ENVIRONMENTAL MONITORING IN PARKS

IOT-PHASE-4

TEAM MEMBERS:
UDHAYAKUMAR N (810021106090)
NISHANTHAN T(810021106053)
SURYAKUMAR R (810021106308)
SANTHOSH T (810021106067)
TARUN MR (810021106086)
VIMAL AS (810021106097)

INTRODUCTION

Environmental monitoring in parks using IoT (Internet of Things) is a modern approach to assess and manage the natural surroundings within these recreational areas. By deploying a network of sensors and connected devices, it enables real-time data collection and analysis. This technology helps track variables like air quality, temperature, humidity, water quality, and wildlife activity. The summarized introduction underscores the significance of IoT in maintaining and safeguarding the ecological balance of our parks, offering better insights and informed decision-making for park management and preservation.

USING WEB DEVELOPMENT CREATE A PLATFORM THAT DISPLAYS ENVIRONMENTAL MONITORING IN PARKS DATA:

HTML CODE:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Environmental Monitoring in Parks</h1>
      <div class="data">
        <h2>Temperature</h2>
        <p id="temperature">28°C(82.4°F)</p>
      </div>
      <div class="data">
        <h2>Humidity</h2>
        <p id="humidity">30%</p>
      </div>
  </div>
  <script src="script.js"></script>
```

```
</body>
</html>
```

CSS CODE:

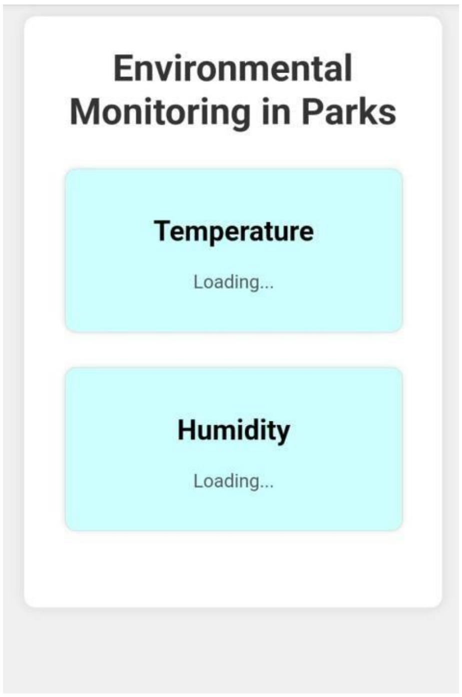
```
 body {     font-family: Arial,
sans-serif;     background-
color: #f0f0f0;
}

.container {     text-align: center;
background-color: #fff;     border-radius:
10px;     padding: 5px;     margin: 10px;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
height:500px;     width:350px;
}

h1 {     color:
#333;
}



.data {
padding:20px;
margin: 30px;
border: 1px solid
#ddd;     border-
radius: 10px;
background-color:
rgb(204,255,255);
box-shadow: 0 0 5px
rgba(0, 0, 0, 0.1);
height:100px;
width:250px;
}

p {     color:
#555;


}

h2 {     font-size:
24px;     font-
weight: bold;
}
```

JAVA SCRIPT:

```
// Function to update temperature and humidity data function
updateData(temperature, humidity) {     const temperatureElement =
document.getElementById("temperature");     const humidityElement =
document.getElementById("humidity");

    temperatureElement.textContent = ${temperature} °C;
humidityElement.textContent = ${humidity}%;
}

// Function to connect to MQTT broker and subscribe to topics
function connectToMQTT() {
   // Replace the following with your MQTT broker details
const brokerUrl = "mqtt://your-mqtt-broker";     const
topicTemperature = "environment/temperature";     const
topicHumidity = "environment/humidity";

   const client = new Paho.MQTT.Client(brokerUrl, "web-client-" + parseInt(Math.random() *
1e9, 10));

   // Set up the callback for a successful connection
client.onConnectionLost = onConnectionLost;     client.onMessageArrived
= onMessageArrived;

   client.connect({
onSuccess: onConnect,
onFailure: onFailure,
   });

   // Callback for successful connection
function onConnect() {
console.log("Connected to MQTT broker");
client.subscribe(topicTemperature);
client.subscribe(topicHumidity);
   }

   // Callback for connection failure     function onFailure(responseObject) {
console.log("Failed to connect to MQTT broker: " + responseObject.errorMessage);
   }

   // Callback for connection loss     function
onConnectionLost(responseObject) {        if (responseObject.errorCode
!== 0) {            console.log("Connection lost: " +
responseObject.errorMessage);
      }
   }

   // Callback for received MQTT messages     function
onMessageArrived(message) {        if (message.topic === topicTemperature)
{           const temperature = parseFloat(message.payloadString).toFixed(2);
updateData(temperature, parseFloat(humidityElement.textContent));
      } else if (message.topic === topicHumidity) {           const humidity =
parseFloat(message.payloadString).toFixed(2);
updateData(parseFloat(temperatureElement.textContent), humidity);
      }
   }
}

// Call the connectToMQTT function to establish MQTT connection connectToMQTT();
```

DESIGN THE PLATFORM TO RECEIVE AND DISPLAY ENVIRONMENTAL MONITORING IN PARKS DATA

SENT BY THE IOT DEVICES



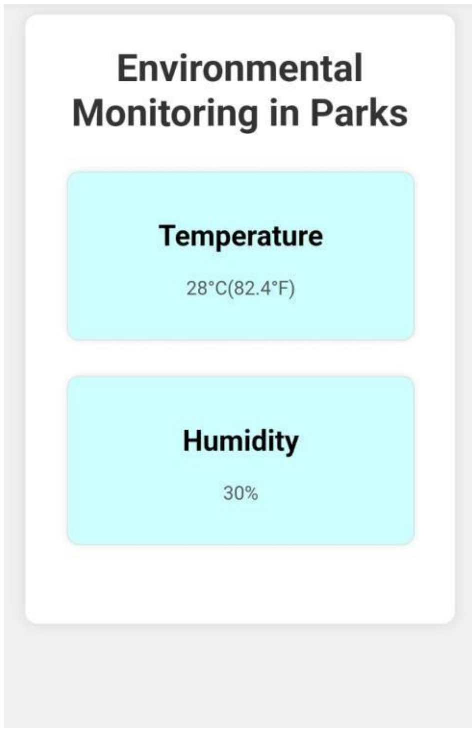
Environmental Monitoring in Parks

Temperature

Loading...

Humidity

Loading...



Environmental Monitoring in Parks

Temperature

28°C(82.4°F)

Humidity

30%

Sensor Nodes:Deploy sensor nodes throughout the park to measure temperature and humidity.Select appropriate IoT sensors, such as DHT22 or BME280, to capture the data.Ensure power sources (e.g., batteries or solar panels) for the nodes.

Communication:Use a communication protocol, such as MQTT or LoRaWAN, to transmit sensor data to a central server.Ensure network connectivity within the park.Central

Server:Set up a central server to collect and store data.Implement data validation and error handling for reliability.Utilize a database (e.g., MySQL or NoSQL) to store historical data.

User Interface:Create a web-based or mobile application for users to access the data.Design an intuitive user interface to display temperature and humidity data.Implement features like real-time updates and historical data retrieval.

Alerts and Notifications:Implement alerting mechanisms for extreme environmental conditions (e.g., heatwaves or high humidity).Send notifications to users or park authorities through email, SMS, or app notifications.

Data Analysis:Incorporate data analytics to identify trends or anomalies in the environmental data.Use tools like Python and data visualization libraries to create graphs and charts.

Security:Ensure data encryption and secure authentication to protect the system from unauthorized access.

Scalability:Design the platform to easily add more sensors or expand to cover larger areas in the park.

Power Management:Optimize power usage in sensor nodes to extend battery life or reduce the need for frequent maintenance.

Maintenance and Support:Establish a maintenance plan for regular checks and sensor calibration.Provide support for users and address issues promptly.

Documentation:Create comprehensive documentation for setup, maintenance, and troubleshooting.

Regulatory Compliance:Ensure compliance with local regulations and data privacy laws when handling sensor data.

Visualization and Reporting:Enable the generation of reports for park management or research purposes.

Community Engagement:Consider involving the local community or park visitors in the project, promoting awareness and engagement.

Conclusion:

This project demonstrates how to utilize the DHT11 Temperature and Humidity Sensor in combination with NodeMCU to send real-time temperature and humidity data to the ThingSpeak IoT server. By doing so, it enables remote monitoring of environmental conditions over the internet, allowing users to access and visualize this data on the ThingSpeak dashboard. This project provides a practical example of how IoT technology can be employed for global environmental data tracking and analysis.