**SMART WATER MANAGEMENT**

|  |  |
| --- | --- |
| **TEAM ID** | **3927** |
| **PROJECT NAME** | **SMART WATER MANAGEMENT USING IOT** |
| **DATE** | **31.10.2023** |

# **Objectives:**

Create a smart water monitoring system that delivers accurate data for informed decisions for minimizing the water level for parks and gardens. The system will focus on water distributing with correct level of water which should be utilize .

**PROJECTS OVERVIEW :**

1. **AUTOMATED WATER MONITORING :**

In this system the flow of water and the water level in the tank will be monitored 24/7 . The water will be refilled when the water level is low. When we start to flow the water in the garden after completing the flow, it will be turn off by our web application.

**2. STATUS INDICATION :**

When the water level is low in the tank , a notification will be sent to the web application. So water will be refilled. In our web application the current water level in the tank will be indicated.

**3. WEB INTERFACE CONTROL:**

In our web application the interface will be user friendly and all the operations like (on, off ) can be done easily. The accurate water level will be shown in this interface.

**4. FUNCTIONALITIES OF SENSOR:**

The ultrasonic sensor and node MCU are the main components for our project. By using ultrasonic sensor we can measure the current water level in the tank. Finaly by using the node MCU all the primary components can be operated.

**5. COMMUNICATION PATH:**

The communication between the web application and water resource can be done by the sensors. All these communications are done by python code which we have inserted in the node MCU. So it acts as a major role in our project.

**IOT DEVICE SETUP:**

**Sensor and data acquisition hardware Technical specifications for ultrasonic sensor**

* Power supply þ5 V DC
* Quiescent current <2 mA
* Working current 15 mA
* Effectual angle <15
* Ranging distance 2–400 cm (1 inch–13 ft)

**Steps:**

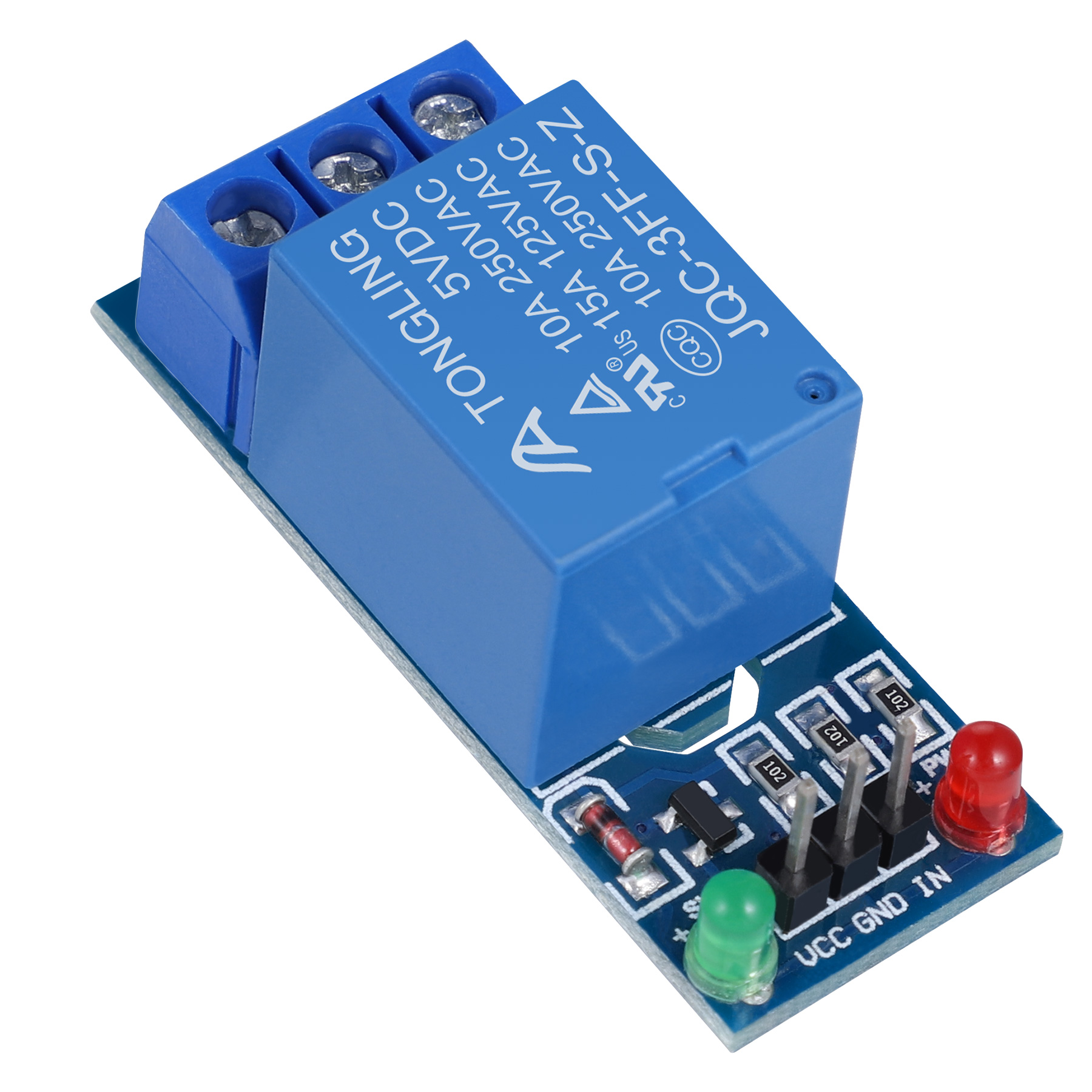
**Ultrasonic Sensor:**



**Connection:**

* VCC to ESP32 5V
* GND to ESP32 GND
* Trig to ESP32 digital pin
* Echo to ESP32 digital pin

**Relay Module:**



**Connection:**

* VCC and GND to ESP32 5V and GND
* IN to ESP32digital pin

**Pump:**



**Connection:**

Ensure the pump is powered appropriately and connected to a relay channel.

**PLATFORM DEVELOPMENT:**

**1. Web Interface:**

* HTML: Create an HTML file for the user interface to control the fountain and display sensor data.
* JavaScript: Use JavaScript to handle button clicks, send commands to the ESP32, and update the interface based on sensor readings.

**2. Server-Side Communication:**

* Node.js/Python (Server Side): Create a server using Node.js or Python to communicate with the ESP32. Use frameworks like Express.js for Node.js or Flask for Python to handle requests.
* WebSocket or REST API: Implement WebSockets or REST API to enable real-time communication between the server and ESP32.

**HARDWARE INTEGRATION:**

**1. Communication:**

Establish serial communication between the ESP32and the server using USB, Bluetooth, or Wi-Fi modules like ESP8266 or ESP32.

**2. Sensor Data Transmission:**

Send sensor data (water level readings) from the ESP32 to the server for display on the web interface.

**OVERALL WORKFLOW:**

* HTML Interface: Users interact with the HTML interface to control the fountain and visualize data.
* JavaScript: Handles button clicks, sends requests to the server, and updates the UI based on received data.
* Server: Communicates with the ESP32, processes requests, and sends data to the HTML/JavaScript front-end.
* Arduino: Receives commands from the server, controls the fountain components, and sends sensor data to the server.

**CODE IMPLEMENTATION**

Here is a list of codes which are used in this project for there function of a Arduino which is going to be implemented in our project.

import RPi.GPIO as GPIO

import time

pH = 0.0

water\_turbidity = 0.0

water\_level = 0

manual\_mode = False

water\_pump = False

**# GPIO pins**

Turbidity = 16

pump = 15

trig = 4

echo = 5

wLevel = 0

sensorPower = 14

**# Thresholds**

lower\_threshold = 420

upper\_threshold = 520

avg\_value = 0

duration = 0

dist\_cm = 0

b = 0.0

buf = [0] \* 10

temp = 0

volt = 0.0

ntu = 0.0

height = 100

def setup():

GPIO.setmode(GPIO.BCM)

GPIO.setup(pump, GPIO.OUT)

GPIO.setup(echo, GPIO.IN)

GPIO.setup(trig, GPIO.OUT)

GPIO.setup(sensorPower, GPIO.OUT)

GPIO.output(sensorPower, GPIO.LOW)

GPIO.setup(wLevel, GPIO.IN)

GPIO.setup(Turbidity, GPIO.IN)

**# Initialize your WiFi connection here**

def loop():

on\_manual\_mode\_change()

time.sleep(0.1)

def on\_manual\_mode\_change():

global manual\_mode

**# Add your code here to act upon Manual Mode change**

if manual\_mode:

on\_water\_pump\_change()

else:

automate()

def on\_water\_pump\_change():

global water\_pump

**# Add your code here to act upon WaterPump change**

if water\_pump:

GPIO.output(pump, GPIO.HIGH)

else:

GPIO.output(pump, GPIO.LOW)

def automate():

global dist\_cm, pH, water\_turbidity, water\_level

**# ULTRASONIC SENSOR**

GPIO.output(pump, GPIO.LOW)

GPIO.output(trig, GPIO.LOW)

time.sleep(0.02)

GPIO.output(trig, GPIO.HIGH)

time.sleep(0.00001)

GPIO.output(trig, GPIO.LOW)

while GPIO.input(echo) == 0:

pulse\_start = time.time()

while GPIO.input(echo) == 1:

pulse\_end = time.time()

duration = pulse\_end - pulse\_start

dist\_cm = duration \* 34300 / 2

if 15 <= dist\_cm <= 20:

print("Sufficient water")

elif 10 <= dist\_cm < 15:

print("Tank only half filled")

elif dist\_cm < 7:

print("Refill tank")

elif dist\_cm < 3:

print("Tank Empty")

GPIO.output(pump, GPIO.HIGH)

**# pH SENSOR**

buf = []

for i in range(10):

buf.append(adc\_read(pH))

time.sleep(0.01)

buf.sort()

avg\_value = sum(buf[2:8])

ph\_value = float(avg\_value) \* 5.0 / 1024 / 6

ph\_value = 3.5 \* ph\_value

pH = ph\_value

print("pH:", round(ph\_value, 2))

time.sleep(0.8)

if ph\_value > 8.5 or ph\_value < 6.5:

print("WARNING: pH level - Extreme")

GPIO.output(pump, GPIO.LOW)

**# TURBIDITY SENSOR**

volt = 0.0

for i in range(800):

volt += float(adc\_read(Turbidity)) / 1023 \* 5

volt /= 800

volt = round\_to\_dp(volt, 2)

if volt < 2.5:

ntu = 0.5

else:

ntu = -1120.4 \* (volt \*\* 2) + 5742.3 \* volt - 4353.8

print(round(ntu, 2), "NTU")

water\_turbidity = ntu

time.sleep(0.1)

if ntu >= 1:

print("Warning: turbidity level - Harmful")

GPIO.output(pump, GPIO.HIGH)

**# Water Level Sensor**

GPIO.output(sensorPower, GPIO.HIGH)

time.sleep(0.1)

level = adc\_read(wLevel)

GPIO.output(sensorPower, GPIO.LOW)

water\_level = level

if level == 0:

print("Water Level: Empty")

elif 0 < level <= lower\_threshold:

print("Water Level: Low")

elif lower\_threshold < level <= upper\_threshold:

print("Water Level: Medium")

elif level > upper\_threshold:

print("Water Level: High")

print("")

time.sleep(3)

def adc\_read(channel):

# Replace this with your ADC reading code for Raspberry Pi

# Example: return GPIO.input(channel)

return 0

def round\_to\_dp(value, decimal\_places):

multiplier = 10 \*\* decimal\_places

return round(value \* multiplier) / multiplier

if \_name\_ == "\_main\_":

setup()

try:

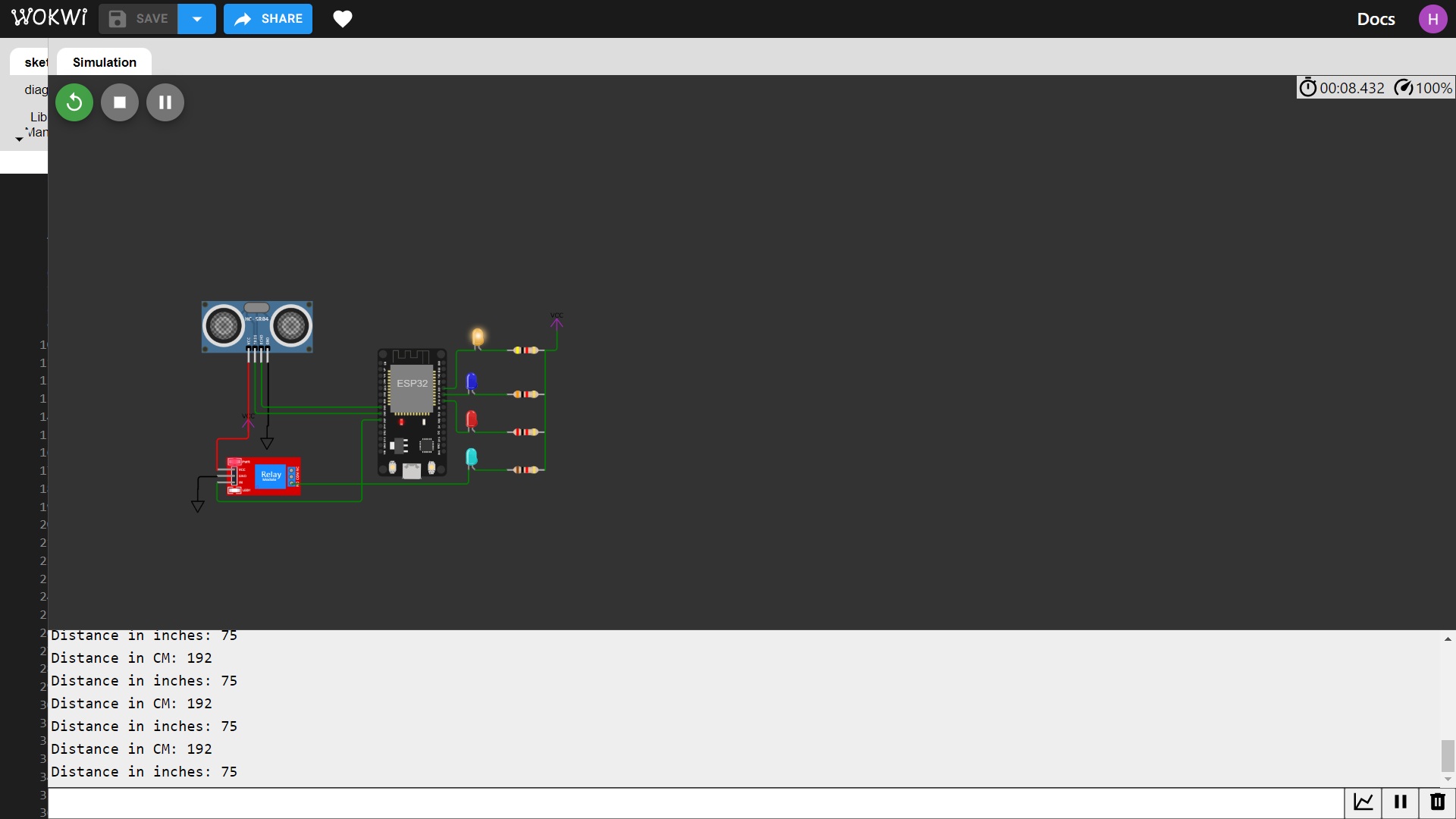
while True:

loop()

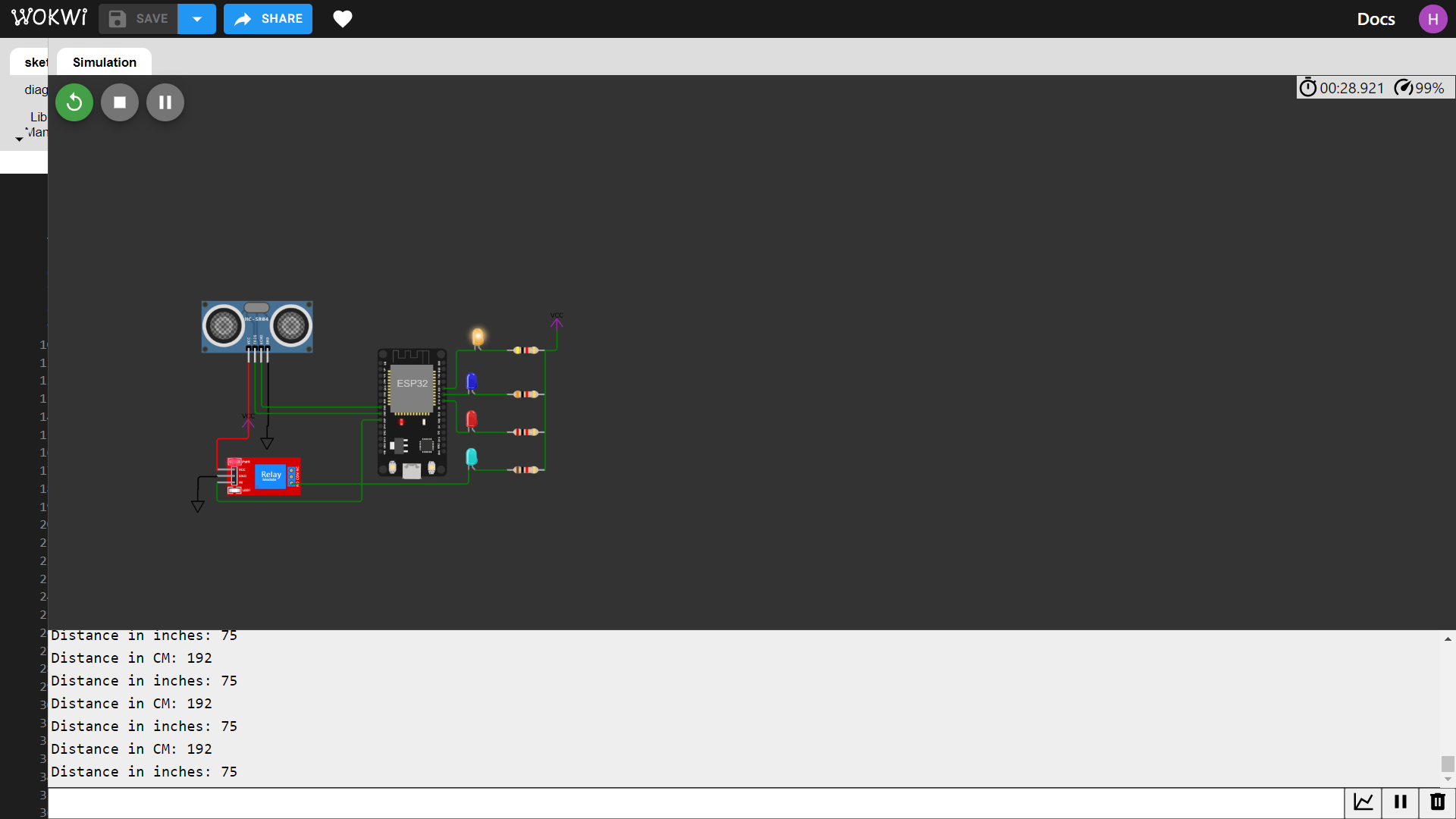
except KeyboardInterrupt:

GPIO.cleanup()

**CIRUIT DIAGRAM :**



**OUTPUT :**



**WEB DEDVELPOMENT:**

**HTML:**

<html lang ="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="widt=device-width, initial-scale=1.0">

<title>Smart Water Management</title>

<style>

body{

font-family: Arial, sans-serif;

background-color: rgb(249, 188, 138);

}

.container{

max width: 800px;

margin: 0 auto;

padding: 20px;

text-align: center;

}

h1{

color: #000000;

}

.status{

font\_size: 1.2em;

margin-top: 20px;

}

.control-buttons{

margin-top: 30px;

}

.control-buttons button{

padding: 10px 20px;

font-size: 1em;

background-color: #ea6161;

color:#fff;

border: none;

cursor: pointer;

}

. control.buttons button: hover{

background-color: #0056b3;

}

</style>

</head>

<body>

<div class="container">

<h1><center>Smart Water Mangement</center></h1>

<div class="status">

<p><center>Current Status: <span id="status">Idle</span></center></p>

</div>

<div class="control-buttons">

<button id="start button">Tank Motor on </button>

<button id="stop button">Tank Motor off </button>

<button id="water level">Water level</button>

<button id="start "> ON </button>

<button id="stop "> OFF </button>

</div>

</div>

<script>

// JavaScript code for handling water

const statusElement = document. getElementById("status");

const startButton = document. GetElementById("startButton");

const stopButton = document. GetElementById("stopButton");

const stopButton = document. GetElementById("start");

const stopButton = document. GetElementById("stop");

startButton.addEventListener("click", () => {

// Send a command to start the garden watering(you can implement this part)

statusElement.textContent = "water flow is on";

});

stopButton.addEventListener("click", () => {

// Send a command to stop the garden watering (you can implement this part)

statusElement.textContent = "water flow is stopped";

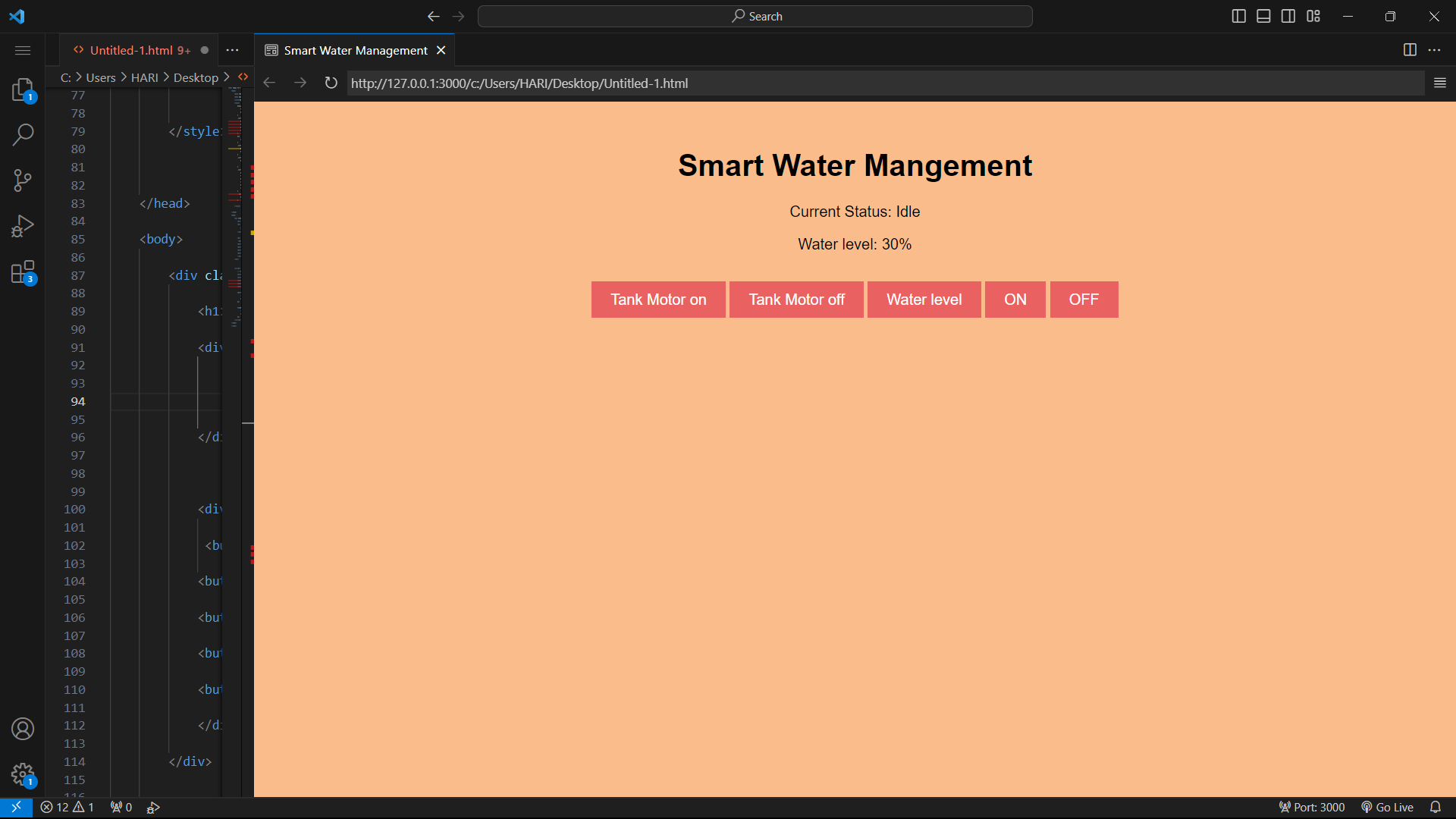
});

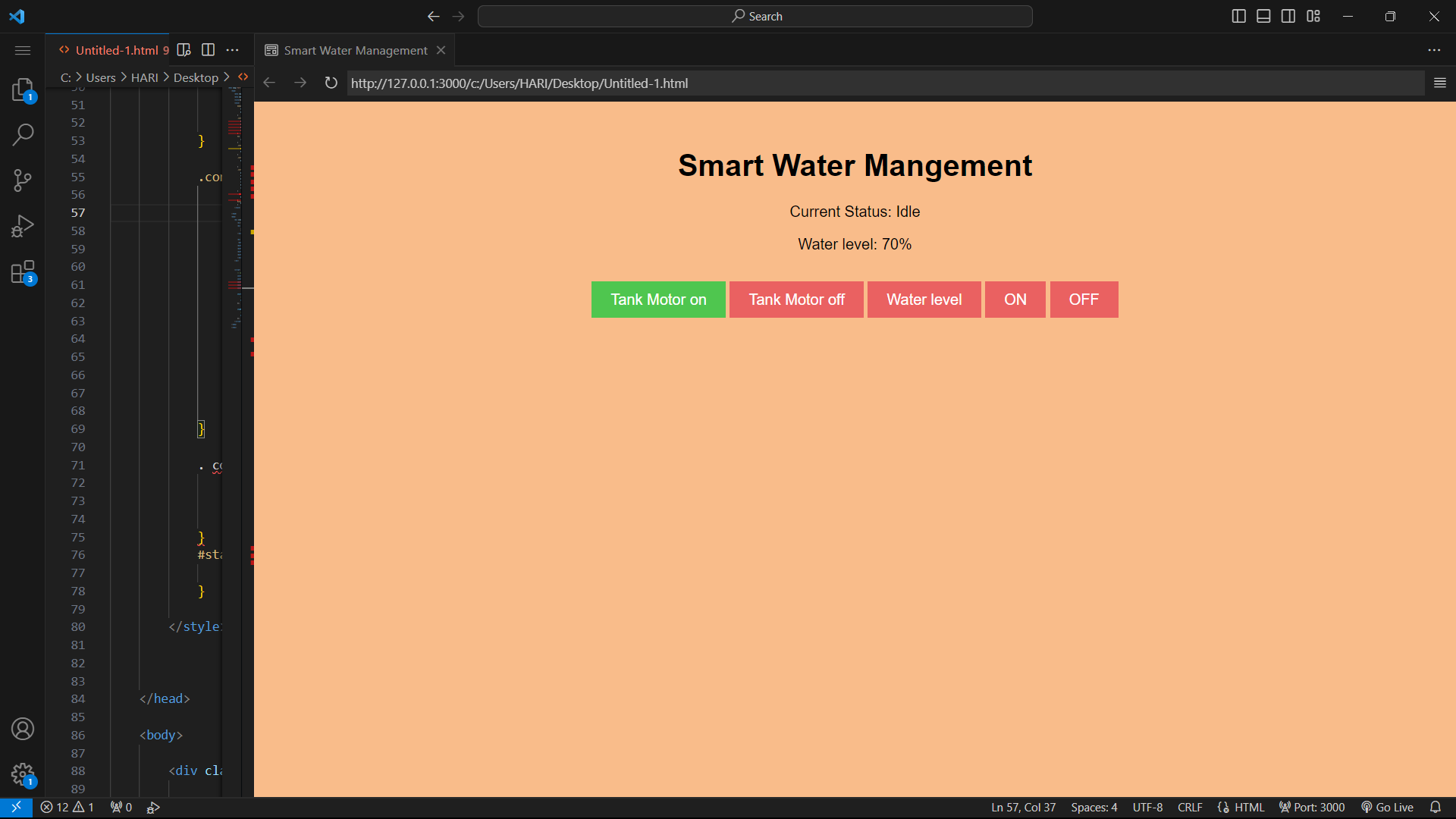
</script>

</body>

</html>

**OUTPUT :**





**PROJECT EXPLANATION IN DETAIL:**

**Components:**

**1. Arduino Board:**

Functions as the core controller, processing sensor data and controlling the connected components.

**2.Ultrasonic Sensor:**

Measures the distance of the water level in the tank.

**3. Relay:**

Enables the control of the water pump or other devices in response to the detected water level.

**4. Servo Motor (Initially):**

Previously used to adjust the level of water for garden but later replaced with pump control.

**Code Overview:**

**1.Setup Function:**

Initializes pins for input and output, setting the initial state of components and initiating serial communication for debugging.

**2.Loop Function:**

* Continuously reads the water level from the ultrasonic sensor.
* If the water level surpasses a predetermined threshold (200 cm), the code activates the relay to start the water flow, turns on the LED to indicate the operational status, and controls the pump to ensure water flow.
* If the water level falls below the threshold, the code deactivates the relay, LED, and pump to stop the operation.

**HTML Interface:**

**1.Status Display:**

Provides information on the current status of the tank, whether it's actively running or stopped, allowing users to visualize its condition.

**2.Distance Measurement:**

Displays the measured water level (distance) detected by the ultrasonic sensor.

**3.Control Buttons:**

Offers users the ability to start or stop the fountain via the web interface by clicking on the corresponding buttons.

OBJECTIVE:

* The project’s primary objective is to create an IoT-based water management system that can autonomously regulate its operation according to the water level, facilitated by the ultrasonic sensor.
* Users can conveniently control the watering remotely through a straightforward web-based interface while also receiving real-time updates about the water status.
* This project hardware integration, sensor data processing, and user interface development to form a comprehensive system for managing a water system in a smart and remotely accessible manner.