Phase 3 project:

Project Title: SMART WATER FOUNTAIN

College Code: 6208

College : Gnanamani College of Technology

Branch: B.E-BIO MEDICAL ENGINEERING

Year: IIIrd year

Team members:

- SIBIRAJ.S (620821121108)
- MOULI.R (620821121069)
- YUGENTHIRAN.S (620821121133)
- VISHNU.S (620821121130)
- PRAVEEN KUMAR.S (620821121085)

- MALLIAPPAN.M (620821121071)

SMART WATER FOUNTAIN

Definition:

A Smart Water Foundation project using IoT refers to an initiative that employs Internet of Things (IoT) technology to enhance the management, conservation, and sustainability of water resources. It involves the deployment of IoT devices equipped with sensors and communication capabilities to collect real-time data on various aspects of water systems, such as quality, quantity, and infrastructure conditions. This data is then processed and analyzed to optimize water usage, improve efficiency, detect issues like leaks or contamination, and promote responsible water management practices. These projects aim to leverage IoT technology to address water-related challenges and contribute to more intelligent, efficient, and sustainable water management.

Components Needed:

1.8051 Microcontroller

2.IoT Module – ESP32

3.Sensors – Ultrasonic Sensors, pH Sensors, Turbidity Sensors, Flow Sensors

4.Actuators

5.Power supply –Solar

6.Communication – MQTT,HTTP

7.IoT platform –Azure IoT, Google Cloud IoT

8.Data storage

PHASE 3

Arduino Code for the IoT Sensor (ESP32/ESP8266):

You'll need to program the ESP32/ESP8266 to read data from the sensors and send it to an MQTT server. Below is an example code snippet using the Arduino IDE and the PubSubClient library to publish data to an MQTT topic

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "your_wifi_ssid";
const char* password = "your_wifi_password";
const char* mqtt_server = "mqtt.yourserver.com"; // Replace with your MQTT broker's address
const int mqtt_port = 1883; // MQTT port

const char* flow_rate_sensor_topic = "water_fountain/flow_rate";
const char* pressure_sensor_topic = "water_fountain/pressure";

WiFiClient espClient;
PubSubClient client(espClient);

float flow_rate_value = 0.0; // Replace with actual flow rate data
float pressure_value = 0.0; // Replace with actual pressure data

void setup() {
```

```
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  client.setServer(mqtt_server, mqtt_port);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  // Simulate sensor readings (replace with actual sensor data)
  flow_rate_value = random(10, 50); // Example flow rate value (adjust as needed)
  pressure_value = random(30, 100); // Example pressure value (adjust as needed)

  // Publish sensor data to MQTT topics
  client.publish(flow_rate_sensor_topic, String(flow_rate_value).c_str());
  client.publish(pressure_sensor_topic, String(pressure_value).c_str());

  delay(60000); // Adjust the delay as needed (60,000 milliseconds = 1 minute)
}
```

```
void reconnect() {
  while (!client.connected()) {
    if (client.connect("water_fountain_sensor")) {
      Serial.println("Connected to MQTT broker");
    } else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" - Retrying in 5 seconds");
      delay(5000);
    }
  }
}
```

This code configures the ESP32/ESP8266 to connect to your Wi-Fi network and the MQTT broker, publish flow rate and pressure data to MQTT topics, and reconnect in case of a connection loss. You should replace the simulated sensor data with actual sensor readings.

Python Script for MQTT Data Processing:

To receive and process the data published by the IoT sensor, you can use a Python script with the Paho MQTT library. Here's a simple example:

```
import paho.mqtt.client as mqtt


def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
```

```
    client.subscribe("water_fountain/#")  # Subscribe to all topics under "water_fountain/"

def on_message(client, userdata, msg):
    topic = msg.topic
    message = msg.payload.decode()
    print(f"Received data on topic '{topic}': {message}")
    # Implement your data processing logic here

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect("mqtt.yourserver.com", 1883, 60)  # Replace with your MQTT broker's address
client.loop_forever()
```

SUBMITTED BY: PRAVEEN KUMAR. S