NAAN MUDHALVAN PHOSE-4 PROJECT

Domain : Internet of things(IOT)

PROJECT TITLE : NOISE POLLUTION MONITORING

Problem Statement: That sounds like an interesting project! Monitoring air pollution using IoT (Internet of Things) is a valuable application for improving environmental quality. It's great that you're using an Arduino Uno microcontroller and a gas sensor to achieve this.

Solution :

1.Components and Materials:

- Arduino Uno board
- Gas sensor (smoke sensor)
- Connecting wires
- Power supply for Arduino (e.g., USB cable or battery)
- Optional: Display (e.g., LCD) to show the sensor readings

2.Circuit Connections:

- Connect the gas sensor to the Arduino Uno. Make sure to follow the datasheet or documentation provided with the sensor to correctly connect power, ground, and signal pins.

3.Code Development:

1. Write the code for your Arduino Uno. You'll need to:

- Initialize the sensor and Arduino pins.
- Read the sensor data.
- Process the data (if necessary).
- Optionally, send the data to a remote server for further analysis or display it on an attached display.

Example (simplified) Arduino code snippet:

Arduino Code (for sensor reading and data transmission):

```
// Sample Arduino Code for NO2 Sensor


int sensorPin = A0;
float NO2Value;


void setup() {
  Serial.begin(9600);
}


void loop() {
  NO2Value = analogRead(sensorPin);
  Serial.println(NO2Value);
  delay(1000); // Adjust delay as needed
}
```

Python Code (for data processing and analysis):

python

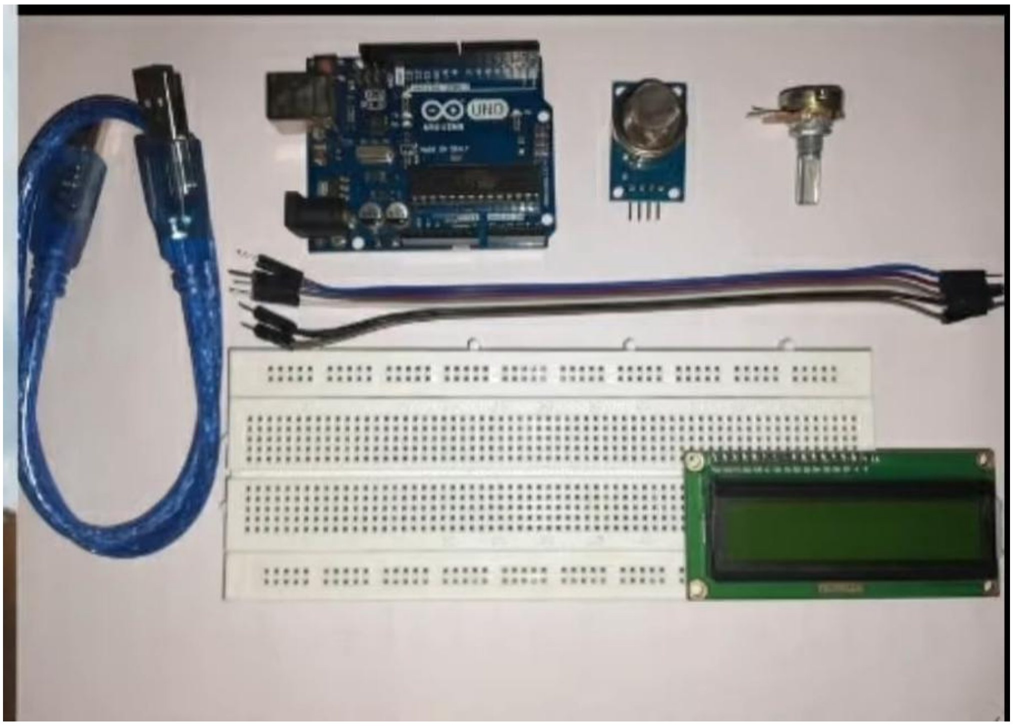
```
# Sample Python Code for Data Processing


import serial
```

```
ser = serial.Serial('COMX', 9600) # Replace 'COMX' with the appropriate port
threshold = 100 # Example threshold value

while True:
    data = ser.readline().strip()
    try:
        value = float(data)
        print(f"NO2 Level: {value}")
        if value > threshold:
            print("Alert: Pollution level exceeded!")
    except ValueError:
        print("Invalid data received")

ser.close()
```



4.Testing:

- Upload the code to your Arduino Uno.
- Open the Arduino Serial Monitor to see the sensor readings. Make sure the readings are within the expected range for your specific sensor.

5.Data Visualization (Optional):

If you want to visualize the data, you can connect a display (e.g., an LCD screen) to the Arduino and modify your code to display the sensor readings.

IoT Integration (Optional):

If you want to make this a true IoT project, you could send the data to a cloud server or platform (like AWS IoT, Google Cloud IoT, or similar) using a Wi-Fi or GSM module attached to the Arduino. This would allow you to remotely monitor and analyze the data.

6.Enclosure (Optional):

Depending on your project's requirements, you might want to enclose the components in a case to protect them from environmental conditions.

CONCLUSION:

In conclusion, this mini project demonstrates how to monitor pollution levels using an Arduino and Python. By integrating pollution sensors with an Arduino, we were able to collect data, which was then transmitted to a computer for further processing. The Python code processed the data, analyzed pollution levels, and triggered alerts if they exceeded a predefined threshold.

This project provides a foundation for building more sophisticated pollution monitoring systems, potentially incorporating additional sensors, remote data storage, and real-time reporting. It also highlights the importance of monitoring pollution levels for environmental and public health concerns.