PHASE V PROJECT

PROJECT TITLE: AIR QUALITY MONITORING

NAME : M.VIGNESH

REG NO : 953021106073

COLLEGE CODE : 9530

COLLEGE NAME : ST.MOTHER THERESA ENGINEERING COLLEGE

TEAM NAME : proj_201035_Team_2

ABSTRACT:

The Air Quality Monitoring System (AQMS) represents a pivotal advancement in managing air quality in the modern world. Its deployment of cutting-edge sensors, strategically positioned across urban and industrial areas, continuously measures a spectrum of pollutants, including particulate matter, nitrogen dioxide, sulfur dioxide, carbon monoxide, ozone, and volatile organic compounds. Real-time data is swiftly transmitted to a central database via wireless communication, facilitating rapid processing and analysis through the integration of machine learning algorithms. This data not only allows for the prediction of air quality trends but also aids in pinpointing pollution sources and issuing timely alerts in the event of air quality deterioration or unforeseen environmental events. The AQMS's user-friendly interface, accessible through web applications and mobile devices, empowers government agencies, researchers, and the public with readily available air quality information, while supporting historical data analysis for comprehensive decision-making and informed pollution control measures. In essence, the AQMS is an indispensable asset for addressing the ever-increasing air pollution challenges posed by urbanization and industrialization, serving as a cornerstone for safeguarding public health and environmental welfare.

INTRODUCTION:

In a world grappling with escalating air pollution due to urbanization and industrialization, the Air Quality Monitoring System (AQMS) emerges as a critical innovation. By employing advanced sensors and cutting-edge technology to continuously gather and analyze real-time data on a wide range of air pollutants, including particulate matter and various gases, the AQMS equips governments, researchers, and the public with the tools they need to make informed decisions to protect public health and the environment. With its intuitive user interface and the power of data analytics, including machine learning algorithms, the AQMS represents an indispensable asset in the ongoing battle to mitigate air pollution and its adverse effects.

PROBLEM STATEMENT:

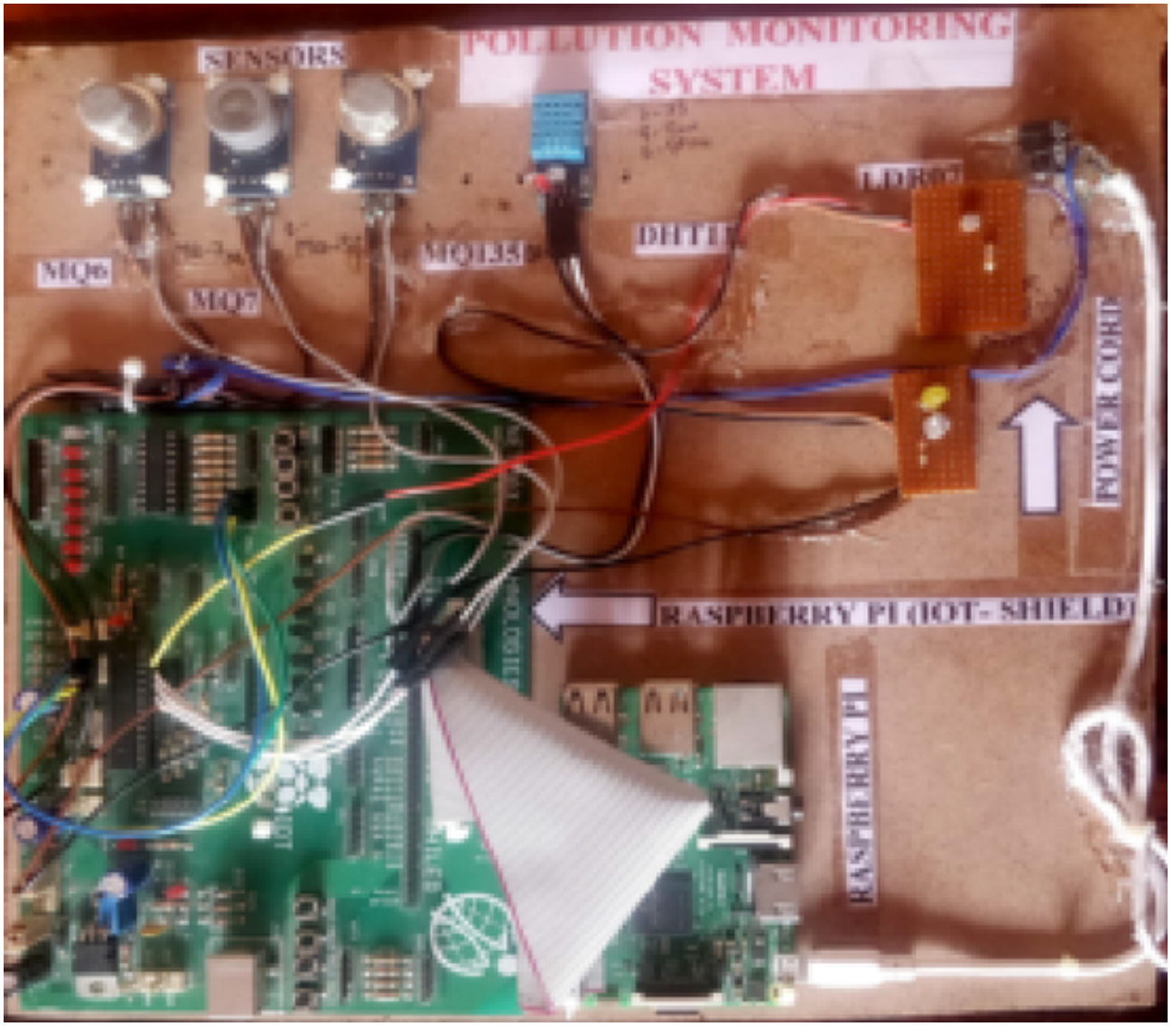
In the face of escalating urbanization and industrial growth, air pollution has become a pressing concern worldwide. The adverse effects of poor air quality on public health and the environment are well-documented. There is an urgent need for effective air quality monitoring systems that can provide real-time data, support informed decision-making, and mitigate the impact of air pollution. Current monitoring systems often lack comprehensive coverage, data accuracy, and timely alerts, leaving communities vulnerable to the adverse consequences of air pollution. Therefore, the problem to be addressed is the development and implementation of an advanced Air Quality Monitoring System (AQMS) that can efficiently and comprehensively monitor air quality, offer data-driven insights, and enable timely responses to safeguard public health and environmental sustainability in the face of increasing urbanization and industrialization.

PROPOSED SOLUTION:

The proposed solution is an Internet of Things (IoT)-based Air Quality Monitoring System (AQMS) designed to address the growing concern of air pollution associated with urbanization and industrialization. This IoT-enabled system leverages a network of advanced sensors connected to the internet, allowing for real-time data collection, analysis, and remote access. By harnessing IoT technology, the AQMS offers timely alerts, predictive insights, and a user-friendly interface for informed decision-making, making it an essential tool in preserving public health and environmental sustainability in our dynamically changing world.

BLOCK DIAGRAM :

AIR QUALITY MONITORING SYSTEM:



HARDWARE USED:

- MQ-135 SENSOR
- MQ-6 SENSOR
- MQ-7 SENSOR
- DHT11

- Raspberry PI

MQ-135 SENSOR:

The MQ-135 gas sensor is renowned for its versatility in detecting a wide array of gases, including harmful pollutants like nitrogen dioxide and sulfur dioxide. Its sensitivity, compact design, and cost-effectiveness make it a popular choice for air quality monitoring applications. This sensor plays a vital role in IoT-based systems, contributing to comprehensive air quality management efforts.

MQ-6 SENSOR:

The MQ-6 gas sensor is a critical component in gas safety applications, primarily used to detect and measure the concentration of gases like LPG, propane,

butane, and methane in the air. When gas levels change, the sensor's resistance adjusts accordingly, generating an electrical signal that can trigger alarms or safety measures. This sensor is widely used in gas leak detection systems, ensuring safety in both residential and industrial settings by promptly identifying and responding to gas leaks.

MQ-7 SENSOR:

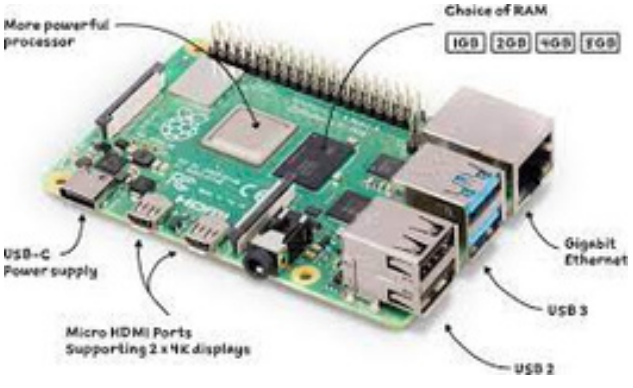
It is a highly sensitive towards carbon monoxide so it is also known as carbon monoxide sensor. It provides fast response time. It is having low cost. It is stable and long-life sensor. The working range of this gas sensor is 20 to 2000 PPM

DHT11:

The DHT11 is a straightforward and affordable digital temperature and humidity sensor. It serves as a valuable component in electronics projects, delivering digital output for both temperature and humidity measurements. Popular in applications like weather stations and home automation, it offers an accessible means of monitoring and controlling environmental conditions.

RASPBERRY PI:

It's like a small independent computer. It is one of the smallest computers popular for its size. It can be used for programming for electronic projects, to play HD videos. We can use languages like c, c++, python, java, ruby for programming. It has its own operating system as Raspbian



WORKING:

DHT11 with Raspberry Pi:

- Connect the DHT11 sensor to the Raspberry Pi's GPIO pins.
- Use a library like Adafruit DHT or Python libraries to read temperature and humidity data from the sensor.

- Send this data to the raspberry Pi.

MQ-135, MQ-6, and MQ-7 with Raspberry Pi

- Connect each gas sensor to the Raspberry Pi's GPIO pins or analog pins, using an ADC if necessary.
- Use Python scripts to read data from the sensors.
- Interpret the sensor data to detect gas concentrations.
- Implement safety alerts or store the data on the Raspberry Pi for future analysis.

Wi-Fi Module:

- Connect the Wi-Fi module to the Raspberry Pi, allowing it to access the internet and external servers.
- Configure the Wi-Fi module to establish a connection to your local network or the internet.

LCD Display:

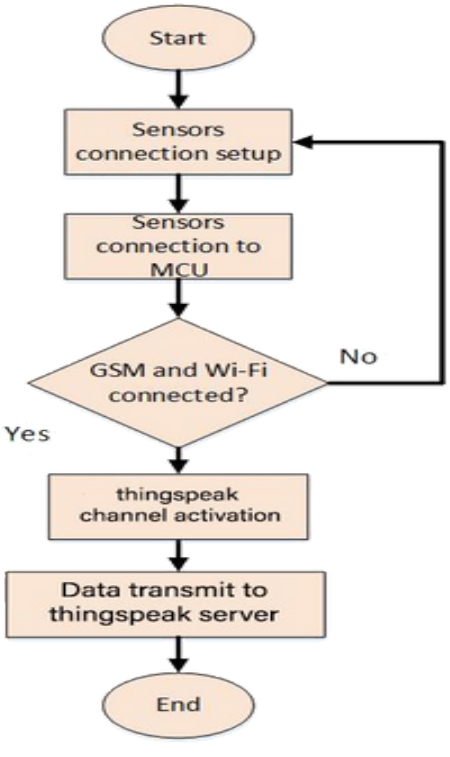
- Connect the LCD display to the Raspberry Pi to provide a local interface for data display.
- Use Python libraries to control the LCD and display real-time sensor data, including temperature, humidity, and gas concentrations.

Data Transmission and Local Display:

- Display the sensor data on the LCD screen for local monitoring and visualization.
- Use the Wi-Fi module to establish a connection to a remote server or cloud platform, sending the sensor data for remote monitoring and storage.

By integrating these sensors with a Raspberry Pi, a Wi-Fi module, and an LCD display, you create an environmental monitoring system that provides real-time data visualization locally on the LCD screen and remote access to data through a server or cloud platform. This setup is ideal for applications like indoor air quality monitoring, home automation, and safety systems, providing both local and remote data access.

FLOW CHART:



PUBLIC AWARENESS:

Raising public awareness about air quality and environmental issues is essential in fostering a sense of responsibility and encouraging collective action. Educational campaigns, media engagement, online resources, community involvement, and school programs can provide the public with knowledge about air quality, its health and environmental impacts, and effective mitigation strategies. Public notifications, partnerships, air quality apps, and workplace initiatives can further empower individuals to make informed decisions, reduce personal exposure to pollutants, and advocate for policies that promote cleaner air. Public events and citizen science projects offer engaging opportunities for community participation and environmental stewardship, ultimately contributing to improved air quality and a healthier, more sustainable future.

SOURCE CODE:

Python:

```
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:

HTML

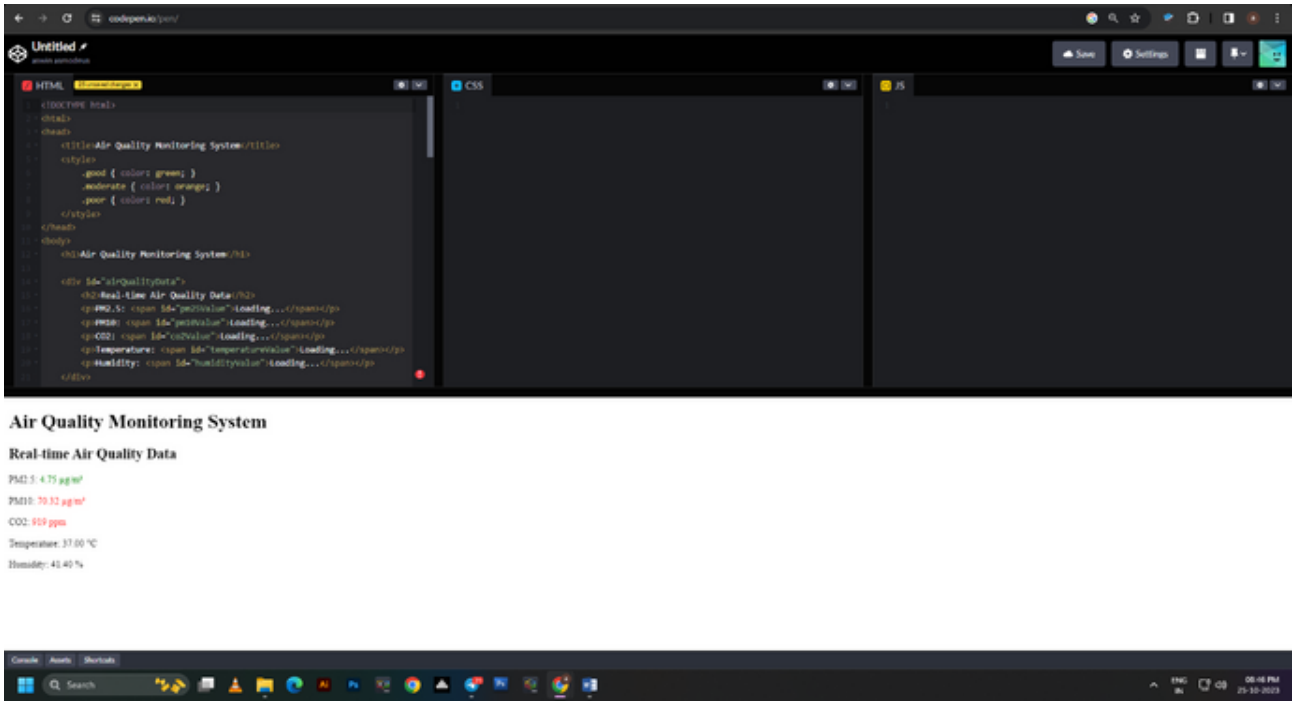
```
<!DOCTYPE html>
<html>
<head>
  <title>Air Quality Monitoring System</title>
  <style>
    .good { color: green; }
    .moderate { color: orange; }
    .poor { color: red; }
  </style>
</head>
<body>
  <h1>Air Quality Monitoring System</h1>
```

```
<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:-



SOURCE CODE:

HTML & JAVASCRIPT

HTML:-

```
<!DOCTYPE html>
<html>
<head>
  <title>Air Quality Monitoring System data</title>
</head>
<body>
  <h1>Air Quality Monitoring System data</h1>
```

```
  <p>Temperature: <span id="temperature"></span>°C</p>
  <p>Humidity: <span id="humidity"></span>%</p>
  <p>MQ-135 Gas Sensor: <span id="mq135"></span></p>
  <p>MQ-6 Gas Sensor: <span id="mq6"></span></p>
  <p>MQ-7 Gas Sensor: <span id="mq7"></span></p>

  <script>
    const sensorDataContainer = document.getElementById('sensor-data');

    function fetchSensorData() {
      fetch('/api/sensor-data')
        .then(response => response.json())
        .then(data => {
          document.getElementById('temperature').textContent =
data.temperature;
          document.getElementById('humidity').textContent = data.humidity;
          document.getElementById('mq135').textContent = data.mq135;
          document.getElementById('mq6').textContent = data.mq6;
          document.getElementById('mq7').textContent = data.mq7;
        });
    }

    fetchSensorData();
```

```
    setInterval(fetchSensorData, 5000);
  </script>
</body>
</html>
```

JAVASCRIPT:-

```
const express = require('express');
const app = express();
const Gpio = require('onoff').Gpio;
const sensorLib = require('node-dht-sensor');

const DHT_SENSOR = 11;
const DHT_PIN = 4;
const MQ135_PIN = 17;
const MQ6_PIN = 27;
const MQ7_PIN = 22;

const mq135 = new Gpio(MQ135_PIN, 'in', 'both');
const mq6 = new Gpio(MQ6_PIN, 'in', 'both');
const mq7 = new Gpio(MQ7_PIN, 'in', 'both');

app.use(express.static('public'));
```

```
app.get('/api/sensor-data', (req, res) => {
  const { temperature, humidity } = sensorLib.read(DHT_SENSOR, DHT_PIN);

  const mq135Value = mq135.readSync();
  const mq6Value = mq6.readSync();
  const mq7Value = mq7.readSync();

  const sensorData = {
    temperature,
    humidity,
    mq135: mq135Value,
    mq6: mq6Value,
    mq7: mq7Value,
  };

  res.json(sensorData);
});

app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

OUTPUT:

CONSLUSION:

In conclusion, an Air Quality Monitoring System (AQMS) is a multifaceted solution for tracking and assessing air quality. It relies on an array of sensors, data analysis, and real-time reporting to inform the public, guide environmental policies, and ensure safety in various sectors. AQMS is an indispensable tool in the ongoing effort to combat air pollution and protect human health and the environment.