IOT-PHASE-5: smart water foundation

Project Objectives:

The objective of this project is to create a real-time water fountain status system using IoT sensors and Raspberry Pi integration. The system aims to promote water efficiency and public awareness by providing real-time data on water fountain usage and availability.

IoT Sensor Setup:

Water Flow Sensor: Measures water flow rate in the fountain's pipeline.

Ultrasonic Sensor: Detects water level in the fountain to determine if it needs refilling.

Temperature and Humidity Sensor: Monitors the environment for optimal fountain conditions.

Mobile App Development:

Develop a mobile app (iOS/Android) to display real-time sensor data. Use libraries like Flask for backend development and React Native for cross-platform app development. Implement graphical representations of sensor data and push notifications for alerts.

Raspberry Pi Integration:

Connect the sensors to a Raspberry Pi board using GPIO pins. Write Python scripts to read data from sensors and send it to a cloud server for further processing.

Code Implementation:

Python

Copy code

```
# Sample Python code to read data from sensors and send it to the cloud server

Import RPi.GPIO as GPIO

Import time

Import requests
```

```
# GPIO setup for sensors
FLOW_SENSOR_PIN = 17
ULTRASONIC_TRIGGER_PIN = 23
ULTRASONIC_ECHO_PIN = 24

GPIO.setmode(GPIO.BCM)
GPIO.setup(FLOW_SENSOR_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(ULTRASONIC_TRIGGER_PIN, GPIO.OUT)
GPIO.setup(ULTRASONIC_ECHO_PIN, GPIO.IN)

# Function to read water flow rate
Def calculate_flow_rate(channel):
    # Implementation for reading flow rate from the sensor
    Pass

# Function to measure water level using ultrasonic sensor
Def measure_water_level():
    # Implementation for measuring water level using ultrasonic sensor
    Pass

# Main loop
While True:
    Flow_rate = calculate_flow_rate(FLOW_SENSOR_PIN)
    Water_level = measure_water_level()

    # Send data to the cloud server
    Data = {
        “flow_rate”: flow_rate,
        “water_level”: water_level,
        “temperature”: read_temperature_sensor(),
        “humidity”: read_humidity_sensor()
    }
    Response = requests.post(http://example.com/api/data, json=data)

    Time.sleep(60)  # Send data every minute
```

Promoting Water Efficiency and Public Awareness:

Water Efficiency: By monitoring flow rate and water levels, the system can optimize fountain usage, ensuring water is not wasted when the fountain is not in demand.

Public Awareness: The mobile app provides real-time information to users, encouraging mindful water usage. Push notifications remind users to use the fountain responsibly. Additionally, data analytics can generate insights on usage patterns, enabling targeted awareness campaigns.

Please note that you would need to integrate this code with a cloud server and implement error handling, security measures, and proper data processing logic for a complete and functional system.

Smart Water Usage: By monitoring water flow and levels in real-time, the system can optimize the fountain’s operation. For instance, the fountain can be turned off during non-peak hours or when the water level is low, saving water resources.

Leak Detection: The system can detect abnormal water flow patterns, indicating leaks in the fountain or the pipeline. Early detection helps in timely repairs, preventing water wastage.

Usage Analytics: Analyzing usage patterns over time can provide valuable insights. For example, understanding peak usage hours enables efficient scheduling, ensuring the fountain operates when people need it the most.

User Awareness: The accompanying mobile app can educate users about responsible water usage. Push notifications can remind users to turn off the fountain after use, fostering a sense of responsibility among the public.

Water Conservation Campaigns: Data collected over time can be used for awareness campaigns. Visualizations and statistics generated from the data can be shared with the public, highlighting the importance of water conservation.

Here's a Python code snippet that demonstrates how you can implement a basic real-time water fountain status system using Flask for the backend and React for the frontend. Note that this is a simplified version within the 200-line limit, and you'll need to expand and customize it according to your specific requirements.

Backend (Flask):

Python

Copy code

```
From flask import Flask, jsonify, request
Import random


App = Flask(__name__)


@app.route('/fountain', methods=['GET'])
Def get_fountain_status():
   # Simulated data for demonstration purposes
   Flow_rate = random.uniform(1, 10)  # Random flow rate between 1 and 10 units
   Water_level = random.uniform(0, 100)  # Random water level percentage


   Return jsonify({'flow_rate': flow_rate, 'water_level': water_level})


If __name__ == '__main__':
   App.run(debug=True)
```

Frontend (React):

Jsx

Copy code

```
Import React, { useState, useEffect } from ‘react’;
Import axios from ‘axios’;

Function App() {
  Const [fountainStatus, setFountainStatus] = useState({});

  useEffect(() => {
    const fetchData = async () => {
      const response = await axios.get(‘/fountain’);
      setFountainStatus(response.data);
    };

    Const interval = setInterval(fetchData, 5000); // Fetch data every 5 seconds

    Return () => clearInterval(interval); // Cleanup on component unmount
  }, []);

  Return (
    <div className=”App”>
      <h1>Water Fountain Status</h1>
      <p>Flow Rate: {fountainStatus.flow_rate} units</p>
      <p>Water Level: {fountainStatus.water_level}%</p>
    </div>
  );
}

Export default App;
```

This example provides a basic foundation. You can further enhance the system by integrating real IoT sensors, implementing user authentication, and adding more features to the mobile app for promoting water efficiency and public awareness.