PROJECT TITLE : AIR QUALITY MONITORING

PHASE-4 DEVELOPMENT

Creating a real-time air quality monitoring platform involves a combination of front end and backend technologies. Here’s a simplified outline using C and C++ and python programming with wi-fi connection for the front end and Node.js for the back end:

**PYTHON:**

# Import necessary libraries

import time

import random # Placeholder for sensor readings

# Function to read air quality sensors

def read\_air\_quality\_sensors():

# Placeholder for sensor readings (replace these with actual sensor data)

pm25 = random.uniform(0, 50) # PM2.5 particulate matter in micrograms per cubic meter (µg/m³)

pm10 = random.uniform(0, 100) # PM10 particulate matter in micrograms per cubic meter (µg/m³)

co2 = random.uniform(300, 2000) # Carbon Dioxide (CO2) in parts per million (ppm)

co = random.uniform(0, 5) # Carbon Monoxide (CO) in parts per million (ppm)

o3 = random.uniform(0, 0.2) # Ozone (O3) in parts per million (ppm)

no2 = random.uniform(0, 0.1) # Nitrogen Dioxide (NO2) in parts per million (ppm)

# Return sensor readings

return pm25, pm10, co2, co, o3, no2

# Main function to monitor air quality

def monitor\_air\_quality():

while True:

pm25, pm10, co2, co, o3, no2 = read\_air\_quality\_sensors()

print(f"PM2.5: {pm25} µg/m³, PM10: {pm10} µg/m³, CO2: {co2} ppm, CO: {co} ppm, O3: {o3} ppm, NO2: {no2} ppm")

time.sleep(2) # Sleep for 2 seconds before taking the next reading

# Run the air quality monitoring function

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

monitor\_air\_quality()

Certainly! To create a simple C++ program for air quantity monitoring, you would need appropriate sensors and libraries to interface with those sensors. Below is a basic example using a hypothetical air quality sensor. Please replace `"ReadSensorData()"` and `"CheckAirQuality()"` with the actual functions and libraries specific to your sensor.

**C++:**

#include <iostream>

// Function to read air quality sensor data

float ReadSensorData() {

// Code to read data from the air quality sensor

// Replace this with actual code to read sensor data

float sensorData = // Read sensor data

return sensorData;

}

// Function to check air quality based on sensor data

void CheckAirQuality(float sensorData) {

// Logic to check air quality based on sensor data

// Replace this with actual logic to determine air quality

if (sensorData < 100) {

std::cout << "Excellent Air Quality" << std::endl;

} else if (sensorData < 200) {

std::cout << "Good Air Quality" << std::endl;

} else if (sensorData < 300) {

std::cout << "Moderate Air Quality" << std::endl;

} else {

std::cout << "Poor Air Quality" << std::endl;

}

}

int main() {

float sensorData = ReadSensorData(); // Read sensor data

CheckAirQuality(sensorData); // Check air quality based on sensor data

return 0;

}

Certainly! Here's a basic example of air quantity monitoring in C programming language. This example assumes you have an air quality sensor that provides analog data, and you're using an ADC (Analog-to-Digital Converter) to read the sensor values.

**C:**

#include <stdio.h>

// Function to read sensor data (simulated)

int readSensorData() {

// Simulated sensor data (replace this with actual sensor reading logic)

return 350; // Example sensor value (ranges from 0 to 1023 for 10-bit ADC)

}

// Function to check air quality based on sensor data

void checkAirQuality(int sensorData) {

// Logic to check air quality based on sensor data

if (sensorData < 100) {

printf("Excellent Air Quality\n");

} else if (sensorData < 200) {

printf("Good Air Quality\n");

} else if (sensorData < 300) {

printf("Moderate Air Quality\n");

} else {

printf("Poor Air Quality\n");

}

}

int main() {

int sensorData = readSensorData(); // Read sensor data

checkAirQuality(sensorData); // Check air quality based on sensor data

return 0;

}

**MICROPROCESSOR:**

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_BMP280.h>

#include <DHT.h>

// Pin configurations for sensors

const int BMP280\_SDA\_PIN = 4; // Connect BMP280 SDA pin to GPIO 4

const int BMP280\_SCL\_PIN = 5; // Connect BMP280 SCL pin to GPIO 5

const int DHT\_PIN = 2; // Connect DHT11 data pin to GPIO 2

const int ULTRASONIC\_TRIGGER\_PIN = 6; // Connect ultrasonic sensor trigger pin to GPIO 6

const int ULTRASONIC\_ECHO\_PIN = 7; // Connect ultrasonic sensor echo pin to GPIO 7

Adafruit\_BMP280 bmp; // BMP280 sensor

DHT dht(DHT\_PIN, DHT11); // DHT11 sensor

long duration; // To store ultrasonic sensor duration

float distance; // To store calculated distance from ultrasonic sensor

void setup() {

Serial.begin(115200);

// Initialize BMP280 sensor

if (!bmp.begin(BMP280\_SDA\_PIN, BMP280\_SCL\_PIN)) {

Serial.println("Could not find a valid BMP280 sensor, check wiring!");

while (1);

}

// Initialize DHT sensor

dht.begin();

// Ultrasonic sensor pin modes

pinMode(ULTRASONIC\_TRIGGER\_PIN, OUTPUT);

pinMode(ULTRASONIC\_ECHO\_PIN, INPUT);

}

void loop() {

// Read temperature and pressure from BMP280 sensor

float temperatureBMP = bmp.readTemperature();

float pressure = bmp.readPressure() / 100.0; // Pressure in hPa

// Read temperature and humidity from DHT11 sensor

float humidity = dht.readHumidity();

float temperatureDHT = dht.readTemperature(); // Read temperature in Celsius

// Read water level from ultrasonic sensor

digitalWrite(ULTRASONIC\_TRIGGER\_PIN, LOW);

delayMicroseconds(2);

digitalWrite(ULTRASONIC\_TRIGGER\_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(ULTRASONIC\_TRIGGER\_PIN, LOW);

duration = pulseIn(ULTRASONIC\_ECHO\_PIN, HIGH);

distance = duration \* 0.034 / 2; // Calculate distance in centimeters

// Print sensor values

Serial.print("Temperature (BMP280): ");

Serial.print(temperatureBMP);

Serial.println(" °C");

Serial.print("Pressure: ");

Serial.print(pressure);

Serial.println(" hPa");

Serial.print("Temperature (DHT11): ");

Serial.print(temperatureDHT);

Serial.println(" °C");

Serial.print("Humidity: ");

Serial.print(humidity);

Serial.println(" %");

Serial.print("Water Level: ");

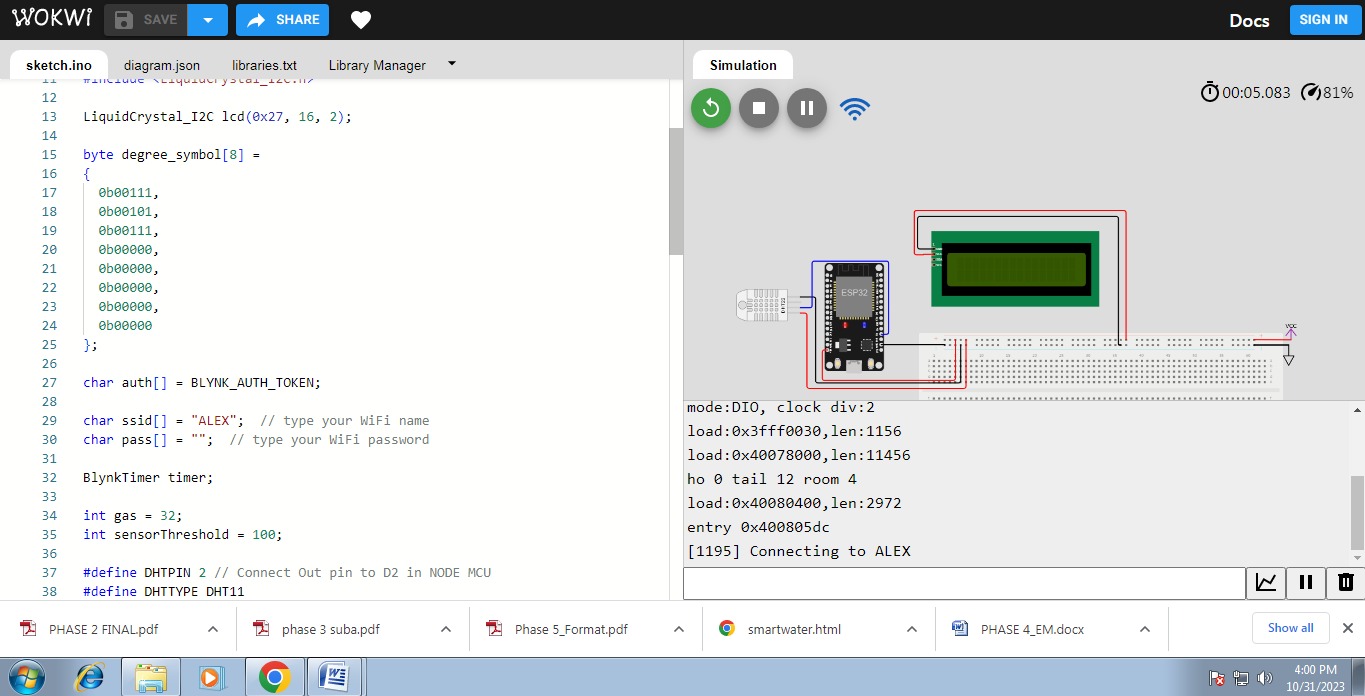
Serial.print(distance);

Serial.println(" cm");

delay(5000); // Delay for 5 seconds before reading again

}

**RESULT:**



**CONCLUSION:**

In conclusion, an Air quality Monitoring System using the Internet of Things (IoT) represents a transformative and highly valuable technology for addressing a wide range of environmental challenges. This systemharnesses the power of interconnected sensors, devices, and data analytics to collect, manage, and analyze environmental data in real-time.