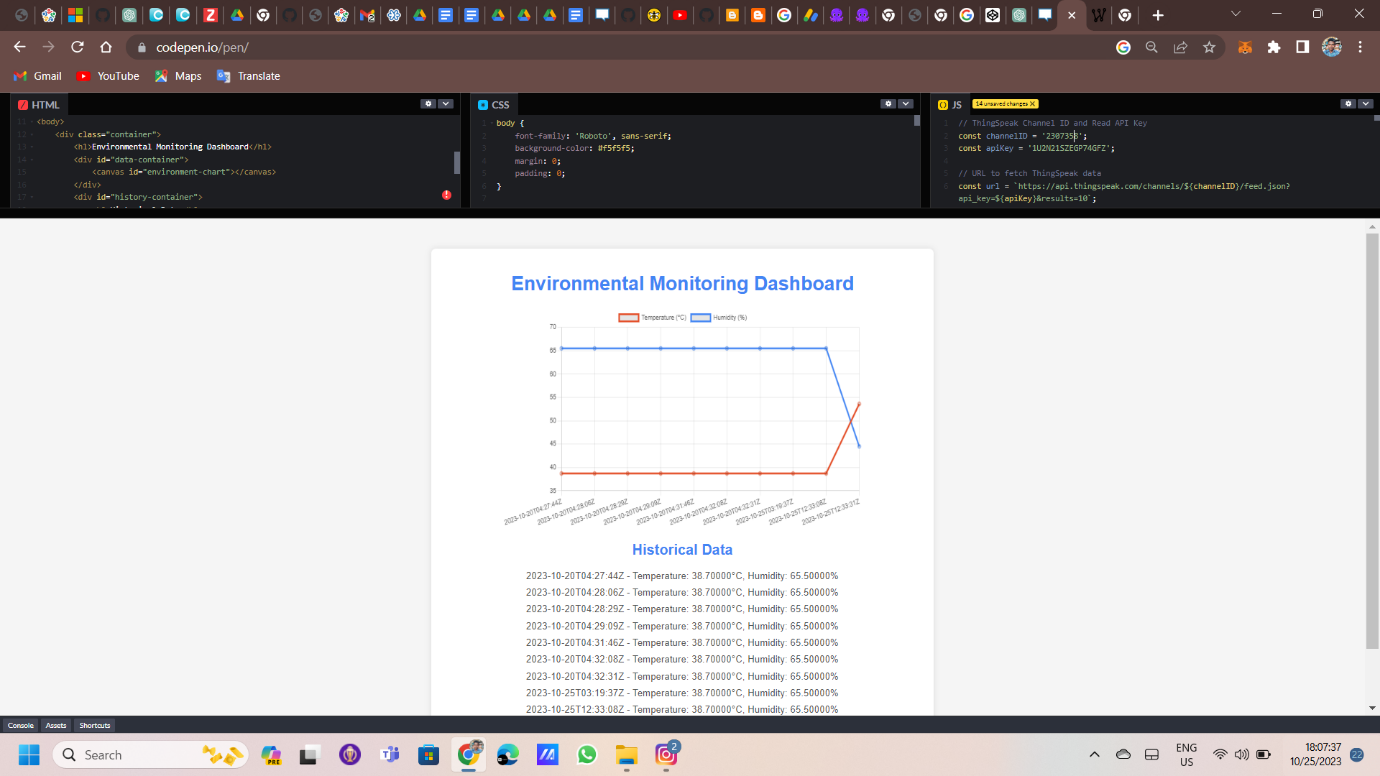
</div>

</div>

<script src="script.js"></script>

</body>

</html>



**CSS: For styling and layout.**

body {

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

background-color: #f5f5f5;

margin: 0;

padding: 0;

}

.container {

max-width: 800px;

margin: 50px auto;

background-color: #fff;

padding: 20px;

border-radius: 8px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

text-align: center;

}

h1, h2 {

color: #4285f4;

}

#data-container,

#analysis-container,

#history-container {

margin-top: 20px;

}

canvas {

width: 100%;

max-width: 600px;

margin: 20px auto;

}

#comparison-chart-container {

margin-top: 20px;

}

#analysis-container label,

#analysis-container button {

margin: 10px;

}

#comparison-chart-container canvas {

width: 100%;

max-width: 600px;

margin-top: 10px;

}

#history-container ul {

list-style: none;

padding: 0;

margin: 0;

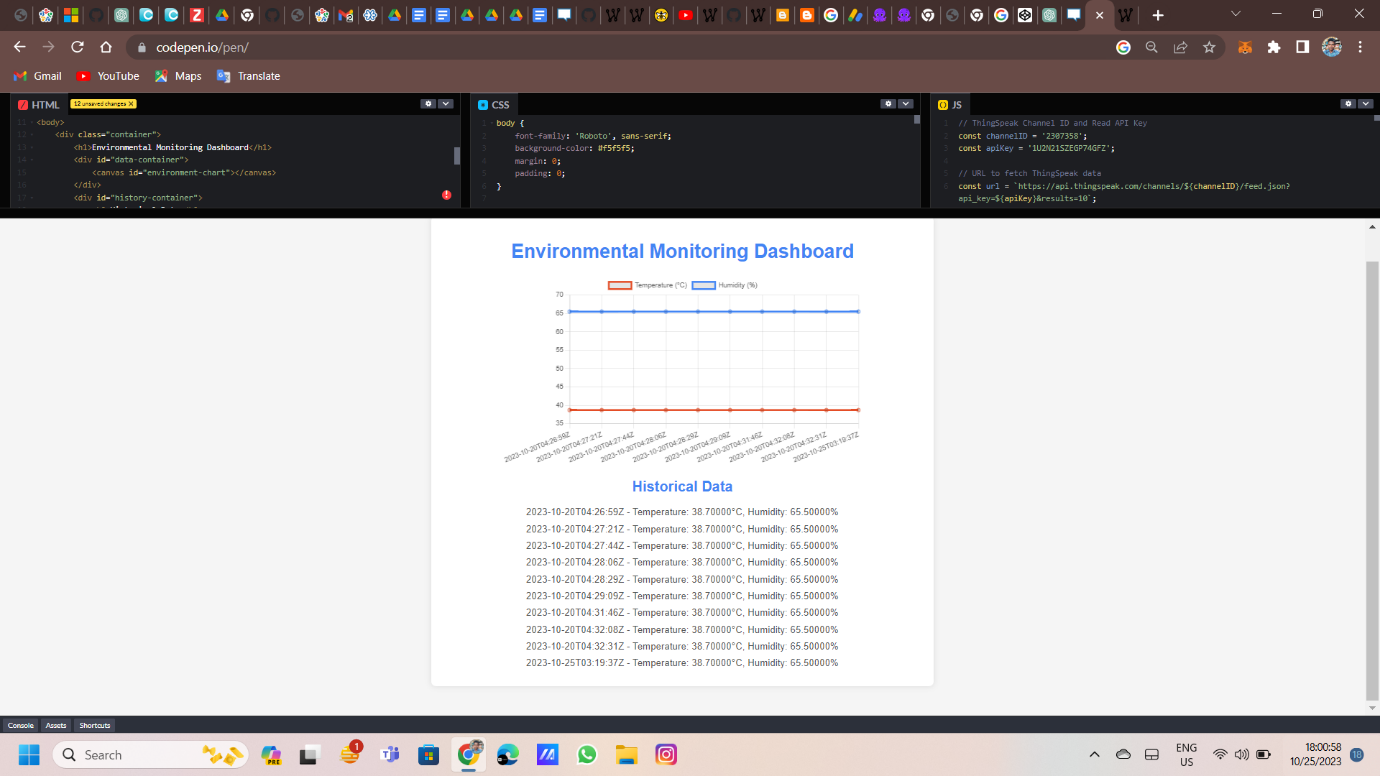
}

li {

margin-bottom: 10px;

color: #555;

}



**JavaScript: For dynamic content, real-time updates, and data analysis.**

**Chart.js: A JavaScript library for creating interactive charts.**

// ThingSpeak 2307358and Read API Key

const channelID = '2307358';

const apiKey = '1U2N21SZEGP74GFZ';

// URL to fetch ThingSpeak data

const url = `https://api.thingspeak.com/channels/${channelID}/feed.json?api\_key=${apiKey}&results=10`;

// Function to fetch data from ThingSpeak and update UI

function fetchDataAndUpdateUI() {

$.ajax({

url: url,

method: 'GET',

dataType: 'json',

success: function (data) {

updateChart(data.feeds);

updateHistory(data.feeds);

},

error: function (error) {

console.error(`Error fetching data: ${JSON.stringify(error)}`);

}

});

}

// Function to update the chart using Chart.js

function updateChart(data) {

const labels = data.map(entry => entry.created\_at);

const temperatureData = data.map(entry => entry.field1);

const humidityData = data.map(entry => entry.field2);

const ctx = document.getElementById('environment-chart').getContext('2d');

new Chart(ctx, {

type: 'line',

data: {

labels: labels,

datasets: [

{

label: 'Temperature (°C)',

borderColor: '#e44d26',

data: temperatureData,

},

{

label: 'Humidity (%)',

borderColor: '#4285f4',

data: humidityData,

},

],

},

});

}

// Function to update the historical data list

function updateHistory(data) {

const historyList = $('#history-list');

historyList.empty();

data.forEach(entry => {

const listItem = `<li>${entry.created\_at} - Temperature: ${entry.field1}°C, Humidity: ${entry.field2}%</li>`;

historyList.append(listItem);

});

}

// Function to compare data over time

function compareData() {

const startDate = $('#startDate').val();

const endDate = $('#endDate').val();

if (!startDate || !endDate) {

alert('Please select both start and end dates.');

return;

}

$.ajax({

url: `${url}&start=${startDate}&end=${endDate}`,

method: 'GET',

dataType: 'json',

success: function (data) {

updateComparisonChart(data.feeds);

},

error: function (error) {

console.error(`Error fetching data for comparison: ${JSON.stringify(error)}`);

}

});

}

// Function to update the comparison chart using Chart.js

function updateComparisonChart(data) {

const labels = data.map(entry => entry.created\_at);

const temperatureData = data.map(entry => entry.field1);

const humidityData = data.map(entry => entry.field2);

const ctx = document.getElementById('comparison-chart').getContext('2d');

new Chart(ctx, {

type: 'line',

data: {

labels: labels,

datasets: [

{

label: 'Temperature (°C)',

borderColor: '#e44d26',

data: temperatureData,

},

{

label: 'Humidity (%)',

borderColor: '#4285f4',

data: humidityData,

},

],

},

});

}

// Update data every 10 seconds

setInterval(fetchDataAndUpdateUI, 10000);

// Initial data fetch on page load

fetchDataAndUpdateUI();

**7. Future Enhancements**

Adding additional environmental sensors for comprehensive monitoring.

Implementing alerts based on predefined thresholds.

Integration with mobile applications for user accessibility.

**8. Conclusion**

The project successfully demonstrates the implementation of an IoT-enabled environmental monitoring system using ThingSpeak. The integration of temperature and humidity sensors provides valuable insights into park conditions. Continuous monitoring and data visualization contribute to environmental awareness and responsible park management.