

1) Configure IOT sensors :

- I) Magnetic Flow Sensor – For analysing water consumption.
- II) pH Sensor – For analysing water contamination level and quality.

I) Configuring Magnetic Flow Sensor :

- a) Get a Magnetic Flow Sensor.
- b) Configure the flow sensor based on our need.

Input:

```
const int flowMeterPin = 2; // Connect the flow meter's output to digital pin
2
unsigned long pulseCount = 0;
float flowRate = 0.0;
float totalFlow = 0.0;
unsigned long previousMillis = 0;
const unsigned long interval = 1000; // Update interval in milliseconds

void setup() {
    pinMode(flowMeterPin, INPUT_PULLUP);
    Serial.begin(9600);
}

void loop() {
    unsigned long currentMillis = millis();

    if (digitalRead(flowMeterPin) == LOW) {
        pulseCount++;
    }

    if (currentMillis - previousMillis >= interval) {
        // Calculate flow rate and total flow
    }
}
```

```

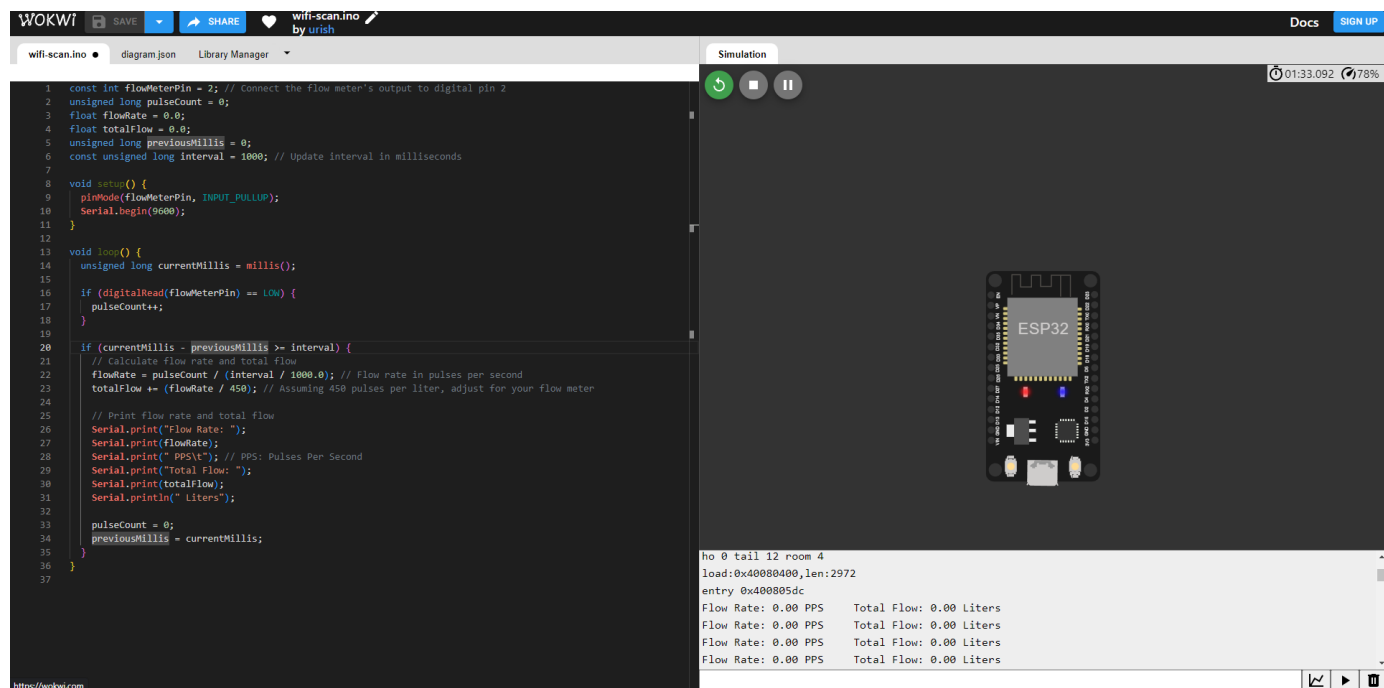
    flowRate = pulseCount / (interval / 1000.0); // Flow rate in pulses per
second
    totalFlow += (flowRate / 450); // Assuming 450 pulses per liter, adjust
for your flow meter

    // Print flow rate and total flow
    Serial.print("Flow Rate: ");
    Serial.print(flowRate);
    Serial.print(" PPS\t"); // PPS: Pulses Per Second
    Serial.print("Total Flow: ");
    Serial.print(totalFlow);
    Serial.println(" Liters");

    pulseCount = 0;
    previousMillis = currentMillis;
}
}

```

Output:



The screenshot shows the Wokwi IDE interface. On the left, the code editor displays the following C++ code:

```

1 const int flowMeterPin = 2; // Connect the flow meter's output to digital pin 2
2 unsigned long pulseCount = 0;
3 float flowRate = 0.0;
4 float totalFlow = 0.0;
5 unsigned long previousMillis = 0;
6 const unsigned long interval = 1000; // Update Interval in milliseconds
7
8 void setup() {
9   pinMode(flowMeterPin, INPUT_PULLUP);
10  Serial.begin(9600);
11 }
12
13 void loop() {
14   unsigned long currentMillis = millis();
15
16   if (digitalRead(flowMeterPin) == LOW) {
17     pulseCount++;
18   }
19
20   if (currentMillis - previousMillis >= interval) {
21     // Calculate flow rate and total flow
22     flowRate = pulseCount / (interval / 1000.0); // Flow rate in pulses per second
23     totalFlow += (flowRate / 450); // Assuming 450 pulses per liter, adjust for your flow meter
24
25     // Print flow rate and total flow
26     Serial.print("Flow Rate: ");
27     Serial.print(flowRate);
28     Serial.print(" PPS\t"); // PPS: Pulses Per Second
29     Serial.print("Total Flow: ");
30     Serial.print(totalFlow);
31     Serial.println(" Liters");
32
33     pulseCount = 0;
34     previousMillis = currentMillis;
35   }
36 }
37

```

On the right, the simulation window shows an ESP32 microcontroller board. Below the board, the output window displays the following text:

```

ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
Flow Rate: 0.00 PPS    Total Flow: 0.00 Liters
Flow Rate: 0.00 PPS    Total Flow: 0.00 Liters
Flow Rate: 0.00 PPS    Total Flow: 0.00 Liters
Flow Rate: 0.00 PPS    Total Flow: 0.00 Liters

```

II) Configuring pH Sensor :

- a) Get a pH Sensor.
- b) Configure the pH sensor based on our need.

Input:

```
const int sensorPin = A0; // Analog input pin for pH sensor
const float offsetVoltage = 0.25; // Offset voltage of the pH sensor
const float calibrationValue = 6.86; // pH calibration value for your specific
sensor

void setup() {
    Serial.begin(9600);
}

void loop() {
    int rawValue = analogRead(sensorPin);
    float voltage = (rawValue / 1023.0) * 5.0; // Convert the analog reading to
voltage

    float pHValue = calibrationValue + ((voltage - offsetVoltage) * (-5.0 /
0.18)); // Calculate pH value

    Serial.print("pH Value: ");
    Serial.println(pHValue, 2); // Print pH value with 2 decimal places

    delay(1000); // Adjust the update interval as needed
}
```

Output:

The screenshot displays the Wokwi online IDE interface. The left pane contains the following C++ code:

```
1 const int sensorPin = A0; // Analog input pin for pH sensor
2 const float offsetVoltage = 0.25; // Offset voltage of the pH sensor
3 const float calibrationValue = 6.86; // pH calibration value for your specific sensor
4
5 void setup() {
6   Serial.begin(9600);
7 }
8
9 void loop() {
10  int rawValue = analogRead(sensorPin);
11  float voltage = (rawValue / 1023.0) * 5.0; // Convert the analog reading to voltage
12
13  float pHValue = calibrationValue + ((voltage - offsetVoltage) * (-5.0 / 0.18)); // Calculate pH value
14
15  Serial.print("pH Value: ");
16  Serial.println(pHValue, 2); // Print pH value with 2 decimal places
17
18  delay(1000); // Adjust the update interval as needed
19 }
20
```

The right pane shows a simulation of an ESP32 microcontroller board. Below the simulation, the serial output is displayed:

```
entry 0x400005dc
pH Value: -42.00
pH Value: -36.70
pH Value: -76.75
pH Value: -79.74
pH Value: -86.26
pH Value: -109.34
```

2) Python script on the IOT sensors to send real-time water consumption data to the data-sharing platform:

Steps:

1. IOT Device: This could be a microcontroller (e.g., Raspberry Pi, Arduino) with a compatible water consumption sensor (e.g., flow sensor).
2. Internet Connectivity: Ensure your IOT device can connect to the internet, either through Wi-Fi, Ethernet, or a cellular module.
3. Data-Sharing Platform: Choose a platform where you want to send the data. Common choices include cloud-based services like AWS, Azure, Google Cloud, or dedicated IOT platforms.
4. Libraries: You may need specific libraries or SDKs to communicate with your chosen data-sharing platform and sensor.

Code:

```
1 import paho.mqtt.client as mqtt
2 import time
3 from random import randint # Simulate water consumption data
4
5 # Define your MQTT broker and topic information
6 broker_address = "mqtt.eclipse.org" # Change to your MQTT broker
7 topic = "water_consumption"
8
9 # Simulate water consumption data (replace this with actual sensor data)
10 def get_water_consumption_data():
11     return randint(1, 10) # Random value between 1 and 10 liters
12
13 # Callback when the client connects to the broker
14 def on_connect(client, userdata, flags, rc):
15     print("Connected with result code " + str(rc))
16     client.subscribe(topic)
17
18 # Initialize the MQTT client
19 client = mqtt.Client()
20 client.on_connect = on_connect
21
22 # Connect to the MQTT broker
23 client.connect(broker_address, 1883, 60)
24
25 try:
26     while True:
27         water_consumption = get_water_consumption_data()
28         print("Water Consumption: {} liters".format(water_consumption))
29
30         # Publish water consumption data to the MQTT topic
31         client.publish(topic, str(water_consumption))
32         time.sleep(60) # Adjust the update interval as needed
33
34 except KeyboardInterrupt:
35     print("Script terminated.")
36
37 client.disconnect()
38
```

Conclusion:

*In this we have developed a program for configuring the IOT sensors like Magnetic Flow Sensor and pH Sensor for checking water consumption and checking the contamination in the water by analysing the pH value.

*Then we have developed a program for sending real-time water consumption data to the data-sharing platform using MQTT services.