SMART WATER MANAGEMENT

PHASE 3 SUBMISSION

Configuring an IoT sensor for measuring water consumption in public places

1. Select the IoT Sensor:

Choose an appropriate water flow sensor that can accurately measure water consumption. Make sure it is compatible with IoT connectivity options like Wi-Fi, LoRa, NB-IoT, or Sigfox.

2. Data Collection and Transmission:

a. Install the sensor in the water supply line of the public place.

b. Configure the sensor to collect and transmit data at regular intervals. Ensure it has a power source, whether it's battery-powered or connected to a power supply.

3. Connectivity:

Choose the appropriate connectivity method based on your infrastructure and location. Wi-Fi is suitable for indoor settings, while LPWAN (Low Power Wide Area Network) technologies like LoRa or NB-IoT are better for outdoor deployments.

4. Data Storage:

Set up a cloud-based database or server to store the sensor data securely. You can use platforms like AWS, Azure, Google Cloud, or dedicated IoT platforms for this purpose.

5. Data Processing and Analysis:

Implement data processing algorithms to clean and analyze the data. This can involve checking for anomalies, aggregating data, and calculating water consumption.

6. User Interface:

Develop a user-friendly interface for administrators to access and monitor the water consumption data. This could be a web portal or a mobile app.

7. Alerts and Notifications:

Configure the system to send alerts or notifications in real-time when unusual water consumption patterns are detected. This can help in identifying leaks or wastage.

8. Integration:

Integrate the IoT system with existing water management systems, if applicable. This ensures that the data can be used effectively for decision-making.

9. Security:

Implement robust security measures to protect the data and the IoT infrastructure from potential threats or breaches.

10. Maintenance and Calibration:

Regularly maintain and calibrate the IoT sensors to ensure their accuracy. This includes changing batteries, cleaning sensors, and verifying data quality.

11. Data Reporting:

Generate reports and insights from the collected data to make informed decisions about water management in public places.

12. Compliance:

Ensure that your IoT system complies with local regulations and privacy standards, especially when dealing with public data.

13. Scaling:

If you plan to deploy these sensors in multiple public places, consider how to efficiently scale the system while maintaining data integrity and security.

Remember that successful implementation of an IoT sensor network for water consumption monitoring involves a combination of hardware, software, and data analysis expertise. It's essential to work with a multidisciplinary team to make this project a success.

Developing a Python script for an IoT sensor to send real-time water consumption data to a data sharing platform:

```
import paho.mqtt.client as mqtt

import time

import random

# MQTT Broker Details

broker_address = "mqtt.example.com"

port = 1883

topic = "water_consumption_data"

# Simulated water consumption sensor (replace with actual sensor logic)

def read_water_consumption():

    return random.uniform(0.1, 2.0)  # Simulated water consumption in gallons

# Create an MQTT client

client = mqtt.Client("WaterConsumptionSensor")

# Connect to the MQTT broker

client.connect(broker_address, port)

try:

    while True:

        # Read water consumption data from the sensor

        water_consumption = read_water_consumption()

       # Create a JSON payload (you can adjust this structure)
```

```
    payload = {
        "timestamp": int(time.time()),
        "water_consumption": water_consumption
      }
     # Publish the data to the MQTT topic
      client.publish(topic, str(payload))
      print("Published: ", payload)
      time.sleep(60)  # Adjust the time interval as needed
    except KeyboardInterrupt:
   client.disconnect()
```

This script does the following:

1. Imports the necessary libraries, including `paho-mqtt` for MQTT communication.

2. Defines the MQTT broker details, including the broker's address, port, and topic where data will be published. Replace these details with your own.

3. Simulates water consumption data with a `read_water_consumption` function. In a real application, replace this with actual sensor data retrieval logic.

4. Initializes an MQTT client and connects to the MQTT broker.

5. In a loop, it reads water consumption data, creates a JSON payload with a timestamp, and publishes it to the MQTT topic.

6. The script continues to run until you manually interrupt it (e.g., by pressing Ctrl+C).

Remember to replace the simulated data with actual sensor data and configure the MQTT broker details according to your specific data sharing platform.