IBM NAN MUDHALVAN –INTERNET OF THINGS

(GROUP-4)

SMART WATER FOUNTAIN

PHASE-III DEVELOPMENT 1

DONE BY:

➢ ARUN KUMAR.R	(au82042106004)
➢ DHINESH.R	(au82042106012)
➢ HARIHARASUDHAN.S	(au82042106014)
➢ MILTON EMIRATE RAJ.A	(au82042106029)
➢ MUKESH.M	(au82042106030)

DEVELOPMENT OF PROJECT BUILDING:

Developing a smart water fountain using IoT (Internet of Things) technology involves integrating sensors, microcontrollers, communication protocols, and mobile or web applications to monitor and control the fountain remotely. Here's a step-by-step guide on how you can develop a smart water fountain:

1. Define the Requirements:

- ✓ Clearly define the objectives and features you want for your smart water fountain. Consider factors like water flow control, water level monitoring, and remote access.

2. Select Hardware Components:

- ✓ Microcontroller: Choose a microcontroller like Arduino, Raspberry Pi, or ESP8266/ESP32 to control the fountain's operation.
- ✓ Water Flow Sensor: Use a flow sensor to measure water flow rate.
- ✓ Water Level Sensor: Use ultrasonic or capacitive sensors to monitor the water level.
- ✓ Pump: Select a suitable water pump for your fountain.
- ✓ Valves: If you want to control the water flow, you'll need valves or solenoids.
- ✓ Power Supply: Ensure a stable power supply for the system.

3. Assemble the Hardware:

- ✓ Connect the selected components according to the circuit diagram.
- ✓ Ensure proper power management and waterproofing where necessary.

4. Develop the Software:

- ✓ Write code for the microcontroller to control the water pump, read sensor data, and communicate with other devices. You can use platforms like Arduino IDE, Python, or Node.js.
- ✓ Implement logic for water level monitoring and flow control.

5. Implement IoT Connectivity:

- ✓ Choose a communication protocol such as Wi-Fi, Bluetooth, Zigbee, or LoRa to connect the fountain to the internet.
- ✓ Set up the microcontroller to send sensor data to a cloud platform or a local server.

6. Cloud Platform Setup:

- ✓ Use IoT cloud platforms like AWS IoT, Google Cloud IoT, or Azure IoT to manage and process the data.
- ✓ Set up devices, data streams, and rules to handle the incoming data.

7. Create a User Interface:

- ✓ Develop a web or mobile application to control the fountain remotely.
- ✓ The app should provide real-time status updates, allow users to turn the fountain on/off, adjust water flow, and monitor water levels.

8. Data Visualization:

- ✓ Implement data visualization tools to display historical and real-time data, such as graphs or dashboards.

9. Security Considerations:

- ✓ Ensure the system is secure, with proper authentication and encryption.
- ✓ Regularly update and patch firmware and software to mitigate security vulnerabilities.

10. Test and Debug:

- ✓ Thoroughly test the smart water fountain for different scenarios and address any issues or bugs.

11. Deployment:

- ✓ Install the smart water fountain in its desired location and connect it to a stable power source and internet connection.

12. Maintenance and Support:

- ✓ Provide ongoing support and maintenance to address any issues and keep the system up to date.

A smart water fountain can be a fun and practical project that showcases the potential of IoT in everyday life. Additionally, it can be integrated into larger smart home or landscape automation systems for a more comprehensive IoT experience.

Python Script for Real-Time Smart Water Fountain:

Creating a real-time smart water fountain system involves both a Python script for control and potentially additional components for data logging and user interface. Below is an example Python script for a real-time smart wate r fountain using an Arduino to control the fountain's operation, a water flow sensor, a water level sensor, and a pump. Note that you'll need to adapt this script to your specific hardware and setup

CODING:

```
import serial

import time

import matplotlib.pyplot as plt

# Initialize the serial connection to the Arduino (change the COM port as
needed)

arduino = serial.Serial('COM3', 9600, timeout=1)

# Initialize data arrays for logging

time_data = []

flow_data = []

level_data = []

# Function to turn the fountain pump on

def turn_on_pump():

arduino.write(b'1')  # Send '1' to Arduino to turn on the pump

# Function to turn the fountain pump off

def turn_off_pump():

arduino.write(b'0')  # Send '0' to Arduino to turn off the pump

# Function to read water flow rate from the sensor

def read_water_flow_rate():

arduino.write(b'2')  # Send '2' to Arduino to request flow rate

return float(arduino.readline().decode().strip())

# Function to read water level from the sensor
```

```
def read_water_level():
arduino.write(b'3')  # Send '3' to Arduino to request water level
return float(arduino.readline().decode().strip())
# Main function
if __name__ == '__main__':
try:
while True:
current_time = time.strftime('%H:%M:%S')
time_data.append(current_time)
# Read water flow rate
flow_rate = read_water_flow_rate()
flow_data.append(flow_rate)
# Read water level
water_level = read_water_level()
level_data.append(water_level)
# Check if the water level is too low, turn on the pump
if water_level < 10:  # Adjust the threshold as needed
turn_on_pump()
else:
turn_off_pump()
# Print and plot the real-time data
print(f'Time: {current_time}, Flow Rate: {flow_rate} L/min, Water Level:
{water_level} cm')
# Plot the data
plt.plot(time_data, flow_data, label='Flow Rate (L/min)')
plt.plot(time_data, level_data, label='Water Level (cm)')
plt.xlabel('Time')
plt.ylabel('Values')
plt.legend()
plt.draw()
plt.pause(1)  # Update the plot every 1 second
except KeyboardInterrupt:
# Close the serial connection and exit gracefully
arduino.close()
```

In this script:

- ✓ We import the necessary libraries for serial communication, time tracking, and real-time plotting.
- ✓ Initialize the serial connection to the Arduino. Make sure to update the COM port based on your setup.
- ✓ Define functions for turning the pump on and off, reading water flow rate, and reading water level.
- ✓ In the main loop, we continuously read the water flow rate and water level, log the data, and update a real-time plot.
- ✓ If the water level falls below a certain threshold (10 cm in this example), the pump is turned on.
- ✓ The script runs in an infinite loop and updates the plot every 1 second.
- ✓ Please note that this script provides real-time monitoring and visualization of the fountain's water flow rate and water level. You can customize it further to include data logging, alerts, and other features based on your specific requirements.