Smart water Fountain

**Procedure:**

1. \*\*Design Your Circuit:\*\*

- Start by designing the circuit for your smart fountain. This circuit may include components like water level sensors, pumps, LEDs, and an Arduino board.

- You can use Wokwi's Circuit Simulator to create your circuit visually. Add components from the component library, connect them, and set up the connections as needed.

2. \*\*Select Sensors:\*\*

- Choose the appropriate water level sensors for your fountain. Common options include ultrasonic sensors, capacitive sensors, or float switches. You can find these sensors in Wokwi's component library.

3. \*\*Write Arduino Code:\*\*

- Open the Arduino IDE and write the code to control your smart fountain. This code will read data from the water level sensors and control the pump and other components accordingly.

- Simulate the code using Wokwi by uploading it to your virtual Arduino board.

4. \*\*Simulate Your Project:\*\*

- In Wokwi, you can simulate your project by running the circuit with the Arduino code you've written. Test various scenarios, such as different water levels, to ensure your fountain functions as expected.

5. \*\*Debug and Refine:\*\*

- If you encounter any issues, debug your code and circuit within the simulation environment. Make necessary adjustments to your code and circuit design as required.

6. \*\*Visualize the Fountain:\*\*

- Use LEDs or other visual components in your circuit to simulate the water flow and fountain lighting. Adjust the code to control these components based on the water level and other conditions.

7. \*\*Monitor and Control:\*\*

- Add a user interface to your application for monitoring and controlling the fountain. You can do this by integrating Bluetooth, Wi-Fi, or other communication modules into your Arduino project, and then simulate the control interface in Wokwi.

8. \*\*Test and Optimize:\*\*

- Continuously test and optimize your smart fountain project within the simulation environment to ensure it operates smoothly and efficiently.

9. \*\*Documentation and Sharing:\*\*

- Document your project and share it with the Wokwi community or your friends to get feedback and showcase your work.

Remember that Wokwi is a great platform for prototyping and simulating your electronic projects. Once your simulation is successful, you can consider implementing the physical version of your smart fountain using real components and sensors.

Be sure to consult the documentation and tutorials available on Wokwi for more specific guidance on using the platform and simulating your projects.

# **Sensors**

We have used the following sensors for our project:

Temperature

Humidity

Water Flow

Water Meter

Water Level

Leak Detection

# **Hardware**

**Microcontrollers:**

They are used to interface with the IoT sensors, process data, and facilitate communication with the data-sharing platform.

**Used:** Arduino,Esp32

**Connectivity:**

**Wi-Fi Module**: Use Wi-Fi for data transmission.

## **Their uses**

**Temperature sensor:**

It is used to measure the temperature of water in real-time. They are useful in detecting changes in temperature that could indicate a problem with the water supply or distribution system.

**Humidity sensor:**

It is used to measure the amount of moisture in the air. They can be used to detect leaks or other issues that could lead to mold growth or other problems.

**Water flow sensor:**

It is used to measure the rate of water flow in pipes and other water distribution systems. They can be used to detect leaks, monitor water usage, and optimize water distribution.

**Water meter:**

It is used to measure the amount of water consumed by households or businesses. They can be used to detect leaks, monitor water usage, and optimize water distribution.

**Water level sensor:**

It is used to measure the level of water in tanks, reservoirs, and other storage containers. They can be used to detect leaks, monitor water usage, and optimize water distribution.

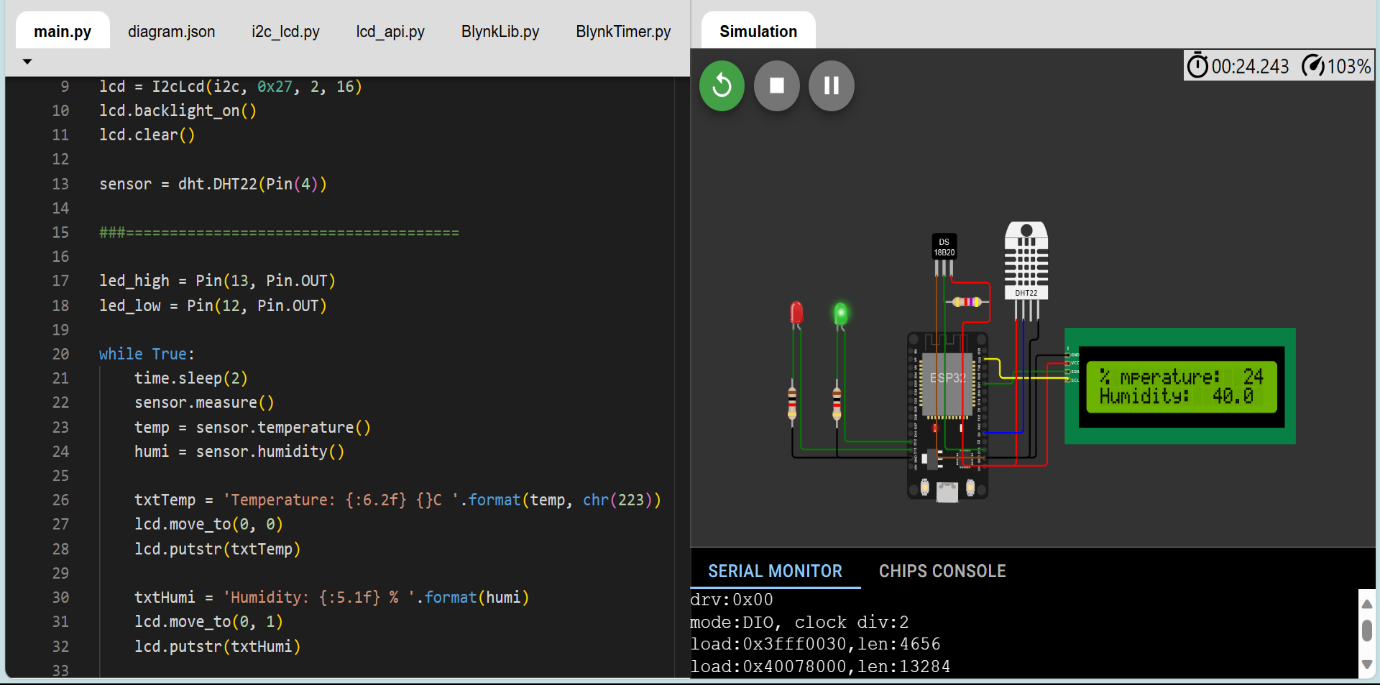
**Leak detection sensor:**

It is used to detect leaks in pipes and other water distribution systems. They can be used to prevent water loss and damage from leaks.

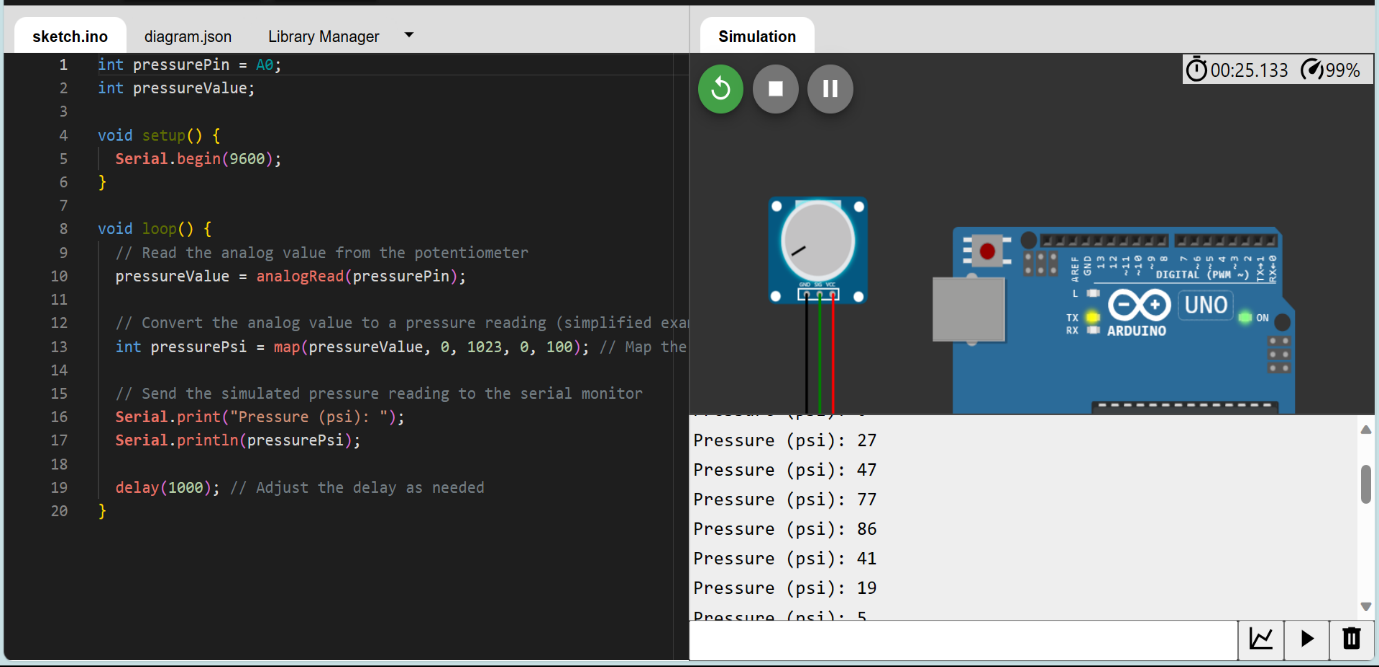
Configured Sensors:

Screenshots:

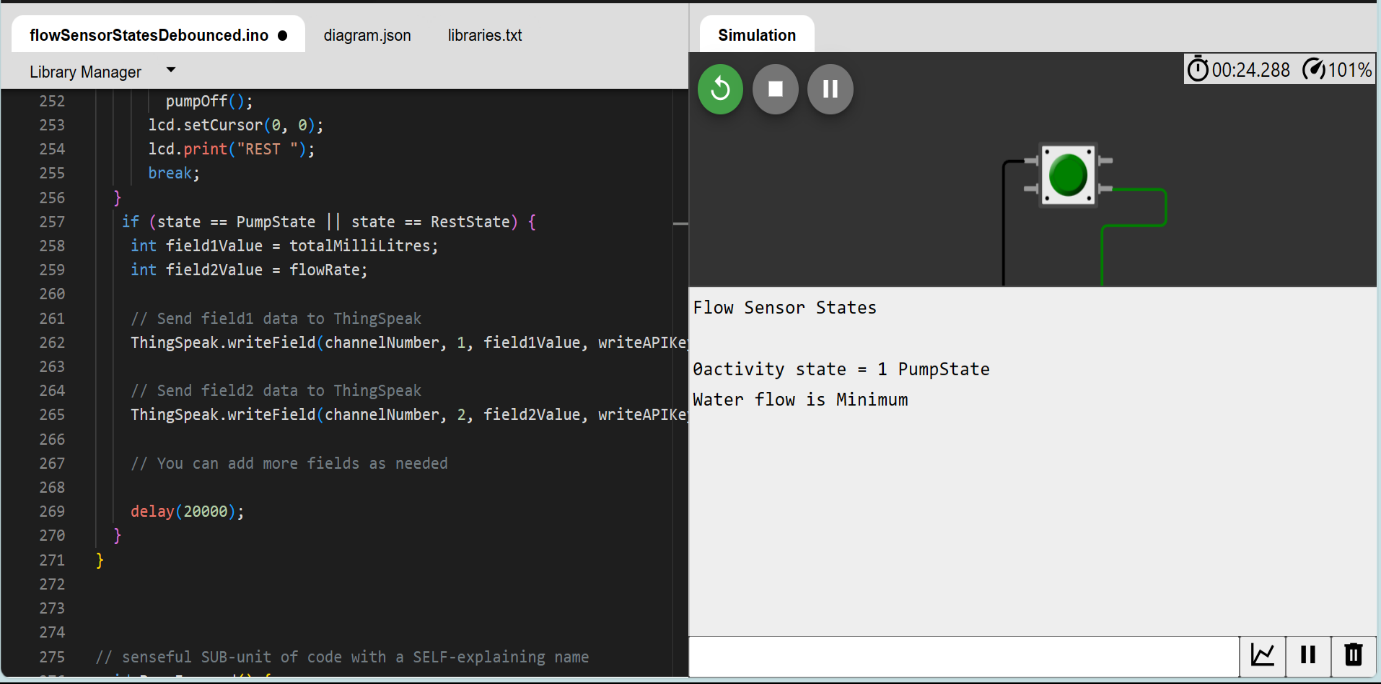
Temperature and Humidity



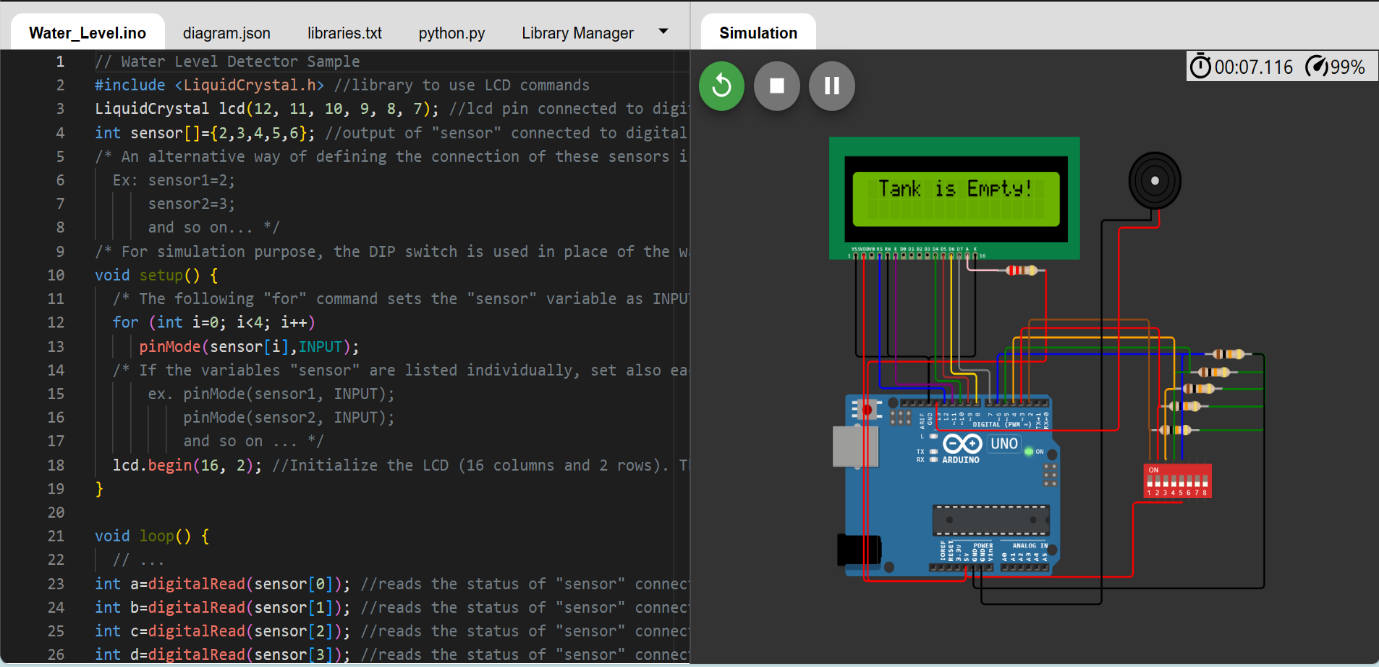
Pressure



Water Flow



Water Level



Codes used available here:

https://drive.google.com/drive/folders/1O8PMTsy5sHP-eAH0xma4c6pbRRohEGnz?usp=share\_link

# Script for data sharing

Python script for sharing data from Temperature and Humidity Sensor

import requests

import time

import json

thingspeak\_url = "https://api.thingspeak.com/update"

api\_key = "6EV4VJEM23TR6EO2"

ssid = "Wokwi-GUEST"

password = ""

DHT\_PIN = 15

TRIG\_PIN = 13

ECHO\_PIN = 12

def get\_distance():

  from machine import Pin

  import dht

  dht\_sensor = dht.DHT22(Pin(DHT\_PIN))

  while True:

    try:

        dht\_sensor.measure()

        temperature = dht\_sensor.temperature()

        humidity = dht\_sensor.humidity()

        distance = get\_distance()

        print("Temperature: {:.2f}°C, Humidity: {:.2f}%, Distance: {:.2f} cm".format(temperature, humidity, distance))

        data = {

            "api\_key": api\_key,

            "field1": temperature,

            "field2": humidity,

            "field3": distance

        }

        response = requests.post(thingspeak\_url, data=data)

        print("Data sent to ThingSpeak. Status code:", response.status\_code)

    except Exception as e:

        print("Error:", str(e))

    time.sleep(15)

Python script for sharing data from WaterFlow Sensor

import requests

import time

import random

channel\_id = "2306722"

write\_api\_key = "J7UB4P9UTY5Z206M"

thing\_speak\_url = f"https://api.thingspeak.com/update?api\_key={write\_api\_key}"

def simulate\_water\_flow\_data():

    return random.uniform(0, 10)

while True:

    try:

        water\_flow\_rate = simulate\_water\_flow\_data()

        data = {

            "field1": water\_flow\_rate

        }

        response = requests.post(thing\_speak\_url, data=data)

        print("Data sent to ThingSpeak. Status code:", response.status\_code)

    except Exception as e:

        print("Error:", str(e))

    time.sleep(15)

Python script for sharing data from Water Meter

import requests

import time

import random

channel\_id = "2306899"

write\_api\_key = "VMMZ4TEZVTV6UX8N"

thing\_speak\_url = f"https://api.thingspeak.com/update?api\_key={write\_api\_key}"

def simulate\_water\_meter\_data():

    return random.uniform(0, 1000)

while True:

    try:

        water\_consumption = simulate\_water\_meter\_data()

        data = {

            "field2": water\_consumption

        }

        response = requests.post(thing\_speak\_url, data=data)

        print("Data sent to ThingSpeak. Status code:", response.status\_code)

    except Exception as e:

        print("Error:", str(e))

    time.sleep(15)

Python script for sharing data from Water Level Sensor

import requests

import time

import machine

from machine import Pin

thingspeak\_url = "https://api.thingspeak.com/update"

channel\_id = "2306899"

api\_key = "VMMZ4TEZVTV6UX8N"

sensor\_pins = [2, 3, 4, 5, 6]

lcd = machine.LCD()

lcd.init()

def read\_sensor\_state():

    sensor\_states = [digitalRead(pin) for pin in sensor\_pins]

    return sensor\_states

def get\_water\_level(sensor\_states):

    if all(sensor\_states):

        return "Overflowing"

    elif sensor\_states[0]:

        return "Tank is Full"

    elif sensor\_states[1]:

        return "Tank is 75% Full"

    elif sensor\_states[2]:

        return "Tank is 50% Full"

    elif sensor\_states[3]:

        return "Tank is 25% Full"

    else:

        return "Tank is Empty"

while True:

    try:

        sensor\_states = read\_sensor\_state()

        water\_level = get\_water\_level(sensor\_states)

        lcd.text(water\_level)

        data = {

            "api\_key": api\_key,

            "field1": water\_level

        }

        response = requests.post(thingspeak\_url, data=data)

        print("Data sent to ThingSpeak. Status code:", response.status\_code)

    except Exception as e:

        print("Error:", str(e))

    time.sleep(15)

Python script for sharing data from Leak Detection Sensor

import requests

import time

import random

channel\_id = "2309000"

write\_api\_key = "28WR12W5X9OSKXY2"

thing\_speak\_url = "https://api.thingspeak.com/update?api\_key={write\_api\_key}"

def simulate\_leak\_data():

    return random.choice([0, 1])

while True:

    try:

        leak\_status = simulate\_leak\_data()

        data = {

            "field1": leak\_status

        }

        response = requests.post(thing\_speak\_url, data=data)

        print("Data sent to ThingSpeak. Status code:", response.status\_code)

    except Exception as e:

        print("Error:", str(e))

    time.sleep(15)

Python script for sharing data from Pressure Sensors

import requests

import time

import random

channel\_id = "2310757"

write\_api\_key = "OL8FZEGPB5KK8DCW"

thing\_speak\_url = f"https://api.thingspeak.com/update?api\_key={write\_api\_key}"

def simulate\_pressure\_data():

    return random.uniform(0, 100)

while True:

    try:

        pressure\_value = simulate\_pressure\_data()

        data = {

            "field1": pressure\_value

        }

        response = requests.post(thing\_speak\_url, data=data)

        print("Data sent to ThingSpeak. Status code:", response.status\_code)

    except Exception as e:

        print("Error:", str(e))

    time.sleep(15)