INTERNET OF THINGS - GROUP 5
PROJECT: SMART WATER SYSTEM

NAME : A.SHARUMITHA NM ID: 752F3756FD0584B825147CBFA4A03AC4

REGNO: 950321104046

PHASE 3: Development Part 1

Smart Water System:

In the first development phase of a Smart Water Management System, the focus should be on setting up the foundational components.

Project Objectives: In this smart water management aim are whether its real-time monitoring, leak detection, water conservation, efficient irrigation, or a combination of these objectives.

NOTE: In this document I declared only the example of the codes. Because these types of coding are used for the smart water management .And it's not an original implementation code.

Microcontroller Programming (e.g., Arduino, Raspberry Pi):

Reading Sensor Data (Arduino Example):

Arduino (C/C++): Arduino boards are widely used for sensor data acquisition and actuator control. You can write C/C++ code using the Arduino IDE to read sensor data, process it, and send it to a central server.

```
#include <Wire.h>

#include <Adafruit_Sensor.h>

#include <Adafruit_BME280.h>

Adafruit_BME280 bme; // BME280 sensor object

void setup() {

  Serial.begin(9600);
  if (!bme.begin(0x76))
 {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }
}
void loop()
{
  float temperature = bme.readTemperature();
  float humidity = bme.readHumidity();
  float pressure = bme.readPressure() / 100.0F;
  Serial.print("Temperature = ");
  Serial.print(temperature);
  Serial.println(" *C");
 Serial.print("Humidity = ");
  Serial.print(humidity);
  Serial.println(" %");
  Serial.print("Pressure = ");
  Serial.print(pressure);
  Serial.println(" hPa");
  delay(2000);
}
```

Server-side Programming (Node.js, Python, etc.):

Controlling Actuators (Arduino Example):

```
const int relayPin = 13;
```

```
void setup()
 {
pinMode(relayPin, OUTPUT);
}
void loop()
 {
  // Turn on the relay (actuator)
  digitalWrite(relayPin, HIGH);
  delay(5000); // Wait for 5 seconds
 // Turn off the relay
  digitalWrite(relayPin, LOW);
  delay(5000); // Wait for 5 seconds
}
```

Database Management (e.g., MongoDB, MySQL):

REST API for Data Transmission (Node.js Example using Express.js):

Node.js (Express.js): Node.js is a JavaScript runtime commonly used for building scalable network applications. Express.js is a popular Node.js framework for building APIs.

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/smartwaterdb', { useNewUrlParser: true, useUnifiedTopology: true });
const SensorDataSchema = new mongoose.Schema({
 temperature: Number,
 humidity: Number,
 pressure: Number,
  timestamp: { type: Date, default: Date.now }
});
```

```
const SensorData = mongoose.model('SensorData', SensorDataSchema);
// Save sensor data to the database
const newData = new SensorData({
temperature: 25.5,
  humidity: 60.2,
  pressure: 1013.2
});
newData.save()
  .then(() => {
    console.log('Data saved successfully.');
  })
  .catch((err) => {
    console.error('Error saving data:', err);
  });
```

Web-based Dashboard (HTML, CSS, JavaScript):

Real-time Data Visualization (JavaScript Example using Chart.js):

HTML, CSS, JavaScript (React, Angular, Vue.js): HTML provides the structure, CSS handles styling, and JavaScript manages interactivity. Modern frontend frameworks like React, Angular, or Vue.js help build dynamic and responsive user interfaces.

```
const ctx = document.getElementById('sensor-chart').getContext('2d');
const sensorChart = new Chart(ctx, {
  type: 'line',
  data: {
    labels: [], // Timestamps
    datasets: [{
      label: 'Temperature (°C)',
      data: [], // Temperature values
      borderColor: 'red',
      fill: false
    }]
  },
  options: {
    scales: {
      x: {
        type: 'linear',
        position: 'bottom'
      }
    }
  }
});
// Update chart with new data
function updateChart(timestamp, temperature) {
  sensorChart.data.labels.push(timestamp);
  sensorChart.data.datasets[0].data.push(temperature);
  sensorChart.update();
}
```

Mobile App Development (React Native, Flutter, etc.):

Mobile App Interface for Monitoring:

React Native, Flutter: Cross-platform frameworks for building mobile apps using JavaScript (React Native) or Dart (Flutter).

```
import React, { useState, useEffect } from 'react';
import { View, Text } from 'react-native';
const App = () => {
  const [temperature, setTemperature] = useState(0);
// Fetch real-time data from the server
  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await fetch('http://example.com/api/sensor-data');
        const data = await response.json();
        setTemperature(data.temperature);
      } catch (error) {
        console.error('Error fetching data:', error);
      }
    };
// Fetch data every 5 seconds
    const interval = setInterval(fetchData, 5000);
 // Clean up interval on component unmount
    return () => clearInterval(interval);
  }, []);
return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Current Temperature: {temperature}°C</Text>
    </View>
  );
};
export default App;
```

CONCLUSION:

The changing smart water distribution and wastewater management will transform the work of local providers. An experienced water technology team will then offer strategies and equipment for improved performance and streamlined data gathering, management, and analysis. Data flowing through smart water systems will help water companies improve their current services and handle future challenges