Flood Monitoring And Early Warning System Using IOT With Sensors

Project Description:

Continue building the project by developing the environmental monitoring platform. Use web development technologies (e.g., HTML, CSS, JavaScript) to create a platform that displays real-time environmental data. Design the platform to receive and display real-time temperature and humidity data from IoT devices.

Gps Sensor:

Ultrasonic sensors are deployed on hundreds of coastal tide gauge platforms that provide tsunami and tropical storm surge warning data.

Benefits:

Timely detection of possible flood risks and floods. Highly reliable and available real-time data. Tailored solution that can be integrated with external developments at any level (device, connectivity, cloud or user application).

Application:

In this system we make use of a raspberry pi with water sensors, rain sensors to predict flood and alert respective authorities and sound instant alarm in nearby villages to instantly transmit information about possible floods using IOT. The water sensors are used to measure water level of 3 different location

1. Define the Project Structure:

Before you start coding, plan your project's structure. Create folders for HTML, CSS, JavaScript, and any other assets (images, icons, etc.). This helps keep your code organized.

2. Design the User Interface:

Design a user-friendly interface to display the real-time data. Consider using a responsive design to make it accessible on different devices. You can use HTML for the structure, CSS for styling, and JavaScript for interactivity.

3. CSS Styling:

Style your platform using CSS to make it visually appealing and user-friendly. You can customize fonts, colors, and layouts according to your preferences.

4. JavaScript for Real-Time Data:

Use JavaScript to fetch real-time data from your IoT devices and update the HTML content accordingly. You can use technologies like WebSockets or AJAX to achieve real-time updates. Here's a basic example using JavaScript and AJAX:

5. Backend and IoT Integration:

On the server-side, you'll need to create an endpoint to fetch data from your IoT devices. You can use a web framework like Node.js with Express to handle these requests. Make sure your IoT devices are set up to provide real-time data through an API.

6. Testing and Deployment:

Test your platform thoroughly, both locally and on different devices, to ensure it works as expected. After testing, deploy your platform to a web server so it's accessible to users.

7. Security:

Ensure that you implement proper security measures to protect your platform from unauthorized access and data breaches, especially if it's dealing with real-time data from IoT devices.

Language: HTML,CSS,Java Script,PHP and MySql.

Hardware: Float switch for water level detector, inverter, rain gauge, GSM module, and microcontroller development board

DEVELOPING THE REAL-TIME TRANSIT INFORMATION PLATFORM BY USING WEBTECHNOLOGY

HTML STRUCTURE:

```
<!DOCTYPE html>
<html>
<head>
   <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
   <header>
      <h1>Environmental Monitoring Platform</h1>
   </header>
   <main>
      <section id="sensor-data">
         <h2>Real-time Data</h2>
```

```
            <div id="temperature">Temperature: <span></span>°C</div>
            <div id="humidity">Humidity: <span></span>%</div>
        </section>
    </main>
    <script src="script.js"></script>
</body>
</html>
```

JavaScript for Real time data:

```
// script.js
// Function to fetch and update temperature and humidity data
function updateData() {
    // Make an AJAX request to your IoT device endpoint
    fetch('/api/environmental-data')
        .then(response => response.json())
        .then(data => {
```

```
            // Update the HTML with real-time data
            document.querySelector('#temperature span').textContent = data.temperature + '°C';
            document.querySelector('#humidity span').textContent = data.humidity + '%';
        })
        .catch(error => {
            console.error('Error fetching data:', error);
        });
}

// Update the data every 10 seconds (adjust as needed)
setInterval(updateData, 10000);
updateData(); // Call initially to load data
```

DESIGN THE PLATFORM TO RECEIVE AND DISPLAY REAL-TIME DATA FROM IOT SENSOR