SMART WATER FOUNTAINS

Project Objectives:

The objectives of the smart water fountain student project can be summarized as follows:

Smart Water Fountain Design: Design and build a smart water fountain that can dispense water in an automated and efficient manner.

- **IoT Integration:**
  Integrate IoT (Internet of Things) technology to monitor and control the water fountain remotely .

- **Water Quality Monitoring:**
  Implement sensors to monitor the quality of the water and ensure it is safe for consumption.

- **Data Logging:**
  Collect and store data related to water consumption and water quality over time.

- **Mobile App or Web Interface:**
  Develop a user-friendly mobile app or web interface for users to interact with and control the smart water fountain.

- **Energy Efficiency:**
  Optimize the system for energy efficiency to reduce operational costs.

- **IoT Device Setup:**

The IoT device setup for the smart water fountain project involves the following components:

- **Microcontroller:**
  Use a microcontroller like Arduino or Raspberry Pi to control the water fountain and interface with sensors and other components.

- **Water Dispensing System**:
  Design a water dispensing mechanism that can be controlled by the microcontroller, ensuring precise and controlled water flow.

- **Water Quality Sensors:**
  Install water quality sensors such as pH sensors, turbidity sensors, or temperature sensors to monitor the quality of the water.

- **Flow Sensors:**
  Use flow sensors to measure the amount of water dispensed.

- **Connectivity Module:**
  Incorporate a Wi-Fi or cellular module to enable communication between the microcontroller and the cloud platform.

- **Power Supply:**
  Provide a power source for the IoT device, which may include battery or mains power, and implement power-saving measures.

- **Platform Development:**
  Developing a cloud platform to manage and control the smart water fountain is a crucial aspect of the project. This platform may involve Cloud Service: Utilize cloud services such as AWS, Azure, or Google Cloud to store data and host the application.

- **Database:**
  Set up a database to store water quality data, usage statistics, and user preferences.

- **User Authentication:**
  Implement secure user authentication to ensure only authorized users can control the fountain.

- **Remote Control:**
  Develop an interface that allows users to remotely control the water fountain, adjust settings, and view water quality data.

- **Notifications:**
  Create a system for sending notifications to users, such as low water levels, filter replacement alerts, or system updates.

Code Implementation:

The code implementation for the smart water fountain project involves programming the microcontroller, creating a user interface, and setting up the cloud platform. Here are some key tasks:

Microcontroller Programming: Write code to control the water dispensing system, interface with sensors, and manage the IoT connectivity.

Sensor Data Processing: Implement code to read and process data from water quality sensors and flow sensors.

IoT Communication: Set up code for sending data from the microcontroller to the cloud platform and receiving commands from the platform.

- **User Interface Development:**
  Create a user-friendly mobile app or web interface to interact with the smart water fountain.

- **Data Analytics:**
  Analyze and visualize the data collected from the fountain to provide insights on water usage and quality.

- **Security Measures:**
  Implement security measures to protect user data and the IoT system from unauthorized access.

- **Testing and Debugging:**
  Thoroughly test the system, identify and fix any bugs, and ensure it functions as intended.

- **Documentation:**
  Document the code and system architecture for future reference and troubleshooting.

Overall, the smart water fountain student project involves designing, building, and programming a smart water fountain, integrating IoT technology, and developing a cloud-based platform for monitoring and control. The project aims to provide a convenient and efficient way

users to access clean water while promoting water conservation and quality monitoring.

1.Smart Water Fountain Components:
Microcontroller (e.g., Raspberry Pi): Central processing unit that controls the fountain and handles data.

Water Dispensing System: Mechanism for controlled water dispensing.Water Quality Sensors (e.g., pH, turbidity): Sensors to monitor water quality.

Flow Sensors: Measure water flow rate.

Connectivity Module: Wi-Fi or cellular module for IoT connectivity.

Power Supply: Battery or mains power source.

2. Schematic Overview:

3. Data Sharing Platform:

To share data from your smart water fountain, you'll need a cloud-based platform. Here's how it works:

Data Collection: The microcontroller collects data from water quality sensors and flow sensors.

Data Transmission: The microcontroller sends this data to the cloud-based platform via the IoT connectivity module (Wi-Fi, cellular, etc.).

Cloud Platform: This platform can be hosted on a service like AWS, Azure, or Google Cloud and includes the following components:

Database: Stores data (e.g., water quality readings, water flow, usage statistics).

User Authentication: Ensures secure access to the platform.

User Interface: Provides a web or mobile app for users to monitor and control the smart water fountain remotely.

Notification System: Sends alerts and notifications to users based on the data (e.g., low water levels, filter replacement alerts).

Data Visualization: The data stored in the cloud platform can be visualized using tools like dashboards, graphs, or charts.

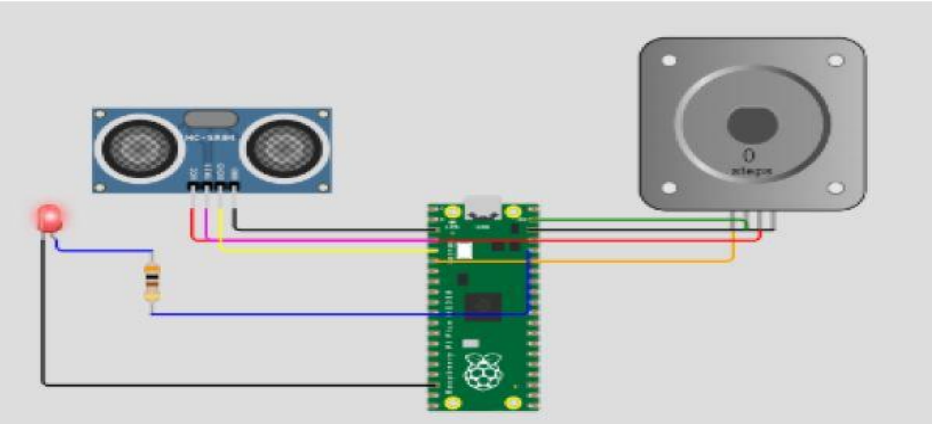
Data Sharing: Users can access the platform via a secure login to view real-time data and historical records of water quality and usage. They can also control the fountain remotely.

Code implementation:

```
import RPi.GPIO as GPIO

Import time
```

```
Import paho.mqtt.client as mqtt
# GPIO setup
GPIO.setmode(GPIO.BCM)
Water_pump_pin = 17
GPIO.setup(water_pump_pin, GPIO.OUT)
# MQTT setup
Mqtt_broker = “your_broker_address”
Mqtt_topic = “water_fountain/control”
Client = mqtt.Client()
# Callback for MQTT message received
Def on_message(client, userdata, message):
If message.payload.decode() == “on”:
GPIO.output(water_pump_pin, GPIO.HIGH)
Else:
GPIO.output(water_pump_pin, GPIO.LOW)
# Connect to MQTT broker
Client.on_message = on_message
Client.connect(mqtt_broker)
Client.subscribe(mqtt_topic)
Client.loop_start()
Try:
While True:
```

```
# Read water level and temperature sensors
Water_level = … # Read water level from sensor
Temperature = … # Read temperature from sensor
# Your logic for controlling the water pump based on sensor readings
If water_level < threshold:
# Water level is too low, turn off the pump
GPIO.output(water_pump_pin, GPIO.LOW)
# Publish sensor data to the MQTT broker
Client.publish(“water_fountain/data”, f”Water Level: {water_level}, Temperature:
{temperature}”)
# Add any other functionality or logic here
Time.sleep(5) # Adjust the interval as needed
Except KeyboardInterrupt:
GPIO.cleanup()
```



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 (C) of an NPN transistor or use a motor driver module to control the pump.
- Connect the emitter (E) 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).

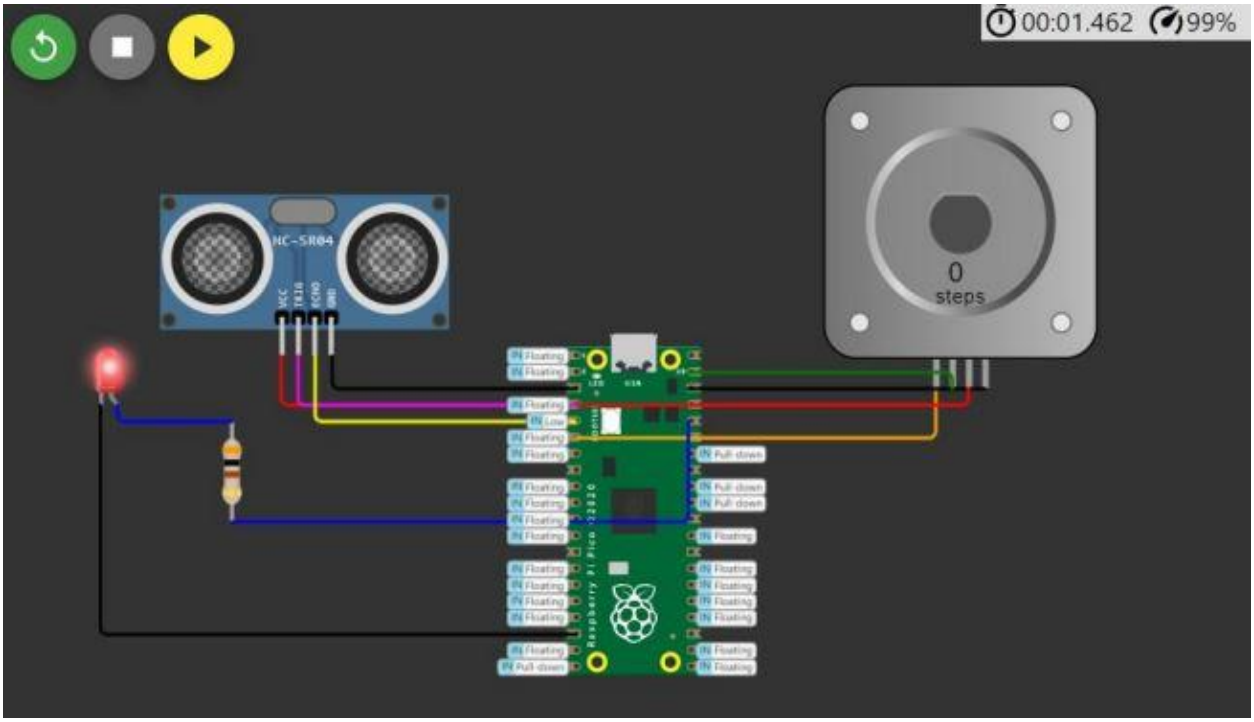
3. LED (with Resistor):

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

RaspberryPi.

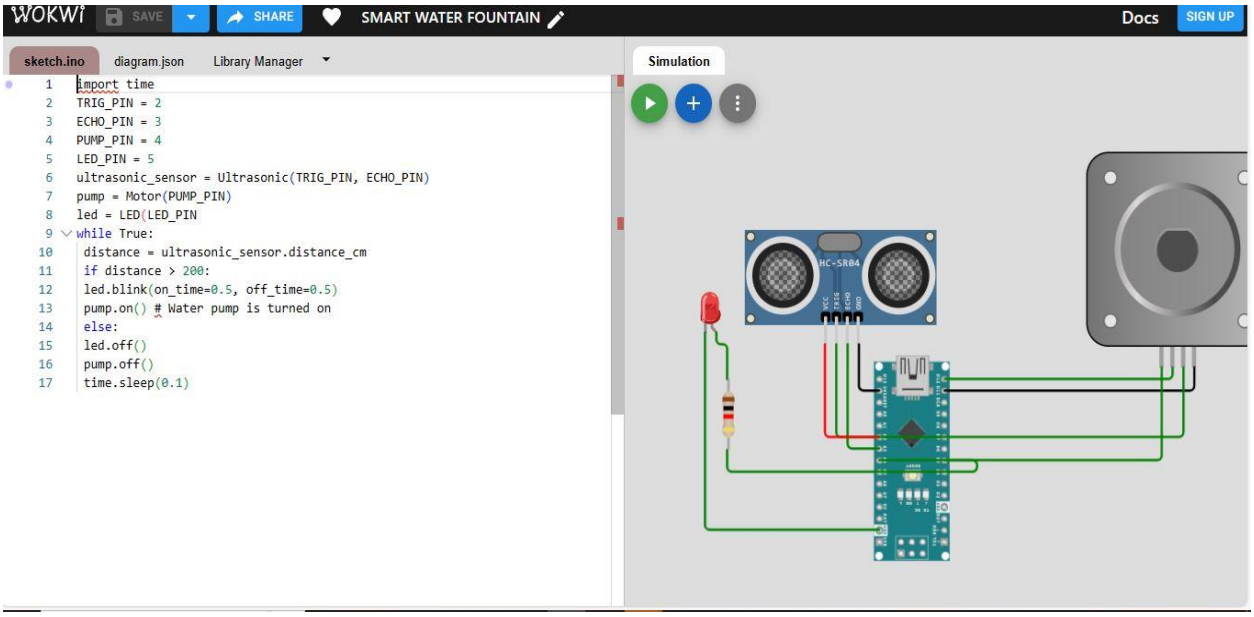
▪ Connect the shorter leg (cathode) of the LED directly to a GND (ground) pin on the Raspberry Pi.

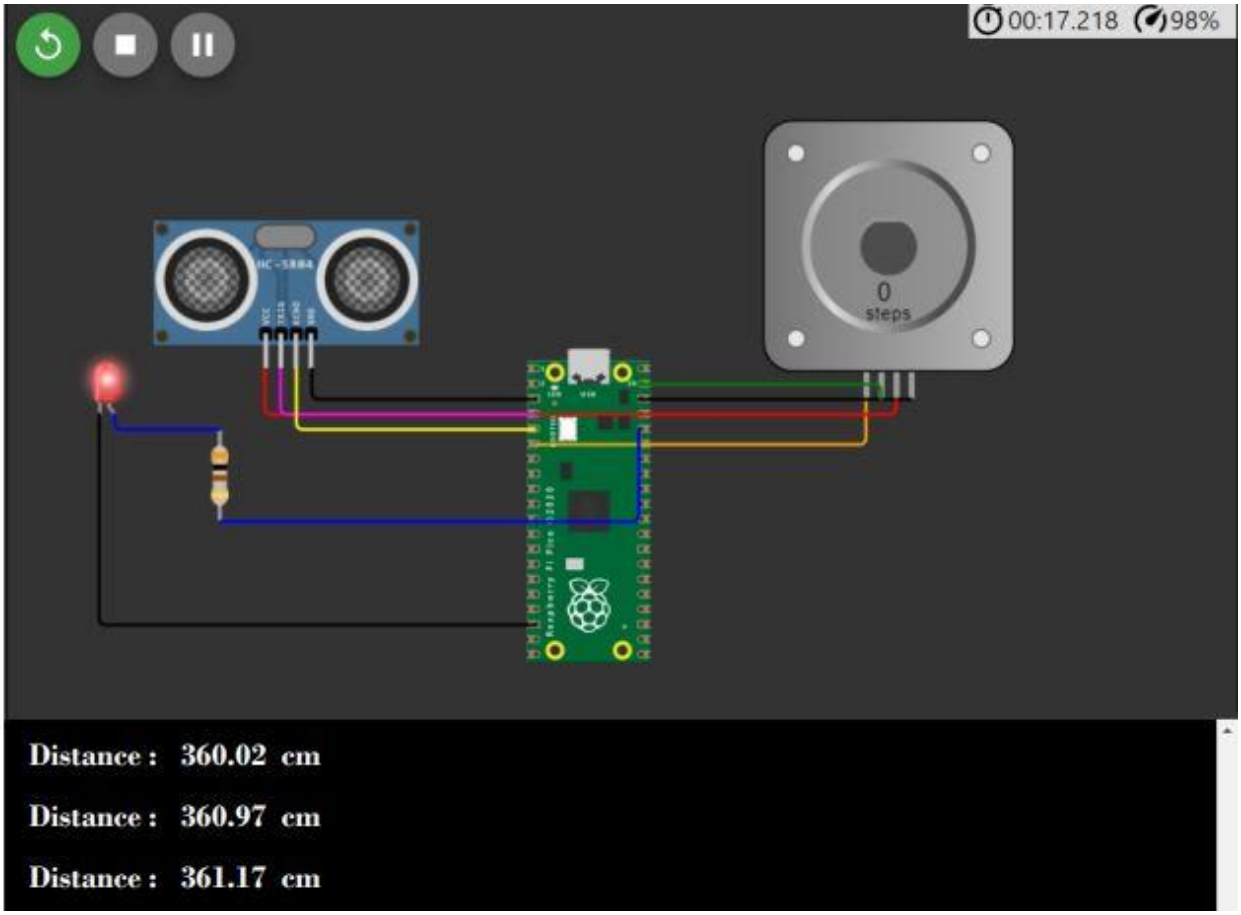


CODE DESCRIPTION:

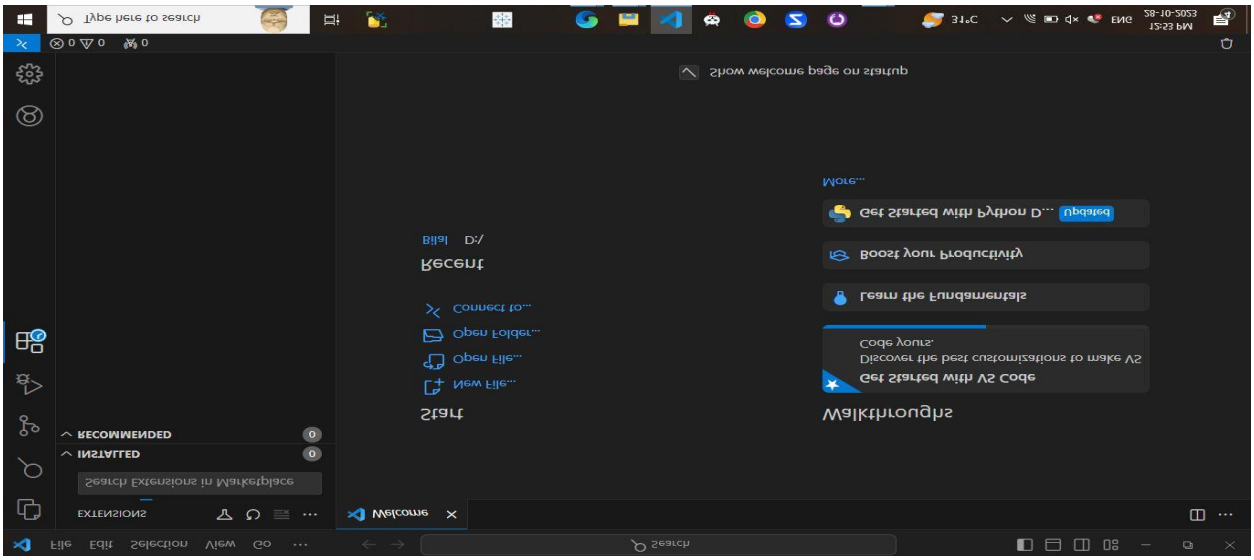
```
import time

TRIG_PIN = 2

ECHO_PIN = 3

PUMP_PIN = 4

LED_PIN = 5

ultrasonic_sensor = Ultrasonic(TRIG_PIN, ECHO_PIN)

pump = Motor(PUMP_PIN)
```

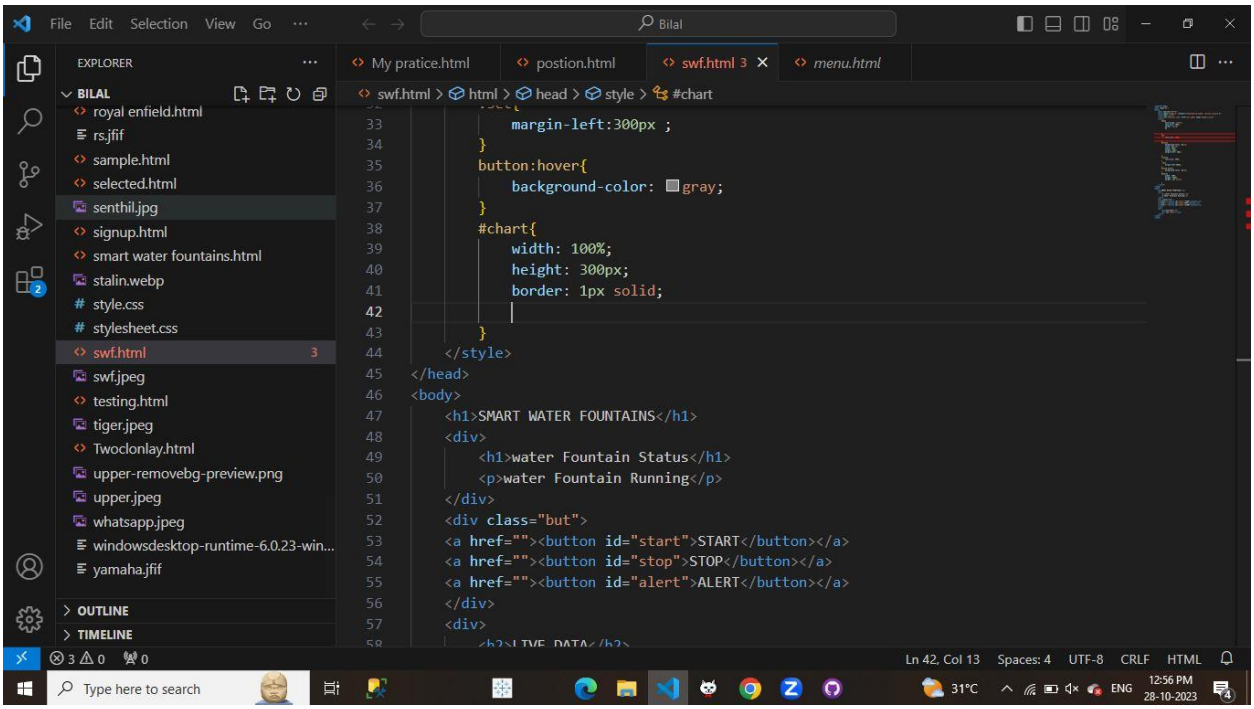
```
led = LED(LED_PIN
while True:
 distance = ultrasonic_sensor.distance_cm
 if distance > 200:
 led.blink(on_time=0.5, off_time=0.5)
 pump.on() # Water pump is turned on
 else:
 led.off()
 pump.off()
 time.sleep(0.1)
```





VISUAL STUDIO CODE:

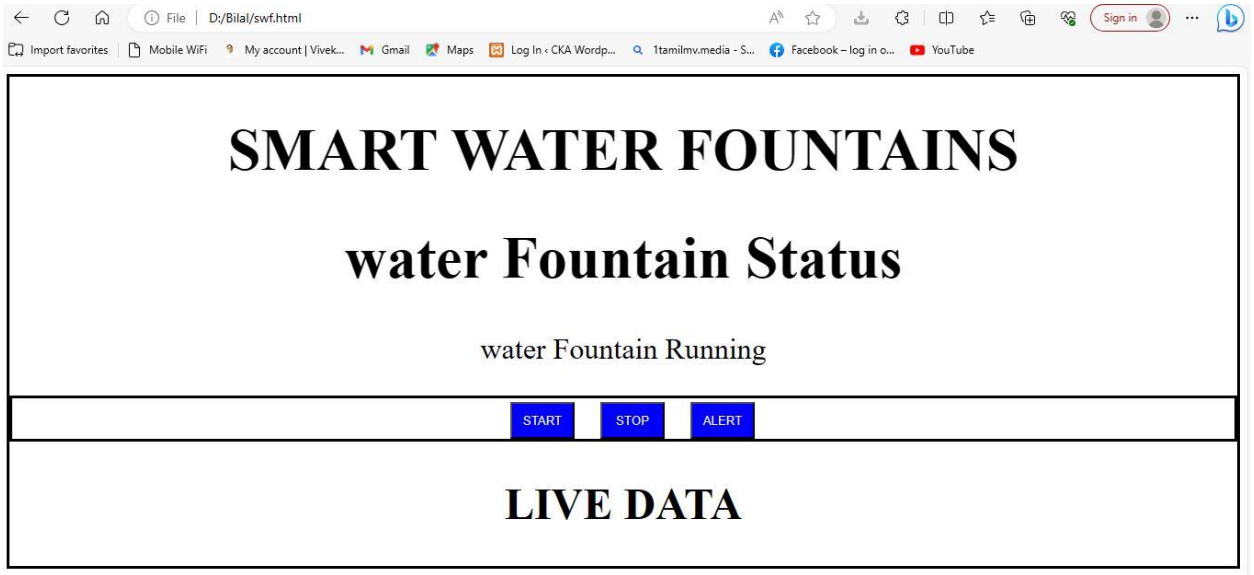



File Edit Selection View Go
Bilal
EXPLORER
BILAL
royal enfield.html
rs.jfif
sample.html
selected.html
senthil.jpg
signup.html
smart water fountains.html
stalin.webp
style.css
stylesheet.css
swf.html
swf.jpeg
testing.html
tiger.jpeg
Twoclonlay.html
upper-removebg-preview.png
upper.jpeg
whatsapp.jpeg
windowsdesktop-runtime-6.0.23-win...
yamaha.jfif
OUTLINE
TIMELINE
My pratice.html
postion.html
swf.html 3
menu.html
swf.html > html > head > style > #chart
margin-left:300px ;
}
button:hover{
background-color: gray;
}
#chart{
width: 100%;
height: 300px;
border: 1px solid;
}
</style>
</head>
<body>
<h1>SMART WATER FOUNTAINS</h1>

<h1>water Fountain Status</h1>
water Fountain Running

<button id="start">START</button>
<button id="stop">STOP</button>
<button id="alert">ALERT</button>

Ln 42, Col 13 Spaces: 4 UTF-8 CRLF HTML
Type here to search
31°C
ENG
12:56 PM
28-10-2023


File | D:/Bilal/swf.html
Sign in
Import favorites
Mobile WiFi
My account | Vivek...
Gmail
Maps
Log In ‹ CKA Wordp...
1tamilmv.media - S...
Facebook – log in o...
YouTube
SMART WATER FOUNTAINS
water Fountain Status
water Fountain Running
START
STOP
ALERT
LIVE DATA