Application

Description automatically generated with low confidence

**SMART WATER FOUNTAINS USING IoT**



**Prepared By,**

Saravanan N,

Meiyarasan S,

Kaviyarasu G,

Ganeshmoorthy E

**Project Description:**

The IoT-enabled Smart Water Fountains System aims to enhance public water fountain efficiency and reliability by deploying IoT sensors to monitor water flow and detect malfunctions in real-time. This project will involve deploying various sensors like flow rate sensors and pressure sensors in public water fountains and developing a Python script to send real-time status data to a central platform.

**Project Stages:**

**Stage 1: Sensor Deployment and Setup**

1.Select Sensors: Choose appropriate flow rate sensors and pressure sensors based on the requirements and compatibility with the water fountains.

2.Install Sensors: Deploy the selected sensors in public water fountains, ensuring they are securely attached and calibrated for accurate measurements.

3.Connect Sensors to Microcontroller: Connect the sensors to a microcontroller (e.g., Raspberry Pi or Arduino) that will process the sensor data.

**Stage 2: Develop Python Script for IoT Sensors**

1. Set Up Microcontroller: Configure the microcontroller and establish a secure connection to the internet via Wi-Fi or Ethernet.

2. Write Python Script: Develop a Python script that reads data from the connected sensors.

**Program Coding**

# Import necessary libraries for sensor reading and communication import sensor library # Import the library for sensor communication

import requests # Library for making HTTP requests

# Function to read sensor data

def read\_sensor\_data():

flow\_rate = sensor\_library.get\_flow\_rate()

pressure = sensor\_library.get\_pressure()

return flow\_rate, pressure

# Function to send data to the central platform

def send\_data\_to\_platform(data):

endpoint\_url = "https://api.smartwaterfountains.com/data"

response = requests.post(endpoint\_url, json=data)

if response.status\_code == 200:

print("Data sent successfully to the platform.")

else:

print("Failed to send data. Status code:", response.status\_code)

# Main function to read sensor data and send it to the platform

def main():

while True:

flow\_rate, pressure = read\_sensor\_data()

data = {

"flow\_rate": flow\_rate,

"pressure": pressure

}

send\_data\_to\_platform(data)

# Implement a suitable delay based on the required data update frequency

time.sleep(60) # Sends data every minute

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

main()

```

3. Data Encryption and Security: Implement data encryption and security measures to protect the transmitted data from unauthorized access.

**Stage 3: Set Up Centralized Platform**

1. Cloud Server: Set up a cloud server (e.g., AWS, Azure) to receive and store the incoming data from the IoT sensors.

2. Database: Create a database to store the received sensor data securely.

3. API: Develop a RESTful API that accepts incoming data from the sensors and stores it in the database.

4. Data Visualization: Implement data visualization tools to provide real-time insights and historical data analysis through a user-friendly dashboard.

**Stage 4: Testing and Deployment**

1.Testing: Thoroughly test the entire system to ensure accurate sensor readings, data transmission, and platform integration.

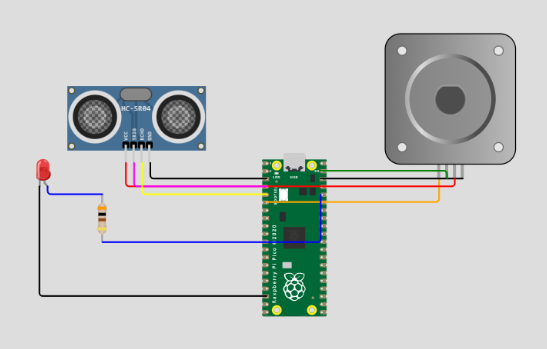
2. Deployment: Deploy the IoT-enabled Smart Water Fountains System in public areas, ensuring proper installation and connectivity.

**Stage 5: Maintenance and Monitoring**

1.Monitoring: Implement a monitoring system to track the performance of sensors and the overall system in real-time.

2.Maintenance: Establish a maintenance schedule to regularly inspect sensors and address any malfunctions promptly.

**Sample Module Smart Water Fountain**

****

import time # Import the time module for time delays

# Define GPIO pin numbers

TRIG\_PIN = 2 # GPIO pin number for the ultrasonic sensor's trigger

ECHO\_PIN = 3 # GPIO pin number for the ultrasonic sensor's echo

PUMP\_PIN = 4 # GPIO pin number for the water pump

LED\_PIN = 5 # GPIO pin number for the LED

# Initialize components (virtual components for Wokwi)

ultrasonic\_sensor = Ultrasonic(TRIG\_PIN, ECHO\_PIN)  # Create an ultrasonic sensor

pump = Motor(PUMP\_PIN)  # Create a water pump

led = LED(LED\_PIN) # Create an LED

while True:

    # Measure distance

    distance = ultrasonic\_sensor.distance\_cm  # Measure distance in centimeters

    if distance > 200:  # Water level is above 200 cm

        # Make the LED blink

        led.blink(on\_time=0.5, off\_time=0.5)  # LED blinks with 0.5 seconds on and off time

        pump.on()  # Water pump is turned on

    else:

        # Water level is below 200 cm

        # Turn off the LED and the pump

        led.off()

        pump.off()

    # Introduce a small delay to control the loop rate

    time.sleep(0.1)  # Sleep for 0.1 seconds

**Conclusion:**

By completing these stages, the IoT-enabled Smart Water Fountains System will provide a reliable and efficient solution for monitoring public water fountains, ensuring their proper functionality, and enhancing user experience. The system will contribute to conserving water resources and promoting sustainable practices in public spaces.