Smart Water Fountain :

Phase 5:

Objectives

The smart water fountain project aims to develop a water fountain that can be monitored and controlled remotely using the Internet of Things (IoT). The project objectives are to:

- Design and implement an IoT sensor setup to monitor the water fountain status in real time, including water level, temperature, and flow.
- Develop a mobile app to allow users to view the water fountain status and control the water flow remotely.
- Integrate a Raspberry Pi into the system to collect and process sensor data, and to communicate with the mobile app.
- Implement a codebase to control the water fountain and to communicate with the mobile app.

IoT Sensor Setup

The following IoT sensors can be used to monitor the water fountain status:

- Water level sensor: To measure the water level in the fountain.
- Temperature sensor: To measure the temperature of the water in the fountain.
- Flow sensor: To measure the flow rate of the water in the fountain.

These sensors can be connected to a microcontroller, such as a Raspberry Pi, to collect and process the sensor data. The microcontroller can then communicate the sensor data to the mobile app using a wireless protocol, such as Wi-Fi or Bluetooth.

Smart Water Fountains:

In our fast paced world, the water fountain will be acting as the natural humidifiers and it also adds moisture to a dry room . This is an overview of smart water fountain that jets the water into the aor in a decorative and dramatic effect.

Smart water fountain are commonly used to promote peace and well being. This smart water fountain uses an arduino which is an open source electronics platform.

The fountain works fully automatically but can also be set up remotely via a mobile app for your convenience. Our smart water fountain will be used with a help of a laptop.

Flow of water from high gravitational potential energy to low gravitational potential energy causes a water fountain to form, due to increasing pressure on the inside of the symstem.

Materials needed:

- 4x 12v mini fountain
- 4 relay module
- Arduino
- 12v power supply for the mini fountain
- Female/Male to jumper cables

Codes:

```
#define RELAY1 9
```

```
#define RELAY2 10
#define RELAY3 11
#define RELAY4 12
Void setup()
{
OUTPUT
pinMode(Relay1,OUTPUT);
pinMode(Relay2,OUTPUT);
pinMode(Relay3,OUTPUT);
pinMode(Relay4,OUTPUT);
}
Void loop()
{
digitalWrite(RELAY1,LOW);
delay(500);
digitalWrite(RELAY4,HIGH);
digitalWrite(RELAY2,LOW);
delay(500);
digitalWrite(RELAY1,HIGH);
digitalWrite(RELAY3,LOW);
delay(500);
digitalWrite(RELAY2,HIGH);
```

```
 digitalWrite(RELAY4,LOW);

delay(500);

 digitalWrite(RELAY3,HIGH);

}
```

Mobile App Development

The mobile app should allow users to view the following water fountain status information:

- Water level
- Temperature
- Flow rate

The app should also allow users to control the water flow remotely, such as turning on or off the fountain, or adjusting the flow rate.

Raspberry Pi Integration

Raspberry Pi is a single-board computer that can be used to collect and process sensor data, and to communicate with the mobile app. The Raspberry Pi can be connected to the IoT sensors using a breadboard and jumper wires.

The Code Implementation

The following Python code can be used to implement the smart water fountain system:

Python:

```
import time

import board

import pwmio
```

```
import adafruit_dht
import paho.mqtt.client as mqtt
# Create a PWM object to control the water pump
pump = pwmio.PWMOut(board.D18)
# Create a DHT object to read the temperature and humidity
dht = adafruit_dht.DHT22(board.D4)

# Create an MQTT client to connect to the cloud
client = mqtt.Client()
# Connect to the MQTT broker
client.connect("localhost", 1883)
# Subscribe to the "water_fountain_status" topic
client.subscribe("water_fountain_status")
def publish_water_fountain_status():
    """Publish the water fountain status to the cloud"""
    # Get the water level
    water_level = dht.temperature
    # Get the temperature
    temperature = dht.humidity
    # Get the flow rate
    flow_rate = pump.duty_cycle
    # Publish the water fountain status to the cloud
```

```
    client.publish("water_fountain_status", json.dumps({
        "water_level": water_level,
        "temperature": temperature,
        "flow_rate": flow_rate
    }))
# Loop forever and publish the water fountain status every 10 seconds
while True:
    publish_water_fountain_status()
    time.sleep(10)
```

- How the Real-Time Water Fountain Status System Promotes Water Efficiency and Public Awareness
- The real-time water fountain status system can promote water efficiency and public awareness in the following ways:
- Water efficiency: The system can help users to conserve water by providing them with real-time information on the water level and flow rate. This information can help users to identify and fix water leaks, and to adjust the water flow rate to avoid water waste.
- Public awareness: The system can help to raise public awareness of water conservation by providing users with information on the water fountain status and their own water usage. This information can help users to understand the importance of water conservation and to take steps to reduce their water consumption.

Smart Water Fountains:

In our fast paced world, the water fountain will be acting as the natural humidifiers and it also adds moisture to a dry room . This is an overview of smart water fountain that jets the water into the aor in a decorative and dramatic effect.

Smart water fountain are commonly used to promote peace and well being. This smart water fountain uses an arduino which is an open source electronics platform.

The fountain works fully automatically but can also be set up remotely via a mobile app for your convenience. Our smart water fountain will be used with a help of a laptop.

Flow of water from high gravitational potential energy to low gravitational potential energy causes a water fountain to form, due to increasing pressure on the inside of the symstem.

Problem: Existing smart water fountains may have issues with water wastage, hygiene, and user engagement.

Solution:

Sensors and AI Integration:Incorporate advanced sensors and AI algorithms to detect when a person approaches the fountain. The AI can analyze foot traffic data to predict peak usage times and adjust water flow accordingly, reducing wastage during off-peak hours.

Hygiene and Safety: To enhance hygiene, use UV-C sterilization technology within the fountain's nozzle area. The UV-C light can automatically disinfect the spout after each use, ensuring clean and safe drinking water.

User-Friendly Interface: Implement a user-friendly touch screen interface with multilingual options, making it easy for people of diverse backgrounds to understand and use the fountain. It can also display water quality metrics in real-time.

Customizable Water Temperature: Allow users to choose between cold, ambient, and warm water options, catering to different preferences and weather conditions.

Reusable Water Bottles: Design a specialized slot for reusable water bottles to encourage eco-friendly practices. Users can place their bottles under the spout, and the fountain can automatically fill them with the desired amount of water.

Mobile App Integration: Develop a companion mobile app that allows users to locate nearby smart water fountains, check water quality, and even pre-set their water temperature and quantity preferences. The app can also gamify water consumption to promote healthy hydration habits.

Sustainability Features: Incorporate a mechanism to collect and filter rainwater, utilizing it as a source for the fountain during rainy seasons, thereby conserving municipal water resources.

Accessibility: Ensure the fountain is ADA-compliant with features like accessible height adjustments, voice commands, and Braille instructions.

Data Analytics: Gather usage data to identify trends and inform maintenance schedules. Proactively address issues and reduce downtime.

Solar Power: Make use of solar panels to power the fountain, reducing energy costs and environmental impact.

Education and Awareness: Use the fountain as an educational tool by displaying facts about water conservation and the environmental impact of single-use plastic bottles.

Aesthetic Design: Create an appealing, modern, and iconic design for the fountain, encouraging more people to use it and making it a focal point in public spaces.

By addressing these aspects of design and innovation, smart water fountains can become more efficient, hygienic, user-friendly, and environmentally conscious, ultimately improving the drinking water experience for people in public spaces.

Platform UI code for Smart Water Fountains:

```
<!DOCTYPE html>
<html>
<head>
<title>Smart Water Fountains</title>
<style>
 /* Style for the water level display */
 #water-level {
 font-size: 24px;
 font-weight: bold;
 }
 /* Style for the submit button */
 #submit-button {
 padding: 10px 20px;
 font-size: 18px;
 }
```

```
</style>

</head>

<body>

<h1>Smart Water Fountains</h1>

<p>Water Level: <span id="water-level">Loading...</span> cm</p>

<button id="submit-button" onclick="sendDataToServer()">Submit Data</button>

<script>

 // Function to update water level data from the server

 function updateWaterLevel() {

 // You can use AJAX or fetch to get data from your server

 // Replace the URL with the actual endpoint that provides water level data

fetch('/getWaterLevelData')

.then(response =>response.json())

.then(data => {

document.getElementById('water-level').textContent = data.waterLevel + " cm";

 })

.catch(error => {

console.error('Error fetching water level data:', error);

 });

 }
```

```
 // Function to send data to the server (e.g., to trigger data collection)
 function sendDataToServer() {
 // You can use AJAX or fetch to send data to your server
 // Replace the URL with the actual endpoint that handles data submission
fetch('/submitData', { method: 'POST' })
.then(response => {
 if (response.status === 200) {
console.log('Data submitted successfully');
 } else {
console.error('Data submission failed with status:', response.status);
 }
 })
.catch(error => {
console.error('Error submitting data:', error);
 });
 }
 // Update water level initially and then at regular intervals
updateWaterLevel();
setInterval(updateWaterLevel, 10000); // Update every 10 seconds
</script>
</body>
```

</html>

Output for above Program: