

## PROJECT TITLE:SMART WATER FOUNTAIN

### PHASE 5: Project documentation and submission

#### OBJECTIVE:

The objective of this smart water fountain project within the Wokwi simulation environment is to design and simulate an interactive water feature using a Raspberry Pi, ultrasonic sensor, and virtual components, with the specific goal of making the LED blink when the water level surpasses 200 cm. This demonstrates the system's ability to dynamically respond to varying water levels and create an engaging visual indicator, offering a practical application for real-world water level monitoring systems and showcasing Wokwi's capabilities for hardware and software integration.

#### WOKWI:

Wokwi is a versatile online platform that allows you to design, simulate, and test electronic circuits in a virtual environment.



Website: (<https://wokwi.com/>)

#### COMPONENTS REQUIRED:

- Raspberry Pi
- Ultrasonic Sensor ▪ Water Pump

- LED

- Resistor

The components required for this smart water fountain project, where the led blinks when the water level surpasses 200 cm, include a raspberry pi for control, an ultrasonic sensor to measure water levels, a water pump to control water flow, an led for visual indication, a resistor for led operation, a breadboard for prototyping, and jumper wires to establish electrical connections, all working together to create an interactive and dynamic water fountain system that autonomously adapts to changing water levels.

#### **WIRING CONNECTIONS: 1. Ultrasonic Sensor:**

**Purpose:** The ultrasonic sensor is used to measure water levels in the fountain.

- Connect the VCC (power) pin to the 5V output of the Raspberry Pi.
- Connect the GND (ground) pin to a GND (ground) pin on the Raspberry Pi. ▪ Connect the TRIG (trigger) pin to GPIO pin 17 on the Raspberry Pi.
- Connect the ECHO (echo) pin to GPIO pin 18 on the Raspberry Pi.

#### **2. Water Pump:**

**Purpose:** The water pump controls the flow of water within the fountain.

- Connect the positive (red) wire of the water pump to an external power supply suitable for the pump's voltage and current requirements.
- Connect the negative (black) wire of the water pump to the collector © of an NPN transistor or use a motor driver module to control the pump.
- Connect the emitter € of the transistor to the GND (ground) of the Raspberry Pi.
- Connect the base (B) of the transistor to GPIO pin 4 on the Raspberry Pi through a current-limiting resistor (220-330 ohms).

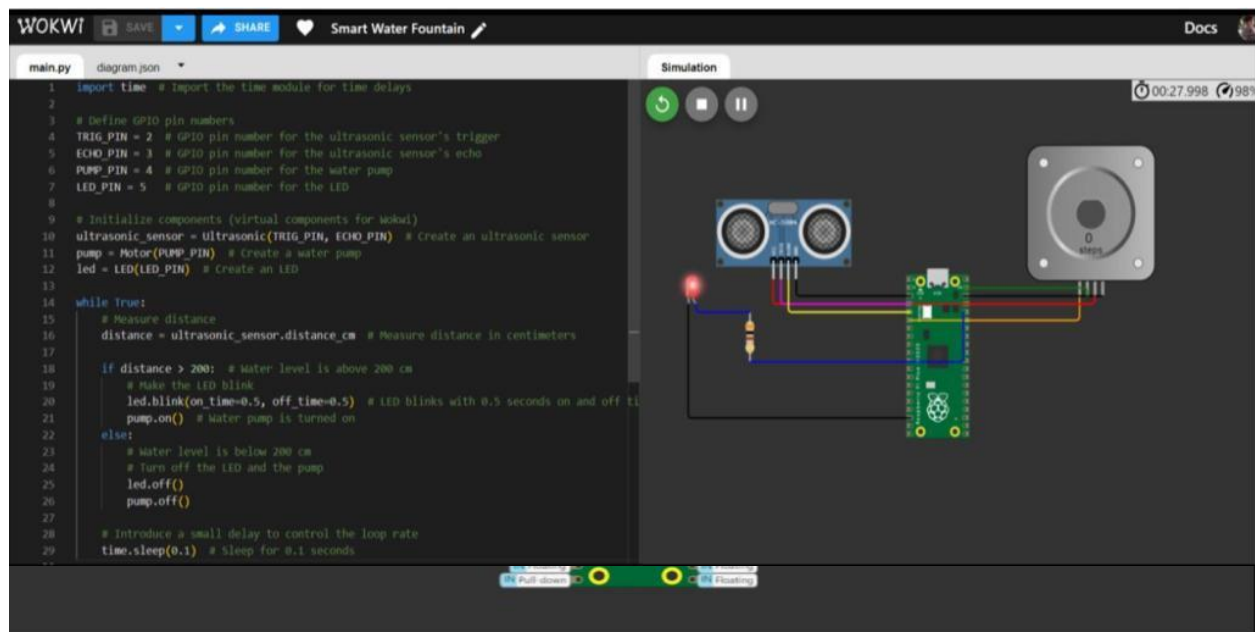
#### **3. LED (with Resistor):**

**Purpose:** The LED serves as a visual indicator of the water level.

- Connect the longer leg (anode) of the LED to a current-limiting resistor (220- 330 ohms).
- Connect the other end of the resistor to GPIO pin 5 on the Raspberry Pi.
- Connect the shorter leg (cathode) of the LED directly to a GND (ground) pin

On the Raspberry Pi.

These connections are essential for the smart water fountain project, allowing the Raspberry Pi to interface with the ultrasonic sensor, water pump, and LED to create a dynamic and interactive water feature.



## CODE DESCRIPTION:

Import time

TRIG\_PIN = 2

ECHO\_PIN = 3

PUMP\_PIN = 4

LED\_PIN = 5

Ultrasonic\_sensor = Ultrasonic(TRIG\_PIN, ECHO\_PIN) pump = Motor(PUMP\_PIN)

Led = LED(LED\_PIN while True:

Distance = ultrasonic\_sensor.distance\_cm if distance > 200:

Led.blink(on\_time=0.5, off\_time=0.5)

Pump.on() # Water pump is turned on else:

Led.off()

Pump.off() time.sleep(0.1)

This code effectively simulates a smart water fountain project in the Wokwi environment, where the LED blinks when the water level is above 200 cm and stops when it falls below that threshold.

WOKWI SAVE SHARE Smart Water Fountain Docs

main.py diagram.json

```

1 import time # Import the time module for time delays
2
3 # Define GPIO pin numbers
4 TRIG_PIN = 2 # GPIO pin number for the ultrasonic sensor's trigger
5 ECHO_PIN = 3 # GPIO pin number for the ultrasonic sensor's echo
6 PUMP_PIN = 4 # GPIO pin number for the water pump
7 LED_PIN = 5 # GPIO pin number for the LED
8
9 # Initialize components (virtual components for Wokwi)
10 ultrasonic_sensor = ultrasonic(TRIG_PIN, ECHO_PIN) # Create an ultrasonic sensor
11 pump = Motor(PUMP_PIN) # Create a water pump
12 led = LED(LED_PIN) # Create an LED
13
14 while True:
15     # Measure distance
16     distance = ultrasonic_sensor.distance_cm # Measure distance in centimeters
17
18     if distance > 200: # Water level is above 200 cm
19         # Make the LED blink
20         led.blink(on_time=0.5, off_time=0.5) # LED blinks with 0.5 seconds on and off time
21         pump.on() # Water pump is turned on
22     else:
23         # Water level is below 200 cm
24         # Turn off the LED and the pump
25         led.off()
26         pump.off()
27
28     # Introduce a small delay to control the loop rate
29     time.sleep(0.1) # Sleep for 0.1 seconds
30

```

Simulation

00:27.998 98%

00:17.218 98%

Distance : 360.02 cm

Distance : 360.97 cm

Distance : 361.17 cm

## Code Implementation:






### Visual Studio Code :

Visual Studio Code, often referred to as VS Code, is a popular and versatile source code editor that's widely used for software development. It's known for flexibility, speed, and a rich ecosystem of extensions, making it a top choice for many developers.

## Visual Studio Code

Editing evolved

### Start

-  New File...
-  Open File...
-  Open Folder...
-  Clone Git Repository...
-  Connect to...






### Recent

expense\_tracker E:\flutter\flutter 3 rd ap;p  
flutter\_application\_second\_app E:\flutter\flutter\_application\_second\_app  
flutter\_application\_first\_app E:\flutter  
TECHNOHACKS E:\original eco  
original eco E\  
More...

### Recommended

-  **GitHub Copilot**  
Supercharge your coding experience for as little as \$10/month with cutting edge AI code generation.

### Walkthroughs

-  **Get Started with VS Code**  
Discover the best customizations to make VS Code yours.
-  **Learn the Fundamentals**
-  **Boost your Productivity**
-  **Get Started with C++ Development** Updated
-  **Get started with React Native development** Updated

## Code Implementation :

The HTML (index.html) file lays the foundation for

The webpage's structure and content. It begins by defining the document type and including a

Reference to an external CSS stylesheet for styling

This screenshot shows the initial state of the `phase4.html` file in VS Code. The Explorer sidebar on the left shows the project structure with `phase4.html`, `script.js`, and `styles.css`. The main editor displays the following HTML code:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8" />
5     <link rel="stylesheet" type="text/css" href="styles.css" />
6   </head>
7   <body>
8     <center><h1 id="a">SMART WATER FOUNTAINS</h1></center>
9     <br />
10    <br />
11    <p />
12    <center>
13      <h3 id="A">
14        Smart Water Management is the activity of planning, developing,
15        distributing and managing the use of water resources using an array of
16        IoT technologies which are designed to increase transparency, and make
17        more reasonable and sustainable usage of these water resources.
18      </h3>
19    </center>
20    <div class="container">
21      <div class="status">
22        <h1>Water Fountain Status</h1>
23        <p id="status-text">Fountain is running</p>
24      </div>
25      <div class="controls">
26        <button id="start-button">Start</button>
27        <button id="stop-button">Stop</button>
28        <button id="alert-button">Alert</button>
29      </div>
30      <div class="live-data">
31        <h2>Live Data</h2>
32        <div id="chart"></div>
33      </div>
34    </div>
35    <script src="script.js"></script>
36  </body>
37 </html>
```

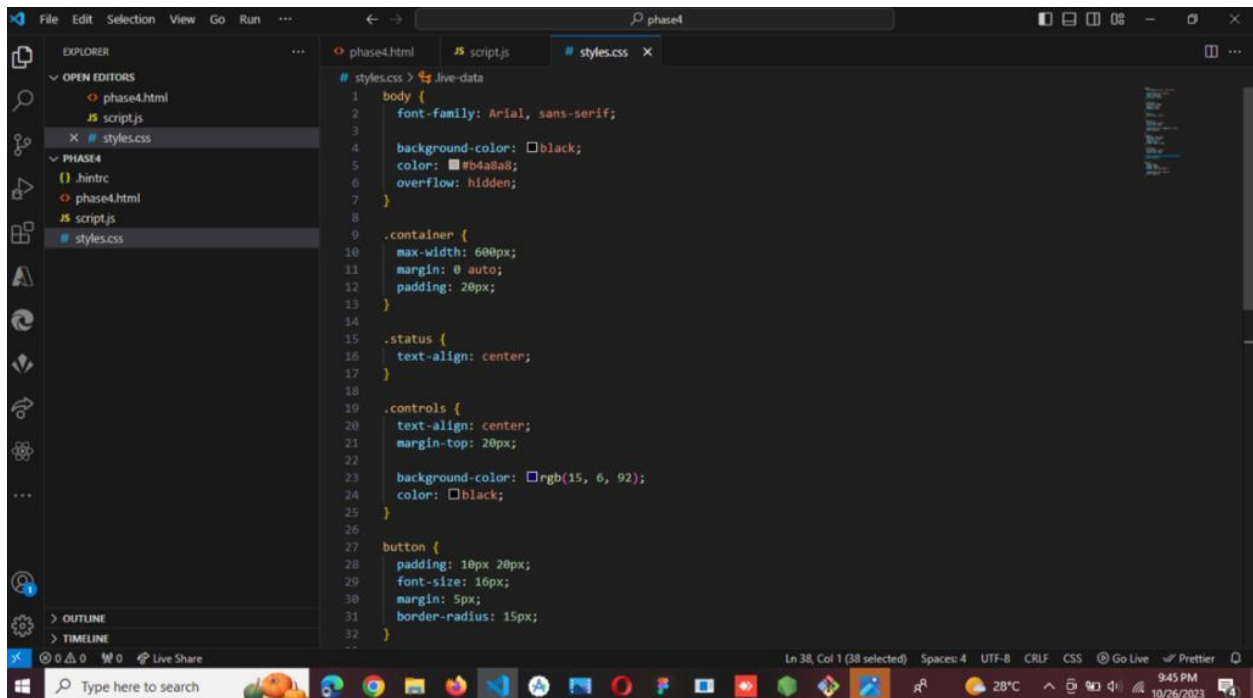
This screenshot shows the final state of the `phase4.html` file in VS Code. The Explorer sidebar on the left shows the project structure with `phase4.html`, `script.js`, and `styles.css`. The main editor displays the following HTML code:

```
22 <h1>Water Fountain Status</h1>
23 <p id="status-text">Fountain is running</p>
24 </div>
25 <div class="controls">
26   <button id="start-button">Start</button>
27   <button id="stop-button">Stop</button>
28   <button id="alert-button">Alert</button>
29 </div>
30 <div class="live-data">
31   <h2>Live Data</h2>
32   <div id="chart"></div>
33 </div>
34 </div>
35 <script src="script.js"></script>
36 </body>
37 </html>
38
```

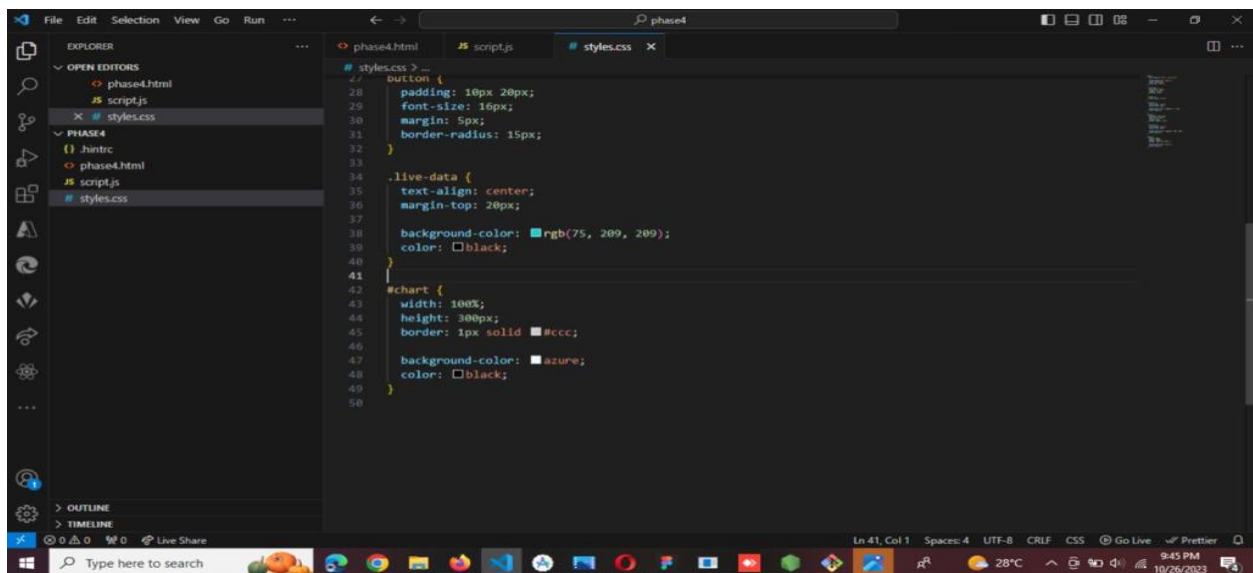
## Code for CSS:

The CSS stylesheet(style.css), linked in the HTML file,

Provides the visual styling for the page.



```
# styles.css > live-data
1 body {
2   font-family: Arial, sans-serif;
3
4   background-color: black;
5   color: #b4a8a8;
6   overflow: hidden;
7 }
8
9 .container {
10  max-width: 600px;
11  margin: 0 auto;
12  padding: 20px;
13 }
14
15 .status {
16  text-align: center;
17 }
18
19 .controls {
20  text-align: center;
21  margin-top: 20px;
22
23  background-color: rgb(15, 6, 92);
24  color: black;
25 }
26
27 button {
28  padding: 10px 20px;
29  font-size: 16px;
30  margin: 5px;
31  border-radius: 15px;
32 }
```



```
# styles.css > ...
27 button {
28  padding: 10px 20px;
29  font-size: 16px;
30  margin: 5px;
31  border-radius: 15px;
32 }
33
34 .live-data {
35  text-align: center;
36  margin-top: 20px;
37
38  background-color: rgb(75, 209, 209);
39  color: black;
40 }
41
42 #chart {
43  width: 100%;
44  height: 300px;
45  border: 1px solid #ccc;
46
47  background-color: #azure;
48  color: black;
49 }
50
```

## **Design and Implementation:**

### **Design Phase:**

#### **Design the System:**

Create a system architecture that illustrates how all the components will connect and interact with each other. Consider power requirements and connectivity options.

#### **User Interface Design:**

If your project includes a user interface (web or mobile app), design its layout and features for controlling and monitoring the smart water fountain

#### **Assemble Hardware:**

Set up the physical components of the water fountain. Ensure secure

Connections and waterproofing where necessary. Microcontroller Programming:

Write the code to control the water pump, read sensors, and manage the fountain's core functionality. Use a suitable programming language (e.g., Python for Raspberry Pi or C/C++ for Arduino).

#### **User Interface Development:**

Create the user interface (web or mobile app) to control and monitor the smart water fountain. Implement the features you designed in the design phase.

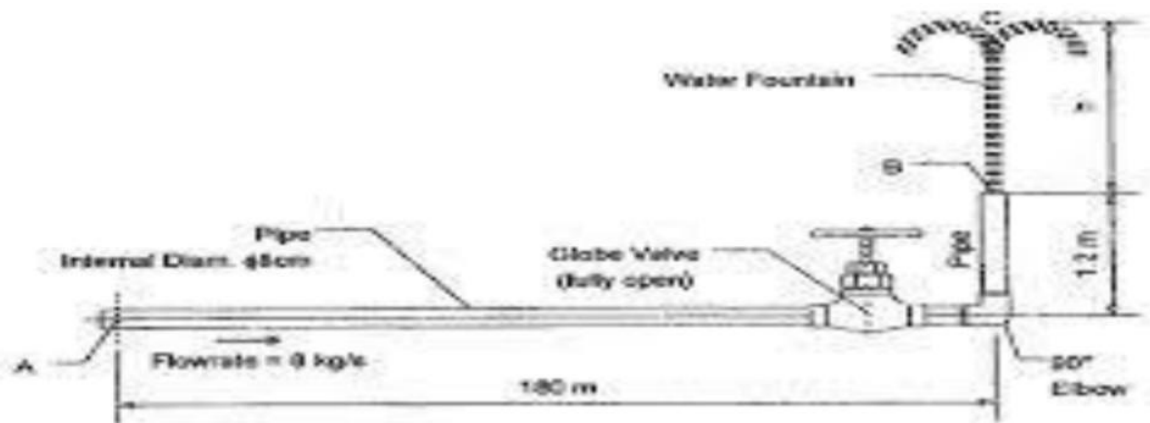
#### **Implement Communication:**

Develop the communication protocols between the microcontroller, user interface, and any other connected devices. Ensure real-time data exchange as needed.

#### **.Website Description:**

\*The smart water fountain system website is a user- friendly platform designed to monitor, control, and manage a water fountain in real time. It provides a seamless and intuitive interface for users to interact with the fountain and stay informed about its status. \*Key Features:\*1. \*Fountain Status:\* Users can instantly check whether the water fountain is running or stopped. The system provides real-time updates on the current status, ensuring users always know the fountain's operational state.2. \*Control Functionality:\* The website allows users to control the fountain with the click of a button. They can easily start or stop the fountain according to their preferences, contributing to efficient water usage.3.





## RESULT ANALYSIS:

This project aims to create a virtual smart water fountain simulation using Wokwi, integrating a Raspberry Pi (or a compatible microcontroller) with virtual components such as an ultrasonic sensor, water pump, and LED. The simulation successfully monitors water levels, causing the LED to blink when the water level surpasses 200 cm and activating the water pump accordingly. This project showcases the ability of virtual components to emulate the functionality of a real-world system within a simulated environment.

## CONCLUSION:

This Wokwi-based smart water fountain simulation project effectively illustrates the control and monitoring of water levels using a Raspberry Pi and virtual components. The project demonstrated the practicality of using Wokwi's virtual environment for hardware simulation, allowing precise testing and visualization of system functionality without physical components. The LED and water pump responded to water level changes as expected, showcasing the potential for virtual hardware modeling.