M.Sc. Project
Report

Design of a Remote Detection System for Harmful Gases

Submitted in partial fulfillment of
the requirements for the award of the degree of

Master of Science
in
Electronics

Submitted by

Roll No	Names of Students
10021017100009	Swani Chatterjee
10021017100010	Saurabh Biswas
10021017100011	Suryashmi Chakraborty

Under the guidance of
Sri Durjoy Roy
Assistant Professor, Dept. of Electronics, Rishi Bankim Chandra College & Visiting Faculty,
West Bengal State University

Department of Electronics
West Bengal State University
Berunanpukuria, Barasat, West Bengal - 700126

Department of Electronics

West Bengal State University, Barasat

Certificate

This is to certify that this is a bonafide record of the project presented by the students whose names are given below during 4th Semester, 2022-23 in partial fulfilment of the requirements of the degree of Master of Science in Electronics.

Roll No	Names of Students
10021017100009	Swani Chatterjee
10021017100010	Saurabh Biswas
10021017100011	Suryashmi Chakraborty

DURJOY ROY
(Project Guide)

ARIJIT ROY
(Course Coordinator)

Date:

Abstract

In this IoT project, we developed a gas detection device to monitor harmful gases in the environment, addressing pressing global air pollution concerns. The device utilizes an Arduino Uno microcontroller, interfacing with gas sensors (MQ-7 for CO, MQ-135 for CO2, and AGSM02A for SO_2, NO_2, NO_3, NH_2, O_3, H_2S. These gases not only threaten human health but also impact butterfly populations and ecosystems. Elevated NO2 and SO2 levels damage host plants, affecting butterfly habitats and life cycles. Ozone exposure causes oxidative stress in butterflies, reducing survival rates and reproductive capabilities. Indirectly, CO and other gases affect nectar quality and flower compositions, impacting pollination. By preserving butterfly populations, we protect ecosystems. The Arduino Uno features a versatile ATmega328P microcontroller, facilitating gas sensor communication through digital and analog pins. Calibration ensures accurate measurements, while the SIM900A GSM Module allows real-time SMS alerts for prompt responses to hazardous situations. The project demonstrates efficiency in monitoring harmful gases, contributing to environmental protection and biodiversity conservation. Future enhancements include advanced sensor integration and data visualization for comprehensive gas monitoring. Overall, this gas detection device showcases the potential of IoT in addressing environmental challenges for a greener and sustainable future

Contents

1 Problem Definition **1**

2 Introduction **2**
2.1 Introduction 2

3 Hardware Description **4**
3.1 Introduction 4
3.2 Arduino UNO 4
3.2.1 Architecture 6
3.2.2 Operation 13
3.3 Sensors 15
3.3.1 What is a sensor? 15
3.3.2 Characteristics of sensor 15
3.3.3 Types of sensors 16
3.4 Sensors used in this Device 17
3.4.1 MQ-7 Sensor 18
3.4.2 MQ-135 Sensor 21
3.4.3 AGS02MA Sensor(TVOC Sensor) 24
3.4.4 DHT11 sensor 27
3.5 GSM Module 30
3.5.1 SIM900A GSM MODULE 33
3.5.2 SIM 900A ARCHITECTURE 36
3.5.3 GSM Interfacing 41
3.6 External Power supply 46
3.6.1 Why do we need it? 47

4 Circuit Design **49**
4.1 Circuit Design 49
4.1.1 Circuit Connection 52

5 Programming **55**
5.1 Designing the Software code 55
5.1.1 The Arduino IDE (Integrated Development Environment) 55
5.1.2 Execution of the Program 61

6 Results and Output **64**
6.1 Results obtained 64

7 Conclusion **66**

8 Future Enhancements **68**

9 Benefits **70**

Acknowledgements **72**

References **73**

List of Figures

3.1 Top and Side views of Arduino Uno 5
3.2 Arduino Uno Architecture 6
3.3 ATmega328P 8
3.4 ATmega328P Pin Diagram 10
3.5 MQ-7 Sensor 17
3.6 MQ-135 21
3.7 AGS02MA TVOC sensor 24
3.8 DHT11 sensor 27
3.9 SIM 900 Module 31
3.10 SIM 900 Module Block Diagram 32
3.11 SIM 900 Connection 34
3.12 SIM900 Module, interfaced with the Arduino Uno 46

4.1 Block Diagram of External Power Suppy Circuit 50
4.2 Block diagram of System Connection 52
4.3 The complete Project Circuit showing all the components interconnected. At the left, the 18650 Li-ion batteries with BMS and charging boards, powering up all the components.at the center, the SIM900 GSM module, and at the extreme right, the Arduino Uno are in the box. The sensors are at the top and the bottom. 54

6.1 Output showing reading of pollutant concentration 65

Chapter 1

Problem Definition

Butterflies are the chief pollinators in the wild. Zoologists are worried about decrease of butterflies in the environment, and are interested in investigating the possible reasons. A possible reason is increase of pollutant gases in the environment. A device that can sense the amount of pollutant gasses in the environment where butterflies live, and send the data to a researcher remotely so that the butterflies do not get distracted or disturbed. Our project is to develop such a handy instrument at very low cost, and components that are easily available in the market.

Chapter 2

Introduction

2.1 Introduction

Air pollution is a global environmental challenge that poses significant threats to human health and the natural ecosystem. Harmful gases emitted into the atmosphere, such as carbon monoxide (CO), nitrogen dioxide (NO_2), sulfur dioxide (SO_2), and ozone (O_3), have far-reaching consequences on both the environment and public well-being.

In response to this pressing concern, we present an Internet of Things (IoT) project aimed at developing an innovative gas detection device capable of monitoring multiple hazardous gases in the environment. The primary objective of this project is to create a robust and efficient system that can continuously monitor the concentration levels of harmful gases in real-time. The device's versatility enables it to detect a wide range of toxic gases, including CO, (CO_2), (SO_2), (NO_2), (NO_3), (NH_2), (O_3), and (H_2S)S. Each of these gases is known to have adverse effects on human health, vegetation, and the delicate balance of ecosystems.

Beyond the immediate focus on human health, our project also takes into account the impact of air pollution on the natural world, particularly on vital pollinators like butterflies. Butterflies play a critical role in pollination, contributing to the reproduction of various plants and sustaining biodiversity. However, exposure to toxic gases can disrupt their life cycles, migratory behaviors, and reproductive patterns, leading to a decline in butterfly populations and the degradation of ecosystems.

At the heart of our gas detection system lies the Arduino Uno microcon-

troller, known for its versatility and widespread usage in IoT applications. Arduino Uno acts as the central hub, efficiently gathering data from a diverse array of gas sensors, including MQ-7 for (CO) detection, MQ-135 for CO2 measurement, and AGSM02A for the detection of multiple harmful gases.

Throughout this documentation, we will delve into the technical aspects of the project, providing detailed insights into the hardware components and circuit connections. The documentation will also elucidate the calibration processes undertaken to ensure accurate and reliable gas measurements.

Furthermore, we will explore the programming aspects, elaborating on the code that facilitates data acquisition from the sensors and enables communication through the SIM900A GSM Module for real-time SMS alerts. This feature allows users to promptly respond to potential hazardous situations, adding a layer of proactive safety to the project's environmental monitoring capabilities.

The project's significance extends beyond the immediate benefits of monitoring harmful gases. By shedding light on the impacts of air pollution on butterflies and other pollinators, we hope to inspire greater awareness and action towards safeguarding the natural environment.

Additionally, the project serves as a stepping stone towards the development of more comprehensive and scalable gas monitoring solutions, leveraging the potential of IoT technology to address pressing environmental challenges.

In conclusion, this IoT-based gas detection project seeks to contribute to the broader efforts of combating air pollution, preserving biodiversity, and promoting ecological balance. By monitoring harmful gases in real-time and understanding their implications on both human health and the natural world, we aim to foster a greener, healthier, and sustainable future for generations to come.

Chapter 3

Hardware Description

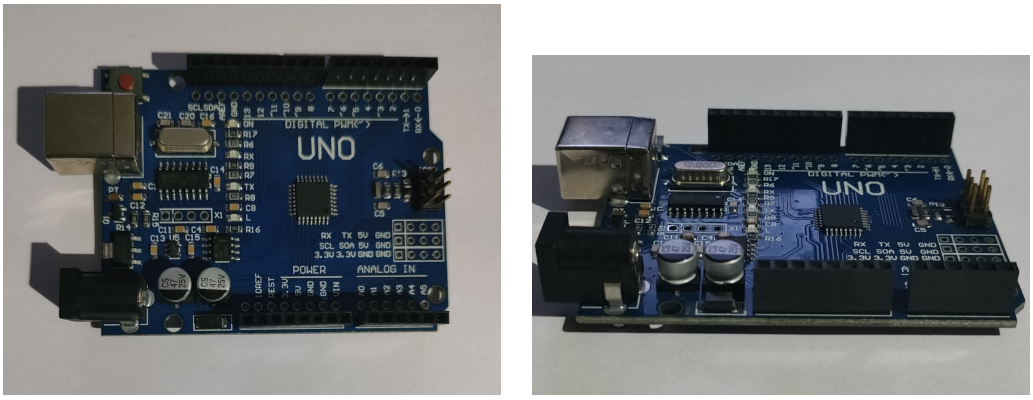
3.1 Introduction

This section will allow users to look at all the hardware information needed to operate the system and additional information on how each and every hardware component works, what is the working principle and etc. The hardware for this project includes an Arduino Uno microcontroller, gas sensors ($MQ-7$ for CO, $MQ-13$ for CO_2, and $AGSM02A$ for SO_2, NO_2, N_O, NH_2, O_3, H_2S), and a SIM900A GSM Module for SMS communication. The Arduino Uno interfaces with the gas sensors to monitor harmful gases. The project aims to develop a cost-effective gas detection device for real-time environmental monitoring and early alerting to protect human health and conserve ecosystems.

3.2 Arduino UNO

It is a microcontroller board developed by Arduino.cc and is based on Atmega328 Microcontroller. The first Arduino project was started in Interaction Design Institute Ivrea in 2003 by David Cuartielles and Massimo Banzi with the intention of providing a cheap and flexible way for students and professionals to learn embedded programming.

The Arduino Uno's user-friendly design makes it an ideal choice for individuals with varying levels of programming and electronics expertise. It features an Atmel ATmega328P microcontroller, which forms the brain of the board and provides processing power for running custom code. The microcontroller operates at a clock speed of 16 MHz and comes with 32KB of



(a) Top View of Arduino Uno (b) Side View of Arduino Uno

Figure 3.1: Top and Side views of Arduino Uno

Flash memory, 2KB of SRAM, and 1KB of EEPROM, providing ample space for storing code and data.

The Arduino Uno's board design includes a set of digital input/output (I/O) pins and analog input pins, allowing users to connect various sensors, actuators, and other electronic components. The board's digital pins can be configured as either input or output, while the analog pins are capable of reading analog voltage levels, making it suitable for a wide range of applications.

Connectivity is another key aspect of Arduino Uno's appeal. It supports serial communication through USB, enabling seamless interaction with computers for code upload, data exchange, and debugging. Moreover, the board is equipped with UART, I2C, and SPI interfaces, facilitating communication with external devices such as sensors, displays, and communication modules.

Arduino Uno's flexibility extends beyond hardware, with its open-source nature encouraging a vast ecosystem of shields, extensions, and add-ons that further expand its capabilities. These shields, which can be stacked on top of the board, provide specialized functionalities for specific projects, enabling users to customize their Arduino-based creations easily.

Arduino Uno stands as a powerful yet approachable microcontroller board that empowers both beginners and experienced makers to bring their electronic ideas to life. Its versatility, user-friendly IDE, extensive community support, and open-source nature make it an indispensable tool in the world

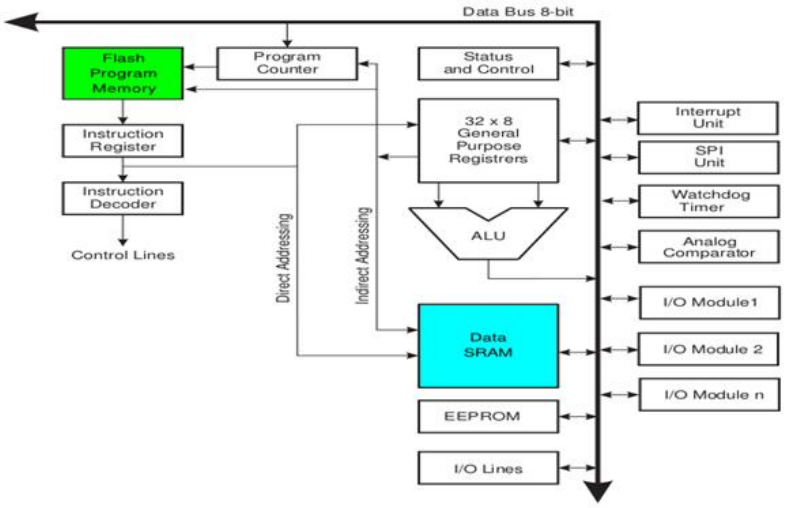


Figure 3.2: Arduino Uno Architecture

of electronics and a driving force behind innovation and creativity. As technology continues to advance, Arduino Uno remains at the forefront, fostering a vibrant community of creators and inspiring a new generation of inventors to explore the exciting world of electronics and programming.

3.2.1 Architecture

Arduino's processor basically uses the Harvard architecture where the program code and program data have separate memory. It consists of two memories- Program memory and the data memory. The code is stored in the flash program memory, whereas the data is stored in the data memory. The Atmega328 has 32 KB of flash memory for storing code (of which 0.5 KB is used for the bootloader), 2 KB of SRAM and 1 KB of EEPROM and operates with a clock speed of 16MHz. A 16 MHz frequency crystal oscillator is equipped on the board.

Clock Source

The microcontroller relies on a crystal oscillator or ceramic resonator as its clock source. In the case of Arduino Uno, it utilizes a 16 MHz crystal oscillator. The clock signal is crucial for synchronizing the operations of the microcontroller and ensuring precise timing of instructions.

Microcontroller ATmega328P

The ATmega328P is a widely-used 8-bit microcontroller that serves as the core component in various electronic projects and development boards, including the popular Arduino Uno. Developed by Microchip Technology, the ATmega328P belongs to the AVR family of microcontrollers, known for their efficient and reliable performance.

At the heart of the ATmega328P is its Reduced Instruction Set Computer (RISC) architecture, designed to optimize the execution of instructions while minimizing power consumption. Operating at a clock speed of 16 MHz, the microcontroller offers impressive processing capabilities suitable for a wide range of applications.

With 32KB of Flash memory, the ATmega328P provides ample space for storing program code. This Flash memory is non-volatile, ensuring that the code remains intact even when the power is turned off. The microcontroller's 2KB of SRAM (Static Random Access Memory) enables efficient data handling during program execution, facilitating temporary storage of variables and data structures.

Additionally, the ATmega328P features 1KB of EEPROM (Electrically Erasable Programmable Read-Only Memory), which provides non-volatile memory for storing data that needs to be retained across power cycles. This makes EEPROM ideal for storing critical system configurations or user-specific settings.

The ATmega328P is equipped with various timers and counters, allowing precise timing and event measurement. These timers are invaluable for tasks such as generating PWM (Pulse-Width Modulation) signals for controlling motors or generating accurate time delays.

The microcontroller boasts a rich set of peripherals, including digital and analog I/O pins, UART, SPI, and I2C interfaces. The 14 digital I/O pins can be configured as either inputs or outputs, enabling seamless interfacing with a plethora of electronic components. Additionally, the 6 analog input pins can accurately read analog voltage levels, making the ATmega328P well-suited for sensor interfacing and analog data acquisition.

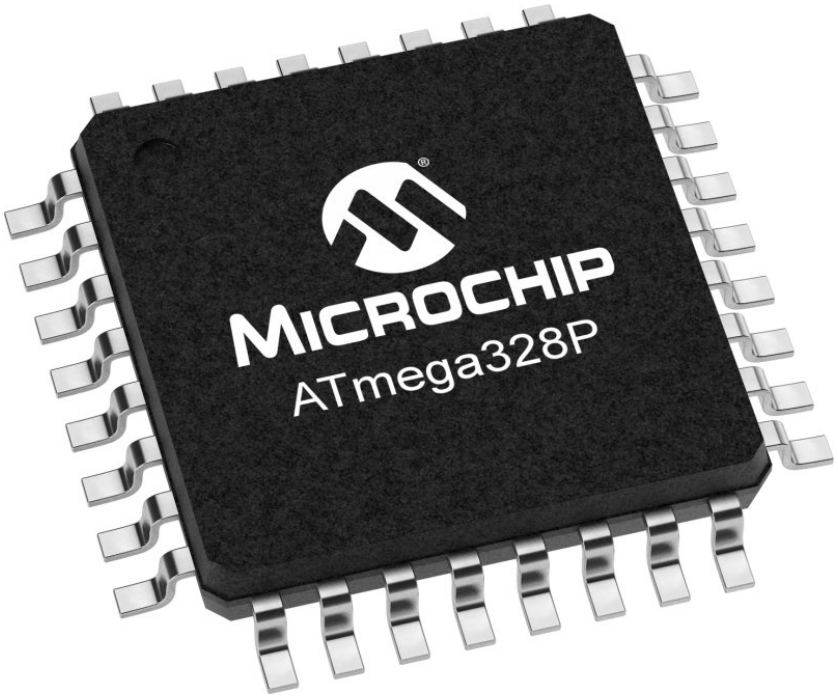


Figure 3.3: ATmega328P

The UART (Universal Asynchronous Receiver/Transmitter) facilitates serial communication, enabling data exchange between the ATmega328P and external devices, such as computers or other microcontrollers. The SPI (Serial Peripheral Interface) and I2C (Inter-Integrated Circuit) interfaces support synchronous and asynchronous communication with external devices, including sensors, displays, and memory chips.

With its versatility, low power consumption, and extensive community support, the ATmega328P has become a cornerstone in the world of embedded systems and DIY electronics. Whether used in commercial products or educational projects, the ATmega328P continues to demonstrate its reliability and flexibility, inspiring a new generation of electronics enthusiasts to explore the endless possibilities of microcontroller-based applications.

Below is the pin configuration of the ATmega328P microcontroller, specifying the functionalities of each pin:

1. Digital I/O Pins (D0 to D13)
    - **D0 to D7:** These pins are general-purpose digital I/O pins that

can be used for both input and output operations. They support features such as digitalRead() and digitalWrite() in Arduino programming.

- **D8 to D13:** Similar to D0 to D7, these pins are also general-purpose digital I/O pins. Additionally, some of these pins have specialized functions, like D9 (PWM) and D10 to D13 (SPI communication).

2. Analog Input Pins (A0 to A5)

   - **A0 to A5:** These pins are analog input pins, used to read analog voltage levels from external sensors or devices. The Arduino programming language provides analogRead() for reading values from these pins.

3. Power Pins

   - **VCC:** The supply voltage pin, typically connected to +5V.
   - **GND:** The ground pin, serving as the reference point for the voltage levels.

4. Reset (RESET)

   - **RESET:** This pin is used to reset the microcontroller. When pulled LOW, it resets the ATmega328P, causing it to restart the program execution from the beginning.

5. Crystal Oscillator Pins (XTAL1 and XTAL2)

   - **XTAL1 and XTAL2:** These are the pins used to connect an external crystal oscillator or ceramic resonator for clock generation. They set the microcontroller's clock frequency.

6. Serial Communication Pins (RX and TX)

   - **RX:** The Receive pin for UART serial communication. It receives data from external devices.
   - **TX:** The Transmit pin for UART serial communication. It sends data to external devices.

7. Special Function Pins

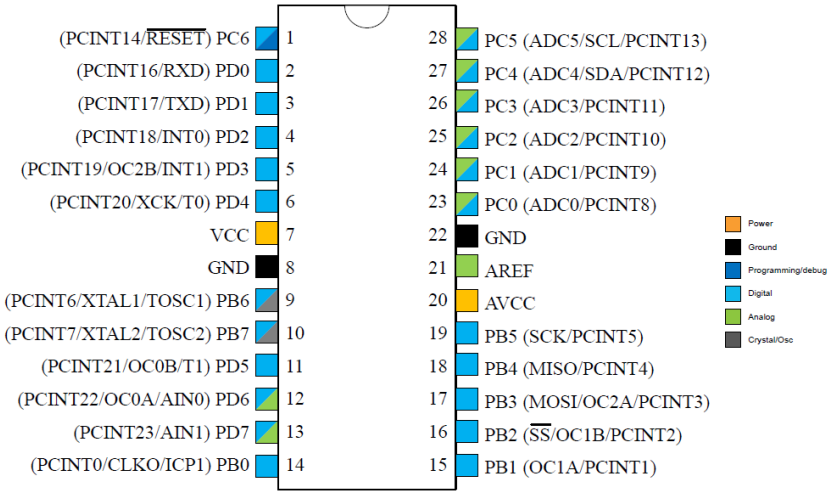


Figure 3.4: ATmega328P Pin Diagram

- **D10 (SS):** This is the Slave Select pin for SPI communication. It is used to select the ATmega328P as the slave device when communicating with other SPI devices.
- **D11 (MOSI):** This is the Master Out Slave In pin for SPI communication. It transmits data from the ATmega328P to other SPI devices.
- **D12 (MISO):** This is the Master In Slave Out pin for SPI communication. It receives data from other SPI devices.
- **D13 (SCK):** This is the Serial Clock pin for SPI communication. It provides the clock signal for synchronizing data transmission between devices.

8. Power Supply Pins

   - **AVCC:** The supply voltage pin for the analog-to-digital converter (ADC). It is typically connected to +5V when using the internal ADC reference voltage.
   - **AREF:** The analog reference voltage pin for the ADC. It can be used to set an external reference voltage for more precise analog-to-digital conversions.

Memory unit

The memory unit of the Arduino Uno is a crucial component that plays a significant role in storing and processing data for the microcontroller. It con-

sists of three main types of memory: Flash memory, SRAM (Static Random Access Memory), and EEPROM (Electrically Erasable Programmable Read-Only Memory).

1. **Flash Memory**: Arduino Uno is equipped with 32KB of Flash memory, which serves as the primary storage for the user's program code. Flash memory is non-volatile, meaning the code remains intact even when the power is turned off. This characteristic ensures that the program will be retained and executed whenever the microcontroller is powered on. Program code is uploaded to the Arduino Uno's Flash memory using the integrated development environment (IDE) and a USB connection. The substantial capacity of Flash memory allows for the implementation of complex programs, making the Arduino Uno suitable for a wide range of applications, from simple blinking LED projects to sophisticated IoT applications.

2. **SRAM (Static Random Access Memory):** The Arduino Uno is equipped with 2KB of SRAM, which serves as the temporary workspace for data during program execution. Unlike Flash memory, SRAM is volatile, meaning it loses its data when the power is turned off. During program runtime, variables, arrays, and other data structures are stored in SRAM, allowing the microcontroller to perform calculations and manipulations efficiently. The limited capacity of SRAM requires careful memory management to avoid running out of memory, especially in more complex projects with extensive data requirements.

3. **EEPROM (Electrically Erasable Programmable Read-Only Memory):** Arduino Uno includes 1KB of EEPROM, a non-volatile memory used for storing data that needs to be retained across power cycles. Unlike Flash memory, which is typically used for program code, EEPROM is intended for storing configuration settings, user preferences, or other critical data that should persist even when the microcontroller is powered off and on again.

In summary, the memory unit of the Arduino Uno consists of Flash memory for storing the program code, SRAM for temporary data storage during program execution, and EEPROM for non-volatile data storage. The synergy between these memory types enables the microcontroller to execute user programs effectively and retain important data across power cycles, making it a powerful and versatile platform for various embedded systems and DIY electronics projects.

I/O ports

The Arduino Uno features a set of versatile Input/Output (I/O) ports that serve as the interface between the microcontroller and external electronic components. These I/O ports are essential for connecting sensors, actuators, displays, and other devices, allowing the Arduino Uno to interact with the external world and execute a wide range of electronic projects.

1. **Digital I/O Ports:** The Arduino Uno has a total of 14 digital I/O pins, numbered D0 to D1
2. These pins can be individually configured as either inputs or outputs, providing flexibility in interfacing with various electronic components. When set as inputs, the digital pins can read the state of external devices, such as buttons or switches. When set as outputs, these pins can drive LEDs, motors, relays, and other actuator devices. Arduino programming enables simple control of digital I/O pins using functions like digitalWrite() and digitalRead().
3. **Analog Input Ports:** Arduino Uno features 6 analog input pins, labeled A0 to A5
4. These pins are capable of measuring analog voltage levels from 0 to 5 volts. Analog input is vital for interfacing with sensors that provide analog output, such as light sensors, temperature sensors, and potentiometers. Arduino programming allows reading analog values from these pins using the analogRead() function.
5. **Power Pins:** The Arduino Uno includes VCC and GND pins, which provide the power supply and ground reference for the board and connected devices. VCC typically receives a +5V supply, while GND serves as the reference for voltage measurements and ensures proper grounding.
6. **Special Function Pins:** Certain digital I/O pins have special functions beyond standard input/output operations. For instance:
    - D0 (RX) and D1 (TX) pins are used for serial communication (UART).
    - D2 (interrupt 0) and D3 (interrupt 1) pins can trigger interrupts to respond to external events.

- D3, D5, D6, D9, D10, and D11 pins are capable of Pulse Width Modulation (PWM) for precise control of motor speed, LED brightness, and more.

The I/O ports of the Arduino Uno are instrumental in enabling the microcontroller to communicate with the external world and respond to inputs, making it a versatile and accessible platform for countless electronic projects and experiments. The combination of digital and analog I/O pins provides a balance between simple digital control and more complex analog data acquisition, allowing users to explore a wide range of applications and unleash their creativity in the world of electronics and programming.

3.2.2 Operation

The Arduino Uno is an open-source microcontroller board based on the ATmega328P microcontroller, known for its simplicity and versatility, making it an ideal platform for a wide range of electronic projects. The operation of the Arduino Uno involves several key processes, including program upload, setup, loop, and interaction with external components through Input/Output (I/O) pins and communication interfaces.

1. **Powering the Arduino Uno:** The Arduino Uno can be powered through either the USB port or an external power source connected to the power jack. When connected to a computer via USB, the board receives power and establishes a serial communication link for program upload and data exchange.

2. **Program Upload (Sketch):** The first step in the operation of the Arduino Uno is to upload the program, also known as a sketch. Users write the program in the Arduino Integrated Development Environment (IDE), which uses the Wiring programming language. The sketch is a set of instructions that define the behavior of the microcontroller and how it interacts with the external world through its I/O pins. Once the sketch is ready, it is uploaded to the Arduino Uno via the USB connection.

3. **Setup Function:** After program upload, the microcontroller begins its operation by executing the setup() function. This function is run once at the start of the program and is used to initialize variables, configure I/O pins, and set up any required communication interfaces.

4. **Loop Function:** Following the setup() function, the Arduino Uno enters the loop() function. This function contains the main code that will be executed repeatedly in a continuous loop. The loop() function forms the heart of the program and controls the board's behavior during its operation.

5. **Digital and Analog Input/Output (I/O) Pins:** The Arduino Uno has a total of 14 digital I/O pins (D0 to D13) and 6 analog input pins (A0 to A5). Digital I/O pins can be configured as either inputs or outputs using the pinMode() function. As outputs, these pins can drive LEDs, motors, and other actuator devices using digitalWrite(). As inputs, they can read the state of buttons, switches, and other sensors using digitalRead().

6. A**nalog Readings:** The analog input pins can read analog voltage levels from 0 to 5 volts using analogRead(). This is crucial for interfacing with analog sensors, such as light sensors or temperature sensors, and performing analog data acquisition.

7. **Serial Communication:** Arduino Uno supports serial communication through its RX (Receive) and TX (Transmit) pins. The Serial library provides functions like Serial.begin() to initialize the serial communication and Serial.print() to send data to a connected computer or external device.

8. **Pulse Width Modulation (PWM):** Several digital pins (D3, D5, D6, D9, D10, and D11) on the Arduino Uno can generate PWM signals using the analogWrite() function. PWM allows precise control of the output's average voltage, making it useful for dimming LEDs, controlling motor speed, and more.

9. **Timers and Interrupts:** The ATmega328P microcontroller on the Arduino Uno has timers that can be used to measure time intervals and generate accurate time delays. Additionally, interrupts can be utilized to respond to external events, such as button presses, without the need for continuous polling.

10. **External Communication:** The Arduino Uno can communicate with other devices using protocols such as I2C, SPI, and UART. These interfaces enable the Arduino Uno to interact with sensors, displays, memory modules, and other microcontrollers or external devices.

11. **Loop Execution:** The *loop*() function continues to execute in a continuous loop until the power is turned off or the microcontroller is reset. The Arduino Uno can interact with external components, read sensors, and perform calculations based on the program logic during this iterative process.

The operation of the Arduino Uno involves powering the board, uploading the program (sketch), executing the setup() and loop() functions, interfacing with external components through digital and analog I/O pins, performing analog readings, utilizing serial communication, and engaging PWM, timers, and interrupts. This combination of features and processes allows users to create diverse and interactive projects, making the Arduino Uno a powerful and accessible platform for electronics enthusiasts, students, and professionals alike.

3.3 Sensors

The primary purpose of sensors is to provide real-time data about the physical world, enabling us to monitor, control, and respond to changes in our surroundings. They play a crucial role in automation, data collection, and feedback systems.

3.3.1 What is a sensor?

A sensor is a device or an instrument that detects and measures physical properties or changes in the environment and converts them into electrical signals or other readable formats. Sensors are fundamental components of various systems, ranging from simple everyday devices to complex industrial applications and scientific instruments.

3.3.2 Characteristics of sensor

Sensors exhibit several important characteristics that define their performance and suitability for specific applications. Some key characteristics of sensors include:

1. **Sensitivity:** Sensitivity refers to the ability of a sensor to detect and respond to small changes in the measured quantity. A highly sensitive

sensor can detect even minor variations, while less sensitive ones require more significant changes to register a response.

2. **Range:** The range of a sensor indicates the minimum and maximum values of the measured property that it can detect accurately. It's essential for a sensor's range to match the expected range of values in the application to avoid saturation or loss of accuracy.

3. A**ccuracy**: Accuracy represents how close the sensor's output is to the actual or true value of the measured property. Highly accurate sensors provide measurements with minimal deviation from the true value.

4. **Precision:** Precision refers to the ability of a sensor to provide consistent and repeatable measurements when exposed to the same input. Even if a sensor is precise, it may not necessarily be accurate if its measurements deviate consistently from the true value.

5. **Resolution:** Resolution is the smallest detectable change in the measured property that a sensor can distinguish. It is related to the sensor's sensitivity and precision. High resolution allows the sensor to detect small changes in the measured property.

3.3.3 Types of sensors

1. **Temperature Sensor:** Measures temperature variations and is used in applications such as weather monitoring, climate control, and industrial processes.

2. **Pressure Sensor:** Detects changes in pressure and finds use in devices like barometers, tire pressure monitors, and industrial pressure monitoring systems.

3. **Proximity Sensor:** Detects the presence or absence of an object without direct contact and is commonly used in touchscreens, automatic doors, and object detection systems.

4. **Light Sensor:** Detects changes in light levels and is used in automatic lighting systems, photography, and ambient light adjustment in electronic devices.

5. **Humidity Sensor (Hygrometer):** Measures the amount of moisture in the air and is used in weather stations, HVAC systems, and environmental monitoring.

Figure 3.5: MQ-7 Sensor

6. **Gas Sensor:** Detects the presence and concentration of specific gases in the environment and is used for gas leak detection, air quality monitoring, and industrial safety.

7. **Infrared Sensor:** Detects infrared radiation and is used in applications like temperature measurement, motion detection, and remote controls.

8. **pH Sensor:** Measures the acidity or alkalinity of a solution and is employed in laboratory analysis, water quality monitoring, and hydroponics.

9. **Image Sensor:** Captures visual information and is used in digital cameras, surveillance systems, and medical imaging devices.

3.4 Sensors used in this Device

Here we are using the Gas sensors. The sensors used for CO detection is MQ-7, for detecting CO_2 is MQ-135 sensor, for detecting SO_2, NO_2, NO, O3, H_2S is AGSM02A (TVOC Sensor) and DHT11 sensor for Temperature and Humidity detection.

3.4.1 MQ-7 Sensor

The MQ-7 sensor is a type of gas sensor widely used to detect carbon monoxide (CO) gas. It is part of the MQ series of gas sensors manufactured by the company Winsen. The MQ-7 sensor is known for its simplicity, ease of use, and cost-effectiveness, making it a popular choice for various applications, including gas leakage detection, industrial safety, and environmental monitoring. The MQ-7 sensor is available in a compact and portable module, making it easy to integrate into various applications. It often comes with a set of three or four pins for power supply, ground connection, and analog output (voltage output). The sensor's analog output voltage varies with the concentration of carbon monoxide gas present in the surrounding environment.

Specifications of MQ-7 sensor

1. **Gas Detected:** Carbon Monoxide (CO).
2. **Sensing Material:** Tin Dioxide (SnO_2).
3. **Heater Voltage:** Typically 5V DC
4. **Heater Power Consumption:** Typically around 350mW
5. **Sensing Resistance:** R_0 (in clean air, typically around $10K\Omega$ to $20K\Omega$)
6. **Target Gas Concentration Range**: Usually 10 to 500 ppm CO
7. **Sensitivity:** Response factor (R_s/R_0) to CO gas concentration
8. **Load Resistance** (R_L)**:** Recommended load resistance for the sensor output circuit (e.g., $10K\Omega$ to $47K\Omega$)
9. **Operating Temperature:** Typically $-10C$ to $+50C$
10. **Output Signal:** Analog voltage output (varies with CO concentration)

Materials Used in MQ-7 Sensor

The main sensing material in the MQ-7 sensor is Tin Dioxide (SnO_2). SnO_2 is a metal oxide semiconductor with excellent gas-sensing properties, particularly for detecting reducing gases like carbon monoxide (CO). The material's

surface interacts with CO gas, causing changes in its electrical conductivity.

The sensor package typically includes an encapsulation to protect the internal components and prevent environmental influences from affecting the sensor's performance.

Internal Circuit and Working Principle

When the heated sensing element comes into contact with the target gas, such as carbon monoxide (CO), the gas molecules are adsorbed onto the surface of the SnO_2 material. This adsorption process changes the conductivity of the SnO_2 semiconductor. The presence of CO gas lowers the resistance of the sensing element, while clean air or other gases result in higher resistance.

The resistance change caused by the adsorption of CO gas is then measured, and the sensor's output voltage varies accordingly. The output voltage can be read using an analog-to-digital converter (ADC) or interfaced directly with microcontrollers or other electronics for further processing and interpretation.

The MQ-7 sensor detects carbon monoxide (CO) gas in the atmosphere through a chemical reaction that occurs on its sensing element, typically made of tin dioxide (SnO_2). When the sensing element is exposed to CO gas, a series of chemical reactions take place, leading to changes in its electrical conductivity. Let's look at the chemical reactions and the resulting changes that enable the MQ-7 sensor to detect CO.

1. **Oxidation of Tin Dioxide (SnO_2):** At room temperature, the sensing element (SnO_2) is in contact with the surrounding air, including oxygen (O_2). SnO_2 is a semiconductor, and in its initial state, it has a certain level of electrical resistance.

   SnO_2 (sensing element) + O_2 (from the air) $\rightarrow$ SnO_2 (unchanged)

2. **Adsorption of Carbon Monoxide** (CO)**:** When carbon monoxide (CO) gas is present in the atmosphere and comes into contact with the sensing element, the CO molecules get adsorbed onto the surface of the SnO_2 semiconductor. This adsorption process changes the conductivity of SnO_2.

$\rightarrow$ SnO_2-CO complex

3. **Change in Conductivity:** The presence of CO on the surface of SnO_2 causes a reduction in the electrical resistance of the sensing element. This is because the adsorbed CO molecules donate electrons to the SnO_2 semiconductor, leading to an increase in the number of charge carriers (electrons) available for conduction.

   With the decrease in resistance, the electrical conductivity of the SnO_2 sensing element increases significantly. This change in conductivity is the key factor that allows the MQ-7 sensor to detect the presence of CO gas.

4. **Output Voltage Variation:** The changes in conductivity of the SnO_2 sensing element result in variations in the output voltage of the MQ-7 sensor. The output voltage is proportional to the concentration of CO gas present in the environment. Higher concentrations of CO lead to more significant changes in conductivity and, consequently, a higher output voltage. By measuring and interpreting the output voltage, the sensor can provide an estimate of the CO gas concentration in parts per million (ppm). The actual relationship between output voltage and CO concentration is calibrated during the sensor's manufacturing process and is specific to each sensor model.

Significance of MQ-7 Sensor:

The MQ-7 sensor holds significant importance in various applications due to its ability to detect carbon monoxide (CO) gas. Carbon monoxide is a colorless, odorless, and tasteless gas that is highly toxic to humans and animals when inhaled in high concentrations. The MQ-7 sensor's ability to detect CO gas is crucial for the following reasons:

1. **Research and Experimentation:** The sensor is widely used in research and experimentation to study the effects of CO exposure and to evaluate the efficiency of CO mitigation strategies.

Figure 3.6: MQ-135

2. **Internet of Things (IoT) Devices:** Due to its low cost, simplicity, and ease of integration, the MQ-7 sensor finds application in IoT devices, enabling smart CO monitoring and control systems.

3.4.2 MQ-135 Sensor

The MQ-135 sensor is a gas sensor commonly used to detect and measure air quality and the presence of various harmful gases in the atmosphere. It is part of the MQ series of gas sensors manufactured by the company Winsen. The MQ-135 sensor is particularly sensitive to gases such as ammonia (NH_3), nitrogen oxides (NO_x), CO_2, benzene, and other volatile organic compounds (VOC).

In MQ-135, it consists of 1k load resistance which we have replaced by using a $22K\Omega$ load resistance in place of it. Then we have made certain modifications in the code as the default code consists of 1k resistance & we inserted the latest value of the global CO_2 level in the code from internet. This is how we make it a CO_2 detecting sensor.

Specifications of MQ-135 sensor

1. **Gas Detected:** Ammonia (NH_3), nitrogen oxides (NO_x), Carbon-dioxide(CO_2), benzene, and other volatile organic compounds (VOCs).

2. **Sensing Material:** Typically made of SnO_2 (tin dioxide), which is a semiconductor sensitive to the target gases.

3. **Heater Voltage:** Typically operates at 5V DC.

4. **Heater Power Consumption:** Typically around 800mW.

5. **Target Gas Concentration Range:** The sensitivity to different gases varies, but it is commonly used for detecting concentrations in the range of several tens to hundreds of parts per million (ppm).

6. **Sensitivity:** The sensor's response factor (Rs/R0) to different target gases at specific concentrations.

7. **Load Resistance (R_L):** Recommended load resistance for the sensor output circuit (changed from $1K\Omega$ to 22Ω).

8. **Operating Temperature:** Typically $-10^0\ C$ to $+50^0 C$.

9. **Output Signal:** Analog voltage output (varies with gas concentration).

Materials Used in MQ-135 Sensor

The primary sensing material in the MQ-135 sensor is Tin Dioxide (SnO_2). The SnO_2 layer's surface provides a platform for gas adsorption, allowing the sensor to detect the presence of various harmful gases in the atmosphere. Additionally, the sensor may contain other materials to enhance its performance and selectivity to specific gases.

To clarify, the MQ-135 sensor is primarily designed to detect gases such as ammonia (NH_3), nitrogen oxides (NO_x), benzene, and other volatile organic compounds (VOCs). It is not specifically sensitive to carbon dioxide (CO_2).

Internal Circuit and Working Principle

The MQ-135 sensor operates based on the principle of gas adsorption and desorption on its sensing material. Inside the sensor, there is a sensitive layer made of SnO_2 (tin dioxide), which is a semiconductor material with high sensitivity to the target gases.

Working Principle

1. **Infrared Light Source:** The CO_2 sensor contains an infrared light source that emits infrared radiation at a specific wavelength, typically around 26 micrometers, which corresponds to the absorption band of CO_2.

2. **Gas Sample Chamber:** The sensor has a gas sample chamber or optical path through which the air containing CO_2 is passed.

3. A**bsorption of Infrared Radiation:** As the gas sample passes through the chamber, the CO_2 molecules present in the air absorb specific wavelengths of infrared light.

4. **Detector:** The sensor also includes a detector that measures the intensity of the infrared light after it passes through the gas sample.

5. **Reference Channel:** CO_2 sensors often include a reference channel that allows the sensor to compensate for changes in the light source's intensity and other environmental factors.

6. **Signal Processing:** The sensor's electronics process the detected infrared light intensity, considering the reference channel's information, and calculate the CO_2 concentration based on the measured absorption.

7. **Output:** The sensor provides an analog or digital output signal that represents the concentration of carbon dioxide (CO_2) in parts per million (ppm).

CO_2 sensors are widely used for indoor air quality monitoring, industrial processes, HVAC systems, and various other applications where accurate and reliable CO_2 measurements are essential.

Significance of MQ-135 Sensor

The MQ-135 sensor holds significant importance in various applications due to its ability to detect a wide range of harmful gases, especially ammonia (NH_3), nitrogen oxides (NO_x), benzene, and other volatile organic compounds (VOCs). Here are some of the key significances of the MQ-135 sensor:

1. **Air Quality Monitoring:** One of the primary applications of the MQ-135 sensor is in air quality monitoring systems. It helps in assessing indoor and outdoor air quality by detecting the presence of harmful gases that can adversely affect human health and the environment.

2. **Research and Experimentation:** The sensor is widely used in research and experimentation to study the effects of different gases on the environment and to evaluate the efficiency of pollution control measures.

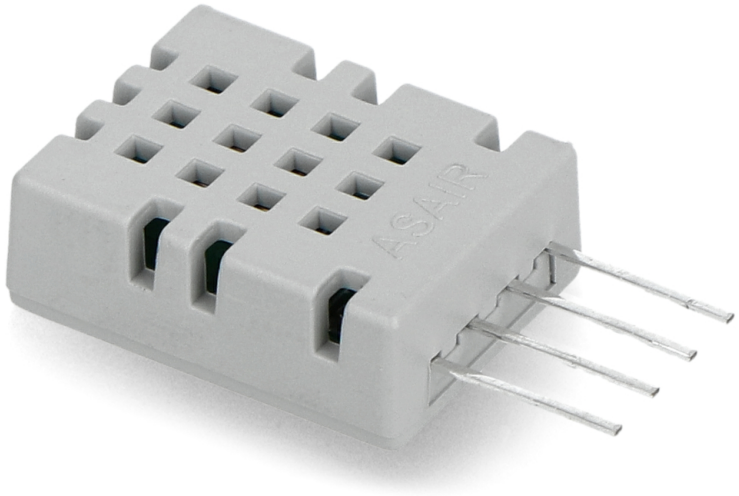


Figure 3.7: AGS02MA TVOC sensor

3. **Internet of Things (IoT) Devices:** Due to its low cost, simplicity, and ease of integration, the MQ-135 sensor finds application in IoT devices, enabling smart gas monitoring and control systems.

4. **Portable Gas Detectors:** The MQ-135 sensor is often used in portable gas detectors, providing a compact and cost-effective solution for personal safety and gas monitoring on-the-go.

3.4.3 AGS02MA Sensor(TVOC Sensor)

The AGS02MA sensor is available as a compact and integrated module, making it easy to incorporate into various gas detection systems and devices. The sensor is designed for low power consumption, making it suitable for battery-operated applications. It is often used in portable gas detectors, industrial safety equipment, and environmental monitoring systems.

Specifications of the AGS02MA sensor:

1. **Gas Detected:** NH_3, NO, NO_2, O_3, SO_2

2. **Sensing Principle:** Catalytic combustion sensor

3. **Heater Voltage:** Typically operates at 5V DC

4. **Heater Power Consumption:** Typically around 800mW
5. **Target Gas Concentration Range:** Typically measures methane concentrations in parts per billion (ppb)
6. **Sensitivity:** The sensor's response factor (Rs/R0) to methane gas at specific concentrations
7. **Warm-up Time:** Time required for the sensor to stabilize after power-on and reach its operational state
8. **Operating Temperature**: The temperature range at which the sensor can operate effectively (e.g., -10℃ to +50℃)
9. **Output Signal:** The type of output signal provided by the sensor, which may be an analog voltage proportional to the concentration of methane gas in the environment consumption make it suitable for various applications requiring reliable and accurate methane gas detection.

Materials Used in AGS02MA Sensor

The AGS02MA sensor utilizes a catalytic combustion element made of platinum or other noble metals, which act as a catalyst for the detection of combustible gases like methane. Additionally, the sensor module may include electronic components for heating the catalytic element and processing the sensor's output signal.

Internal Circuit and Working Principle

The AGS02MA sensor operates based on the principle of a catalytic combustion sensor for detecting combustible gases like methane (CH4). The internal circuit of the sensor typically includes the following components:

1. **Catalytic Combustion Element:** The sensor contains a catalytic combustion element made of platinum or other noble metals. This element is sensitive to combustible gases like methane.
2. **Heater Element:** The catalytic combustion element is heated to a specific temperature by a built-in heater element. The heater helps to maintain a constant temperature of the catalytic element.

3. **Electrodes:** The sensor has electrodes connected to the catalytic element. When the target gas (e.g., methane) comes into contact with the catalytic element, it undergoes catalytic combustion, leading to a change in electrical resistance.

Working Principle for Methane (CH4) Detection

1. **Detection of Methane:** When methane gas (CH4) comes in contact with the heated catalytic element, it reacts with oxygen (O_2) in the air, resulting in catalytic combustion.

   CH_4 (methane) + $2O_2$ (oxygen) $\rightarrow$ CO_2 (carbon dioxide) + $2H_2O$ (water) + Heat

2. **Resistance Change:** The catalytic combustion of methane on the surface of the catalytic element causes a change in its electrical resistance.

3. **Signal Processing:** The sensor's internal circuit measures the change in resistance and converts it into an electrical signal.

4. **Output Signal:** The sensor provides an output signal, which is usually an analog voltage proportional to the concentration of methane gas in the environment.

Significance of AGS02MA Sensor

1. **Gas Leak Detection:** The sensor's sensitivity to methane gas makes it valuable for gas leak detection applications in various settings, including homes, commercial buildings, and industrial facilities. Detecting methane leaks early can prevent potential hazards and minimize environmental impact.

2. **Portable Gas Detectors:** The AGS02MA sensor is used in portable gas detectors, providing a compact and cost-effective solution for personal safety and gas monitoring on-the-go. These detectors are essential for workers in potentially hazardous environments.

3. **Smart Home Automation:** The AGS02MA sensor can be integrated into smart home automation systems to monitor methane levels and trigger alerts or shut off gas supply lines in case of gas leaks, ensuring enhanced safety in residential settings.

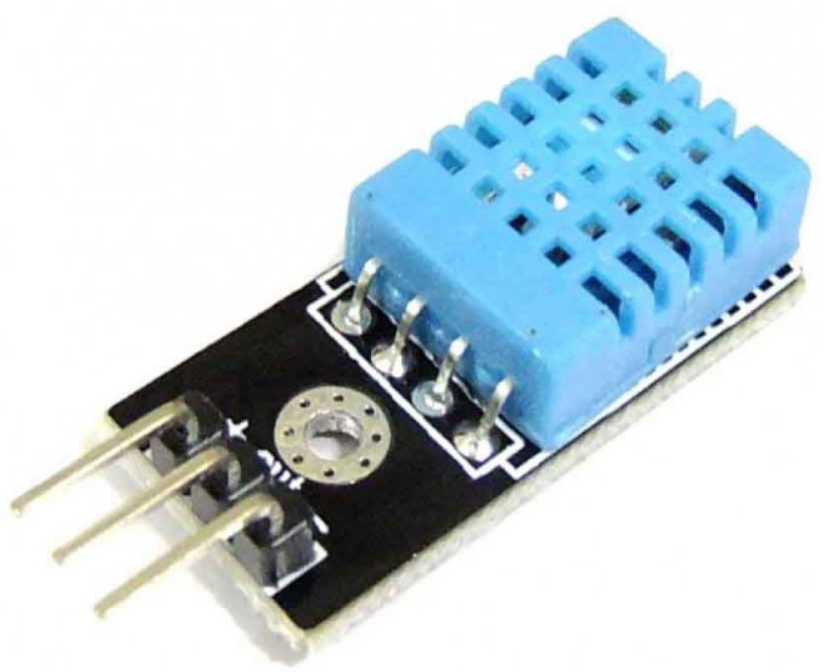


Figure 3.8: DHT11 sensor

4. **Internet of Things (IoT) Devices:** Due to its low power consumption and compact size, the AGS02MA sensor is suitable for IoT devices, enabling smart gas monitoring and control systems.

5. **Research and Experimentation:** The AGS02MA sensor is also used in research and experimentation to study the behavior of methane gas in various conditions and environments.

3.4.4 DHT11 sensor

The DHT11 sensor is available as a compact module with three or four pins, depending on the variant. The three-pin module includes pins for power supply (VCC), ground (GND), and data (data pin). The four-pin module includes an additional pin for a digital output signal (data pin).

The DHT11 sensor operates at a low voltage, typically around 3V to 5V DC, making it suitable for use with microcontrollers and other electronic devices. The sensor provides relatively basic accuracy for temperature and humidity measurements, making it suitable for applications where high precision is not critical.

The DHT11 sensor's simplicity, low cost, and ease of use make it a popular choice for hobbyists, students, and DIY projects. It is commonly used in home automation, weather monitoring systems, environmental monitoring, and various Internet of Things (IoT) applications.

Specifications of the DHT11 sensor:

1. **Sensor Type:** Digital Temperature and Humidity Sensor
2. **Sensing Parameters:** Measures both temperature and relative humidity
3. **Temperature Measurement Range:** 0°C to 50°C (32°F to 122°F)
4. **Temperature Measurement Accuracy:** ±2°C (±
5. 6°F)
6. **Humidity Measurement Range**: 20% RH to 90% RH
7. **Humidity Measurement Accuracy:** ±5% RH
8. **Operating Voltage:** 3V to 5V DC
9. **Operating Current**: Typically around
10. 5mA during active measurement
11. **Signal Interface:** Single-wire digital interface (one data pin)
12. **Response Time:** The time taken to provide a stable reading after a change in the environmental conditions.

Materials Used in DHT11 Sensor

The DHT11 sensor's sensing element is typically made of a moisture-absorbing polymer material. The temperature sensor component is a thermistor, which can be made from various semiconductor materials like metal oxides or ceramics. Other materials used in the sensor's construction include the integrated circuit (IC) for signal processing and the single-wire communication interface.

Internal Circuit and Working Principle

1. **Sensing Element:** The DHT11 sensor uses a humidity-sensitive element, typically made of a moisture-absorbing polymer. This element changes its electrical resistance based on the amount of moisture present in the air.

2. **Temperature Sensor:** The sensor also includes a thermistor, which is a type of resistor that changes its resistance with temperature variations. The thermistor is used to measure the ambient temperature.

3. **Analog-to-Digital Converter (ADC):** The sensor's internal circuitry includes an ADC to convert the analog signals from the humidity and temperature sensors into digital data that can be processed by microcontrollers or other electronic devices.

4. **Signal Processing:** The DHT11 sensor's onboard electronics process the digital data from the ADC and apply calibration and compensation algorithms to calculate the temperature and humidity values.

5. **Timing and Communication:** The DHT11 sensor communicates with external devices using a single-wire interface. It sends data in a time-division multiplexing manner, where it transmits 40 bits of data in a specific time sequence.

6. **Integrated Pull-Up Resistor:** The sensor includes a built-in pull-up resistor that keeps the communication line (data pin) at a high logic level when not actively transmitting data.

Significance of DHT11 Sensor

1. **Cost-Effectiveness:** The DHT11 sensor is highly affordable and readily available, making it accessible to hobbyists, students, and DIY enthusiasts. Its low cost allows for widespread use in various projects and applications.

2. **Simplicity and Ease of Use:** The DHT11 sensor features a simple interface with only a few pins, making it easy to integrate into electronics projects and microcontroller-based systems. Its straightforward operation allows even beginners to utilize it effectively.

3. **Environmental Monitoring:** The DHT11 sensor is commonly used in environmental monitoring systems to measure temperature and humidity levels in indoor and outdoor environments. Monitoring these

parameters helps in assessing comfort levels, identifying potential issues like high humidity causing condensation or low humidity leading to dryness, and implementing appropriate measures for improvement.

4. **Home Automation:** In home automation projects, the DHT11 sensor is employed to monitor indoor temperature and humidity, providing data for climate control and optimizing energy consumption in heating, ventilation, and air conditioning (HVAC) systems.

5. **Weather Stations:** The DHT11 sensor is suitable for entry-level weather stations used by hobbyists and educational institutions. It allows users to collect basic weather data, including temperature and humidity, to gain insights into local weather conditions.

6. **Internet of Things (IoT)** Devices: Due to its low cost and ease of use, the DHT11 sensor is popular in IoT projects where temperature and humidity measurements are essential for various smart home and industrial applications.

7. **Indoor Air Quality (IAQ) Monitoring:** The DHT11 sensor aids in indoor air quality assessment by measuring temperature and humidity. Understanding indoor environmental conditions is crucial for creating a healthier living and working environment.

8. **Remote Sensing:** The DHT11 sensor can be used in remote sensing applications, such as data loggers and environmental monitoring in hard-to-reach locations.

3.5 GSM Module

A GSM module is a hardware device used to enable communication over the Global System for Mobile Communications (GSM) network. GSM is the most widely used standard for mobile communication around the world. GSM modules are commonly used in various electronic projects and applications that require wireless communication capabilities.

The GSM module typically consists of a GSM modem, which is responsible for transmitting and receiving data over the GSM network, and other necessary components like a SIM card holder, antenna connector, and serial communication interface (such as UART or USB) for connecting to other devices.

Figure 3.9: SIM 900 Module

KEY FEATURES OF GSM MODULE
Key features of a GSM module include:

1. **Communication: GSM modules can send and receive SMS messages, make and receive voice calls, and connect to the internet through GPRS** (General Packet Radio Service) for data transmission.

2. **SIM Card Support:** The module requires a valid GSM SIM card, just like a mobile phone, to operate on the cellular network.

3. **Power Supply:** GSM modules usually operate at low power and can be powered by various sources, including batteries and external power supplies.

4. **AT Commands:** GSM modules communicate with the controlling device (such as a microcontroller or a computer) using AT (Attention) commands, which are simple text-based instructions.

The GSM module's remote sensing capability offers a significant advantage in monitoring gas concentrations in distant locations without human

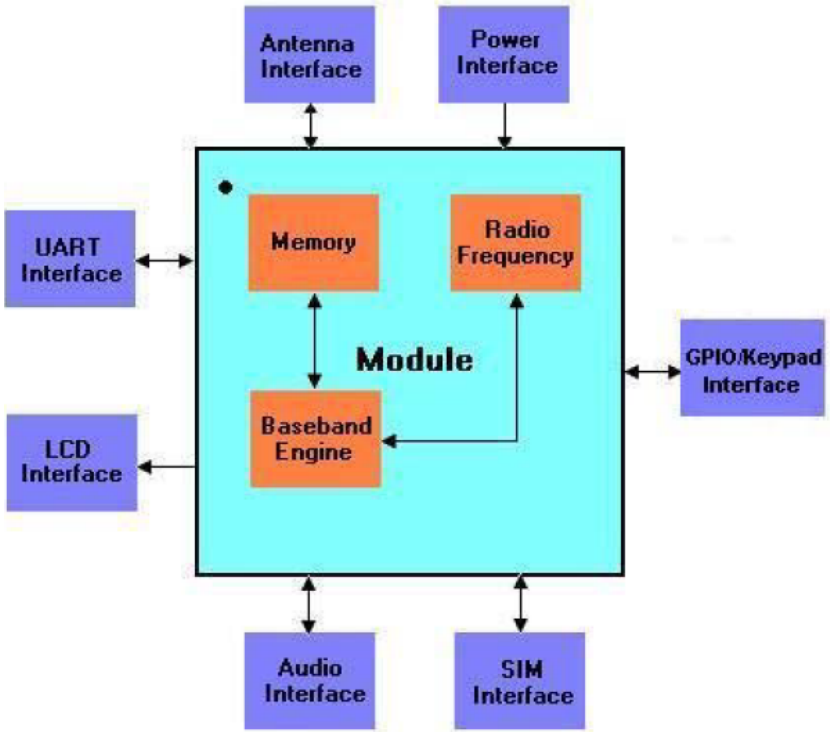


Figure 3.10: SIM 900 Module Block Diagram

intervention. When placed at a remote site, the GSM module can efficiently provide gas readings, allowing researchers to assess the habitat's suitability for pollinators without disturbing their living environment.

In contrast, the IoT gas monitoring device goes beyond remote sensing by offering real-time data transmission and continuous monitoring capabilities. It not only detects harmful gases but also provides insights into their temporal variations, trends, and potential pollutant sources. This level of detail enables comprehensive analysis of air quality and its impact on pollinators, aiding in the formulation of targeted conservation measures. Additionally, the IoT device can alert relevant authorities and stakeholders to rapidly respond to sudden changes in gas concentrations, ensuring timely actions to safeguard pollinator habitats and biodiversity. That's why we will be using SIM900A GSM module for this project.

3.5.1 SIM900A GSM MODULE

The SIM900A is a popular GSM (Global System for Mobile Communications) module manufactured by SIMCom. It is an improved version of the SIM900 module and is widely used in various applications that require GSM communication capabilities. The SIM900A module offers features like SMS messaging, voice calling, and GPRS data transmission, making it suitable for a wide range of applications. Key features of the SIM900A GSM module include:

1. **Quad-Band Support:** The SIM900A module supports four GSM frequency bands, making it compatible with GSM networks worldwide. These bands are 850 MHz, 900 MHz, 1800 MHz, and 1900 MHz.
2. **Communication Interfaces:** The module typically communicates with other devices using standard serial communication protocols like UART (Universal Asynchronous Receiver-Transmitter).
3. **SIM Card Holder:** The SIM900A module requires a valid GSM SIM card to operate. The SIM card holder is usually integrated into the module.
4. **AT Commands:** Like most GSM modules, the SIM900A is controlled using AT (Attention) commands, which are simple text-based instructions sent over the serial interface.
5. **Voice Calls:** The module can make and receive voice calls, allowing two-way voice communication between users.
6. **SMS Messaging:** It supports SMS (Short Message Service) messaging, enabling the sending and receiving of text messages.
7. **GPRS Data Transmission:** The module can connect to the internet using GPRS, allowing data transmission for applications like sending and receiving data over the internet.
8. **GPIO Pins:** The SIM900A module typically includes GPIO (General Purpose Input/Output) pins that can be used to interface with external devices and sensors.

Applications of the SIM900A GSM module include:

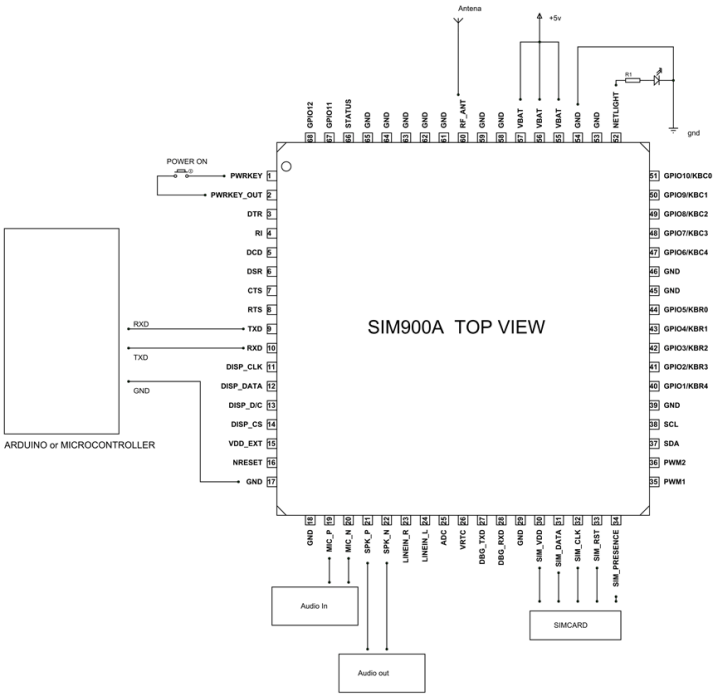


Figure 3.11: SIM 900 Connection

1. **Remote Monitoring and Control:** The module can be used in applications where remote monitoring and control are required, such as home automation, industrial automation, and security systems.

2. **IoT Devices:** It can be integrated into IoT devices to enable communication and data transmission over GSM networks.

3. **Vehicle Tracking:** The SIM900A module can be used in GPS-based vehicle tracking systems for real-time tracking and communication.

4. **Weather Stations:** GSM modules are sometimes used in weather stations to send weather data to a central server or a user's mobile device.

5. **Alarm Systems:** The module can be used in alarm systems to send alert messages via SMS in case of intrusion or other emergencies.

SIM900A MICROCONTROLLER:

The SIM900A GSM module is typically equipped with a dedicated microcontroller that handles the communication tasks and the processing of AT

commands. The specific microcontroller used in the SIM900A module is not publicly disclosed by the manufacturer SIMCom, and it may vary between different module versions and revisions. However, I can provide you with some general characteristics of the microcontroller typically found in GSM modules like SIM900A:

1. **Embedded Microcontroller:** The microcontroller used in the SIM900A module is an embedded system designed to operate in resource-constrained environments. It is optimized for low-power consumption and efficient execution of communication-related tasks.

2. **AT Command Interpreter:** The microcontroller firmware is programmed to include an AT command interpreter. The AT commands are standard commands used for controlling and configuring GSM modules. These commands are sent to the module via a serial interface (usually UART), and the microcontroller interprets and executes them accordingly.

3. **Communication Protocols:** The microcontroller handles the communication protocols required to connect to mobile networks. It manages tasks like network registration, data transmission, SMS handling, and call management.

4. **SIM Card Interface:** The microcontroller is responsible for interfacing with the SIM card inserted in the SIM900A module. It communicates with the SIM card to perform tasks like reading SMS messages, accessing phonebook contacts, and obtaining subscriber information.

5. **UART Interface:** The microcontroller has built-in Universal Asynchronous Receiver/Transmitter (UART) hardware that facilitates serial communication with the host system or other peripherals.

6. **GPIOs and Peripheral Interfaces:** The microcontroller may include General Purpose Input/Output (GPIO) pins and other peripheral interfaces to enable external device connections and interactions.

7. **Low-Level Hardware Control:** The microcontroller firmware manages low-level hardware control, such as power management, voltage regulation, and communication with other components on the module.

8. **Real-Time Clock (RTC):** Some GSM modules have an embedded RTC to keep track of time, which is essential for time-stamping events like SMS reception and call logs.

3.5.2 SIM 900A ARCHITECTURE

The SIM900A module is a widely used GSM/GPRS (2G) module developed by SIMCom. It is designed for various embedded applications that require wireless communication capabilities such as sending and receiving SMS messages, making phone calls, and connecting to the internet over GPRS.

Here's an overview of the SIM900A module's architecture:

1. **Microcontroller:** The SIM900A module is equipped with an embedded commands. The microcontroller firmware is stored in its internal memory.
2. **GSM/GPRS Modem:** The heart of the module is the GSM/GPRS modem, which allows it to connect to mobile networks. It supports quad-band GSM frequencies (850/900/1800/1900 MHz), enabling it to work on different cellular networks worldwide.
3. **SIM Card Holder:** The module features a SIM card holder where you insert the GSM SIM card with an active plan to enable communication and data services.
4. **Antenna Connector:** There's an antenna connector on the module to attach an external antenna. The antenna helps improve signal reception and transmission.
5. **Serial Communication Interface:** The primary method to communicate with the SIM900A module is through a serial interface (usually UART). You can send AT commands to control and configure the module's functionality.
6. **Power Supply:** The module requires a power supply typically in the range of 4V to 5V, although some variants might have different voltage requirements.
7. **I/O Pins:** The module provides several General Purpose Input/Output (GPIO) pins that allow you to interface it with external devices or sensors.

FEATURES	DETAILS
Power Input	4V to 5V
Operating Frequency	EGSM900 and DCS1800
Transmitting Power Range	2W for EGSM900 and 1W for DCS1800
Data Transfer Link	Download: 86kbps, Upload:4 8kbps
SMS	MT, MO, CB, Text and PDU mode
Antenna Support	Available
Audio Input/output	Available
Serial Port	I^2C and UART
Serial Debug Port	Available

SIM900A GSM Module Main Features

PIN CONFIGURATION OF SIM900A

The Module SIM900A looks like a single chip but it has a bunch of features that can help to build almost many commercial applications. Although, there are a total of 68 pins on SIM900A and using these pins helps to build the applications. But we will need few pins if you we use a module for interfacing with Arduino.

The SIM900A GSM module's pin configuration is as follows:

1. **VBAT:** Power supply voltage input (typically 4V to 5V).
2. **GND:** Ground (0V reference).
3. **VDD_EXT:** External voltage supply input (typically 4V to 5V).
4. **RST:** Reset pin. Pulled low to reset the module.
5. **VRTC:** Real-time clock backup voltage input (typically 0V).
6. **NETLIGHT:** Network status indicator output.
7. **STATUS:** Module status indicator output.
8. **RI:** Ring indicator input for incoming call indication.
9. **DTR:** Data Terminal Ready. Connect to logic high (e.g., 3V or 5V) for normal operation.
10. **DCD:** Data Carrier Detect input for detecting carrier signals.

11. **DSR:** Data Set Ready output for indicating the module's readiness.
12. **RTS:** Request to Send output for hardware flow control.
13. **CTS:** Clear to Send input for hardware flow control.
14. **TXD:** Serial transmit data (connect to the RXD of the microcontroller or other device).
15. **RXD:** Serial receive data (connect to the TXD of the microcontroller or other device).
16. **GND:** Ground (0V reference).
17. **MIC-:** Negative terminal of the microphone (for voice calls).
18. **MIC+**: Positive terminal of the microphone (for voice calls).
19. **SPK-:** Negative terminal of the speaker (for voice calls).
20. **SPK+:** Positive terminal of the speaker (for voice calls).
21. **ANT:** GSM antenna connector (connect to the GSM antenna).

We will list details of pinout diagram in next section.

1. **Status Pin:** The module has two status pins which help to indicate two different kinds of status. The first one is the working status of the module and the second for communication status. Net status means either the module is connecting to the network or other network functions, etc. Both these pins can't operate LED directly. They always act with a combination of a transistor.
    - STATUS – Pin52
    - NIGHTLIGHT – Pin66
2. **SIM900A Display Interface Pins:** The device offers a 4pin display interface with itself. The display isn't necessary, it is only in case of requirement. The use of interface helps to get the visualization with the module and make it an application. All display pins are:
    - DISP_DATA – Pin12 – For Display Data
    - DISP_CLK – Pin11 – For Clock Input

- DISP_CS – Pin14 – To enable the display
- DISP_D/C – Pin13 – To select between data and command

3. **I2C Pins:** SIM900A has multiple kinds of communication and I2C is one of them due to is popularity. The module has a single I2C protocol pin, which helps to build the application with any module with that communication.

   - SCL – Pin38
   - SDA – Pin37
   - SDA for data and SCL for clock pulse.

4. **SIM900A GSM Module Keypad interface Pins:** The two-pin keypad is interfaceable with the module. The module will take the keypad data as a 2D matrix value from the KCB pins for each value. The keypad interface pins in the module are:

   - KBR0 KBR4 (ROWS) – Pin40 Pin44
   - KBC0 KBC4 (COLUMN) – Pin47 Pin51

5. **Serial Port:** The UART serial interface uses the two pins for proper data communication, which are RX and TX. Both pins have no independence on any other pins or modules. In SIM900A these pins are available but it also has some other pins for status/indication of data. By combining these pins, the serial port helps to generate the RS-232 connector too. All the serial pins are:

   - RXD – Pin10 – To receive the data
   - TXD – Pin 9- To send the data
   - RTS – Pin8 – To send the request of data transmission
   - CTS – Pin7 – To clear the send request
   - RI – Pin4 – Ring indicator
   - DSR – Pin6 – To indicate that data set ready
   - DCD – Pin5 – To indicate data carry detect
   - DTR – Pin3 – To indicate data terminal ready

6. **Debug Interface:** Debugging helps the developers to debug the module and update its firmware. In this module, there are sperate serial interface pins for debugging. Both pins are:

- DBG_TXD – Pin27 – For Data Transmission
- DBG_RXD – Pin28 – For Data receiving

7. **SIM Interface:** As we know that module SIM900A is a GPRS/GSM module. The module is dependent on some devices for some of its features. The most important one is the SIM. The SIM needs to connect with the module for GPRS/GSM functions to fully operate. All the sim interface of the module is:

    - SIM_VDD – Pin30 – Power Supply of the SIM
    - SIM_DATA – Pin31 – For data output
    - SIM_CLK – Pin32 – For clock pulse
    - SIM_PRESENCE – Pin34 – To detect the SIM

8. **SIM900A Analog to Digital converter Pins:** The module has only a single pin to detect and convert the analog signal to digital for SIM900A. The voltage range on the ADC pin is from 0 to 3 only.

    - ADC – Pin25

9. **PWM Pins:** The PWM is mostly in microcontrollers for industrial applications but due to IoT, the module offers two PWM pins which helps to make the IoT and PWM based device without using any third interface.

    - PWM1 – Pin35
    - PWM2 – Pin36

10. **Audio Interface:** The audio interface will help to connect the mic and speaker with SIM900A. The connection of Line, Audio and Speaker will help to make the calls through the modules.

    - MIC_P – Pin19
    - MIC_N – Pin20
    - SPK_P – Pin21
    - SPK_N – Pin22
    - LINEIN_R – Pin23
    - LINE_L – Pin24

11. **Control Pin:** There is power on pins on the device, which helps to turn it on using external signals. There is two power on pins. The first one is PWRKEY which requires a LOW signal to power on/off the system. To do that, the pins require an input signal for a little bit long time. The second pin is PWRKEY_OUT, which gets short with the PWRKEY pin and turn on/off the device.

    - PWRKEY – Pin1
    - PWRKEY_OUT – Pin2

12. **Reset pins:** The device has an external LOW input signal reset pin to reset the device with the use of an external signal.

    - NRESET – Pin16

13. **SIM900A GSM Module RF Antenna:** To extend the range of the SIM900A the antenna pin needs to connect with an external wire. The official antenna is also available for the module.

    - RF_ANT – Pin60

14. **Power Pins:** The module SIM900A has multiple types of power pin. Some works as input and some as output. The most important one to understand is VRTC, which acts as a backup for the internal RTC of the device. All power and ground pins of the module are:

    - VBAT(Input) – Pin55, Pin56, Pin57
    - VRTC (Input/Output) – Pin26
    - VDD_EXT(OUTPUT) – Pin15
    - GND – Pin17, Pin18, Pin29, Pin39, Pin45, Pin46, Pin53, Pin54, Pin58, Pin59, Pin61, Pin62, Pin63, Pin64, Pin6

3.5.3 GSM Interfacing

GSM interfacing refers to the process of integrating a GSM module with a microcontroller or another electronic device to enable communication over the Global System for Mobile Communications (GSM) network. This allows the device to send and receive data, make voice calls, and send SMS messages over the cellular network.

Here are the general steps involved in GSM interfacing:

1. Selecting the GSM Module: Choose a suitable GSM module that meets your project requirements. Consider factors such as frequency bands supported, communication interfaces (UART, USB, etc.), power requirements, and available features like voice calling, SMS, and GPRS.

2. Hardware Connections: Connect the GSM module to the microcontroller or the host device. The specific connections depend on the module's pin configuration and the microcontroller's communication interfaces (usually UART or USART). Ensure proper power supply connections, connect the SIM card to the GSM module, and attach the GSM antenna.

3. Power Supply: Provide the appropriate power supply to the GSM module. Most GSM modules require a stable power supply within a specified voltage range.

4. SIM Card: Insert a valid GSM SIM card into the SIM card holder of the GSM module. The SIM card is essential for network registration and authentication.

5. AT Commands: GSM modules are controlled using AT (Attention) commands. These are simple text-based commands sent over the serial communication interface (e.g., UART) to the GSM module. By sending AT commands, you can instruct the module to perform various tasks, such as making a call, sending an SMS, or establishing a GPRS data connection.

6. Initialization: Before using the GSM module, it needs to be initialized. This typically involves sending specific AT commands to set the module's configuration and establish communication with the GSM network.

7. Coding: Write the necessary code on the microcontroller or host device to handle the communication with the GSM module. This includes sending appropriate AT commands, parsing responses from the GSM module, and implementing the desired functionalities like making calls, sending SMS messages, or handling incoming messages/calls.

8. Testing and Integration: Once the code is ready, test the GSM interfacing thoroughly. Verify that the module can make calls, send and receive SMS messages, and connect to the internet (if GPRS is supported). Integrate the GSM functionality into your overall project as required.

9. Error Handling: Implement error handling and recovery mechanisms in your code to handle situations like network disconnections, failed communications, or errors in the AT command responses.

10. Security: If your project involves sensitive information or requires secure communication, consider using encryption protocols for data transmission over the GSM network.

Operation of SIM900A

The SIM900A is a GSM module that enables communication over the Global System for Mobile Communications (GSM) network. It allows devices to send and receive data, make voice calls, and send SMS messages. Below is a general outline of the operation of the SIM900A GSM module:

Fig. 14 Connection of Arduino and SIM900A

1. Powering On: When the module is powered on, it initializes and performs self-tests to ensure proper functionality. During this phase, the module may also register with the GSM network to establish a connection.

2. AT Command Mode: The SIM900A is controlled using AT (Attention) commands. It typically starts in AT command mode, waiting to receive AT commands from an external device, such as a microcontroller or a computer.

3. Sending AT Commands: An external device can send AT commands to the SIM900A module over a serial communication interface (e.g., UART). These commands are simple text-based instructions that instruct the module to perform specific tasks.

4. Making Voice Calls: To make a voice call, the external device sends the appropriate AT command containing the phone number to dial. The SIM900A module then initiates a call to the specified number, and if successful, establishes a voice connection.

5. Receiving Voice Calls: When a voice call is received by the module (the module's phone number should be known and accessible to the caller), it can notify the external device about the incoming call using a ring indicator or through the serial interface. The external device can then decide to answer or reject the call by sending the corresponding AT commands.

6. Sending SMS Messages: To send an SMS message, the external device sends an AT command containing the recipient's phone number and the message content. The SIM900A module processes the command and sends the SMS to the specified number.

7. Receiving SMS Messages: When an SMS message is received by the module, it can notify the external device through the serial interface or by using a specific indicator pin. The external device can then issue AT commands to read and process the received SMS messages.

8. Internet Connectivity: The SIM900A module supports GPRS (General Packet Radio Service) for internet connectivity. It can establish a GPRS data connection to enable data transmission over the internet. This allows the module to send and receive data, such as sensor readings or other information, over the internet.

9. Error Handling: The external device should implement error handling and response parsing to handle possible issues and errors that may occur during communication with the SIM900A module. This ensures the system can recover from errors and continue functioning properly.

10. Powering Off: When the module is no longer needed or needs to be powered down, the external device can send an appropriate AT command to shut down the SIM900A module safely.

How to send message in mobile phone using SIM900A GSM Module:

To send an SMS message using the SIM900A GSM Module, we need to use AT commands via a microcontroller or a terminal program (such as Arduino, Raspberry Pi, or a PC) to communicate with the module. Here's a step-by-step guide on how to send an SMS using the SIM900A GSM Module:

1. **Hardware Setup:**

   - Connect the SIM900A module to your microcontroller or computer via a serial communication interface (usually UART). Ensure proper power supply connections, and connect the GSM antenna.
   - Insert a valid GSM SIM card into the SIM card holder on the SIM900A module.

2. **Power Up the SIM900A Module:**

- Power on the SIM900A module, and wait for it to complete its initialization process. The module will typically indicate its status through the status indicator pin.

3. **Enter AT Command Mode:**

   - Send "AT" (Attention) command to the SIM900A module. It should respond with "OK," indicating that the module is ready to accept AT commands.

4. **Set SMS Text Mode:**

   - By default, the module may be in PDU (Protocol Data Unit) mode for SMS messages. To set it to Text mode, send the following command: objectivecCopy code AT+CMGF=1
   - The module should respond with "OK" to confirm that it has switched to Text mode.

5. Specify Recipient and Message:

   - Now, you can specify the recipient's phone number and the content of the SMS message.
   - Use the following command to set the recipient's phone number (replace "XXXXXXXXXX" with the recipient's phone number): Objective Copy code AT+CMGS="XXXXXXXXXX"
   - After sending this command, the module will respond with a ">" prompt, indicating that it is waiting for you to enter the message text.

6. **Enter the Message Text:**

   - Type the SMS message text that you want to send. End the message with the ASCII control character for "Ctrl+Z" (hex code 1A) to indicate the end of the message.

7. **Send the SMS:**

   - Press "Ctrl+Z" (hex code 1A) to send the SMS message. The module will process the message and respond with a message reference number and "OK."

8. **Verify Message Sent:**

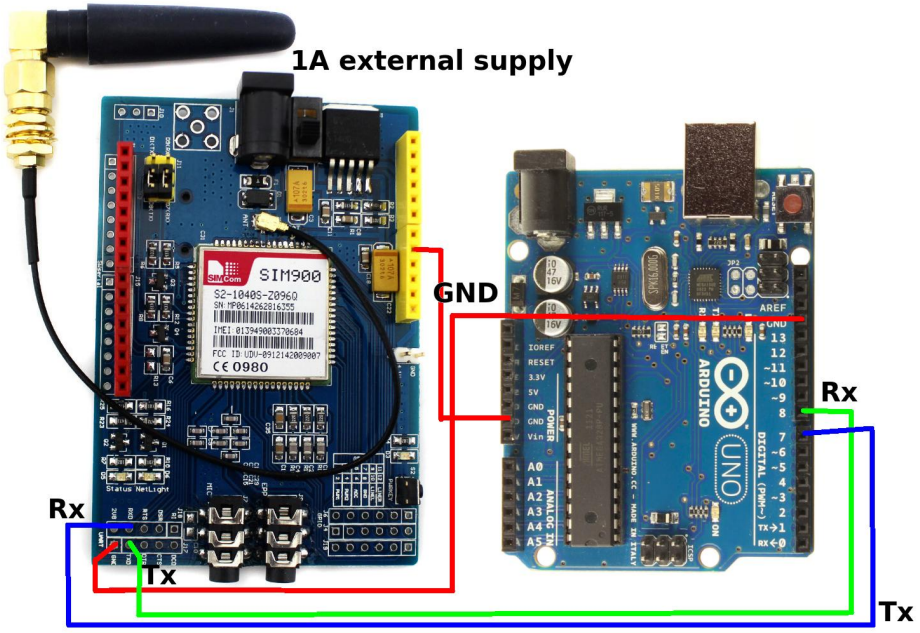


Figure 3.12: SIM900 Module, interfaced with the Arduino Uno

- If the message was sent successfully, the GSM network will handle its delivery to the recipient's phone. The SIM900A module will typically respond with a "+CMGS: [message reference number]" notification.

9. **Handle Errors:**

   - Implement error handling in your code to handle situations like network issues or errors in the AT command responses. The module may return specific error codes for certain failures. Remember to wait for the appropriate responses from the module before proceeding to the next command to ensure reliable communication. Additionally, consider implementing proper delays between commands to allow the module to process each step correctly.

3.6 External Power supply

External power supply for the project is essential to ensure reliable and continuous operation of the Arduino Uno and other components. While the Arduino Uno can be powered via USB, using an external power source is

advantageous in scenarios where the project needs to operate independently from a computer. An external power supply, such as a battery pack or a regulated power adapter, provides stable and consistent power to the project, making it suitable for long-term deployments in remote or mobile setups. Moreover, with gas monitoring projects like this, having an external power supply enhances safety by reducing the risk of power interruptions and ensuring the gas detection device functions seamlessly in real-world environments.

3.6.1 Why do we need it?

Having an external power supply is crucial for the successful operation of the gas detection project using the Arduino Uno and gas sensors. There are several compelling reasons why an external power supply is needed:

1. **Independence from USB Connection:** While the Arduino Uno can be powered through a USB connection from a computer, relying solely on USB power limits the project's mobility and independence. With an external power supply, such as a battery pack or a regulated power adapter, the project can function autonomously, free from the constraints of being tethered to a computer. This is particularly important for remote monitoring applications or projects that need to operate in areas without easy access to a computer or power outlet.

2. **Continuous and Stable Power:** External power sources, especially regulated power adapters or high-capacity battery packs, provide a stable and consistent power output to the Arduino Uno and other components. This ensures reliable and continuous operation of the gas detection device, without the risk of sudden power fluctuations that could disrupt data collection or compromise the accuracy of gas measurements.

3. **Safety Considerations:** In gas monitoring projects, safety is of paramount importance. An external power supply adds an extra layer of safety by reducing the risk of potential power interruptions that could lead to data loss or malfunctioning of the gas sensors. A consistent power supply helps maintain the continuous monitoring and alerting functions, ensuring timely detection and response to harmful gas levels.

4. **Extended Deployment Time:** An external power supply with higher capacity, such as a larger battery pack, allows the gas detection device

to operate for extended periods without the need for frequent battery changes or recharging. This is particularly beneficial in scenarios where continuous, long-term monitoring is required, such as in environmental studies or industrial monitoring applications.

5. **Flexibility and Versatility:** By using an external power supply, the gas detection project gains flexibility in its deployment options. It can be installed in various locations, both indoors and outdoors, without being limited by the availability of USB connections or power outlets. This versatility enhances the project's adaptability to different environments and use cases. An external power supply is essential for the gas detection project's success, offering independence, continuous and stable power, safety, extended deployment time, and flexibility. It ensures that the gas detection device operates reliably and effectively, making it a valuable tool for environmental monitoring, safety applications, and various projects where mobility, long-term operation, and independence are critical factors.

Chapter 4

Circuit Design

4.1 Circuit Design

The circuit described above is a power supply system designed to provide a stable and continuous power source for the gas detection system. Let's break down the components and their functions in detail:

1. **Power Source:** The power source for this circuit consists of two 18650 Li-ion batteries connected in parallel. Connecting the batteries in parallel increases the overall capacity of the power supply, allowing for extended operating time between recharges. Li-ion batteries are commonly used in portable electronic devices due to their high energy density and rechargeable nature.

2. **Battery Charging Module (TP-4056):** The TP-4056 charging module serves the purpose of charging the Li-ion batteries. It is a compact and efficient charging module designed to provide safe and controlled charging to the batteries. The module includes overcharging and over-discharging protection circuits, ensuring the batteries are charged within safe limits and prolonging their lifespan.

3. **Type-C and Micro USB Ports:** The addition of both Type-C and Micro USB ports provides flexibility in charging the batteries. Users can choose to charge the batteries using either a Type-C or Micro USB cable, depending on their preference and availability of charging cables. These ports act as input interfaces for the charging module, allowing easy connection to external power sources for recharging the batteries.

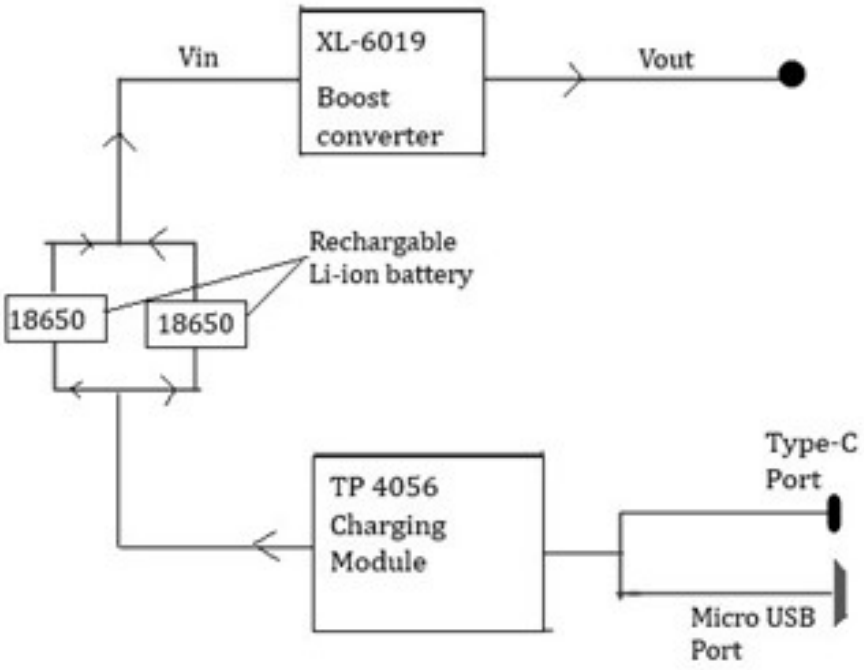


Figure 4.1: Block Diagram of External Power Suppy Circuit

4. **Switch:** The switch added in series to the circuit serves as an ON/OFF switch, enabling users to control the power supply to the gas detection project. When the switch is ON, the power supply is active, and the batteries provide power to the rest of the circuit. When the switch is OFF, the power supply is disconnected, conserving battery power and preventing any unwanted power consumption.

5. **Boost Converter (XL6019):** The XL6019 is a step-up (boost) converter used to increase the voltage output from the batteries to a stable and suitable level for powering the Arduino Uno, gas sensors, and other components. The boost converter ensures that the voltage output remains constant even as the batteries discharge, preventing fluctuations in power that could affect the accuracy of gas measurements or the proper functioning of the Arduino Uno.

The power supply circuit with the TP-4056 charging module, 18650 Li-ion batteries, Type-C and Micro USB ports, switch, and XL6019 boost converter provides a robust and flexible power source for the gas detection project. It ensures continuous and stable power to the Arduino Uno and gas sensors, allowing for reliable gas monitoring and real-time alerts. The addition of

overcharging and over-discharging protection through the TP-4056 module enhances safety, while the use of Li-ion batteries with a parallel configuration increases the overall capacity for extended operation. The circuit's simplicity and versatility make it an excellent choice for powering the gas detection device in various environmental monitoring and safety applications.

On measuring we got the total current required = 0.16 to 0.18A

full charge voltage of Li-ion battery = 4.2V

Nominal Voltage of Li-ion Battery = 3.7V

Discharge cut-off voltage of Li-ion Battery = 3V

Maximum Output current allowed by BMS = 3A

Therefore, Power(P) = Voltage(V) * Current(I)

$P = 3.7 \times 3$ (Taking the nominal voltage of battery and Current of BMS)

P = 11.1W

Since, each battery has capacity of 2600mAh, therefore total capacity of the power supply

= 2 * 2600 mAh

= 5,200 mAh

Therefore, total power capacity of the battery = 3.7V * 5.2Ah

= 19.24 Wh

If the avg. power consumption rate of the system = 1.92 Wh

Hence, total operation time of power supply = 19.24/1.92

= 10.02h

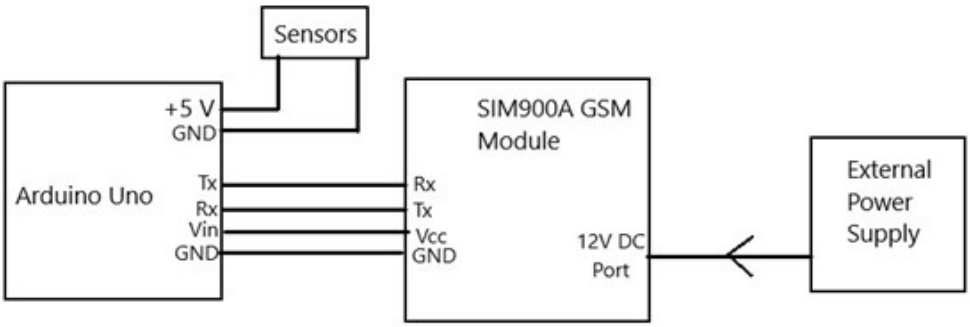


Figure 4.2: Block diagram of System Connection

4.1.1 Circuit Connection

The circuit for the gas detection project using the Arduino Uno, gas sensors (MQ-135 for CO_2, MQ-7 for CO, and AGS02A for SO_2, H_2S, NO_2, NO, O_3), and the SIM900A GSM Module is described on the previous page.

1. **Arduino Uno Connections:** The Arduino Uno acts as the central microcontroller for the project. It is connected to the gas sensors and the GSM Module through its digital and analog I/O pins. A connecting wire from the SIM900A Vcc was connected to the Arduino Uno's power PIN Vin to provide power to the microcontroller.

2. **Gas Sensors (MQ-135 and MQ-7) Connections:** The MQ-135 gas sensor for CO2 and the MQ-7 gas sensor for CO are interfaced with the Arduino Uno. The sensors' analog outputs are connected to the Arduino Uno's analog pins (A0 and A1, respectively). The gas sensors are calibrated during the setup phase to establish the appropriate sensitivity and baseline resistance values for accurate gas measurements.

3. **AGS02A (TVOC Sensor) Connection:** The AGS02A sensor is utilized to detect various harmful gases, including O_2,H_2S, NO_2, NO, and O_3. The sensor communicates with the Arduino Uno through the I2C communication protocol, using the Wire library. The I2C pins (SDA and SCL) of the AGS02A sensor are connected to the corresponding I2C pins of the Arduino Uno (A4 and A5).

4. **DHT11 Temperature and Humidity Sensor:**The DHT11 temperature and humidity sensor is utilized to measure environmental conditions. It is connected to digital pin 2 of the Arduino Uno. The Adafruit Unified Sensor and DHT libraries are employed to obtain temperature and humidity data from the sensor.

5. **XL6019 Boost Converter and Switch:** The XL6019 is a step-up (boost) converter used to increase the voltage output from the parallel-connected Li-ion batteries. The XL6019 is connected to the batteries in series with the charging module. The switch in series with the boost converter allows the user to control the power supply to the gas detection system.

6. **TP-4056 Charging Module:** The TP-4056 charging module is used to charge the two 18650 Li-ion batteries connected in parallel. It incorporates overcharging and over-discharging protection circuits for safe and efficient battery charging. The Type-C and Micro USB ports of the external power supply are connected to the charging module for recharging the batteries.

7. **SIM900A GSM Module:** The SIM900A GSM Module enables the gas detection system to send real-time gas concentration alerts via SMS. The module communicates with the Arduino Uno using serial communication through the RX and TX pins. The GSM module is powered through the Arduino Uno's power supply.

In operation, the gas sensors continuously monitor the environment for CO_2, CO, SO_2, H_2S, NO_2, NO, and O_3 gas levels. The Arduino Uno processes the sensor data and calculates the correction factor based on temperature and humidity readings from the DHT11 sensor. The GSM module is triggered to send SMS alerts when harmful gas concentrations exceed predefined thresholds. The power supply, comprising the Li-ion batteries, TP-4056 charging module, XL6019 boost converter, and switch, ensures continuous and stable power to the gas detection system.

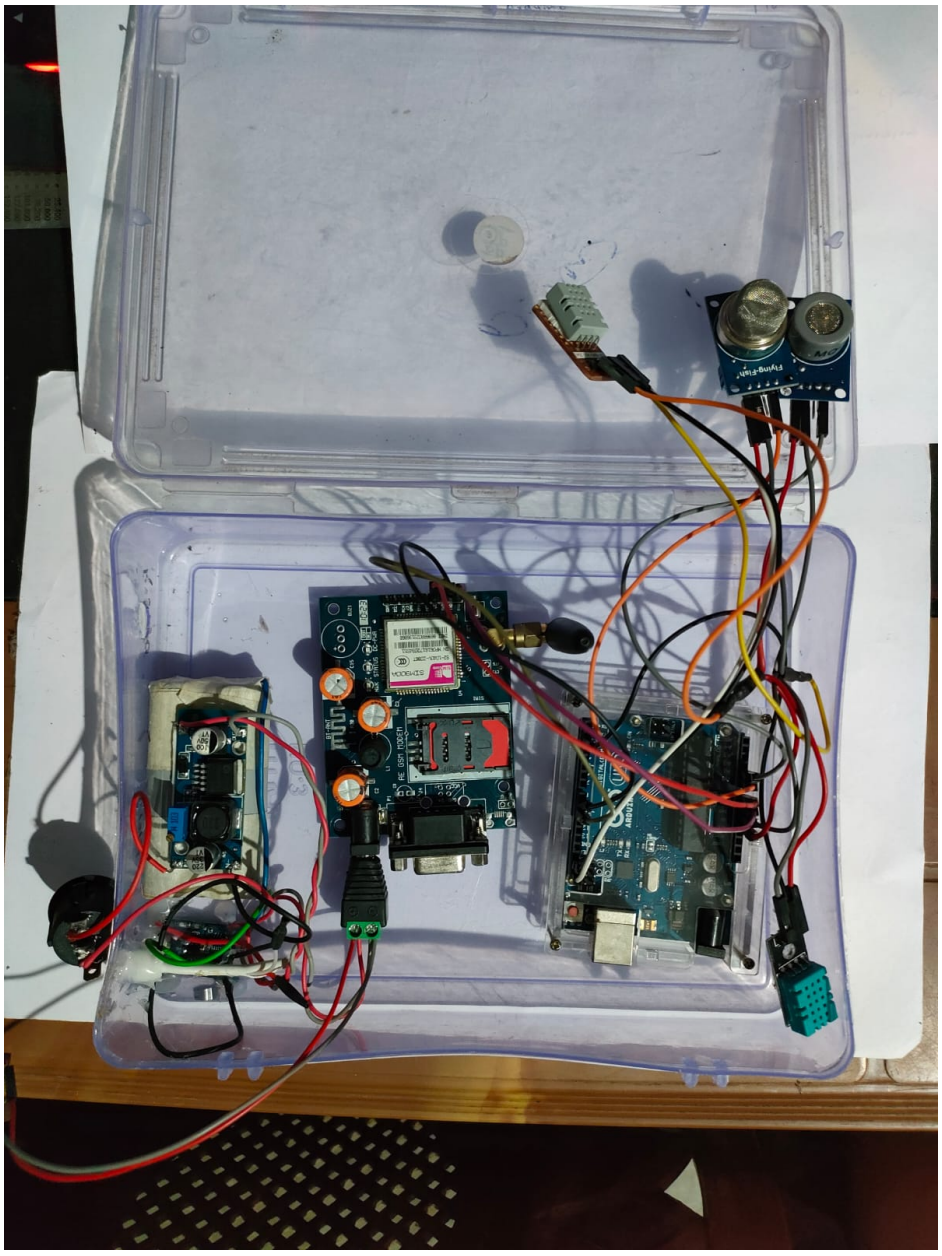


Figure 4.3: The complete Project Circuit showing all the components interconnected. At the left, the 18650 Li-ion batteries with BMS and charging boards, powering up all the components. At the center, the SIM900 GSM module, and at the extreme right, the Arduino Uno are in the box. The sensors are at the top and the bottom.

Chapter 5

Programming

5.1 Designing the Software code

Programming is the process of writing code to instruct a computer or microcontroller to perform specific tasks or functions. In the context of the gas detection project using the Arduino Uno, programming is fundamental to control the gas sensors, process data, and trigger communication with the GSM module for real-time alerts.

5.1.1 The Arduino IDE (Integrated Development Environment)

The Arduino IDE is a user-friendly software platform designed specifically for programming Arduino boards, including the Arduino Uno. It provides a simplified and intuitive interface that makes it accessible to beginners and experienced developers alike. Here are some key features of the Arduino IDE:

1. **Code Editor:** The Arduino IDE includes a code editor where users write their Arduino sketches. It supports syntax highlighting and auto-completion, making it easier to write and debug code.

2. **Library Manager:** Arduino libraries are collections of pre-written code that simplify complex tasks. The IDE comes with a built-in library manager that allows users to easily find, install, and update libraries required for their projects.

3. **Serial Monitor:** The Serial Monitor is a valuable tool for debugging and communication. It allows users to send and receive data between the Arduino Uno and a connected computer via the USB port, aiding in monitoring sensor readings and debugging the code.

4. **Board Manager:** The Board Manager is used to install and manage different board packages, including the Arduino Uno. It enables users to select the correct board and processor settings for their specific Arduino model.

5. **Sketch Upload:** The Arduino IDE provides a simple and straightforward process to upload sketches to the Arduino Uno. With just a click of a button, the compiled code is transferred to the microcontroller through the USB connection.

6. **Examples and Tutorials:** The IDE includes a vast collection of example sketches and tutorials that help users understand Arduino programming concepts and demonstrate how to use various sensors and modules.

The Arduino IDE's user-friendly interface and powerful features make it an ideal platform for us to program Arduino boards effectively. Whether for simple LED blinking projects or complex IoT applications, the Arduino IDE provides the tools necessary to turn ideas into reality, empowering users to create innovative electronic projects and explore the exciting world of embedded systems programming.

Given below is the code written for the Arduino UNO to get the sensor data and send a real-time SMS via SIM900A module to get remote access to the data.

Arduino Code for the project

```
/*
Arduino Code for Gas Sensor
Written By  Swani Chatterjee, Saurabh Biswas & Suryashmi Chakraborty
for M.Sc. Project in Electronics
under supervision of Durjoy Roy
*/


#include <MQUnifiedsensor.h>
#include <SoftwareSerial.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#include "AGS02MA.h"              // These are include directives to include necessary libraries for working with various
```

```
15  //sensors and devices.

17  #define placa "Arduino UNO"
    #define Voltage_Resolution 5
19  #define pinMQ135 A0                                    //
     configuring Arduino analog Pin A0 for MQ-135
    #define RL_MQ135 22                                   //
     changed the value of RL from 1(default) to 22(changed)
21  #define pinMQ7 A1
     // configuring Arduino analog Pin A1 for MQ-7
    #define RL_MQ7 1               // these are preprocessor
     macro definitions used to assign values to specific names
     (placa,
23  // Voltage_Resolution, pinMQ135, etc.) for easier code
     readability and maintenance.

25  #define MQ135_DEFAULTPPM 421.72               //
     preprocessor macro definitions with specific values
    //for MQ135 sensor calibration and calculation.
27  #define MQ135_DEFAULTRO 68550
    #define MQ135_SCALINGFACTOR 116.6020682
29  #define MQ135_EXPONENT -2.769034857

31  #define DHTPIN 2
        // configuring Arduino Digital pin 2 for DHT11 sensor
    #define DHTTYPE DHT11                                  //
     defining the DHT sensor type
33
    #define PWMPin 5
35
    SoftwareSerial mySerial(9, 10);
37
    AGS02MA AGS(26);
39  DHT_Unified dht(DHTPIN, DHTTYPE);   // Object instantiation
      for AGS02MA and DHT_Unified using specific pins
    // and types.
41  unsigned long delayMS;              // Declaration of an
     unsigned long variable to hold the delay time in
     milliseconds.

43  MQUnifiedsensor MQ135(placa, Voltage_Resolution, 10,
     pinMQ135, "MQ-135");
    MQUnifiedsensor MQ7(placa, Voltage_Resolution, 10, pinMQ7,
     "MQ-7");
45  // Object instantiation for MQ7 & MQ-135 sensors with
     specific parameters.

47  float getCorrectionFactor(float t, float h);   // Function
     prototype for the getCorrectionFactor function, which
```

```
// calculates a correction factor based on temperature and humidity

void setup() {
        // Initialization code executed once at the beginning
  Serial.begin(9600);
  // Starts serial communication with a baud rate of 9600.
  dht.begin();
        // Initializes the DHT sensor.
  sensor_t sensor;

  MQ135.setRegressionMethod(1);             // Setting up MQ135 sensor calibration parameters
  MQ135.setA(110.47);
  MQ135.setB(-2.862);
  MQ135.init();
  MQ135.setRL(RL_MQ135);

  Serial.print("Calibrating MQ-135, please wait...");
  float calc135R0 = 0;
  for (int i = 1; i <= 10; i++) {
    MQ135.update();
    calc135R0 += MQ135.calibrate(3.6);
  }
  MQ135.setR0(calc135R0 / 10);
  Serial.println(" done!");
     // print "done" once the calibration is successful.

  MQ7.setRegressionMethod(1);                          // Setting up MQ7 sensor calibration parameters
  MQ7.setA(99.042);
  MQ7.setB(-1.518);
  MQ7.init();
  MQ7.setRL(RL_MQ7);

  Serial.print("Calibrating MQ-7, please wait...");
  float calc7R0 = 0;
  for (int i = 1; i <= 10; i++) {
    MQ7.update();
    calc7R0 += MQ7.calibrate(27.5);
  }
  MQ7.setR0(calc7R0 / 10);
  Serial.println(" done!");
     // print "done" once the calibration is successful.

  Wire.begin();
             // Initializes the I2C communication (needed for AGS02MA).
```

```
    AGS.begin();
87  AGS.setPPBMode();
      // Initializes AGS02MA sensor and sets it to PPB (parts
    per billion) mode.

89  Serial.println("Initialization complete.");

91  delayMS = 500;
          // Assigns a delay time of 500 milliseconds to
   the delayMS variable.
}
93
void loop() {
                // Main code that runs in a loop after setup
   () is executed.
95  delay(delayMS);
          // Pauses the program for 'delayMS' milliseconds.

97  sensors_event_t event;
    dht.temperature().getEvent(&event);           // Reading
   temperature and humidity from the DHT.
99
    float cFactor = 0;
101 if (!isnan(event.temperature) && !isnan(event.
   relative_humidity))
    cFactor = getCorrectionFactor(event.temperature, event.
   relative_humidity);
103
    MQ135.update();
105 float CO2 = MQ135.readSensor(false, cFactor);
         // Reading CO2 concentration from the MQ135 sensor
   .

107 MQ7.update();
    float CO = MQ7.readSensor();
                        // Reading CO concentration from
   the MQ7 sensor.
109 float SO2 = (AGS.readPPB())*0.00355;
    float H2S = (AGS.readPPB())*0.00666;
111 float NO2 = (AGS.readPPB())*0.00494;
    float NO = (AGS.readPPB())*0.00757;
113 float O3 = (AGS.readPPB())*0.00473;
    float tCO2 = CO2 + MQ135_DEFAULTPPM;
115
    Serial.print("CO2(ppm):  ");
        // Printing the sensor readings to the serial
   monitor
117 Serial.print(tCO2);
    Serial.print("       ");
```

```
119
    Serial.print("CO(ppm):  ");
121 Serial.print(CO);
    Serial.print("       ");
123
    Serial.print("SO2(ppm):  ");
125 Serial.print(SO2);
    Serial.print("       ");
127
    Serial.print("H2S(ppb):  ");
129 Serial.print(H2S);
    Serial.print("       ");
131
    Serial.print("NO2(ppb):  ");
133 Serial.print(NO2);
    Serial.print("       ");
135
    Serial.print("NO(ppb):  ");
137 Serial.print(NO);
    Serial.print("       ");
139
    Serial.print("O3(ppb):  ");
141 Serial.print(O3);
    Serial.print("       ");
143
    Serial.print("Temp.(C): ");
145 Serial.print(event.temperature);
    Serial.print("       ");
147
    Serial.print("Humi.(%): ");
149 Serial.print(event.relative_humidity);
    Serial.println();
151
    String Data;
153 Data = "CO2(ppm):  " + String(tCO2) + "       " +
    "CO(ppm):  " + String(CO) + "       " +
155 "SO2(ppm):  " + String(SO2) + "       " +
    "H2S(ppb):  " + String(H2S) + "       " +
157 "NO2(ppb):  " + String(NO2) + "       " +
    "NO(ppb):  " + String(NO) + "       " +
159 "O3(ppb):  " + String(O3) + "       " +
    "Temp.(C):  " + String(event.temperature) + "       " +
161 "Humi.(%): " + String(event.relative_humidity);

163 if (Serial.available()>0)
    switch(Serial.read())
165 {
      case 's':
```

```
167       mySerial.println("AT+CMGF=1");
                                                    //Sets the GSM
    Module in Text Mode
          delay(1000);

               // Delay of 1 second
169       mySerial.println("AT+CMGS=\"+918372015720\"\r");
                  // mobile number to which the data will be send
          delay(1000);
171       mySerial.println(Data);
                                                              // The
    Data to send
          delay(100);
173       mySerial.println((char)26);
                    // ASCII code of CTRL+Z for saying the end of
    SMS to  the module
          delay(1000);
175       break;

177       case 'r':
          mySerial.println("AT+CNMI=2,2,0,0,0");              //
    AT Command to receive a live SMS
179       delay(1000);
          break;
181      }
         if (mySerial.available()>0)
183      Serial.write(mySerial.read());
       }
185    float getCorrectionFactor(float t, float h) {
                 // Function to calculate a correction factor
         based on
         //temperature and humidity.
187      const float CORA = 0.00035;
         const float CORB = 0.02718;
189      const float CORC = 1.39538;
         const float CORD = 0.0018;
191      return CORA * t * t - CORB * t + CORC - (h - 33.0) * CORD
       ;                          // Returns the calculated
         correction factor.
       }
193
```

Listing 5.1: gassensor.ino

5.1.2 Execution of the Program

1. **Library Inclusion:** The sketch begins with including necessary libraries for working with sensors and devices. These libraries provide

predefined functions to interact with the sensors easily.

2. **Macro Definitions:** The sketch uses preprocessor macro definitions to assign specific values to meaningful names. This enhances code readability and makes it easier to maintain.

3. **Global Variable Declarations:** Global variables are declared at the beginning of the code, including sensor objects and variables to store delay time and correction factors.

4. **Setup Function**: The '*setup*()' function is executed once at the beginning of the program. It initializes the serial communication, sensors, and calibration parameters for the MQ135 and MQ7 sensors.

5. **Calibration of Sensors:** The calibration of both MQ135 and MQ7 sensors is performed to obtain accurate gas readings. Calibration involves reading the sensor values in a known environment to determine sensor-specific parameters.

6. **Sensor Readings in Loop:** The '*loop*()' function runs continuously after the '*setup*()' function. In the loop, the program does the following:

   - Reads temperature and humidity from the DHT sensor using the "$\backslash dht.temperature().getEvent(\&event)$" function.
   - Calculates a correction factor for the gas sensors based on temperature and humidity.
   - Updates the MQ135 and MQ7 sensors to obtain gas readings.
   - Calculates gas concentrations ((CO_2), CO, (SO_2), (H_2S), (NO_2), NO, (O_3)) using calibration data and the "$readSensor()$" function.
   - Prints the sensor readings to the Serial Monitor for real-time monitoring.

7. **Data Sending via GSM Module:** The program listens to the Serial Monitor for user input. If the user sends 's', it initiates sending an SMS with the gas concentration data. It sets up the GSM module in text mode, specifies the recipient number, and sends the gas concentration data as an SMS.

8. **Data Receiving via GSM Module:** If the user sends 'r', the program sets up the GSM module to receive live SMS messages.

9. *getCorrectionFactor* **Function:** This function calculates a correction factor for the gas sensors based on temperature and humidity values. The correction factor is used to compensate for variations in gas readings caused by environmental conditions.

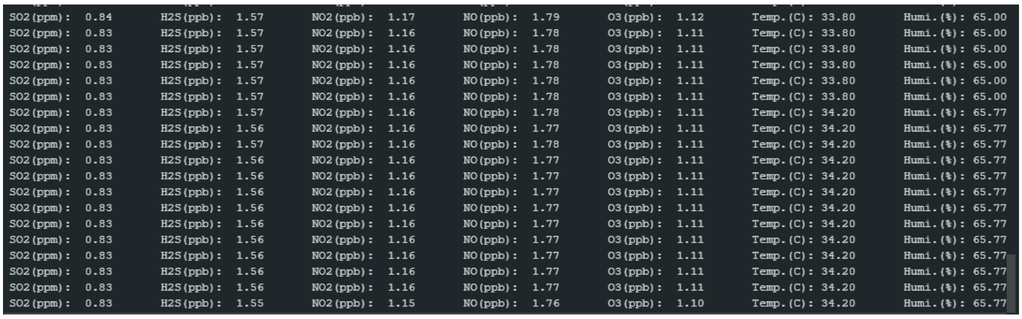
In summary, the program reads gas concentrations ((CO_2), CO, (SO_2), (H_2S), (NO_2), NO, (O_3)) from the MQ135 and MQ7 sensors along with temperature and humidity from the DHT sensor. It calibrates the sensors for accuracy, calculates the correction factor, and continuously prints the gas concentration data to the Serial Monitor.

Chapter 6

Results and Output

6.1 Results obtained

The output of the system was recorded once all the sensors were working properly i.e. when all the sensors were giving more or less accurate reading though in electronics nothing has a precision or accuracy of 100%, we just try to be as accurate and precise as possible.



```
SO2(ppm):  0.84   H2S(ppb):  1.57   NO2(ppb):  1.17   NO(ppb):  1.79   O3(ppb):  1.12   Temp.(C): 33.80   Humi.(%): 65.00
SO2(ppm):  0.83   H2S(ppb):  1.57   NO2(ppb):  1.16   NO(ppb):  1.78   O3(ppb):  1.11   Temp.(C): 33.80   Humi.(%): 65.00
SO2(ppm):  0.83   H2S(ppb):  1.57   NO2(ppb):  1.16   NO(ppb):  1.78   O3(ppb):  1.11   Temp.(C): 33.80   Humi.(%): 65.00
SO2(ppm):  0.83   H2S(ppb):  1.57   NO2(ppb):  1.16   NO(ppb):  1.78   O3(ppb):  1.11   Temp.(C): 33.80   Humi.(%): 65.00
SO2(ppm):  0.83   H2S(ppb):  1.57   NO2(ppb):  1.16   NO(ppb):  1.78   O3(ppb):  1.11   Temp.(C): 33.80   Humi.(%): 65.00
SO2(ppm):  0.83   H2S(ppb):  1.57   NO2(ppb):  1.16   NO(ppb):  1.78   O3(ppb):  1.11   Temp.(C): 33.80   Humi.(%): 65.00
SO2(ppm):  0.83   H2S(ppb):  1.57   NO2(ppb):  1.16   NO(ppb):  1.78   O3(ppb):  1.11   Temp.(C): 34.20   Humi.(%): 65.77
SO2(ppm):  0.83   H2S(ppb):  1.56   NO2(ppb):  1.16   NO(ppb):  1.77   O3(ppb):  1.11   Temp.(C): 34.20   Humi.(%): 65.77
SO2(ppm):  0.83   H2S(ppb):  1.57   NO2(ppb):  1.16   NO(ppb):  1.78   O3(ppb):  1.11   Temp.(C): 34.20   Humi.(%): 65.77
SO2(ppm):  0.83   H2S(ppb):  1.56   NO2(ppb):  1.16   NO(ppb):  1.77   O3(ppb):  1.11   Temp.(C): 34.20   Humi.(%): 65.77
SO2(ppm):  0.83   H2S(ppb):  1.56   NO2(ppb):  1.16   NO(ppb):  1.77   O3(ppb):  1.11   Temp.(C): 34.20   Humi.(%): 65.77
SO2(ppm):  0.83   H2S(ppb):  1.56   NO2(ppb):  1.16   NO(ppb):  1.77   O3(ppb):  1.11   Temp.(C): 34.20   Humi.(%): 65.77
SO2(ppm):  0.83   H2S(ppb):  1.56   NO2(ppb):  1.16   NO(ppb):  1.77   O3(ppb):  1.11   Temp.(C): 34.20   Humi.(%): 65.77
SO2(ppm):  0.83   H2S(ppb):  1.56   NO2(ppb):  1.16   NO(ppb):  1.77   O3(ppb):  1.11   Temp.(C): 34.20   Humi.(%): 65.77
SO2(ppm):  0.83   H2S(ppb):  1.56   NO2(ppb):  1.16   NO(ppb):  1.77   O3(ppb):  1.11   Temp.(C): 34.20   Humi.(%): 65.77
SO2(ppm):  0.83   H2S(ppb):  1.56   NO2(ppb):  1.16   NO(ppb):  1.77   O3(ppb):  1.11   Temp.(C): 34.20   Humi.(%): 65.77
SO2(ppm):  0.83   H2S(ppb):  1.56   NO2(ppb):  1.16   NO(ppb):  1.77   O3(ppb):  1.11   Temp.(C): 34.20   Humi.(%): 65.77
SO2(ppm):  0.83   H2S(ppb):  1.56   NO2(ppb):  1.16   NO(ppb):  1.77   O3(ppb):  1.11   Temp.(C): 34.20   Humi.(%): 65.77
SO2(ppm):  0.83   H2S(ppb):  1.55   NO2(ppb):  1.15   NO(ppb):  1.76   O3(ppb):  1.10   Temp.(C): 34.20   Humi.(%): 65.77
```

(a) Output with DHT reading

```
CO2(ppm):  423.64   CO(ppm):  0.61   SO2(ppm):  1.01   H2S(ppb):  1.90   NO2(ppb):  1.41   NO(ppb):  2.16   O3(ppb):  1.35
CO2(ppm):  423.64   CO(ppm):  0.62   SO2(ppm):  1.01   H2S(ppb):  1.90   NO2(ppb):  1.41   NO(ppb):  2.16   O3(ppb):  1.35
CO2(ppm):  423.64   CO(ppm):  0.60   SO2(ppm):  1.01   H2S(ppb):  1.90   NO2(ppb):  1.41   NO(ppb):  2.16   O3(ppb):  1.35
CO2(ppm):  423.64   CO(ppm):  0.60   SO2(ppm):  1.01   H2S(ppb):  1.90   NO2(ppb):  1.41   NO(ppb):  2.16   O3(ppb):  1.35
CO2(ppm):  423.60   CO(ppm):  0.60   SO2(ppm):  1.01   H2S(ppb):  1.90   NO2(ppb):  1.41   NO(ppb):  2.16   O3(ppb):  1.35
CO2(ppm):  423.55   CO(ppm):  0.61   SO2(ppm):  1.01   H2S(ppb):  1.89   NO2(ppb):  1.40   NO(ppb):  2.15   O3(ppb):  1.34
CO2(ppm):  423.55   CO(ppm):  0.60   SO2(ppm):  1.01   H2S(ppb):  1.89   NO2(ppb):  1.40   NO(ppb):  2.15   O3(ppb):  1.34
CO2(ppm):  423.55   CO(ppm):  0.60   SO2(ppm):  1.01   H2S(ppb):  1.89   NO2(ppb):  1.40   NO(ppb):  2.15   O3(ppb):  1.34
CO2(ppm):  423.64   CO(ppm):  0.61   SO2(ppm):  1.01   H2S(ppb):  1.89   NO2(ppb):  1.40   NO(ppb):  2.15   O3(ppb):  1.34
CO2(ppm):  423.73   CO(ppm):  0.60   SO2(ppm):  1.00   H2S(ppb):  1.88   NO2(ppb):  1.40   NO(ppb):  2.14   O3(ppb):  1.34
CO2(ppm):  423.64   CO(ppm):  0.60   SO2(ppm):  1.00   H2S(ppb):  1.88   NO2(ppb):  1.40   NO(ppb):  2.14   O3(ppb):  1.34
CO2(ppm):  423.55   CO(ppm):  0.60   SO2(ppm):  1.00   H2S(ppb):  1.88   NO2(ppb):  1.40   NO(ppb):  2.14   O3(ppb):  1.34
CO2(ppm):  423.51   CO(ppm):  0.60   SO2(ppm):  1.00   H2S(ppb):  1.88   NO2(ppb):  1.40   NO(ppb):  2.14   O3(ppb):  1.34
CO2(ppm):  423.51   CO(ppm):  0.60   SO2(ppm):  1.00   H2S(ppb):  1.88   NO2(ppb):  1.40   NO(ppb):  2.14   O3(ppb):  1.34
CO2(ppm):  423.55   CO(ppm):  0.60   SO2(ppm):  1.00   H2S(ppb):  1.88   NO2(ppb):  1.40   NO(ppb):  2.14   O3(ppb):  1.34
CO2(ppm):  423.55   CO(ppm):  0.60   SO2(ppm):  1.00   H2S(ppb):  1.88   NO2(ppb):  1.40   NO(ppb):  2.14   O3(ppb):  1.34
CO2(ppm):  423.60   CO(ppm):  0.62   SO2(ppm):  1.00   H2S(ppb):  1.88   NO2(ppb):  1.40   NO(ppb):  2.14   O3(ppb):  1.34
CO2(ppm):  423.55   CO(ppm):  0.62   SO2(ppm):  1.00   H2S(ppb):  1.88   NO2(ppb):  1.39   NO(ppb):  2.13   O3(ppb):  1.33
CO2(ppm):  423.55   CO(ppm):  0.60   SO2(ppm):  1.00   H2S(ppb):  1.88   NO2(ppb):  1.39   NO(ppb):  2.13   O3(ppb):  1.33
```

(b) Output without DHT reading

Figure 6.1: Output showing reading of pollutant concentration

Chapter 7

Conclusion

Our Internet of Things (IoT) project focused on developing a gas monitoring system using Arduino Uno and various sensors has proved to be a significant step towards addressing the pressing global concern of air pollution caused by harmful gases. By harnessing the power of IoT technology, we aimed to create an efficient and reliable device capable of detecting and measuring multiple hazardous gases in the environment.

Throughout the project, we successfully designed a gas detection device that incorporates sensors like MQ-7, MQ-135, and AGS02A (TVOC Sensor) to monitor carbon monoxide (CO), carbon dioxide (CO_2), sulfur dioxide (SO_2), hydrogen sulfide (H_2S), nitrogen dioxide (NO_2), nitrogen trioxide(NO_3), and ozone (O3). The gas sensors' precise working principles and sensitivities to specific gases were meticulously studied, allowing us to calibrate and validate the accuracy of the gas concentration measurements.

The core of the system, the Arduino Uno microcontroller, acted as the central hub, interfacing with the sensors and the SIM900A GSM Module for real-time data transmission. The Arduino Uno's versatility and extensive community support made it an ideal choice for seamless integration and efficient data processing.

One of the project's significant achievements lies in recognizing the harmful effects of these gases on not only human health but also the delicate ecosystems of butterflies and other pollinators. As crucial contributors to pollination and biodiversity, butterflies' vulnerability to air pollution underscores the broader ecological impact of gas emissions. By monitoring these gases, we contribute to safeguarding biodiversity and promoting a sustainable environment.

Moreover, the incorporation of an external power supply using TP-4056 1S charging module, 18650 Li-ion batteries, and XL6019 Boost converter ensures continuous device operation and data availability even in remote locations.

The system's functionality was thoroughly tested, demonstrating its efficiency in continuous gas monitoring and real-time data transmission via SMS. This capability empowers individuals and communities to take proactive measures to protect human health and the environment by responding promptly to potential hazardous situations.

In conclusion, our gas monitoring IoT project marks a substantial advancement in addressing air pollution concerns. By harnessing the potential of Arduino Uno, multiple gas sensors, and IoT technology, we have successfully created a cost-effective, reliable, and scalable solution to combat environmental challenges. The project's potential enhancements, such as cloud-based data storage and data visualization interfaces, pave the way for even more comprehensive gas monitoring and analysis. Our efforts contribute to building a greener, healthier, and sustainable future by mitigating air pollution and preserving the delicate balance of ecosystems and the well-being of all living beings.

Chapter 8

Future Enhancements

The current gas monitoring IoT project lays a solid foundation for addressing air pollution concerns, but several future enhancements can be implemented to further improve its capabilities and functionality. These enhancements are aimed at making the system more versatile, user-friendly, and efficient in tackling environmental challenges. Some potential future improvements include:

1. **Advanced Sensor Integration:** Explore the integration of more advanced gas sensors with higher accuracy and sensitivity. Emerging sensor technologies can provide more comprehensive gas detection capabilities, enabling the system to monitor additional harmful gases and pollutants.

2. **Wireless Connectivity:** Upgrade the system to use wireless communication protocols such as Wi-Fi or Bluetooth in addition to or instead of the GSM module. This would enable data transmission to a centralized cloud-based platform, making it easier to access and analyse gas concentration data remotely.

3. **Cloud-Based Data Storage:** Implement cloud-based data storage for long-term data retention and analysis. Storing gas concentration data in the cloud allows for historical trend analysis and the generation of insightful reports to understand air pollution patterns over time.

4. **Data Visualization:** Develop a user-friendly data visualization interface, such as a web-based dashboard or a mobile app. This would enable users to monitor gas concentrations graphically and receive real-time alerts when certain thresholds are exceeded.

5. **Geolocation Tagging:** Integrate GPS capabilities into the system to enable geolocation tagging of gas concentration data. Geotagging would help identify specific pollution hotspots and support targeted interventions in areas with severe air quality issues.

6. **Power Optimization:** Implement power-saving mechanisms to extend the device's battery life when operating on external power or during low gas concentration periods. Efficient power management ensures continuous monitoring without frequent battery replacements.

7. **Sensor Calibration and Auto-Calibration:** Develop an automated sensor calibration mechanism to periodically recalibrate the gas sensors for optimal accuracy. Auto-calibration can account for sensor drift and ensure reliable gas concentration readings over extended periods.

8. **Real-Time Alerts:** Enhance the SMS alert system to provide more detailed and context-specific notifications. Alerts can be customized based on gas concentration thresholds and user preferences, ensuring timely responses to potential environmental hazards.

9. **Machine Learning Algorithms:** Integrate machine learning algorithms to predict air pollution levels based on historical data, meteorological conditions, and other relevant factors. Machine learning models can aid in forecasting air quality trends and guiding long-term pollution control measures.

Chapter 9

Benefits

The gas monitoring IoT device developed using Arduino Uno and various sensors offers numerous benefits that contribute to addressing air pollution and promoting a healthier and sustainable environment. Some of the key advantages of this device are:

1. **Environmental Protection:** The primary benefit of the gas monitoring device is its ability to detect and measure harmful gases in the environment. By continuously monitoring gas concentrations, the device helps identify areas with poor air quality and enables timely interventions to mitigate pollution and protect the environment.

2. **Public Health Safety:** Air pollution poses significant health risks to humans, causing respiratory and cardiovascular problems. This device provides real-time gas concentration data, allowing individuals and authorities to take proactive measures to safeguard public health and reduce the incidence of pollution-related illnesses.

3. **Pollution Control and Regulation:** The device offers valuable data to environmental regulatory bodies, aiding them in understanding pollution trends and formulating effective policies for pollution control and abatement. This information is essential for implementing targeted strategies to reduce harmful gas emissions.

4. **Biodiversity Conservation:** The device's monitoring of harmful gases' impact on butterflies and other pollinators helps raise awareness about the broader ecological consequences of air pollution. By preserving butterfly populations and their habitats, the device indirectly supports biodiversity conservation.

5. **Data-Driven Decision Making:** The gas monitoring system generates comprehensive and accurate gas concentration data. This data-driven approach empowers decision-makers with critical information to plan and execute pollution control measures based on real-time and historical data.

6. **Early Warning System:** With the capability to send real-time SMS alerts, the device acts as an early warning system, notifying authorities and individuals about sudden spikes in gas concentrations. Rapid alerts allow for swift responses and emergency protocols to be activated.

7. **Cost-Effective Solution:** The device, built using readily available components and open-source software, offers a cost-effective solution for gas monitoring compared to expensive commercial systems. It enables wider adoption in both urban and remote areas where air pollution monitoring is crucial.

8. **Scalability and Versatility:** The modular design of the device allows for easy integration of additional sensors and functionalities. This scalability ensures adaptability to different environments and potential future enhancements.

9. **User-Friendly Interface:** The implementation of a data visualization interface, such as a web-based dashboard or a mobile app, makes the device user-friendly. Users can easily access, interpret, and share gas concentration data for informed decision-making.

Acknowledgments

We would like to take this opportunity to express our sincere gratitude to all those who have contributed to the successful completion of our M.Sc. project and the preparation of this report.

First and foremost, We are deeply thankful to our supervisor, Sri Durjoy Roy, for his unwavering guidance, invaluable insights, and continuous support throughout the duration of this project. His expertise, patience, and mentorship played a pivotal role in shaping the direction and outcomes of our research.

We extend our heartfelt appreciation to the faculty members of Department of Electronics, West Bengal State University, Barasat for providing a conducive academic environment.

We are grateful to our family and friends for their unwavering belief in us. Sri Prithwish Biswas, our classmate in M.Sc., deserves a special mention for his immense knowledge, and the helping hand he has extended towards this project.

This project has been a journey of growth, learning, and self-discovery. The collective efforts and support from numerous individuals have made this endeavor both meaningful and fulfilling.

Thank you to everyone who played a role in making this project a reality.

August 2023
West Bengal State University

Swani Chatterjee	Saurabh Biswas	Suryashmi Chakraborty
10021017100009	10021017100010	10021017100011

References

[1] Internet of Things (IoT): Principles, Paradigms and Applications of IoT by Dr Kamlesh Lakhwani, https://www.researchgate.net/publication/341214595_Internet_of_Things_IoT_Principles_Paradigms_and_Applications_of_IoT

[2] Arduino for Beginners by makerspaces, https://www.makerspaces.com/wp-content/uploads/2017/02/Arduino-For-Beginners-REV2.pdf

[3] Programming with Arduino by Hans-Petter Halvorsen, https://www.halvorsen.blog/documents/technology/resources/resources/Arduino/Programming%20with%20Arduino.pdf

[4] SIM900A Details, https://microcontrollerslab.com/sim900a-gsm-module-pinout-examples-applications-datasheet/

[5] Arduino Code Optimization from ChatGPT, https://openai.com/gpt-4