***SMART WATER FOUNTAIN***

**PHASE 3:** Deploy IoT sensors (e.g., temperature 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.

**SMART WATER FOUNTAIN:**

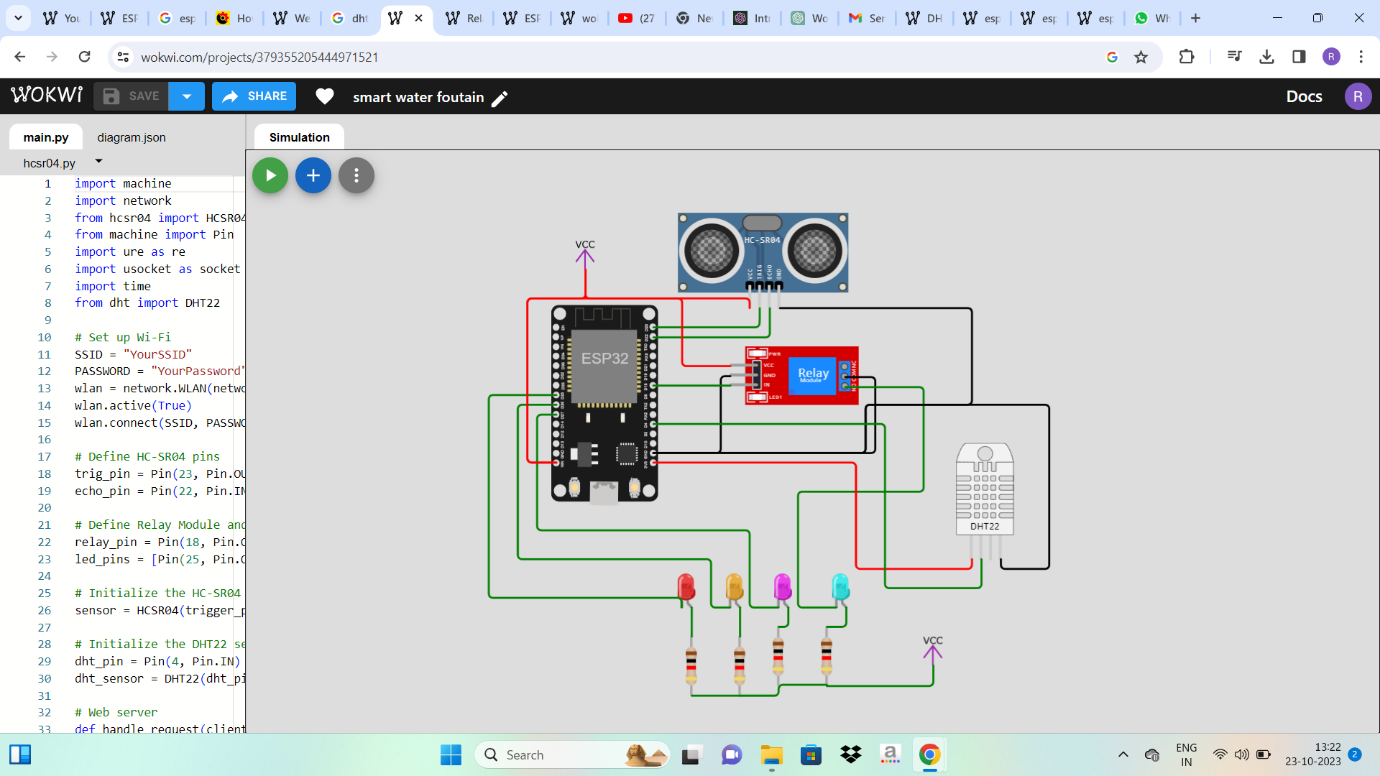
**HARDWARE SETUP:**

* ESP32 development board
* HC-SR04 ultra sonic distance sensor
* DHT22
* Relay module
* LED

**ABOUT ESP32:**

**The ESP32 is a popular WiFi and Bluetooth-enabled microcontroller, widely used for IoT Projects**

**SIMULATION DIAGRAM**



**ABOUT CIRCUIT LAYOUT:**

* ESP8266:

- Connect to Relay Module (Control Pin)

- Connect to Ultrasonic Sensor (Trigger and

Echo pins)

* Relay Module:

- GND to GND

- VCC to VCC

-Vin to D21

* Ultrasonic Sensor (HC-SR04):

- VCC to VCC

- GND to GND

- Trig to Node MCU GPIO (D23)

- Echo to Node MCU GPIO ( D22)

* LED-(D14,13,12)

**PYTHON SCRIPT:**

import machine

import network

from hcsr04 import HCSR04

from machine import Pin

import ure as re

import usocket as socket

import time

from dht import DHT22

# Set up Wi-Fi

SSID = "YourSSID"

PASSWORD = "YourPassword"

wlan = network.WLAN(network.STA\_IF)

wlan.active(True)

wlan.connect(SSID, PASSWORD)

# Define HC-SR04 pins

trig\_pin = Pin(23, Pin.OUT)

echo\_pin = Pin(22, Pin.IN)

# Define Relay Module and LED pins

relay\_pin = Pin(18, Pin.OUT)

led\_pins = [Pin(25, Pin.OUT), Pin(26, Pin.OUT), Pin(27, Pin.OUT)]

# Initialize the HC-SR04 sensor

sensor = HCSR04(trigger\_pin=trig\_pin, echo\_pin=echo\_pin)

# Initialize the DHT22 sensor

dht\_pin = Pin(4, Pin.IN)

dht\_sensor = DHT22(dht\_pin)

# Web server

def handle\_request(client):

request = client.recv(1024).decode('utf-8')

if 'GET /on' in request:

relay\_pin.on()

elif 'GET /off' in request:

relay\_pin.off()

distance = sensor.distance\_cm()

water\_level = "High" if distance < 10 else "Low"

dht\_sensor.measure()

temperature = dht\_sensor.temperature()

humidity = dht\_sensor.humidity()

response="HTTP/1.1200OK\r\nContent-Type: text/html\r\n\r\n"

response+=f"<html><body><h1>Water Level and Temperature/Humidity Monitoring</h1>"

response += f"<p>Distance: {distance} cm</p>"

response += f"<p>Water Level: {water\_level}</p>"

response+=f"<p>Temperature:{temperature}°C</p>"

response+=f"<p>Humidity:{humidity}%</p>" response+="<p><ahref='/on'>TurnPumpOn</a></p>"

response+="<p><ahref='/off'>TurnPumpOff</a></p>"

response += "</body></html>"

client.send(response)

client.close()

def run\_server():

s=socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.bind(('', 80))

s.listen(5)

while True:

client, addr = s.accept()

handle\_request(client)

# Main loop

while True:

distance = sensor.distance\_cm()

dht\_sensor.measure()

temperature = dht\_sensor.temperature()

humidity = dht\_sensor.humidity()

print("Distance:", distance, "cm")

print("Temperature:", temperature, "°C")

print("Humidity:", humidity, "%")

# Control the water pump based on distance

if distance < 10:

relay\_pin.on()

else:

relay\_pin.off()

# Indicate water level using LEDs

if distance < 10:

for i in range(3):

led\_pins[i].on()

else:

for i in range(3):

led\_pins[i].off()

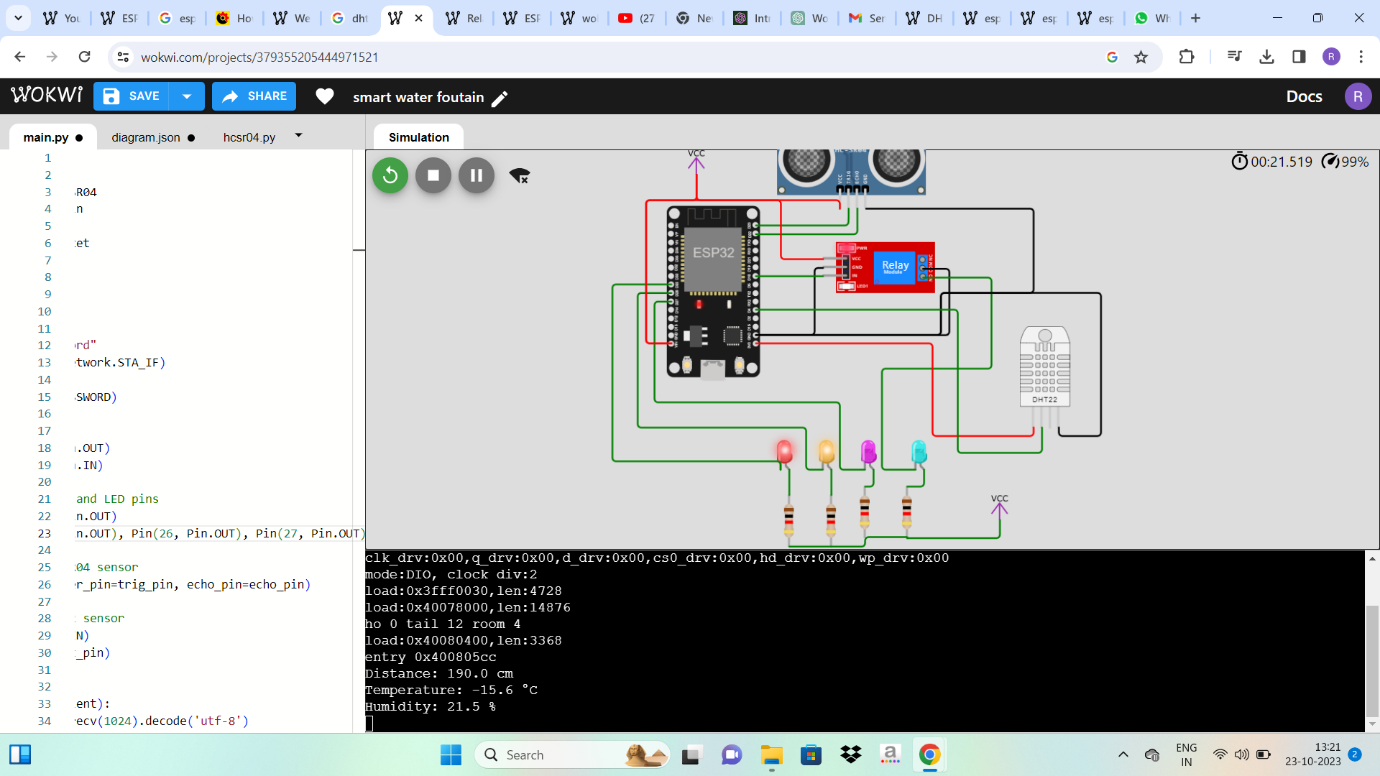
# Run the web server

run\_server()

# Delay for a while to avoid excessive measurements

time.sleep(2)

**OUTPUT OF SIMULATION:**



* The distance measured by the HC-SR04 sensor (in centimeters) is printed.
* The temperature (in °C) and humidity (%) measured by the DHT22 sensor are printed.
* The relay is turned on or off based on the water level detected by the HC-SR04 sensor.
* The LEDs indicate the water level; they are turned on or off accordingly .The web server provides information about water level, temperature, and humidity when accessed via a web browser.
* The program essentially creates a monitoring and control system for water level and environmental conditions, accessible through a web interface.
* The output includes printed data and, when accessed via a web browser, the sensor data presented in an HTML format.