**SMART WATER FOUNTAIN**

**PHASE 5: Project Documentation & Submission**

* **In this part you will document your project and prepare it for submission.**
* **Document the Smart Water Fountains project and prepare it for submission.**

**What is a smart water fountain?**

Smart water fountain is a technologically advanced device that provides access to drinking water and incorporates various features to enhance user experience, convenience, and efficiency. These features typically include touch less operation, water filtration, temperature control, customization options, connectivity to mobile apps, and sometimes even water quality monitoring and data collection capabilities. Smart water fountains are designed to offer improved hygiene, accessibility, and user control compared to traditional water fountains.

**IOT SENSOR AND HARDWARE SETUP OF SMART WATER FOUNTAIN CIRCUIT:**

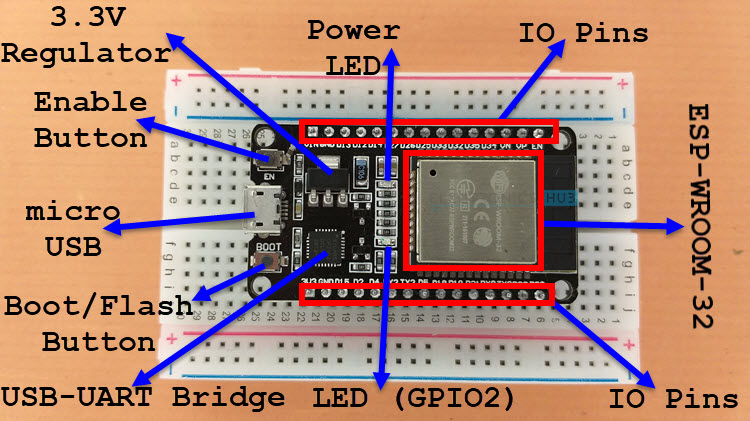
* ESP32 development board
* HC-SR04 ultra sonic distance sensor
* DHT22
* Relay module
* LED

**ESP32 DEVELPMENT BOARD:**

ESP32 is the name of the chip that was developed by Espressif Systems. This provides Wi-Fi (and in some models) dual-mode Bluetooth connectivity to embedded devices. While ESP32 is technically just the chip, modules and development boards that contain this chip are often also referred to as “ESP32” by the manufacturer.

What are the components of ESP32 development board?

* ESP-WROOM-32 Module.
* Two rows of IO Pins (with 15 pins on each side)
* CP2012 USB – UART Bridge IC.
* micro–USB Connector (for power and programming)
* AMS1117 3.3V Regulator IC.
* Enable Button (for Reset)
* Boot Button (for flashing)
* Power LED (Red)



**HC-SR04 ultra sonic distance sensor:**

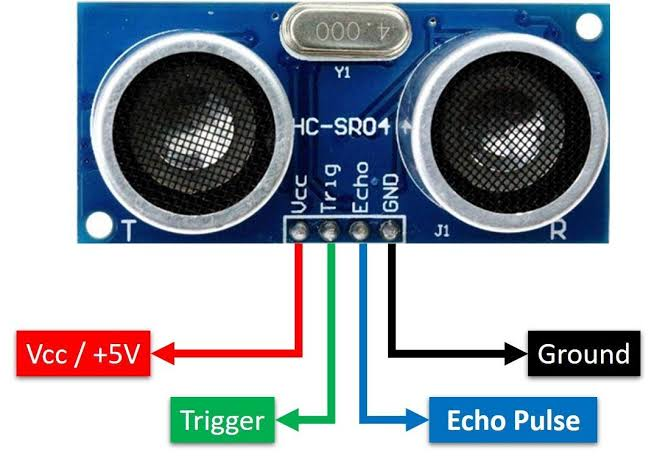
The ultrasonic sensor works on the principle of SONAR and RADAR system which is used to

determine the distance to an object.

An ultrasonic sensor generates the high-frequency sound (ultrasound) waves. When this ultrasound hits the object, it reflects as echo which is sensed by the receiver .

By measuring the time required for the echo to reach to the receiver, we can calculate the distance.

This is the basic working principle of Ultrasonic module to measure distance.



**DHT22:**

The DHT-22 uses a polymer capacitor to sense the temperature and humidity, measuring the temperature of the air between –40 and 80 degrees Centigrade (which Arduino can convert to Fahrenheit), and the relative humidity between 0 and 100%.

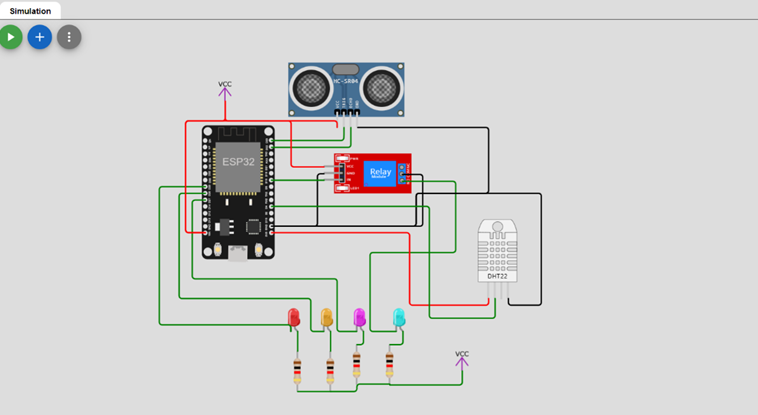
**Relay module:**

**The IoT Power Relay is a controllable power relay equipped with four outputs that help you create an Internet of Things project with safe, reliable power control. With the IoT Power Relay you can easily control the power going to a device with an Arduino, Raspberry Pi or other single-board computer or microcontroller.**



**URL for WOWKI Project of Smart Water Fountains:** [**https://wokwi.com/projects/380205266817380353**](https://wokwi.com/projects/380205266817380353)

**SCREENSHOT OF SMART WATER FOUNTAIN CIRCUIT:**



**PYTHON SCRIPT RUN CODE FOR ABOVE CIRCUIT**

import json

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)

if 'GET /data' in request:

# Measure the distance, water level, temperature, and humidity

distance = sensor.distance\_cm()

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

dht\_sensor.measure()

temperature = dht\_sensor.temperature()

humidity = dht\_sensor.humidity()

# Create a dictionary to hold the data

data = {

"distance": distance,

"water\_level": water\_level,

"temperature": temperature,

"humidity": humidity

}

# Return the data as JSON

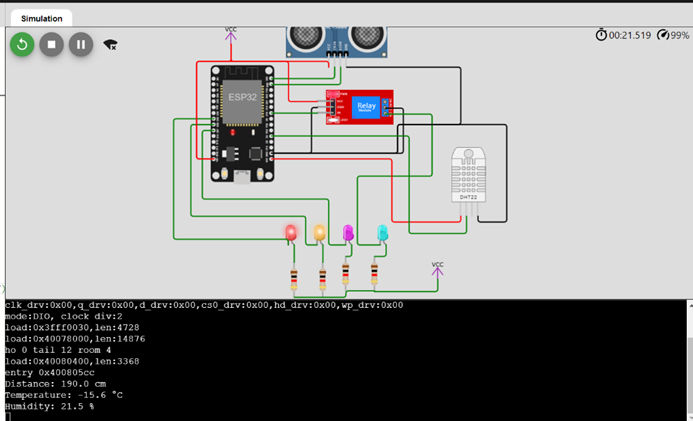
response = "HTTP/1.1 200 OK\r\nContent-Type: application/json\r\n\r\n"

response += json.dumps(data)

client.send(response)

client.close()

**SAMPLE OUTPUT SCREENSHOT OF THE OBOVE CIRCUIT:**



**WORKING OF THE ABOVE PYTHON SCRIPT:**

The provided Python program is designed to work with various IoT sensors, including an HC-SR04 ultrasonic sensor for measuring distance, a DHT22 sensor for measuring temperature and humidity, and a relay module for controlling a pump or similar device. It also sets up a simple web server to monitor and control the system remotely.

**Wi-Fi Setup**: The program begins by setting up a Wi-Fi connection with your SSID and PASSWORD. It uses the network module to connect to the Wi-Fi network.

**Sensor and Pin Definitions:** It defines the pins for various sensors and actuators (like the relay and LEDs), as well as initializes the HC-SR04 and DHT22 sensors.

**Web Server Handling:**The program includes a function called handle\_request that handles incoming HTTP requests. It checks the request, and if it's "GET /on," it turns on the relay, and if it's "GET /off," it turns off the relay.

**Sensor Measurements:** It continuously measures the distance using the HC-SR04 sensor, temperature, and humidity using the DHT22 sensor. It also sets the water level to "High" or "Low" based on the distance measurement.

**Web Server Response:** It constructs an HTML response that includes the distance, water level, temperature, and humidity measurements. It also includes links to turn the pump on and off.

**Server Initialization:** The run\_server function initializes a socket server listening on port 80 and accepts incoming client connections. When a client connects, it calls the handle\_request function to process the HTTP request.

**Main Loop:** In the main loop, it repeatedly measures the distance, temperature, and humidity.It controls the water pump (relay) based on the distance. If the distance is less than 10 cm, it turns on the relay; otherwise, it turns it off.

It indicates the water level using LEDs. If the distance is less than 10 cm, it turns on the LEDs; otherwise, it turns them off.

**Additional Web Server Handling:**The program also includes a block of code that responds to "GET /data" requests. It returns a JSON object containing the current distance, water level, temperature, and humidity.

This program effectively turns your ESP32 board into an IoT sensor system. It continuously measures environmental data, controls a pump based on the water level, and provides a web interface for monitoring and controlling the system remotely. Users can access this information by making HTTP requests to the ESP32's IP address.

**Platform UI code for Smart Water Fountains:**

<!DOCTYPE html>

<html>

<head>

<title>Water Level and Temperature/Humidity Monitoring</title>

</head>

<body>

<h1>SMART WATER FOUNTAIN USING IOT</h1>

<h2>Water Level and Temperature/Humidity Monitoring</h2>

<p>Distance: <span id="distance">Waiting for data...</span> cm</p>

<p>Water Level: <span id="waterLevel">Waiting for data...</span> </p>

<p>Temperature: <span id="temperature">Waiting for data...</span>° C</p>

<p>Humidity: <span id="humidity">Waiting for data...</span>%</p>

<p><a href="/on">Turn Pump On</a></p>

<p><a href="/off">Turn Pump Off</a></p>

<script>

**// JavaScript code to update the data on the HTML page**

function updateData() {

fetch('/data') // You should create an endpoint to fetch the data from your Python code

.then(response => response.json())

.then(data => {

document.getElementById('distance').innerText = data.distance + " cm";

document.getElementById('waterLevel').innerText=data.water\_level;

document.getElementById('temperature').innerText = data.temperature + "°C";

document.getElementById('humidity').innerText = data.humidity + "%";

});

} // Call the updateData function initially

updateData();

// Periodically update the data (e.g., every 5 seconds)

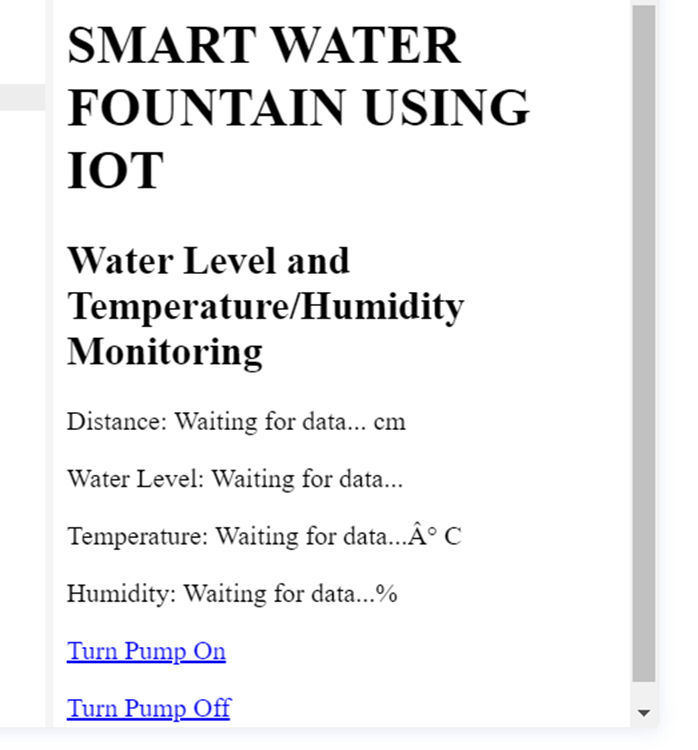
setInterval(updateData, 5000);

</script>

</body>

</html>

OUTPUT OF THE UI CODE:



**WORKING OF ABOVE PLATFORM UI CODE:**

The provided HTML program is designed to work as a web page interface for monitoring and controlling a smart water fountain using IoT. It interacts with the ESP32-based IoT sensor system you mentioned in the previous Python program.

**HTML Structure:** The HTML document is structured with title tags and body tags. It sets the title to "Water Level and Temperature/Humidity Monitoring" and includes headings and paragraphs to display the monitored data.

**JavaScript Code:**

The JavaScript code within the <script> tags is responsible for updating the data displayed on the HTML page and interacting with the IoT system.

**updateData():**

This function sends an HTTP request to the /data endpoint, expecting to receive JSON data from the ESP32-based IoT system. The fetch function is used for this purpose. The received JSON data contains information about distance, water level, temperature, and humidity. The JavaScript code updates the content of the HTML elements with the received data. For example, it updates the content of the <span> elements with IDs like "distance," "water Level," "temperature," and "humidity". The set Interval function is used to periodically call the updateData function every 5 seconds. This ensures that the data displayed on the web page is regularly updated with the latest information from the IoT system.

**HTML Elements:**

The HTML elements with IDs like "distance," "water Level," "temperature," and "humidity" are used to display the corresponding data received from the IoT system. Initially, they display "Waiting for data..." until the JavaScript code updates them.

**Hyperlinks:**

The HTML page includes two hyperlinks: "Turn Pump On" and "Turn Pump Off." When a user clicks on these links, they send GET requests to the /on and /off endpoints, respectively. These endpoints are expected to be handled by your ESP32-based IoT system, allowing you to control the water pump remotely.In summary, this HTML program provides a user-friendly web interface for monitoring water level, temperature, and humidity data from the IoT system. It uses JavaScript to periodically fetch data from the IoT system and update the displayed values. Additionally, it allows users to control the water pump by clicking on the provided hyperlinks, which trigger GET requests to the IoT system.

**CONCLUSION::** The project aims to enhance public water fountains by implementing IOT sensors to control water flow and detect malfunctions. The primary objective is to provide real-time information about water fountain status to residents through a public platform. This project includes defining objectives, designing the IOT sensor system, developing the water fountain status platform, and integrating them using IOT technology and Python.

**THANK YOU**