# IOT_PHASE-03

# SMART WATER FOUNTAINS

REG NO:610821106309

NAME: PUSHPARAJ.J

## IoT sensors in public water fountains to monitor water flow and detect malfunctions.

1. **Define Objectives**:
    - Clearly define the goals of the project, such as improving water fountain efficiency, reducing water wastage, and ensuring timely maintenance.
2. **Sensor Selection**:
    - Choose appropriate IoT sensors for the project. In this case, flow rate sensors and pressure sensors are essential. You might also consider water quality sensors to detect contamination.
3. **Connectivity**:
    - Ensure that the sensors have connectivity capabilities (e.g., Wi-Fi, LoRa, cellular) to transmit data to a central monitoring system.
4. **Power Source**:
    - Decide on the power source for the sensors. Options include batterypowered sensors, solar panels, or power from the fountain itself.
5. **Sensor Placement**:

- Strategically place the sensors within the fountain. Flow rate sensors should be positioned to measure water inflow and outflow, while pressure sensors should be placed to monitor system pressure.

6. **Data Management**:
    - Set up a data management system to collect, store, and analyze sensor data. Cloud-based solutions are common for IoT projects.
7. **Real-time Monitoring**:
    - Implement a real-time monitoring system that allows you to track the performance of water fountains remotely. This can be achieved through a web-based dashboard or a mobile app.
8. **Malfunction Detection**:
    - Define criteria for detecting malfunctions, such as significant changes in water flow or pressure. Set up alerts or automated actions when malfunctions are detected.
9. **Data Analysis**:
    - Analyze historical data to identify patterns and make informed decisions regarding maintenance and improvements.
10. **Maintenance Scheduling**:
    - Create a maintenance schedule based on data analysis to ensure the water fountains are serviced regularly and efficiently.

## Python script that simulates sending water fountain status data to an MQTT broker

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


# MQTT broker details broker_address =
"your.mqtt.broker.address" port = 1883
username = "your_username" password
= "your_password"

# MQTT topics topic =
"water_fountain/status"


# Function to simulate water fountain status data
def get_water_fountain_status():
   # Replace this with your actual sensor data retrieval logic
return random.choice  (["ON", "OFF"])
# MQTT callback functions def
on_connect(client, userdata, flags, rc):
   if rc == 0:
      print("Connected to MQTT broker")
   else:
      print("Connection to MQTT broker failed with code: " + str(rc))
def on_publish(client, userdata, mid):
   print("Data published to MQTT broker")

# Create an MQTT client
```

```
client = mqtt.Client()
client.username_pw_set(username, password)

# Set the callback functions client.on_connect
= on_connect client.on_publish = on_publish

# Connect to the MQTT broker
client.connect(broker_address, port, keepalive=60)

# Start the MQTT loop client.loop_start()

try:    while
True:
      water_fountain_status = get_water_fountain_status()       #
Publish the status data to the MQTT broker   client.publish(topic,
water_fountain_status)       print(f"Published status:
{water_fountain_status} to topic: {topic}")       time.sleep(5)  # Adjust
the interval as needed except KeyboardInterrupt:    print("Script
terminated")

# Disconnect from the MQTT broker
client.disconnect()
```