TECHNOLOGY : INTERNET OF THINGS

PROJECT TITLE : SMART WATER FOUNTAINS

NAME : BALAGAJARAJ P

Main theme:

Continue building the project by developing the water fountain status platform. Use web development technologies (e.g., HTML, CSS, JavaScript) to create a platform that displays real-time water fountain status.

Hardware Selection:

Choose the appropriate sensors and hardware for monitoring the water fountain. You might need water level sensors, a microcontroller (e.g., Arduino or Raspberry Pi), and possibly a camera for visual feedback.

Software Development:

Develop the software to interface with the sensors and hardware. You'll need to write code to collect data and send it to the platform.

Data Storage:

Decide how and where you'll store the data. This could be a database or a cloud-based solution, depending on your preferences.

Assumes you have a backend system providing real-time data through JavaScript (AJAX).

```
<!DOCTYPE html>

<html>

<head>

  <title>Water Fountain Status</title>

</head>
```

```
<body>
  <h1>Water Fountain Status</h1>
  <p id="status">Loading...</p>

  <script>
    // Function to update the fountain status
    function updateStatus() {
      // Make an AJAX request to your backend to get the real-time data
      const xhr = new XMLHttpRequest();
      xhr.open('GET', '/api/getFountainStatus', true);

      xhr.onload = function () {
        if (xhr.status === 200) {
          const statusData = JSON.parse(xhr.responseText);
          const statusElement = document.getElementById('status');
          statusElement.innerHTML = `Status: ${statusData.status}`;
        } else {
          console.error('Failed to retrieve status.');
        }
      };

      xhr.send();
    }

    // Update the status initially and then every 5 seconds
    updateStatus();
    setInterval(updateStatus, 5000);
  </script>
</body>
```

```
</html>
```

We have a simple HTML structure with a title, a header, and a paragraph where the real-time status will be displayed.

JavaScript is used to make an AJAX request to a hypothetical /api/getFountainStatus endpoint on your backend. This endpoint should return JSON data containing the current fountain status.

The updateStatus function retrieves the data from the backend and updates the status paragraph accordingly.

We use setInterval to periodically call the updateStatus function, which keeps the status updated at regular intervals (every 5 seconds in this case)

Frontend (HTML, CSS, JavaScript):

User Interface Design: Design a user-friendly web page with sections for displaying real-time data and alerts.

HTML Structure:

Creating an HTML structure for your page, including headings, divs, and placeholders for data.

```
<!DOCTYPE html>
<html>
<head>
  <title>Smart Water Fountain Dashboard</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <header>
    <h1>Smart Water Fountain Dashboard</h1>
  </header>

  <section class="real-time-data">
```

```
        <h2>Real-time Data</h2>
        <div class="data-card">
            <h3>Flow Rate</h3>
            <p id="flow-rate">Loading...</p>
        </div>
        <div class="data-card">
            <h3>Temperature</h3>
            <p id="temperature">Loading...</p>
        </div>
    </section>

    <section class="alerts">
        <h2>Alerts</h2>
        <div class="alert-card">
            <h3>Malfunction Alert</h3>
            <p id="malfunction-alert">No alerts</p>
        </div>
    </section>

    <script src="script.js"></script>
</body>
</html>
```

CSS Styling:

By CSS to style the page, making it visually appealing and responsive for different devices.

```
body {
    font-family: Arial, sans-serif;
}
header {
    background-color: #007BFF;
    color: white;
    text-align: center;
    padding: 20px;
}
h1 {
    margin: 0;
}
.section {
    margin: 20px;
    padding: 20px;
    border: 1px solid #E0E0E0;
    border-radius: 5px;
    background-color: #F7F7F7;
}
.data-card, .alert-card {
    margin: 10px;
    padding: 10px;
    border: 1px solid #D0D0D0;
    border-radius: 5px;
    background-color: white;
}

.alert-card {
    background-color: #FF8888;
}

.data-card h3, .alert-card h3 {
    margin: 0;
```

```
}
h2 {
    margin-top: 0;
}
```

Javascript:

By Using JavaScript to fetch real-time data and update the page dynamically.

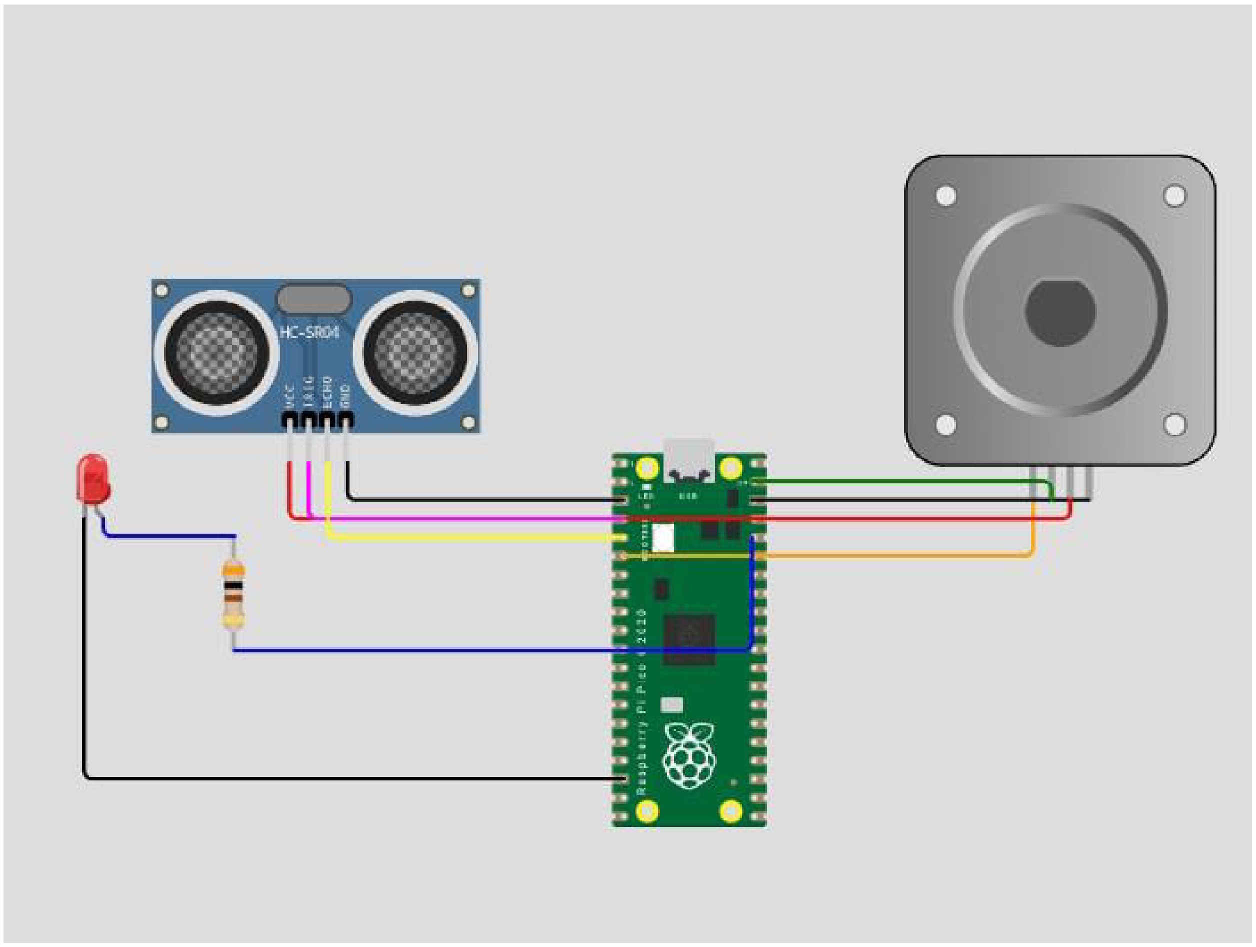
```
// Simulated real-time data for demonstration
function updateData() {
    document.getElementById('flow-rate').textContent = (Math.random() * 10 + 20).toFixed(2) + ' L/min';
    document.getElementById('temperature').textContent = (Math.random() * 10 + 15).toFixed(2) + ' °C';

    const malfunctionAlert = Math.random() > 0.8; // Simulated malfunction alert
    document.getElementById('malfunction-alert').textContent = malfunctionAlert ? 'Malfunction Detected!' : 'No alerts';
}
// Update data every 5 seconds (simulated data in this example)
setInterval(updateData, 5000);

// Initial data update
updateData();
```

To display real-time water fountain data, including water flow rate and malfunction alerts by python:

```
import time
def initialize_flow_sensor()
    pass
```

```
def read_flow_rate(sensor):
    flow_rate = sensor.read()
    return flow_rate
def control_fountain(flow_rate)
    pass
if __name__ == '__main__':
    flow_sensor = initialize_flow_sensor()
    try:
        while True:
            flow_rate = read_flow_rate(flow_sensor)
            control_fountain(flow_rate)
            time.sleep(1)  # Adjust the monitoring interval as needed
    except KeyboardInterrupt:
        pass
```

Simulation diagram of smart water fountain


HC-SR04
VCC
TRIG
ECHO
GND
Raspberry Pi Pico © 2020