# Overview of Smart Water Fountains

## Introduction:

Start building the IoT-enabled Smart Water Fountains system. Deploy IoT sensors (e.g., flow rate sensors, pressure sensors) in public water fountains to monitor water flow and detect malfunctions. Develop a Python script on the IoT sensors to send real-time water fountain status data to the platform.

## Define Requirements:

Clearly define the requirements for your Smart Water Fountains system, such as the number of fountains, their location, the information you want to gather, and the features you want to implement (e.g., water quality monitoring, remote control).

## Hardware Selection:

Choose IoT hardware components such as microcontrollers (e.g., Arduino, Raspberry Pi), sensors (e.g., water quality sensors, flow sensors), and connectivity modules (e.g., Wi-Fi, LoRa, GSM).

## Design the System:

Create a system architecture that includes the fountains, sensors, data processing, and user interfaces.

## Sensor Integration:

Connect and integrate sensors to measure parameters like water level, quality, and temperature.

## Microcontroller Programming:

Program the microcontrollers to collect data from sensors and transmit it to a central server or cloud platform.

*1.Flow Rate Sensors:*

These sensors can measure the rate of water flow, helping detect unusual variations or leaks.

*2.Pressure Sensors:*

Pressure sensors can monitor the water pressure in the fountain, ensuring it’s within safe and operational limits.

*3.Leak Detection:*

Sudden drops in flow rate or pressure can indicate leaks, allowing for timely repairs to conserve water.

*4.Remote Monitoring:*

IoT technology enables remote monitoring, alerting maintenance teams to issues and reducing response times.

*5.Data Analytics:*

Accumulated data can be analyzed to predict maintenance needs and optimize water fountain operation.

By implementing such a system, municipalities can save water, reduce maintenance costs, and enhance public infrastructure management.

## Install Necessary Libraries:

Install Python libraries that are required for your selected hardware. For example, if you’re using a Raspberry Pi, you might need RPi.GPIO for GPIO control.

*Python*

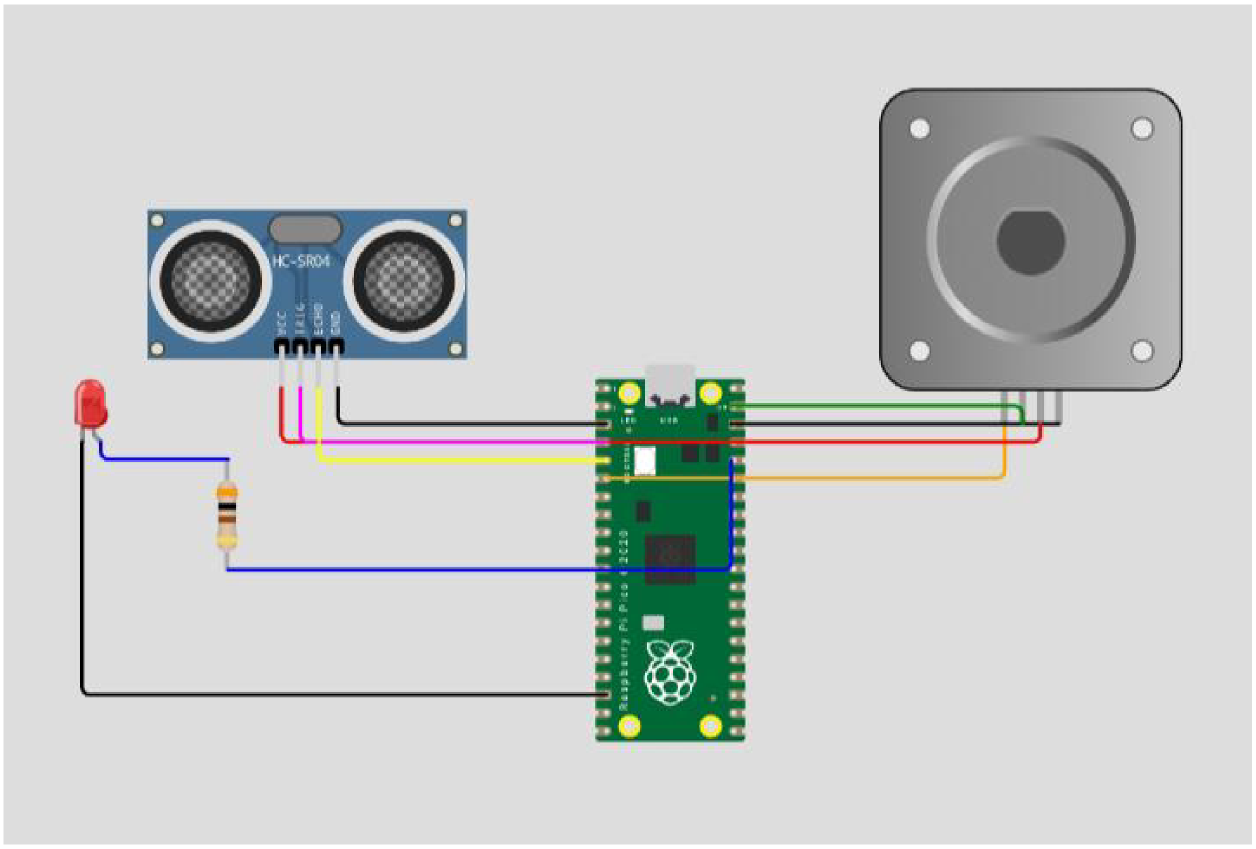
```
Import RPi.GPIO as GPIO

Import time

From AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient


# Set up AWS IoT

myMQTTClient = AWSIoTMQTTClient(“myClient”)
```

```
myMQTTClient.configureEndpoint("your-iot-endpoint.amazonaws.com", 8883)
myMQTTClient.configureCredentials("root-CA.crt", "private-key.pem", "cert.pem")

# Set up GPIO pin for your water level sensor
Water_level_pin = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(water_level_pin, GPIO.IN)

# Function to publish data to AWS IoT
Def publish_data():
  Water_level = GPIO.input(water_level_pin)
  Message = {
    "water_level": water_level
  }
  myMQTTClient.publish("water-fountain-status", json.dumps(message), 1)

# Connect to AWS IoT
myMQTTClient.connect()

# Main loop to continuously send data
Try:
  While True:
    Publish_data()
    Time.sleep(10)  # Send data every 10 seconds

Except KeyboardInterrupt:
  Pass

# Clean up GPIO and disconnect from AWS IoT
```

GPIO.cleanup()

myMQTTClient.disconnect()

Run the Script: Run the Python script on your IoT device. It will read the water level sensor and send the data to the IoT platform periodically.



SUMMITED BY,

BALAGAJARAJ P