

Project Title: Smart Water Fountains

Project Overview:

The "Smart Water Fountains" project aims to create an intelligent and environmentally friendly system for managing and enhancing the functionality of water fountains in public spaces, parks, and commercial establishments. This project combines IoT technology, data analytics, and user interaction to achieve water conservation, efficient maintenance, and an improved user experience.

Objectives:

Water Conservation: The primary goal of this project is to optimize water usage in public fountains by monitoring water levels and adjusting water flow accordingly to minimize wastage.

User Interaction: Implement user-friendly interfaces that allow users to interact with the fountains, trigger special effects, and receive information about water quality and temperature.

Data-Driven Maintenance: Utilize data analytics to predict maintenance needs and improve the longevity of fountain components.

Energy Efficiency: Implement energy-efficient water pumps and valves to reduce power consumption.

Data Sharing: Create a data-sharing platform to monitor and control the fountains remotely and collect data for analysis.

Hardware Components:

Raspberry Pi or Arduino: A microcontroller serves as the brain of the smart fountain, controlling sensors, actuators, and data transmission.

Water Level Sensors: Capacitive or ultrasonic sensors to measure water levels in the fountain.

Temperature Sensors: Waterproof temperature sensors to monitor water temperature.

Proximity Sensors: Ultrasonic or infrared sensors to detect user presence.

Submersible Water Pumps: To control water flow and create different fountain effects.

Solenoid Valves: For regulating water flow.

Wi-Fi Module: For internet connectivity.

Power Supply: Stable 5V power supply and battery backup for power redundancy.

Enclosures: Weatherproof enclosures to protect hardware components from environmental factors.

Software Configuration:

Programming Environment: Raspberry Pi or Arduino configured to run Raspbian or similar OS, with code written in Python or C/C++.

Sensor Data Acquisition: Python scripts to read sensor data through GPIO pins or sensor-specific libraries.

Actuator Control: Code for controlling water pumps and valves based on sensor readings and user input.

Data Preprocessing: Minimal data preprocessing for data formatting and calibration.

Data Transmission:

Communication Protocol: MQTT for lightweight and efficient data transmission.

Data Security: MQTT communication is encrypted with TLS/SSL and includes authentication mechanisms to ensure data security.

Platform Development:

Cloud-Based Platform: Hosted on cloud services like AWS or Azure.

Server Setup: A web server (Nginx or Apache), a message broker (Mosquitto MQTT), and a database server (MySQL) for data storage.

Database Configuration:

Database Technology: MySQL is chosen for its reliability and scalability.

Database Schema: Tables for water level, temperature, user interactions, and timestamps.

Data Storage Strategy: Historical data retention for analysis and monitoring.

User Interface (UI):

Screenshots and Diagrams: User-friendly web-based dashboard for monitoring and controlling fountains.

User Interaction Flow: Users can view real-time data, control water flow, and receive information about water quality.

Data Visualization: Real-time charts and graphs for visualizing sensor data.

Data Analytics:

Analytics Models: Predictive maintenance models to predict component failures based on historical data.

Data Processing: Data preprocessing for missing values and feature engineering.

Output Presentation: Display results and alerts on the platform's user interface.

Implementation and Workflow:

IoT Device Setup: Hardware components are assembled, and Raspberry Pi (or Arduino) is configured with sensor and actuator controls.

Data Collection: Sensors continuously monitor water levels, temperature, and user presence.

User Interaction: Proximity sensors detect users, allowing them to interact with the fountains through the platform.

Data Transmission: Sensor data is published to the MQTT broker, ensuring real-time updates to the platform.

Data Ingestion: The data-sharing platform subscribes to MQTT topics, ingests sensor data, and stores it in the MySQL database.

Data Visualization: Real-time data is presented to users through the web-based UI, providing insights into water levels and quality.

User Control: Users can manually control water flow and initiate special effects through the platform's interface.

Data Analytics: Historical data is analyzed to predict maintenance needs, improving fountain longevity and efficiency.

Alerts and Notifications: Users are notified of maintenance predictions and any anomalies in fountain behaviour.

Energy Efficiency: The system optimizes water pump and valve operations to reduce power consumption.

Smart Water Fountain using the Wokwi simulator

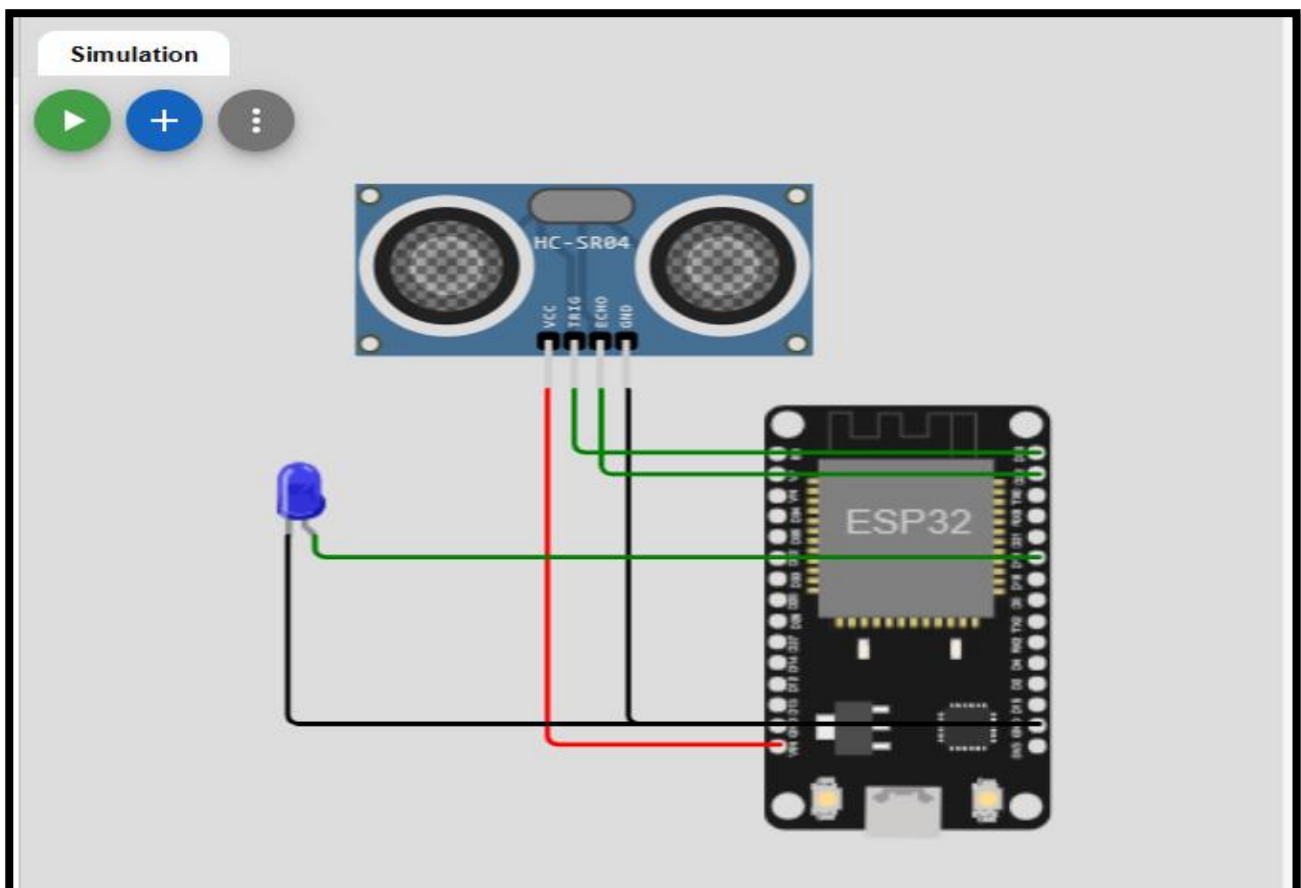
code: main.py

```
main.py  diagram.json  Library Manager  ▼
1  import machine
2  import time
3
4  # Pin assignments for the ultrasonic sensor
5  TRIGGER_PIN = 23 # GPIO23 for trigger
6  ECHO_PIN = 22    # GPIO22 for echo
7
8  # Pin assignment for the LED
9  LEAK_LED_PIN = 19 # GPIO19 for the LED
10
11 # Set the pin modes
12 trigger = machine.Pin(TRIGGER_PIN, machine.Pin.OUT)
13 echo = machine.Pin(ECHO_PIN, machine.Pin.IN)
14 leak_led = machine.Pin(LEAK_LED_PIN, machine.Pin.OUT)
15
16 # Function to measure distance using the ultrasonic sensor
17 def measure_distance():
18     # Generate a short trigger pulse
19     trigger.value(0)
20     time.sleep_us(5)
21     trigger.value(1)
22     time.sleep_us(10)
23     trigger.value(0)
24
25     # Measure the echo pulse duration to calculate distance
26     pulse_start = pulse_end = 0
27     while echo.value() == 0:
28         pulse_start = time.ticks_us()
29     while echo.value() == 1:
30         pulse_end = time.ticks_us()
31
32     pulse_duration = pulse_end - pulse_start
33
34     # Calculate distance in centimeters (assuming the speed of sound is 343 m/s)
35     distance = (pulse_duration * 0.0343) / 2 # Divide by 2 for one-way travel
36
37     return distance
38
39 # Function to check for a water leak
40 def check_for_leak():
41     # Measure the distance from the ultrasonic sensor
42     distance = measure_distance()
43
44     # Set the threshold distance for detecting a leak (adjust as needed)
45     threshold_distance = 10 # Adjust this value based on your tank setup
46
47     if distance < threshold_distance:
48         # If the distance is less than the threshold, a leak is detected
49         return True
50     else:
51         return False
52
53 # Main loop
54 while True:
55     if check_for_leak():
56         # Blink the LED to indicate a leak
57         leak_led.value(1) # LED ON
58         time.sleep(0.5)
59         leak_led.value(0) # LED OFF
60         time.sleep(0.5)
61     else:
62         leak_led.value(0) # LED OFF
63
64     time.sleep(1) # Delay between measurements
65
```

Sdiagram.json

```
main.py  diagram.json  Library Manager
1  {
2    "version": 1,
3    "author": "Uri Shaked",
4    "editor": "wokwi",
5    "parts": [
6      { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -14.5, "left": 81.4, "attrs": {} },
7      { "type": "wokwi-led", "id": "led2", "top": 6, "left": -111.4, "attrs": { "color": "blue" } },
8      { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": -113.7, "left": -71.3, "attrs": {} }
9    ],
10   "connections": [
11     [ "esp:TX0", "$serialMonitor:RX", "", [] ],
12     [ "esp:RX0", "$serialMonitor:TX", "", [] ],
13     [ "ultrasonic1:GND", "esp:GND.2", "black", [ "v0" ] ],
14     [ "ultrasonic1:ECHO", "esp:D22", "green", [ "v0" ] ],
15     [ "ultrasonic1:TRIG", "esp:D23", "green", [ "v0" ] ],
16     [ "ultrasonic1:VCC", "esp:VIN", "red", [ "v0" ] ],
17     [ "led2:A", "esp:D19", "green", [ "v0" ] ],
18     [ "led2:C", "esp:GND.1", "black", [ "v0" ] ]
19   ],
20   "dependencies": {}
21 }
```

CIRCUIT :-



STIMULATION OUTPUT:-

main.py

diagram.json

Library Manager

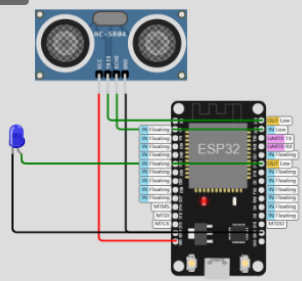
```
1 import machine
2 import time
3
4 # Pin assignments for the ultrasonic sensor
5 TRIGGER_PIN = 23 # GPIO23 for trigger
6 ECHO_PIN = 22    # GPIO22 for echo
7
8 # Pin assignment for the LED
9 LEAK_LED_PIN = 19 # GPIO19 for the LED
10
11 # Set the pin modes
12 trigger = machine.Pin(TRIGGER_PIN, machine.Pin.OUT)
13 echo = machine.Pin(ECHO_PIN, machine.Pin.IN)
14 leak_led = machine.Pin(LEAK_LED_PIN, machine.Pin.OUT)
15
16 # Function to measure distance using the ultrasonic sensor
17 def measure_distance():
18     # Generate a short trigger pulse
19     trigger.value(0)
20     time.sleep_us(5)
21     trigger.value(1)
22     time.sleep_us(10)
23     trigger.value(0)
24
25     # Measure the echo pulse duration to calculate distance
26     pulse_start = pulse_end = 0
27     while echo.value() == 0:
28         pulse_start = time.ticks_us()
29     while echo.value() == 1:
30         pulse_end = time.ticks_us()
```

Simulation

00:45.079

99%

Resume



```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4728
load:0x40078000,len:14876
ho 0 tail 12 room 4
load:0x40080400,len:3368
entry 0x400805cc
```

The "Smart Water Fountains" project represents a significant advancement in the management and functionality of public fountains, incorporating IoT technology, data analytics, and user interaction to achieve water conservation, efficient maintenance, and an improved user experience.

This project's success extends beyond the efficient management of resources. It provides an exemplar of how technology can be harnessed to enrich public spaces and engage communities. The data-driven approach not only conserves water but also offers opportunities for educational and artistic experiences.

In conclusion, the "Smart Water Fountains" project has not only demonstrated the feasibility of an innovative and sustainable approach to fountain management but also paves the way for further exploration of IoT applications in public spaces. By embracing modern technology, we contribute to a more responsible and enjoyable urban environment while setting the stage for future advancements in IoT and environmental conservation.