College code :4212 421221106021 Phase 4

Environmental Monitoring System With Temperature Measurement

Project Description:

The Environmental Monitoring System with Temperature Measurement is a comprehensive project aimed at monitoring and controlling various environmental parameters with a primary focus on temperature. This system is designed to provide real-time data on temperature levels, enabling users to make informed decisions for various applications, including climate control, energy efficiency, and environmental conservation.

Certainly, designing a platform to receive and display real-time data from IoT sensors involves a comprehensive approach. Here's a design outline for such a platform:

IoT Sensors:

Start with defining the IoT sensors we will use. These sensors should be capable of measuring environmental parameters such as temperature and humidity.

Data Collection:

Set up the IoT sensors to send data to a central data collection point. This could be a microcontroller or a gateway device.Ensure that the data transmission protocol (e.g., MQTT, HTTP) is established for the sensors to communicate with the central point.

Data Processing:

Design a data processing component to validate, filter, and process incoming data. Remove any anomalies or outliers.

Data Storage:

Create a database to store the received data. Consider using databases like MySQL, PostgreSQL, or NoSQL databases like MongoDB, depending on project's requirements.

Server-side:

Develop a server or backend application to receive data from IoT sensors. Use a web framework such as Node.js, Django, or Flask to handle incoming data and store it in the database.

API Development:

Create RESTful APIs or WebSocket endpoints that IoT devices can send data to.

Implement APIs for querying historical data and real-time data.

Real-time Data:

Use WebSocket or server-sent events to provide real-time data updates. This will enable the platform to push data to the user interface as it's received.

User Interface (UI):

Build a user interface using HTML, CSS, and JavaScript to display real-time data.

Include charts, graphs, and visualizations to make the data more comprehensible.

Add user-friendly features like a dashboard, data history, and notifications.

Client-side:

Develop the client-side scripts to connect to the server and receive real-time data updates.Use JavaScript libraries like Socket.io for real-time communication with the server.

Security:

Implement security measures to protect the data in transit and at rest. Use HTTPS for secure data transmission. Implement authentication and authorization to restrict access to the platform.

Scalability:

Design the platform to be scalable, allowing it to handle a growing number of IoT sensors and users.

Testing:

Rigorously test the entire system for data accuracy, responsiveness, and scalability.

Deployment:

Deploy the platform on a cloud server or a dedicated web server for accessibility over the internet.

Monitoring and Maintenance:

Implement monitoring tools to keep an eye on system performance and resolve any issues that may arise.Regularly update and maintain the platform to ensure its reliability.

Languages: C, PHP, JavaScript, Ajax, Bootstrap, HTML, CSS

Hardware: Arduino UNO, SIM808, GPS antenna

DEVELOPING THE REAL-TIME ENVIRONMENTAL MONITORING SYSTEM WITH TEMPERATURE MEASUREMENT BY USING WEB TECHNOLOGY

Html :

```
<!DOCTYPE html>
<html>
<head>
    <title>Environmental Monitoring</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <h1>Real-time Environmental Data</h1>
    <div id="temperature">Temperature: - °C</div>
    <div id="humidity">Humidity: - %</div>
    <script src="script.js"></script>
```

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

CSS:

```
body {
    text-align: center; font-family:
    Arial, sans-serif;
}
h1 { color:
   #333;
}
```

Javascript:

```
// Function to update temperature data function
updateTemperature() { fetch('/getTemperatureData') // Replace
with your API endpoint
      .then(response => response.json())
      .then(data => {
        document.getElementById('temperature').textContent = `Temperature:
${data.temperature} °C`;
      })
      .catch(error => { console.error('Error fetching temperature
         data:', error);
      });
```

```
}

// Fetch temperature data every 5 seconds (adjust this interval as needed)
setInterval(updateTemperature, 5000);
```

Applications of environmental monitoring system with temperature measurement:

Home climate control

Greenhouse temperature regulation

Industrial temperature monitoring

Weather stations

Energy-efficient HVAC systems

Environmental conservation projects

Environmental Monitoring IoT Platform.