SMART WATER FOUNTAIN

TEAM MEMBER

920321104014- Hearam Kishore M

PHASE 3 -SUBMISSION DOCUMENT

Project Title: Smart water Fountain

Introduction:

Determine the purpose and features of your smart water fountain.
Decide on the sensors, actuators, and connectivity options.

Gather Hardware and Components:

Raspberry Pi or similar single-board computer.
Water pump.
Water level sensor.
Motion sensor (optional).
Camera module (optional).
WiFi or Ethernet module for connectivity.
Power source (e.g., batteries or a power adapter).

Install and Configure Raspberry PI:

Install a suitable IoT-oriented operating system (e.g., Raspbian).

Configure the WiFi or Ethernet connection.

Develop IoT Software:

Write code to control the water pump and read sensor data.
Use a programming language like Python.
Implement IoT communication protocols such as MQTT or HTTP for data transmission.

Connect to the Cloud:

Configure your Raspberry Pi to communicate with the IoT cloud platform.
Send sensor data (e.g., water level, motion) to the cloud.

Implement User Interface:

Develop a web or mobile app for users to control the fountain remotely.
Integrate features like turning the fountain on/off, checking water levels, and viewing camera feeds (if applicable).

Implement Automation:

Create rules or scripts in the cloud platform to automate fountain actions based on sensor data.
For example, you can set rules to refill the fountain when water levels are low.

Security Measures:

Implement security best practices to protect your IoT system from cyber threats.

Testing and Debugging:

Test the entire system to ensure it functions as intended. Debug any issues that arise.

Deployment:

Install the smart water fountain in the desired location. Ensure it has a reliable power source and internet connectivity.

Python script for IOT sensor:

Setup:

First, make sure you have the necessary Python libraries installed. You'll need the Paho MQTT client library, which can be installed using pip:

```
pip install paho-mqtt
```

Python Script:

```
import paho.mqtt.client as mqtt
import random
import time

# Define the MQTT broker and topic information
```

```
mqtt_broker = "mqtt.eclipse.org"  # Update with your
MQTT broker
mqtt_topic = "water_fountain/status"  # Update with your
topic

# Simulate water fountain status (replace with real sensor
data)
def get_water_fountain_status():
    return random.choice(["On", "Off"])

# Create an MQTT client
client = mqtt.Client("WaterFountainClient")

# Connect to the MQTT broker
client.connect(mqtt_broker, 1883)

try:
    while True:
        # Get the current water fountain status
        status = get_water_fountain_status()

        # Publish the status to the MQTT topic
        client.publish(mqtt_topic, status)

        print(f"Published: Water Fountain Status - {status}")
        time.sleep(5)  # Adjust the interval as needed
except KeyboardInterrupt:
    client.disconnect()
```

Replace mqtt_broker with your MQTT broker's address, and mqtt_topic with the desired topic for publishing the water fountain status.

Explanation:

The script uses the Paho MQTT library to communicate with an MQTT broker.
The get_water_fountain_status function is a placeholder for retrieving real sensor data. In this example, it generates random "On" or "Off" statuses. Replace it with your actual sensor data retrieval logic.
The script connects to the MQTT broker, publishes the status to the specified topic, and repeats the process at a defined interval.

Platform Integration:

You'll need to set up an MQTT broker and configure your platform to subscribe to the water_fountain/status topic to receive and process the real-time data.

Remember that this is a basic example, and real IoT deployments often involve additional security, error handling, and other considerations. Ensure that your IoT devices and platform are properly configured for your specific use case.