ENVIRONMENT MONITERING

Phase 4 : Development Part 2

TEAM MEMBERS : MAHASRINATH.M 814821106020

Introduction

An Environmental Monitoring System using the Internet of Things (IoT) is a cutting-edge solution that leverages interconnected devices, sensors, and data analytics to gather, manage, and analyze environmental data in real-time. This technology plays a crucial role in addressing environmental challenges, tracking the state of natural resources, and ensuring a sustainable future. It can be applied in various contexts, from urban areas to remote wilderness, helping to assess and manage environmental conditions more effectively.

Overview

1. The use of IoT in an EMS allows for the monitoring and control of various environmental parameters, such as air quality, water quality, energy consumption, waste management, and more. Sensors and devices can be deployed to collect data on these parameters, which is then transmitted to a central system for analysis and action.

2. The real-time nature of IoT data allows organizations to respond quickly to environmental incidents or deviations from set targets. For instance, if a sensor detects a sudden increase in air pollution levels, an alert can be generated, enabling immediate action to be taken to mitigate the issue.

3. In summary, incorporating IoT into an EMS provides organizations with the ability to collect and analyze real-time environmental data, leading to improved environmental performance, resource efficiency, and sustainability. It allows for proactive environmental management, reduces costs, and enhances overall operational effectiveness.

VARIOUS TOOLS USED FOR THIS PROJECT

To create a platform that displays real-time environmental data using web development technologies (HTML, CSS, JavaScript), follow these steps:

1. Set Up Your Project

Create a project folder and set up basic files (e.g., index.html, style.css, script.js) inside it.

2.HTML Structure

Open `index.html` and create the basic structure of your platform. This may include headers, navigation, and a section for displaying data.

```
html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

```
    <title>Environmental Monitoring Platform</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <h1>Environmental Monitoring Platform</h1>
    </header>
    <section class="data-display">
        <!-- Real-time data will be displayed here -->
    </section>
    <script src="script.js"></script>
</body>
</html>
```

3. CSS Styling

Open `style.css` and apply basic styling. Customize as per your design preferences.

```
css
body {
    font-family: Arial, sans-serif;
}

header {
    background-color: #333;
    color: #fff;
    padding: 1em;
    text-align: center;
}
```

```
.data-display {
    padding: 2em;
}
```

4. JavaScript for Real-time Data

Open `script.js` and start working on the JavaScript code to handle real-time data.

```
javascript
// Simulated real-time data (replace this with actual data handling)
function generateRandomData() {
    const temperature = Math.random() * 30 + 10; // Random temperature between 10°C and 40°C
    const humidity = Math.random() * 50 + 40; // Random humidity between 40% and 90%
    return { temperature, humidity };
}

function updateDataDisplay() {
    const data = generateRandomData(); // Replace with actual data retrieval logic
    const displayElement = document.querySelector('.data-display');

    // Update the HTML to display real-time data
    displayElement.innerHTML = `
        <h2>Real-time Data</h2>
        <p>Temperature: ${data.temperature.toFixed(2)}°C</p>
        <p>Humidity: ${data.humidity.toFixed(2)}%</p>
    `;
}
```

```
// Update data every 2 seconds (adjust as needed)
setInterval(updateDataDisplay, 2000);
```

5. Testing

Open the `index.html` file in a web browser. You should see a basic platform with simulated real-time data.

6. Connect with Actual Data Source

Replace the `generateRandomData` function with actual code to fetch real-time data from your IoT devices. This may involve using AJAX requests, WebSockets, or other technologies depending on your setup.

7.Further Enhancements

As your project progresses, you can add features like data visualization, user authentication, and additional environmental parameters.

HTML

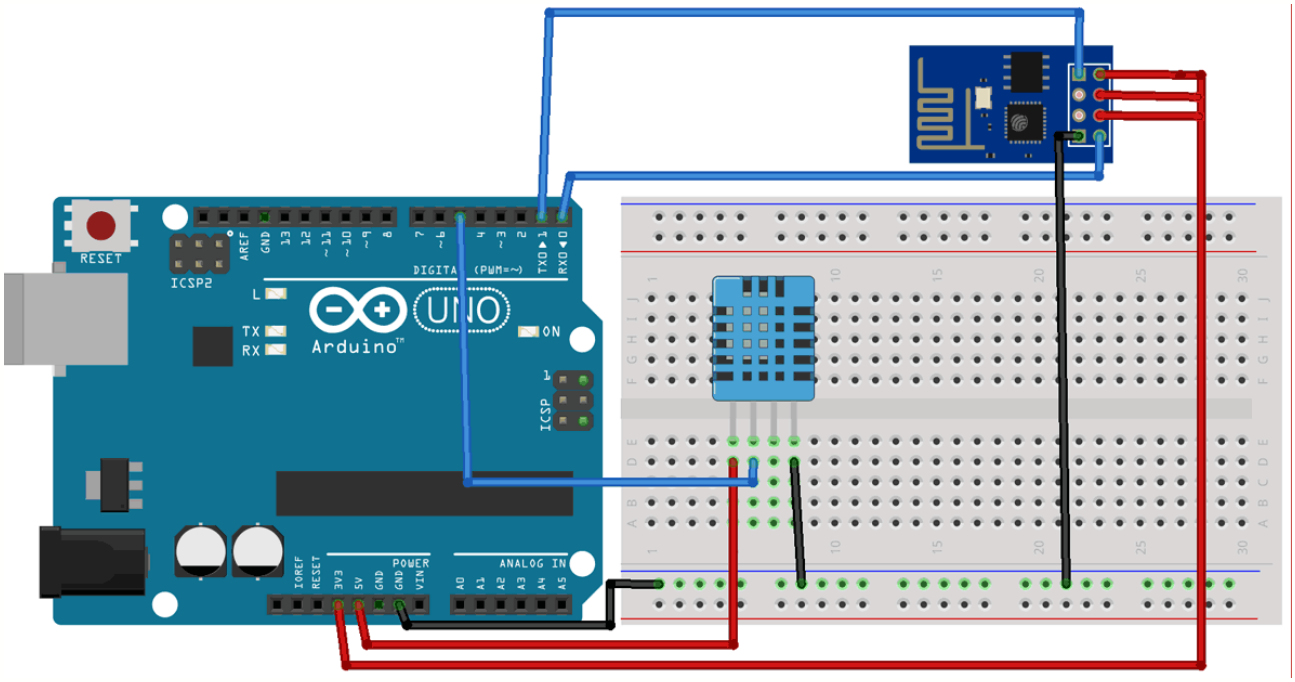
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Real-time Water Level and Flood Warnings</title>
    <link rel="stylesheet" href="styles.css">
```

```
</head>
<body>
    <div class="container">
        <h1>Real-time Water Level Data</h1>
        <div id="waterLevel"></div>
        <h2>Flood Warnings</h2>
        <div id="floodWarning"></div>
    </div>
    <script src="script.js"></script>
</body>
</html>

body {
    font-family: Arial, sans-serif;
    text-align: center;
}

.container {
    max-width: 800px;
    margin: auto;
    padding: 20px;
}

#waterLevel, #floodWarning {
    font-size: 24px;
    margin-bottom: 20px;
}
```

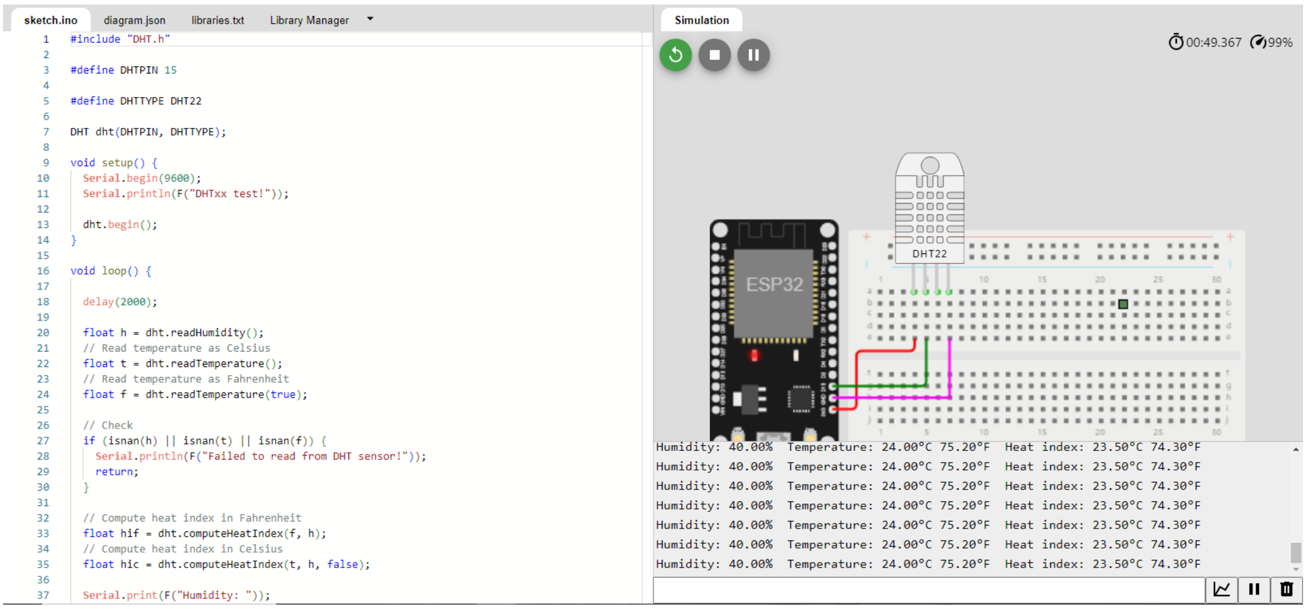
```
// Simulated real-time data (replace with actual data source)
function getRandomWaterLevel() {
    return (Math.random() * 10).toFixed(2); // Generating random water level between 0 and 10
}

function getRandomFloodWarning() {
    return Math.random() > 0.8; // Simulating a 20% chance of flood warning
}

function updateData() {
    const waterLevelElement = document.getElementById('waterLevel');
    const floodWarningElement = document.getElementById('floodWarning');

    const waterLevel = getRandomWaterLevel();
    const floodWarning = getRandomFloodWarning();

    waterLevelElement.innerText = `Water Level: ${waterLevel} meters`;
    floodWarningElement.innerText = floodWarning ? 'Flood Warning: Active' : 'Flood Warning:
Inactive';
 if (floodWarning) {
        floodWarningElement.style.color = 'red';
    } else {
        floodWarningElement.style.color = 'green';
```

```
    }
}
// Update data every 5 seconds (adjust as needed)
setInterval(updateData, 5000);
```

Circuit Design



Display real-time temperature and humidity data



Conclusion

In conclusion, environmental monitoring through Arduino IoT is an accessible and versatile way to understand and manage natural resources. With the help of modern technology and the Arduino platform's extensive possibilities, we can effectively monitor various environmental parameters and work towards a better future. We encourage readers to explore A**rduino IOT**and the world of environmental science to make a positive impact on our planet.

THANK YOU