Phase 5:Project Documentation & Submission

IOT:

1. **IoT Device Setup:**
    - **Objective 1:** Develop or select appropriate IoT devices and sensors for water quality monitoring, usage tracking, and touchless dispensing.
    - **Objective 2:** Ensure seamless integration of IoT devices into the water fountain system, enabling real-time data collection and communication with the central platform.
2. **Platform Development:**
    - **Objective 3:** Create a centralized platform that collects and processes data from the IoT devices. This platform could be cloud-based and capable of handling large amounts of data.
    - **Objective 4:** Implement a user-friendly interface accessible via web or mobile for users to interact with the fountain, check water quality, receive updates, and customize settings.
    - **Objective 5:** Develop an analytics dashboard for administrators to monitor fountain usage, water quality trends, and system performance.
3. **Code Implementation:**
    - **Objective 6:** Develop the necessary firmware and software for IoT devices to collect, transmit, and manage data effectively.
    - **Objective 7:** Implement security measures to protect the data and communication between

devices and the platform, ensuring user privacy and system integrity.

- **Objective 8:** Create an API or SDK for potential third-party integration, allowing for future expansions or collaborations with other systems.

4. **Integration and Testing:**
   - **Objective 9:** Integrate the IoT devices with the developed platform and ensure seamless communication and data exchange.
   - **Objective 10:** Perform extensive testing for reliability, accuracy, and security of the entire system, both in controlled environments and real-world scenarios.
5. **Scalability and Maintenance:**
   - **Objective 11:** Design the system with scalability in mind, ensuring it can accommodate additional features or an increased number of connected fountains in the future.
   - **Objective 12:** Develop a maintenance plan to regularly update software, replace components, and ensure the system's long-term functionality.

SMART WATER FOUNTAION

1. Planning and Conceptualization:

- **Project Scope Definition:** Define the objectives and scope of the smart water fountain project. Determine features like touchless operation, water quality monitoring, and user interaction capabilities.

- **Requirement Gathering:** Identify the required components, including IoT devices, sensors, and connectivity technology.

2. Design and Component Selection:

- **Select IoT Devices and Sensors:** Choose appropriate sensors for water quality (pH, turbidity, temperature), flow sensors, and motion sensors for touchless operation.
- **Device Architecture Design:** Plan the connection and communication structure between IoT devices, the fountain, and the central IoT platform.

3. IoT Device Setup:

- **Acquisition and Installation:** Procure the selected IoT devices and sensors. Install and configure them within the water fountain structure.
- **Integration:** Ensure proper integration and functionality of the devices for seamless data collection and transmission.

4. IoT Platform Development:

- **Backend Infrastructure:** Develop a robust cloud-based backend infrastructure to handle the data generated by the fountain’s IoT devices.
- **Database Setup:** Establish a database structure for storing and managing the sensor data.
- **User Interface Development:** Create a user-friendly web or mobile interface for user interaction and data visualization.

5. Code Implementation:

- **Firmware Development:** Develop firmware for the IoT devices to collect, process, and transmit data efficiently.
- **Platform Software Development:** Build software for the platform to manage and display data for users and administrators.

6. Integration and Testing:

- **Devices-Platform Integration:** Ensure seamless communication between the IoT devices and the central platform.
- **Thorough Testing:** Conduct comprehensive testing to ensure reliability, accuracy, and security of the system, including stress testing and real-world simulations.

7. Deployment and User Experience:

- **Installation:** Install the smart water fountain in the intended location.
- **User Training:** Educate users about the features and functionality of the fountain.
- **Feedback Collection:** Gather user feedback for iterative improvements.

8. Maintenance and Upkeep:

- **Scheduled Maintenance:** Create a maintenance schedule for regular updates, component replacements, and system checks to maintain the fountain's functionality.
- **Security Updates:** Continuously monitor and update security measures to protect user data and the system from potential vulnerabilities.

9. Documentation and Scaling:

- **Documentation:** Record the development process, component details, codes, and maintenance protocols for future reference and troubleshooting.
- **Scalability Plan:** Prepare the system for potential scalability, considering additional features and expanding the network to accommodate more fountains.

10. Sustainability and Environmental Impact:

- **Water Conservation Analysis:** Utilize the collected data to analyze water usage patterns and identify areas for conservation.
- **Energy Efficiency Measures:** Implement energy-saving practices, potentially integrating renewable energy sources for a sustainable operation

Smart Water Fountain using the Wokwi simulator

code: main.py

main.py diagram.json Library Manager

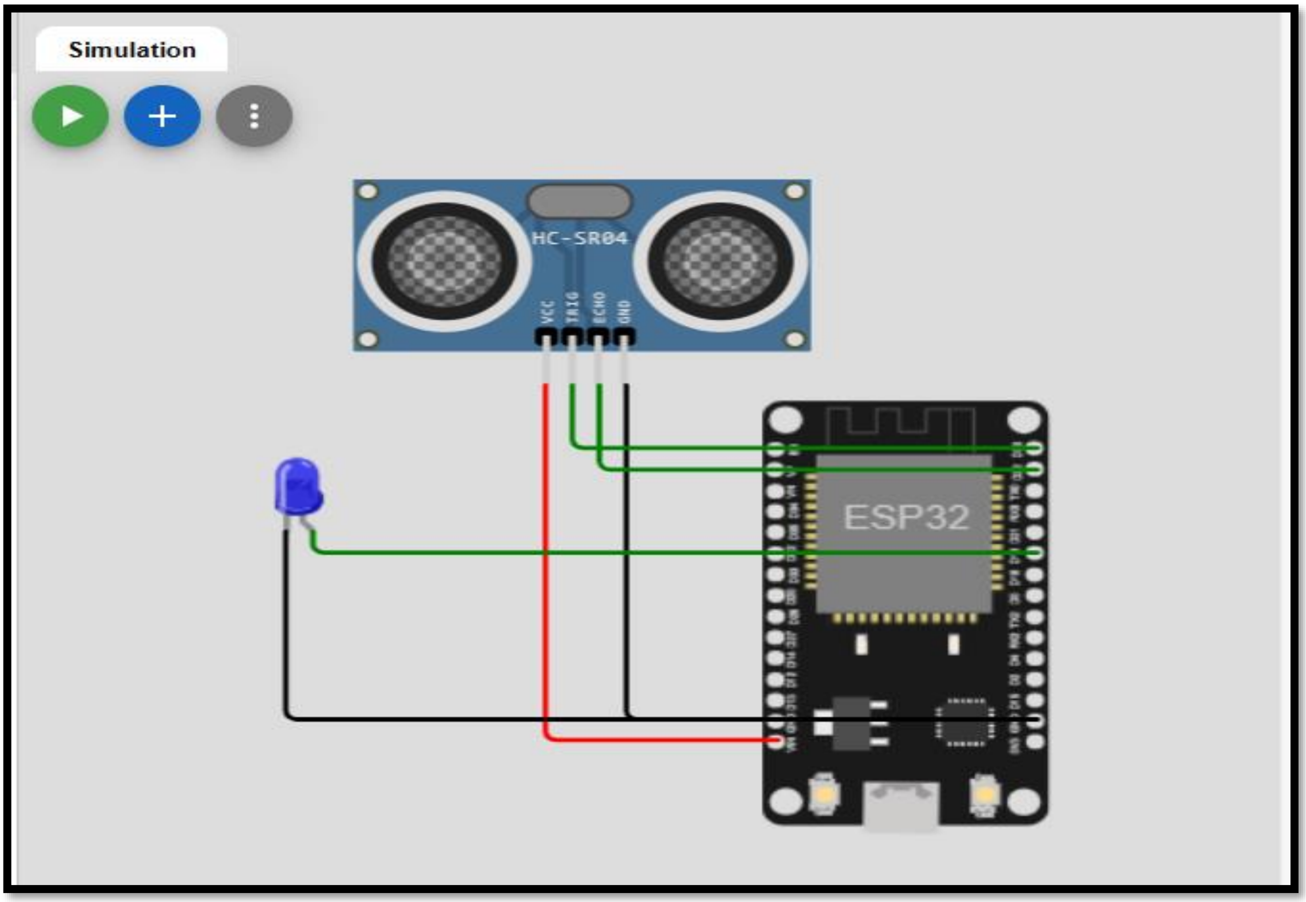
```
import machine
import time

# Pin assignments for the ultrasonic sensor
TRIGGER_PIN = 23  # GPIO23 for trigger
ECHO_PIN = 22     # GPIO22 for echo

# Pin assignment for the LED
LEAK_LED_PIN = 19  # GPIO19 for the LED

# Set the pin modes
trigger = machine.Pin(TRIGGER_PIN, machine.Pin.OUT)
echo = machine.Pin(ECHO_PIN, machine.Pin.IN)
leak_led = machine.Pin(LEAK_LED_PIN, machine.Pin.OUT)

# Function to measure distance using the ultrasonic sensor
def measure_distance():
    # Generate a short trigger pulse
    trigger.value(0)
    time.sleep_us(5)
    trigger.value(1)
    time.sleep_us(10)
    trigger.value(0)

    # Measure the echo pulse duration to calculate distance
    pulse_start = pulse_end = 0
    while echo.value() == 0:
        pulse_start = time.ticks_us()
    while echo.value() == 1:
        pulse_end = time.ticks_us()

    pulse_duration = pulse_end - pulse_start

    # Calculate distance in centimeters (assuming the speed of sound is 343 m/s)
    distance = (pulse_duration * 0.0343) / 2  # Divide by 2 for one-way travel

    return distance

# Function to check for a water leak
def check_for_leak():
    # Measure the distance from the ultrasonic sensor
    distance = measure_distance()

    # Set the threshold distance for detecting a leak (adjust as needed)
    threshold_distance = 10  # Adjust this value based on your tank setup

    if distance < threshold_distance:
        # If the distance is less than the threshold, a leak is detected
        return True
    else:
        return False

# Main loop
while True:
    if check_for_leak():
        # Blink the LED to indicate a leak
        leak_led.value(1)  # LED ON
        time.sleep(0.5)
        leak_led.value(0)  # LED OFF
        time.sleep(0.5)
    else:
        leak_led.value(0)  # LED OFF

    time.sleep(1)  # Delay between measurements

```

Sdiagram.json

main.py | diagram.json | Library Manager

```
{
  "version": 1,
  "author": "Uri Shaked",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -14.5, "left": 81.4, "attrs": {} },
    { "type": "wokwi-led", "id": "led2", "top": 6, "left": -111.4, "attrs": { "color": "blue" } },
    { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": -113.7, "left": -71.3, "attrs": {} }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "ultrasonic1:GND", "esp:GND.2", "black", [ "v0" ] ],
    [ "ultrasonic1:ECHO", "esp:D22", "green", [ "v0" ] ],
    [ "ultrasonic1:TRIG", "esp:D23", "green", [ "v0" ] ],
    [ "ultrasonic1:VCC", "esp:VIN", "red", [ "v0" ] ],
    [ "led2:A", "esp:D19", "green", [ "v0" ] ],
    [ "led2:C", "esp:GND.1", "black", [ "v0" ] ]
  ],
  "dependencies": {}
}
```

CIRCUIT :-



STIMULATION OUTPUT:-