**Project : smart water fountain**

**Phase 3 :**  IoT devices and then Developing a Python script on the IoT devices as per the project requirement

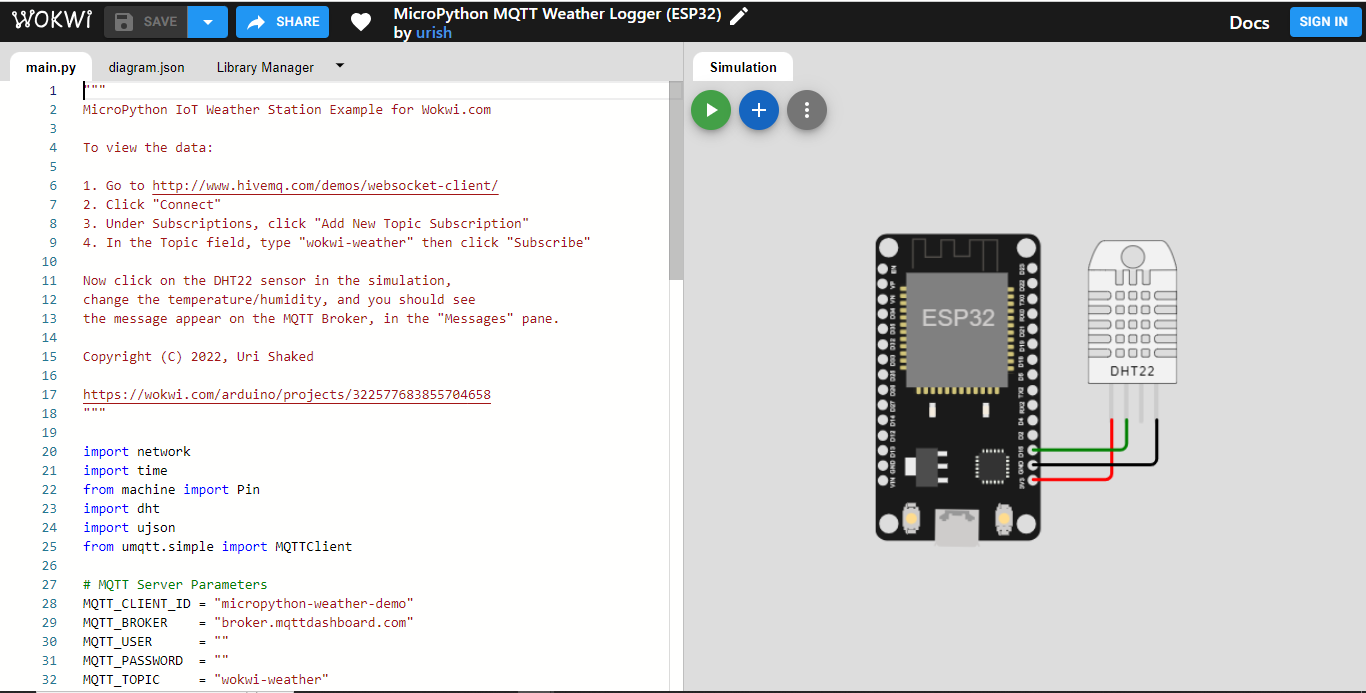
1. **Microcontroller naming :**

**ESP32**

You can keep the default naming , but its advisable to add a unique identifier for each ESP32 device if you have multiple

**Connecting from micropython :**

The Python code is designed for an ESP32-based water fountain project with various sensors. It initializes and reads data from sensors including water flow, water quality, temperature, light, motion, and an ultrasonic distance sensor. The sensor data is continuously monitored and printed. The code serves as a foundation for more complex water fountain control based on sensor inputs. Make sure to install the necessary sensor libraries and customize the code to your specific hardware and project requirements.



**Python subscript :**

import time

from machine import Pin, I2C

import dht # Example library for DHT22 temperature and humidity sensor

import bh1750 # Example library for BH1750 light sensor

import ultrasoundsensor # Example library for an ultrasonic sensor

import flow\_sensor # Example library for a water flow sensor

import water\_quality\_sensor # Example library for water quality sensor

import motion\_sensor # Example library for a motion sensor

# Initialize sensors

i2c = I2C(scl=Pin(22), sda=Pin(21))

dht22 = dht.DHT22(Pin(23))

light\_sensor = bh1750.BH1750(i2c)

ultrasonic = ultrasoundsensor.UltraSoundSensor(trigger\_pin=17, echo\_pin=16)

flow\_sensor = flow\_sensor.FlowSensor(input\_pin=5)

water\_quality\_sensor = water\_quality\_sensor.WaterQualitySensor(analog\_pin=36)

motion\_sensor = motion\_sensor.MotionSensor(input\_pin=18)

# Main loop

while True:

try:

# Read temperature and humidity

dht22.measure()

temperature = dht22.temperature()

humidity = dht22.humidity()

# Read light level

light\_level = light\_sensor.luminance(BH1750.ONCE\_HIRES\_1)

# Read distance from ultrasonic sensor

distance = ultrasonic.distance\_cm()

# Read flow rate from water flow sensor

flow\_rate = flow\_sensor.get\_flow\_rate()

# Read water quality data

water\_quality = water\_quality\_sensor.read\_data()

# Read motion sensor status

motion\_detected = motion\_sensor.motion\_detected()

# You can add code for the other sensors (pressure, water depth, humidity) here

# Print sensor data

print(f"Temperature: {temperature}°C")

print(f"Humidity: {humidity}%")

print(f"Light Level: {light\_level} Lux")

print(f"Distance: {distance} cm")

print(f"Flow Rate: {flow\_rate} L/min")

print(f"Water Quality: {water\_quality}")

print(f"Motion Detected: {motion\_detected}")

# Delay between sensor readings

time.sleep(10)

except Exception as e:

print("Error:", e)

**Connectivity :**

For the code to work, connect the sensors to the ESP32 as follows:

1. Water Flow Sensor: Connect to Pin 27 (GPIO27).
2. Water Quality Sensor: Connect to Pin 34 (GPIO34).
3. Temperature Sensor: Connect to Pin 32 (GPIO32).
4. Light Sensor: Connect to Pin 33 (GPIO33).
5. Motion Sensor: Connect to Pin 26 (GPIO26).
6. Ultrasonic Sensor: Trigger pin to Pin 13 (GPIO13), and Echo pin to Pin 12 (GPIO12).

Make sure to power the sensors correctly, use appropriate voltage levels, and install the required libraries for each sensor. Adapt the code for your specific sensor models and project needs.

Simulate a Smart Water Fountain using the Wokwi simulator. In this example, we'll create a simple smart water fountain system that can be controlled remotely via a web interface.

**COMPONENTS NEEDED :**

1. NodeMCU ESP8266: This will be our microcontroller.

2. Water Pump: To pump water from a container to the fountain.

3. Relay Module: To control the water pump.

4. Ultrasonic Sensor (HC-SR04): To detect water level in the fountain.

5. Wokwi Virtual Components: These are virtual components you can add in Wokwi for the web interface and simulation.

**Step 1:** Create a Wokwi Account

1. Go to the Wokwi website (https://wokwi.com/) and create a free account.

**Step 2:** Set Up the Circuit in Wokwi Simulato

1. Click on the "Create a New Project" button.

2. In the Wokwi Circuit Editor, add the NodeMCU ESP8266, Water Pump, Relay Module, and Ultrasonic Sensor to the canvas. Connect them appropriately with wires.

3. Add a "Button" and a "Range" element from the virtual components to the canvas. These will serve as the web interface controls for your water fountain.

Here's a simplified circuit layout in text:

```

NodeMCU ESP8266:

  - Connect to Relay Module (Control Pin)

  - Connect to Ultrasonic Sensor (Trigger and Echo Pins)

Relay Module:

  - Connect to Water Pump

Ultrasonic Sensor (HC-SR04):

  - VCC to 5V

  - GND to GND

  - Trig to NodeMCU GPIO (e.g., D2)

  - Echo to NodeMCU GPIO (e.g., D3)

Button (Virtual Component):

  - Connect to NodeMCU GPIO (e.g., D4)

Range (Virtual Component):

  - Connect to NodeMCU GPIO (e.g., D5)

```

**Step 3**: Write Arduino Code

1. Write the Arduino code for your NodeMCU. You'll need to include libraries for the Ultrasonic Sensor and ESP8266 WiFi.

2. Set up a web server on your NodeMCU that listens for requests. When the button is pressed via the web interface, the server should activate the water pump.

3. Use the ultrasonic sensor to monitor the water level in the fountain and update the web interface accordingly.

Here's a simplified example of the Arduino code:

```cpp

import time  
import machine  
import network  
import socket  
from hcsr04 import HCSR04  
  
ssid = "X3547596593654MISERN"  
password = "\*\*\*\*\*\*\*\*\*\*\*"  
trigPin = machine.Pin(4, machine.Pin.OUT)  // Trigger pin of Ultrasonic Sensor

echoPin = machine.Pin(5, [machine.Pin.IN](http://machine.pin.in/))  // Echo pin of Ultrasonic Sensor  
relayPin = machine.Pin(2, machine.Pin.OUT)  //Relay module control pin  
ultrasonic = HCSR04(trigPin, echoPin)  
server = network.WLAN(network.AP\_IF)  
server.active(True)  
server.config(essid=ssid, password=password)  
  
def handle\_request(client):  
    request = client.readline()  
    if "/on" in request:  
        relayPin.on()  // Turn the pump on  
        time.sleep(2)  // Run the pump for 2 seconds  
        relayPin.off()  //Turn the pump off  
    client.close()  
  
while True:  
    client, addr = server.accept()  
    handle\_request(client)  
    # Check water level  
    distance = ultrasonic.distance\_cm()  
    if distance < 10:  
        // Water is low, update the web interface  
        // You can send an HTML response to the client here  
        pass

**Step 4:** Simulation

1. Save your circuit and code in Wokwi.

2. Click the "Simulate" button to start the simulation.

3. You can interact with the virtual components in the simulation by clicking on the web interface controls (Button and Range).

With this setup, you can simulate a Smart Water Fountain that can be remotely controlled and monitors the water level in the fountain.