SMART WATER FOUNTAINS

NAME: M.Gopinath

NM ID: au422721106018

Mail id: marsalgopi1212@gmail.com

PROJECT TITLE: Smart Water Fountains

DEPLOYING IOT SENSORS

1.Sensor Selection:

Choose appropriate sensors tailored to the specific requirements of monitoring public water fountains. Select sensors with the capacity to withstand outdoor conditions and provide accurate readings.

2.Site Assessment:

Conduct a thorough site assessment to determine optimal sensor placement. Identify key locations such as the water supply line, outlets, and other critical points in the water fountain system.

3.Power Supply:

Decide on the power source for the sensors. In public areas, it's often practical to use low-power or battery-powered sensors. Consider using solar panels for sustainable power if feasible.

4.Sensor Placement:

Install flow rate sensors in the water supply lines and pressure sensors near the fountain outlets to measure water pressure. Ensure that sensors are securely mounted and protected from vandalism and weather conditions.

5.Sensor Calibration:

Calibrate the sensors to provide accurate readings. This calibration process may vary depending on the sensor type and manufacturer.

6.Wiring and Connectivity:

Set up the necessary wiring and connectivity for the sensors. This may include cables and connectors. Consider using wireless technologies like LoRa, Wi-Fi, or cellular for data transmission.

7.Data Logging and Transmission:

Connect the sensors to data loggers or microcontrollers capable of reading and processing the sensor data. Set up the devices to transmit the data to a central hub or IoT platform. Ensure data encryption for security.

8.IoT Platform Integration:

Integrate the data collection system with an IoT platform. The platform should be capable of securely receiving, storing, and processing the sensor data.

9.Real-time Monitoring:

Develop a real-time monitoring system that can display the water fountain status. This may include web-based dashboards, mobile apps, or on-site display screens.

10.Alerting Mechanisms:

Implement alerting mechanisms based on predefined thresholds. When abnormal flow rates or pressure levels are detected, the system should send alerts to designated personnel or authorities via email, SMS, or push notifications.

11. **Data Storage and Analysis**:

    Store sensor data for historical analysis and reporting. Use data analytics tools or machine learning algorithms to identify trends and anomalies in the data.

12. **Regular Maintenance**:

    Establish a maintenance schedule for the sensors and related equipment. This includes checking sensor accuracy, changing batteries, and ensuring proper sensor hygiene.

13. **Regulatory Compliance**:

    Ensure that your data collection and monitoring system complies with relevant regulations and privacy laws, especially when operating in public spaces.

14. **Documentation**:

    Maintain comprehensive documentation of the sensor deployment, sensor locations, data transmission methods, and any relevant system configurations. This documentation aids in troubleshooting and future system upgrades.

Deploying IoT sensors in public water fountains offers multiple benefits, including water conservation, early malfunction detection, and efficient maintenance. A well-planned and executed deployment can significantly improve the management of these public resources.

Develop a Python script on the IoT sensors to send real-time water fountain status data to the platform.

PROGRAM:

```
import paho.mqtt.client as mqtt

import time

import json

import random


# Define your MQTT broker information

mqtt_broker = "127.0.0.1"

mqtt_port = 8883

mqtt_topic = "Water-Fountain-Status"


# Create a unique identifier for your sensor

sensor_id = "water-fountain-sensor-1"


# Initialize the MQTT client

client = mqtt.Client(sensor_id)


# Set up a username and password if required

# client.username_pw_set(username="hivemq.webclient.1697645821355",
password="3vs2CQAc9ML4.&@Gb*ek")
```

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

try:
    while True:
        # Simulate sensor data (replace this with actual sensor data)
        flow_rate = random.uniform(0.5, 2.0)  # Sample flow rate data
        pressure = random.uniform(20, 80)  # Sample pressure data

        # Create a JSON message with sensor data
        sensor_data = {
            "flow_rate": flow_rate,
            "pressure": pressure,
            "timestamp": time.time()
        }

        # Convert the data to JSON format
        json_data = json.dumps(sensor_data)

        # Publish the data to the MQTT topic
        client.publish(mqtt_topic, json_data)
        # Print the sent data for debugging (you can remove this in production)
        print(f"Published data: {json_data}")

        # Wait for a specific interval before sending the next data
        time.sleep(10)  # Adjust the interval as needed

except KeyboardInterrupt:
    client.disconnect()
    print("Disconnected from the MQTT broker")
```

THANK YOU