INTERNET OF THINGS

PROJECT MANUAL GUIDE

13. Air quality and gas monitor system.

TABLE OF CONTENTS

MODULE 1 - INTRODUCTION

The Internet of Things (IoT) has evolved into a new era of innovation. Things are becoming smart through the convergence of the digital and physical worlds. Researchers estimate that by 2025, the number of active wireless connected devices will reach 75.4 billion. The explosive growth of the “Internet of Things” is changing our world, and the rapid drop in price for typical IoT components is allowing people to innovate new designs and products.

1.1 PROBLEM STATEMENT

In urban environments and industrial settings, monitoring air quality is essential for maintaining safe and healthy conditions for residents and workers. Pollutants such as carbon monoxide (CO), nitrogen dioxide (NO_2), sulfur dioxide (SO_2), and particulate matter (PM) can pose serious health risks if present in high concentrations. Additionally, monitoring for the presence of combustible gases (like methane) is critical to prevent potential hazards.

1.2 Aim

To design a real-time air quality and gas monitoring system that detects and records levels of harmful gases, alerts users when concentrations exceed safe thresholds, and provides insights into air quality trends over time to ensure health and safety in various environments

MODULE 2 - HARDWARE

Hardware refers to any physical components/particulars of a system containing ICs, electronics, sensors, and circuit boards. Without hardware, an IoT system cannot exist, and the software developed won't be able to run.

2.1 COMPONENTS REQUIRED

- Arduino Uno Board
- Ultrasonic Sensor
- Gas Sensors
- Servo Motor
- DHT11 Sensor
- RGB LED
- Buzzer
- Resistor
- Power Supply

A.ARDUINO UNO:

Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

Fig.1.0 Arduino UNO

B.BREADBOARD:

Breadboards are one of the most fundamental pieces when learning how to build circuits. Breadboards are commonly utilized while prototyping temporary circuits. It is useful to designers because it allows components to be removed and replaced easily

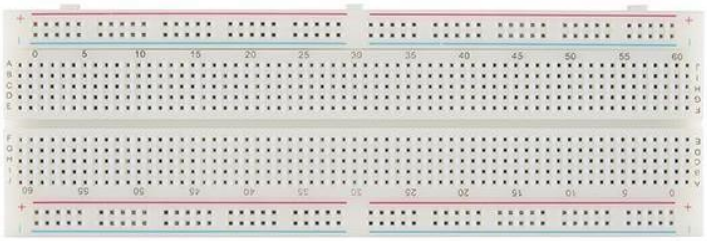


Figure 1.1 – Breadboard

C.ULTRASONIC SENSOR:

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal.

Figure 1.2 - Ultrasonic sensor

D.Servo Motor:

A servo motor is a rotary actuator that allows precise control of angular position, velocity, and acceleration.

Figure 1.3 - Micro Servo

E.DHT11 Sensor:

DHT11 is a low-cost, digital temperature and humidity sensor.

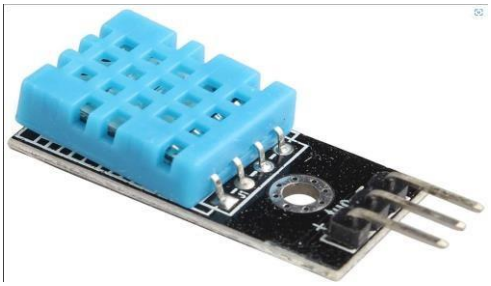
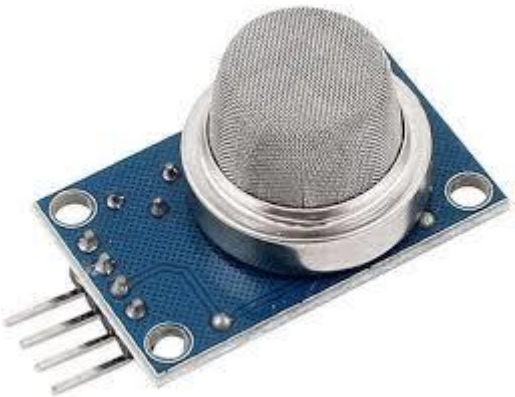


Figure 1.4 – DHT11 Sensor

F. Buzzer:

A buzzer is an electronic component that produces sound or audio signals to provide alerts or feedback in response to certain conditions.

Fig 1.5 Buzzer



- **Gas Sensors**

Gas sensors are devices that help us understand the amount of gas in the environment and the natura state of its movement.

MODULE 3 - SOFTWARE

Software is a generic term to refer to the scripts and programs that run on a microprocessor or microcontroller and execute specific tasks.

3.1 PROGRAM

```
// Define the pin numbers for the sensors and buzzer
const int airQualityPin = A0;  // MQ-135 (Air Quality) connected to A0
const int alcoholPin = A1;      // MQ-3 (Alcohol) connected to A1
const int gasPin = A2;          // MQ-2 (Gas) connected to A2
const int buzzerPin = 9;        // Buzzer connected to digital pin 9

// Define thresholds (these values need calibration based on your environment)
const int airQualityThreshold = 400;  // Example threshold for Air Quality (MQ-135)
const int alcoholThreshold = 300;      // Example threshold for Alcohol (MQ-3)
const int gasThreshold = 500;          // Example threshold for Gas (MQ-2)

void setup() {
  Serial.begin(9600);           // Start Serial communication
  pinMode(buzzerPin, OUTPUT);    // Set buzzer pin as output
}

void loop() {
  // Read the analog values from each sensor
  int airQualityValue = analogRead(airQualityPin);
  int alcoholValue = analogRead(alcoholPin);
  int gasValue = analogRead(gasPin);

  // Print the values to the Serial Monitor
  Serial.print("Air Quality Value: ");
  Serial.println(airQualityValue);
```

```
  Serial.print("Alcohol Value: ");
Serial.println(alcoholValue);
  Serial.print("Gas Value: ");
  Serial.println(gasValue);

  // Initialize a flag for buzzer status
  bool alarmTriggered = false;

  // Check for harmful conditions based on thresholds
  if (gasValue > gasThreshold) {
    Serial.println("Warning: Harmful gas detected!"); // Indicate harmful gas
    alarmTriggered = true; // Set flag to indicate alarm should be triggered
  }

  if (airQualityValue > airQualityThreshold) {
    Serial.println("Warning: Poor air quality detected!"); // Indicate poor air quality
    alarmTriggered = true; // Set flag to indicate alarm should be triggered
  }

  if (alcoholValue > alcoholThreshold) {
    Serial.println("Warning: Alcohol detected!"); // Indicate alcohol presence
    alarmTriggered = true; // Set flag to indicate alarm should be triggered
  }

  // Activate buzzer if any alarm is triggered
  if (alarmTriggered) {
    digitalWrite(buzzerPin, HIGH);  // Turn on the buzzer
    delay(500);                     // Buzzer on for 500 ms
  } else {
digitalWrite(buzzerPin, LOW);   // Turn off the buzzer
    Serial.println("No harmful gas detected."); // Indicate no harmful gas
```

```
  }


  delay(5000); // Wait for a second before the next reading
}
```

Implementation Guidelines

- **Hardware Components**: Use appropriate gas sensors, microcontroller (e.g., Arduino, Raspberry Pi), communication module, power supply, and possibly a backup battery for continuous monitoring.
- **Software Components**: Develop the server software (for data storage, processing, and alerts), frontend (dashboard), and backend services (data API, alert system).
- **Data Storage**: Consider using a database to store historical data, like PostgreSQL for structured data or NoSQL solutions for scalability.
- **Machine Learning (Optional)**: Implement a predictive model to analyze trends and forecast future air quality levels.

Success Criteria

- Real-time detection and alerting of gas concentrations above threshold levels.
- Historical data availability and ability to generate insights and reports.
- Scalable and reliable system that can monitor multiple sensors and locations.
- User-friendly interface for monitoring and managing settings.

OUTPUT:

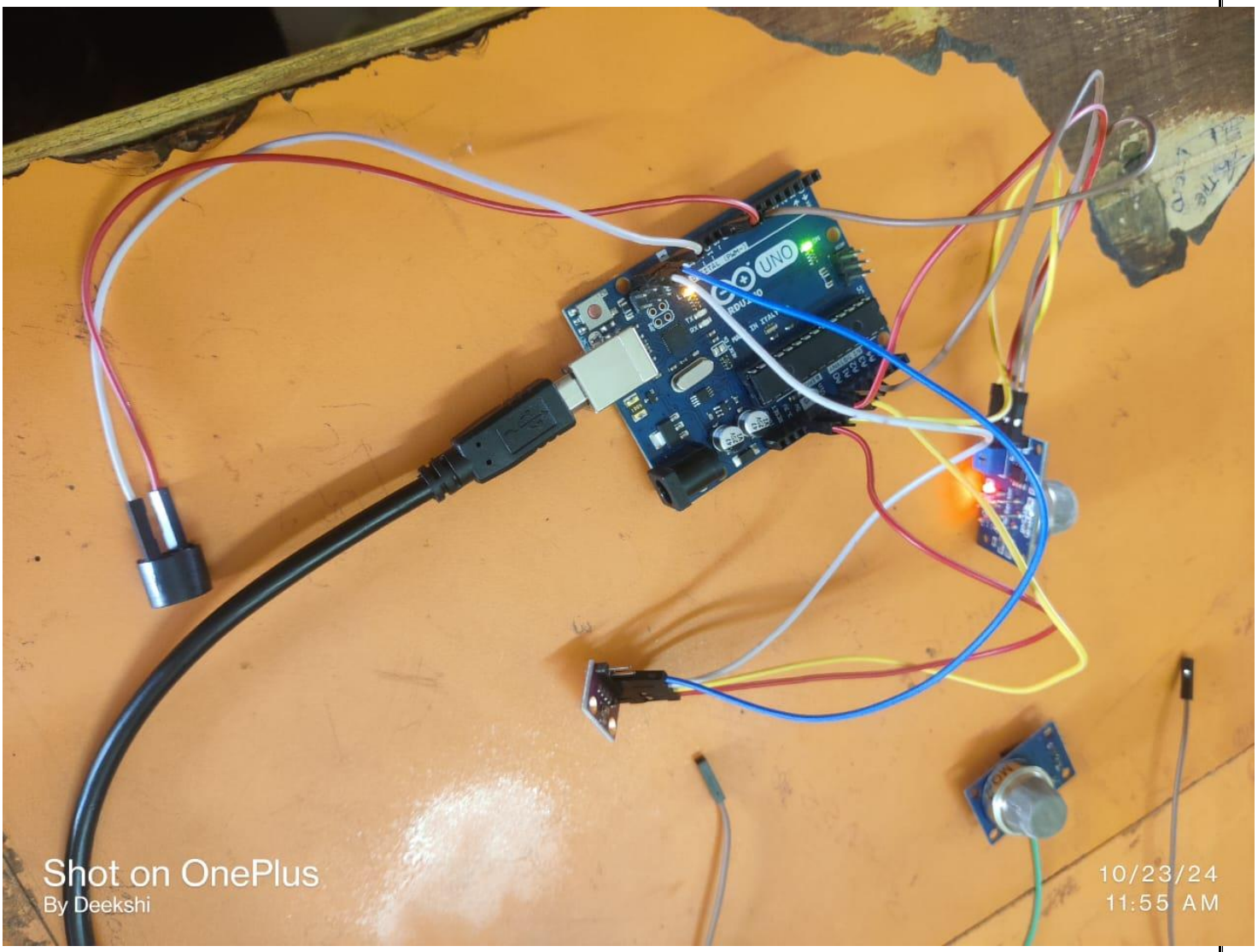


Fig 1.7 Output diagram

MODULE 5 - CONCLUSION

Hence, The Air Quality and Gas Monitoring System effectively detects and monitors harmful gases and particulate matter in real-time, providing immediate alerts when levels exceed safe thresholds. With its simple design and essential components, this system helps ensure a safer and healthier environment by enabling timely awareness and action against air pollution hazards.