**Flow of the**

**1. Libraries Included**

**#include "SoftwareSerial.h"**

**#include "MHZ19.h"**

**#include "PMS.h"**

**#include "MQ131.h"**

**#include "dht.h"**

**#include "DS3231.h"**

* **SoftwareSerial.h –** Enables serial communication on digital pins, often used for interfacing with sensors and modules**.**
* **MHZ19.h –** Library for the MH-Z19 CO₂ sensor, which measures carbon dioxide levels.
* PMS.h – Used for Plantower PMS5003/PMS7003 air quality sensors, which measure particulate matter (PM2.5, PM10, etc.).
* **MQ131.h –** For the MQ-131 ozone gas sensor, used to detect ozone (O₃) concentrations in the air.
* **dht.h –** Library for DHT11/DHT22 temperature and humidity sensors.
* **DS3231.h –** Used for the DS3231 RTC (Real-Time Clock) module, which provides timekeeping functionality.

***These libraries are for serial communication, CO2 sensor, particulate matter sensor, ozone sensor, temperature/humidity sensor, and real-time clock.***

**2. Pin Definitions**

**#define led 13**

**#define tvocPin 7 // VOC sensor activation**

**#define dht22 5 // DHT22 temperature and humidity sensor**

* Assigns pins for the sensors and an LED.

**3. Sensor and Module Initialization**

dht DHT;

DS3231 rtc(SDA, SCL);

MHZ19 myMHZ19;

SoftwareSerial co2Serial(2, 3);

SoftwareSerial pmsSerial(8, 9);

PMS pms(pmsSerial);

* Initializes objects for different sensors.

**4. Variables for Sensor Data Storage**

unsigned long dataTimer = 0;

int temp, hum, CO2, o3, tvoc, pm25;

int hours, minutes, previousMinutes = 1;

String receivedData = "Z";

* Stores sensor readings and timestamps.

**5. Setup Function**

void setup() {

Serial.begin(9600);

pinMode(6, OUTPUT);

pinMode(tvocPin, OUTPUT);

digitalWrite(6, HIGH);

digitalWrite(tvocPin, HIGH);

delay(20000); // Warm-up sensors

digitalWrite(6, LOW);

digitalWrite(tvocPin, LOW);

rtc.begin();

co2Serial.begin(9600);

pmsSerial.begin(9600);

myMHZ19.begin(co2Serial);

myMHZ19.autoCalibration(false);

MQ131.begin(6, A0, LOW\_CONCENTRATION, 1000000);

}

* Configures sensor connections and prepares the system.

**6. Loop Function**

void loop() {

readDHT = DHT.read22(dht22);

temp = DHT.temperature;

hum = DHT.humidity;

digitalWrite(tvocPin, HIGH);

delay(5000);

tvoc = analogRead(A1);

digitalWrite(tvocPin, LOW);

co2Serial.listen();

dataTimer = millis();

while (millis() - dataTimer <= 3000) {

CO2 = myMHZ19.getCO2();

}

pmsSerial.listen();

dataTimer3 = millis();

while (millis() - dataTimer3 <= 1000) {

pms.readUntil(data);

pm25 = data.PM\_AE\_UG\_2\_5;

}

MQ131.sample();

o3 = MQ131.getO3(PPB);

t = rtc.getTime();

hours = t.hour;

minutes = t.min;

storeData();

sendDataToNextion();

}

* Reads data from sensors and stores it.

**7. Storing Data for Graphing**

void storeData() {

if ((minutes - previousMinutes) >= 15) {

memmove(tempData, &tempData[1], sizeof(tempData));

tempData[sizeof(tempData) - 1] = temp;

// Similar operations for humidity, TVOC, CO2, PM2.5, and ozone data

previousMinutes = minutes;

}

}

* Saves sensor readings for the past 24 hours.

**8. Sending Data to Display**

void sendDataToNextion() {

Serial.print("co2V.val=");

Serial.print(CO2);

Serial.write(0xff); Serial.write(0xff); Serial.write(0xff);

}

* Sends sensor data to a Nextion display.

**9. Handling Incoming Data**

void checkForIncomingData() {

if (Serial.available() > 0) {

receivedData = Serial.readString();

delay(30);

if (receivedData == "0") { r = 0; }

if (receivedData == "1") { r = 1; }

}

}

* Checks for incoming data from the Nextion display.

**10. Generating Graph Y-Axis Values**

void getYAxisValues() {

maxV = 0;

if (r == 0) {

for (int i = 0; i < sizeof(pm25Data); i++) {

if (maxV < map(pm25Data[i], 0, 255, 0, 1000)) {

maxV = map(pm25Data[i], 0, 255, 0, 1000);

}

}

}

}

* Determines the graph scaling based on sensor readings.