**PROJECT TITLE: AIR QUALITY MONITORING**

**NAME :** T .Jones

**REG NO** : 953021106028

**COLLEGE CODE :** 9530

**COLLEGE NAME :** ST.MOTHER THERESA ENGINEERING COLLEGE

**TEAM CODE : proj\_201035\_Team\_1**

**SOURCE CODE :-**

import time

import serial

import RPi.GPIO as GPIO

import Adafruit\_CharLCD as LCD

# Initialize the LCD

lcd\_rs = 25

lcd\_en = 24

lcd\_d4 = 23

lcd\_d5 = 17

lcd\_d6 = 21

lcd\_d7 = 22

lcd\_columns = 16

lcd\_rows = 2

lcd = LCD.Adafruit\_CharLCD(lcd\_rs, lcd\_en, lcd\_d4, lcd\_d5, lcd\_d6, lcd\_d7, lcd\_columns, lcd\_rows)

# Initialize the SDS011 sensor

ser = serial.Serial('/dev/ttyUSB0', baudrate=9600, timeout=2)

ser.flushInput()

def read\_sensor\_data():

try:

while True:

while ser.in\_waiting < 10:

time.sleep(1)

data = ser.read(10)

if data[0] == 170 and data[1] == 192:

pm25 = (data[2] + data[3] \* 256) / 10.0

pm10 = (data[4] + data[5] \* 256) / 10.0

return pm25, pm10

except Exception as e:

print(f"Error reading from the sensor: {e}")

def display\_air\_quality(pm25, pm10):

lcd.clear()

lcd.message('PM2.5: {:.2f} ug/m3\n'.format(pm25))

lcd.message('PM10: {:.2f} ug/m3'.format(pm10))

if \_\_name\_\_ == '\_\_main\_\_':

try:

while True:

pm25, pm10 = read\_sensor\_data()

display\_air\_quality(pm25, pm10)

time.sleep(10) # Update every 10 seconds

except KeyboardInterrupt:

lcd.clear()

GPIO.cleanup()

**SOURCE CODE :-**

<!DOCTYPE html>

<html>

<head>

<title>Air Quality Monitoring System</title>

<style>

.good { color: green; }

.moderate { color: orange; }

.poor { color: red; }

</style>

</head>

<body>

<h1>Air Quality Monitoring System</h1>

<div id="airQualityData">

<h2>Real-time Air Quality Data</h2>

<p>PM2.5: <span id="pm25Value">Loading...</span></p>

<p>PM10: <span id="pm10Value">Loading...</span></p>

<p>CO2: <span id="co2Value">Loading...</span></p>

<p>Temperature: <span id="temperatureValue">Loading...</span></p>

<p>Humidity: <span id="humidityValue">Loading...</span></p>

</div>

<script>

class AirQualityComponent {

constructor() {

this.pm25Value = document.getElementById("pm25Value");

this.pm10Value = document.getElementById("pm10Value");

this.co2Value = document.getElementById("co2Value");

this.temperatureValue = document.getElementById("temperatureValue");

this.humidityValue = document.getElementById("humidityValue");

}

update(data) {

this.pm25Value.textContent = data.pm25 + " µg/m³";

this.pm10Value.textContent = data.pm10 + " µg/m³";

this.co2Value.textContent = data.co2 + " ppm";

this.temperatureValue.textContent = data.temperature + " °C";

this.humidityValue.textContent = data.humidity + " %";

this.setAirQualityIndicator(this.pm25Value, data.pm25, 20, 50, 100);

this.setAirQualityIndicator(this.pm10Value, data.pm10, 20, 50, 100);

this.setAirQualityIndicator(this.co2Value, data.co2, 400, 800, 1000);

}

setAirQualityIndicator(element, value, good, moderate, poor) {

if (value <= good) {

element.className = "good";

} else if (value <= moderate) {

element.className = "moderate";

} else {

element.className = "poor";

}

}

}

function simulateAirQualityData() {

return {

pm25: (Math.random() \* 100).toFixed(2),

pm10: (Math.random() \* 100).toFixed(2),

co2: Math.floor(Math.random() \* 1200),

temperature: (Math.random() \* 30 + 15).toFixed(2),

humidity: (Math.random() \* 60 + 30).toFixed(2)

};

}

function updateAirQualityComponent(airQualityComponent) {

const data = simulateAirQualityData();

airQualityComponent.update(data);

setTimeout(() => updateAirQualityComponent(airQualityComponent), 5000);

}

const airQualityComponent = new AirQualityComponent();

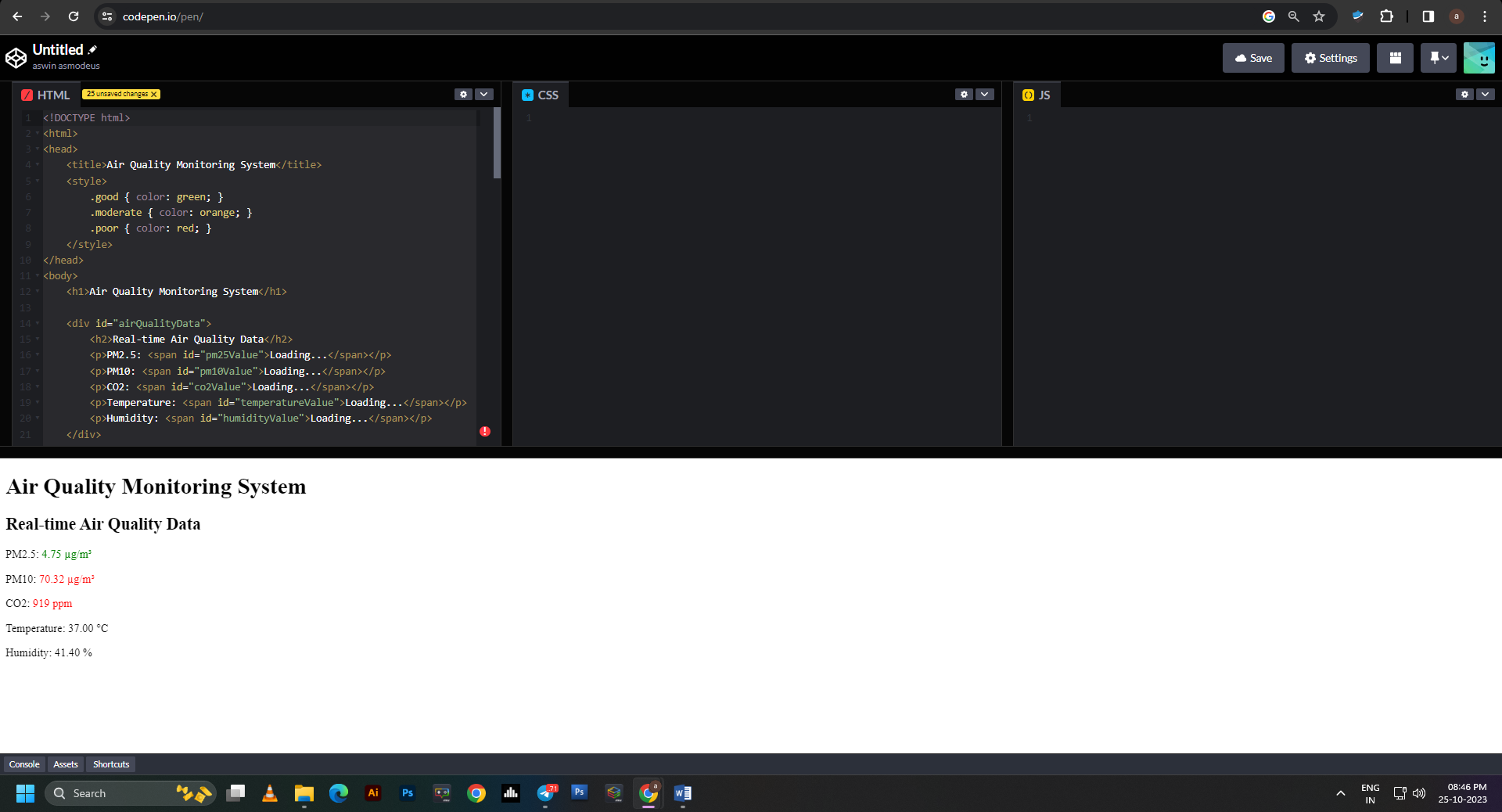
updateAirQualityComponent(airQualityComponent);

</script>

</body>

</html>

**OUTPUT :-**

****