DEVELOPMENT PART-1

SMART WATER FOUNTAINS

Building an IoT-enabled Smart Water Fountains system involves multiple steps. Here's a high-level overview of the process, and I'll provide a Python code snippet for sending real-time data from IoT sensors to a platform:

1. **IoT Sensor Selection**: Choose the appropriate IoT sensors like flow rate sensors and pressure sensors that suit your application and the specific water fountain's needs.

2. **IoT Hardware Selection**: Select the right hardware components, such as microcontrollers (e.g., Raspberry Pi, Arduino), communication modules, and power sources for the IoT sensors.

3. **Sensor Integration**: Connect the selected sensors to the microcontroller, ensuring proper wiring and configuration.

4. **Python Script for IoT Sensors**: Here's a simplified Python script example to read data from sensors and send it to a platform using MQTT (Message Queuing Telemetry Transport), a common IoT communication protocol:

```python
Import paho.mqtt.client as mqtt
Import time

# Initialize MQTT client
Client = mqtt.Client("WaterFountainSensor")
Broker_address = "your_mqtt_broker_address"
Port = 1883

# Connect to the MQTT broker
Client.connect(broker_address, port)
```

```
While True:
    # Read data from sensors (replace with your actual sensor reading code)
    Flow_rate = 0.5  # Example flow rate reading
    Pressure = 30.0  # Example pressure reading

    # Create a JSON payload with the sensor data
    Sensor_data = {
        "flow_rate": flow_rate,
        "pressure": pressure
    }

    # Publish sensor data to a specific MQTT topic
    Client.publish("water_fountain/data", json.dumps(sensor_data))

    Time.sleep(60)  # Adjust the timing based on your requirements (e.g., every minute)
```

5. **Data Collection Platform** - Set up an MQTT broker or use an existing IoT platform (e.g., AWS IoT, Google Cloud IoT Core) to receive and store data from your IoT sensors.

- Subscribe to the "water_fountain/data" topic to collect incoming sensor data.

6. **Data Processing and Monitoring**:Implement data processing to monitor water flow and detect malfunctions. This could involve setting thresholds and triggers for alerts when data falls outside acceptable ranges.

7. **User Interface**: Create a user-friendly dashboard or web interface to visualize real-time and historical data from the water fountains.

8. **Alerts and Notifications**:Implement a notification system to alert relevant personnel or authorities in case of malfunctions or anomalies.

9. **Power Management**:Ensure your IoT sensors have a reliable source of power, which may involve using batteries, solar panels, or a local power source.

10. **Security**: Implement security measures to protect the IoT system from unauthorized access and data breaches. This includes securing MQTT communication and the data storage platform.

Remember that this is a simplified example, and a production system may require more complex code and considerations for scalability, reliability, and robustness. Additionally, take into account the privacy and legal aspects of collecting data from public areas and ensure compliance with relevant regulations.