**SMART WATER MANAGEMENT**

**IoT\_Phase5**

Reg no:610821106003

Name: Afrin.G

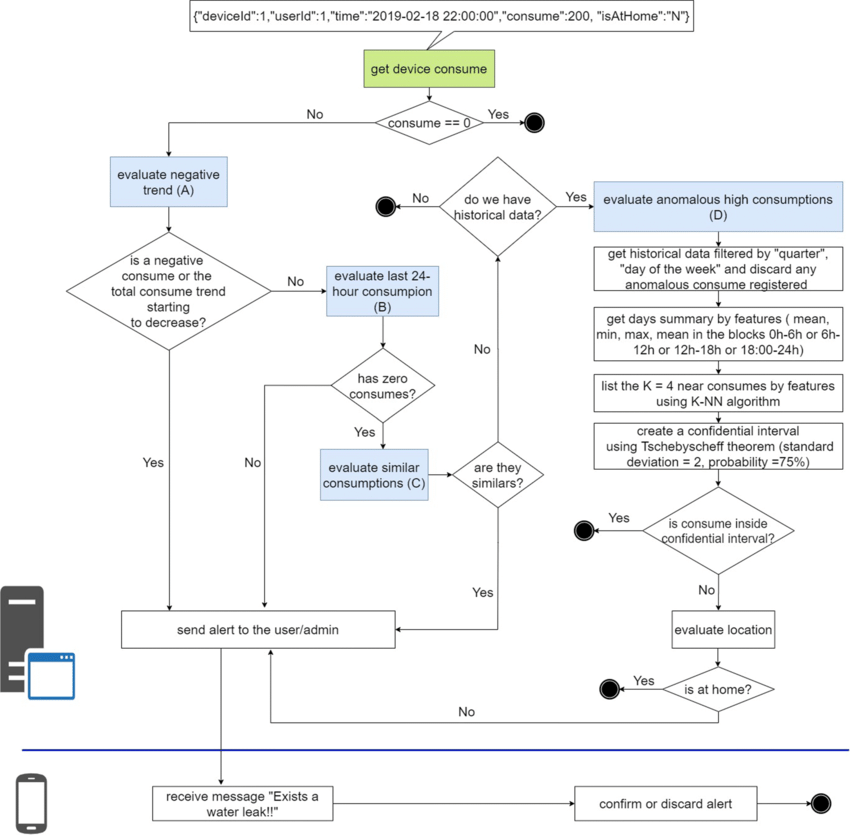
**Objectives:**

* To stop wastage of water due to overflow and leakage.
* To monitor real-time water consumption.
* To create public awareness.
* To enhance the sustainable resource management.

**Water Detection Algorithm:**

The algorithm detects the possible existence of a water leak considering four scenarios, for this it takes the input parameters:

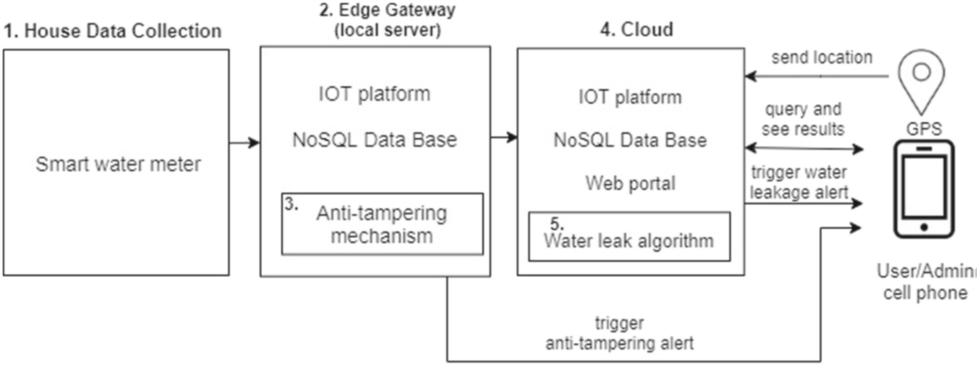
* Device ID
* User ID
* Time T2
* Consumption In T2
* The Location
* Scenario A verifies if the consumption received has a negative value or the total consumption accumulated in the last 24 h has a negative trend; This could be due to failures in the smart meters when capturing consumption.
* Scenario B verifies if there is a continuous flow of water consumption in the last 24 h, since there was no consumption at any zero time, which is highly unlikely for normal consumption.
* Scenario C verifies if there is a high consumption outside its historical behavior.

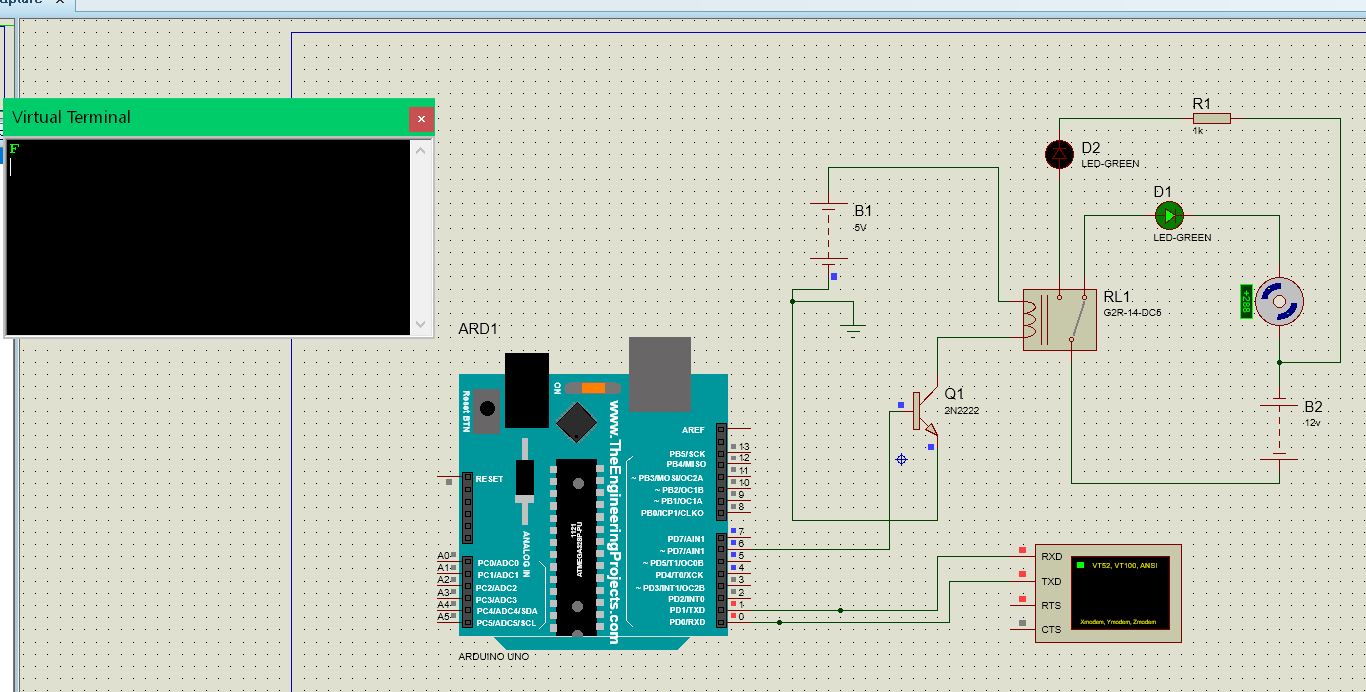


**IoT Device setup:**

The below figure shows the five main components of the system, which allow the collection, storage, analysis and visualization of water consumption.

**Code implementation:**

Python script on IoT sensors to send real-time water consumption data to the data-sharing platform.



import paho.mqtt.client as mqtt

import json

import time

from random import uniform

# Define MQTT parameters

broker\_address = "your\_broker\_address"

port = 1883

topic = "water\_consumption"

# Function to simulate water consumption data

def generate\_water\_data():

return {"timestamp": int(time.time()), "flow\_rate": round(uniform(0.5, 5.0), 2)}

# Callback when the client connects to the broker

def on\_connect(client, userdata, flags, rc):

print("Connected with result code "+str(rc))

client.subscribe(topic)

# Callback when a message is published to the topic

def on\_publish(client, userdata, mid):

print("Message Published")

# Main script

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_publish = on\_publish

# Connect to the broker

client.connect(broker\_address, port, 60)

try:

while True:

water\_data = generate\_water\_data()

payload = json.dumps(water\_data)

# Publish the data to the topic

client.publish(topic, payload)

time.sleep(10) # Adjust the interval based on your requirements

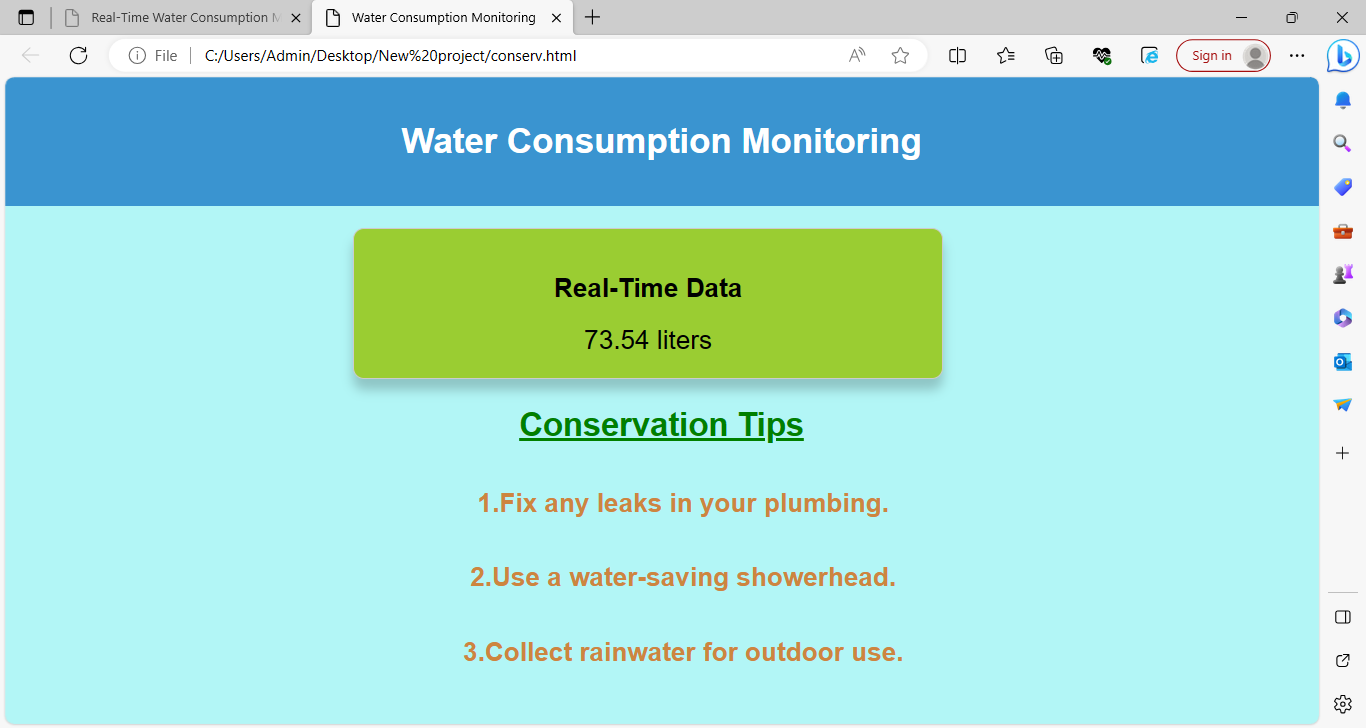
except KeyboardInterrupt:

print("Script terminated by user.")

client.disconnect()

**Data sharing platform:**

Designing the platform to display water consumption data from IoT sensors and promote water conservation efforts (using html, CSS and JS).



The webpage displays the real time data of water level based on the information given by the sensors.

**HTML:**

<!DOCTYPE html>

<html>

<head>

<title>Water Consumption Monitoring</title>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<header>

<h1>Water Consumption Monitoring</h1>

</header>

<main>

<div class="data-container">

<h2>Real-Time Data</h2>

<div class="water-consumption" id="water-consumption">

Loading...

</div>

</div>

<div class="conservation-tips">

<h2>Conservation Tips</h2>

<ul>

<li>Fix any leaks in your plumbing.</li>

<li>Use a water-saving showerhead.</li>

<li>Collect rainwater for outdoor use.</li>

</ul>

</div>

</main>

<script src="script.js"></script>

</body>

</html>

**CSS:**

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

header {

background-color: #3498db;

color: #fff;

text-align: center;

padding: 20px;

}

main {

text-align: center;

padding: 20px;

}

.data-container {

border: 1px solid #ccc;

padding: 20px;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

}

.water-consumption {

font-size: 24px;

}

.conservation-tips {

margin-top: 20px;

}

ul {

list-style-type: disc;

}

ul li {

margin: 10px 0;

}

**JavaScript:**

document.addEventListener("DOMContentLoaded", function () {

const consumptionElement = document.getElementById("water-consumption");

function updateWaterConsumption() {

const randomConsumption = (Math.random() \* 100).toFixed(2);

consumptionElement.textContent = randomConsumption + " liters";

}

setInterval(updateWaterConsumption, 5000);

// Initial data load.

updateWaterConsumption();

});

Using the above frontend code I built the basic prototype of the data sharing platform.

**Benefits of the system :**

* Real-time water consumption analysis
* Reduced maintenance costs
* Predicting potential failures
* Remote monitoring
* Interactive reports
* Reduced risks
* Reducing leakage