Ahmedabad
University

Project Report-1
Group-5
Deadline: 12/09/2024

Instructor: Anurag Lakhlani **Teaching Assistant:** Priyesh Chaturvedi

Enrolment No	Name
AU2240118	Tirth Teraiya
AU2240080	Namra Maniar
AU2240244	Saumya Shah
AU224075	Henish Trada

Chapter 04

Circuit Diagram and Explanation

Introduction

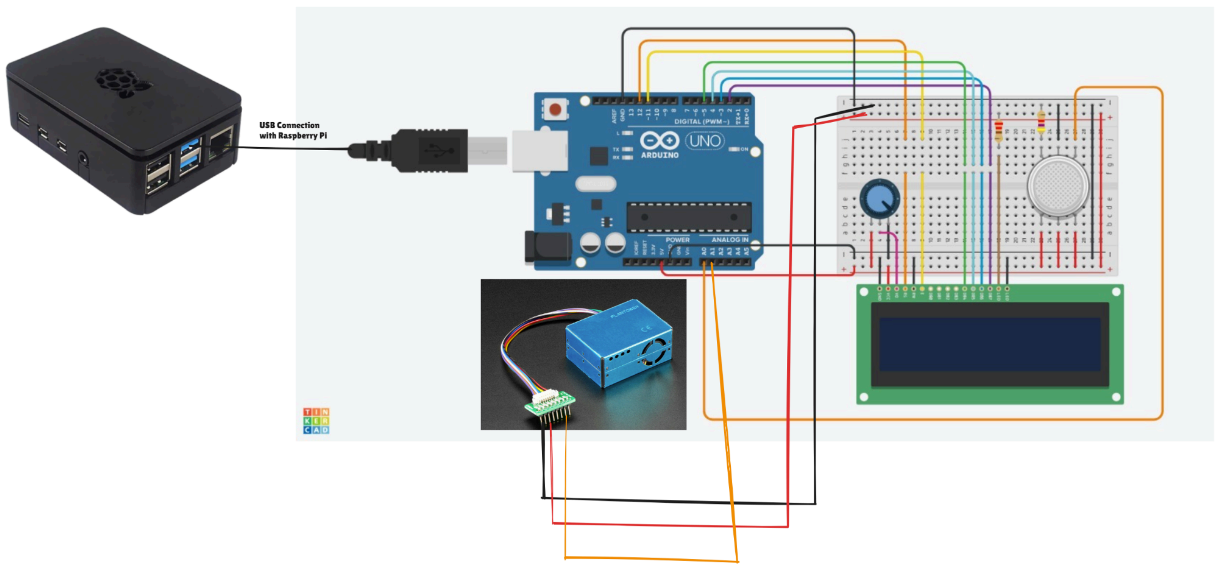
This chapter presents the detailed electronic circuit design for the IoT-based Environmental Air Quality Monitoring and Alerting System. It outlines the overall system schematic, provides a comprehensive justification for the selection of each hardware component, and analyzes the technical inputs and outputs of the integrated system.

4.1 Overall System Schematic

The system is architected around a Raspberry Pi 3 Model B, which serves as the central processing and communication hub. A critical design consideration is the Raspberry Pi's lack of native analog-to-digital conversion capabilities, which necessitates the inclusion of an external Analog-to-Digital Converter (ADC) Arduino UNO to interface with the MQ135 gas sensor and PM2.5 particle sensor. We will display the reading on LCD display

The complete wiring diagram is detailed below, describing the connections between the Raspberry Pi and all peripheral components.

Circuit Diagram :



Component Connections:

1. **Power Distribution:**
   - The Raspberry Pi's 5V pins supply power to the MCP3008 ADC, MQ135 Gas Sensor, PMS5003 PM Sensor, 16x2 I2C LCD, and the Active Buzzer module.
   - All components share a common ground (GND) connected to the Raspberry Pi's GND pins.
2. **Arduino UNO to Raspberry Pi (USB Serial Interface):** The Arduino UNO is essential for converting the analog signal from the MQ135 sensor into a digital format and sending it to the Raspberry Pi.
   - Arduino UNO's USB port is connected to one of the Raspberry Pi's USB ports.
3. **MQ135 Gas Sensor to MCP3008:**
   - MQ135 VCC to Raspberry Pi 5V.
   - MQ135 GND to Raspberry Pi GND.
   - MQ135 AOUT (Analog Output) to MCP3008 CH0 (Channel 0).
4. **PMS5003 Particulate Matter Sensor to Raspberry Pi (UART Interface):**
   - PMS5003 VCC to Raspberry Pi 5V.
   - PMS5003 GND to Raspberry Pi GND.
   - PMS5003 TXD (Transmit) to Raspberry Pi RXD (GPIO15).
   - The sensor's logic level is 3.3V, making it directly compatible with the Raspberry Pi's GPIO pins.
5. **16x2 I2C LCD to Raspberry Pi (I2C Interface):**
   - LCD VCC to Raspberry Pi 5V.
   - LCD GND to Raspberry Pi GND.
   - LCD SDA (Serial Data) to Raspberry Pi SDA (GPIO2).
   - LCD SCL (Serial Clock) to Raspberry Pi SCL (GPIO3).
6. **Active Buzzer Module to Raspberry Pi (Digital GPIO):**
   - Buzzer VCC to Raspberry Pi 5V.
   - Buzzer GND to Raspberry Pi GND.
   - Buzzer I/O (Signal) to Raspberry Pi GPIO17.

4.2 Component Selection and Justification

The selection of each component was a deliberate process, balancing performance requirements, cost, interfacing complexity, and alignment with the project's objectives as defined in Project Report 1. The rationale behind each choice is detailed in Table 4.1.

Component	Key Specifications	Selection Rationale for this Project

Raspberry Pi 3 B	1.2GHz Quad-Core CPU, 1GB RAM, Wi-Fi/BLE	The project requires processing data from multiple sensors concurrently and transmitting it to a cloud platform. The Raspberry Pi's powerful processor, ample RAM, and built-in Wi-Fi make it an ideal choice for this application.
Arduino UNO	ATmega328P, 6 Analog Inputs, USB Interface	The Raspberry Pi lacks native analog inputs, making it incompatible with the MQ135's analog output. An Arduino UNO is used as a cost-effective and programmable ADC. It reads the analog voltage from the sensor, converts it to a digital value, and transmits it to the Raspberry Pi via a simple USB serial connection, leveraging its built-in ADC and straightforward serial libraries.
MQ135 Sensor	Detects NH3, NOx, alcohol, Benzene, smoke, CO2 Detection Range : 100-1000ppm	This sensor directly addresses the project's primary goal of monitoring a wide range of harmful gases commonly associated with poor air quality.
PMS5003 Sensor	Measures PM1.0, PM2.5, PM10 via laser scattering	Particulate matter is a critical indicator of air pollution and a key parameter mentioned in the project's introduction. The PMS5003 was chosen for its accuracy, reliability, and its simple UART interface, which can be easily managed by the Raspberry Pi's serial port.
16x2 I2C LCD	2-line, 16-char display	This component fulfills the local display requirement of the "Processing and Smart Alerting System". The I2C version was specifically chosen to minimize the number of GPIO pins required for operation (only two data pins), preserving the Raspberry Pi's limited GPIO resources for other sensors and actuators.
Active Buzzer	Built-in oscillator, ~2.5kHz tone, low-level trigger	The buzzer provides the audible alert functionality specified in the block diagram. An active buzzer was selected for its simplicity; it generates a tone with a simple DC signal from a GPIO pin.

Table 4.1 : Component Selection Rationale

4.3 Technical Input/Output Analysis

4.3.1 System Inputs

- **Environmental Data: These are the raw physical parameters being measured from the ambient atmosphere.**
  - Concentration of gaseous compounds (e.g., ammonia, nitrogen oxides, benzene, smoke, carbon dioxide).
  - Density of suspended particulate matter, specifically PM1.0, PM2.5, and PM10.
  - Ambient air temperature.
  - Relative humidity.
- **Electrical Signals: These are the transduced signals that the Raspberry Pi processes.**
  - **Serial Data Stream (USB/UART from Arduino):** The MQ135 sensor outputs a variable analog voltage to the Arduino UNO. The Arduino converts this to a 10-bit digital value and transmits it as a serial data stream over USB to the Raspberry Pi.
  - **Serial Data Stream (UART from PMS5003):** The PMS5003 sensor transmits a structured 32-byte data frame via its TXD pin at a baud rate of 9600 bps. This frame contains discrete values for PM1.0, PM2.5, and PM10 concentrations.

4.3.2 System Outputs

The system generates three primary types of outputs to fulfill its monitoring and alerting functions.

- **Visual Information:** Real-time data is presented to the user on the 16x2 character LCD. This includes:
  - Numerical values for PM2.5 (in µg/m^3), temperature (in °C), and humidity (in %).
  - A qualitative classification of the current air quality based on a calculated Air Quality Index (AQI), such as "Good," "Moderate," or "Hazardous".
- **Audible Alerts:** An audible alarm is generated by the active buzzer module.
  - This output consists of a continuous, single-frequency tone of approximately 2.5 kHz.
  - The alert is triggered programmatically when the calculated AQI surpasses a predefined "Hazardous" threshold, providing an immediate local warning of unsafe conditions.

Chapter 05

Arduino UNO/Mega OR Raspberry Pi Features

This chapter presents the detailed electronic circuit design for the IoT-based Environmental Air Quality Monitoring and Alerting System. It outlines the overall system schematic, provides a comprehensive justification for the selection of each hardware component, and analyzes the technical inputs and outputs of the integrated system.

5.1 Raspberry Pi 3 Model B: Core Features and Project Role

The Raspberry Pi 3 Model B is a credit card-sized single-board computer (SBC) that offers a powerful combination of processing capability, memory, and integrated connectivity, making it a popular choice for complex IoT projects.

5.1.1 Key Features Utilized in Project

Processor: The board is equipped with a Quad-Core 1.2GHz Broadcom BCM2837 64-bit ARMv8 processor.

Memory: It includes 1GB of LPDDR2 SDRAM.

Connectivity: The on-board BCM43438 wireless LAN (Wi-Fi 802.11n) and Bluetooth Low Energy (BLE) are cornerstone features for this IoT project.

GPIO (General Purpose Input/Output): The 40-pin extended GPIO header provides the physical interface to all sensors and actuators. This project utilizes the header's support for multiple protocols:

- **I2C pins (SDA, SCL):** For the 16x2 LCD module.
- **UART pins (TXD, RXD):** For the PMS5003 sensor.
- **Standard Digital I/O pins:** For Active Buzzer.

USB Ports: The four USB 2.0 ports are used for interfacing with the Arduino UNO, which acts as an ADC and provides data over a serial connection.

5.1.2 Strengths and Weaknesses for this Project

Strengths:

- **Full-fledged Operating System:** Running Raspberry Pi OS (a Debian derivative) greatly simplifies development. It allows for easy installation of programming languages (Python), libraries (e.g., RPi.GPIO, smbus), remote access via SSH for debugging, and robust management of network connections.
- **Vast Community and Library Support:** The Raspberry Pi has one of the largest and most active communities in the maker space. This translates to extensive, well-maintained Python libraries for virtually every common sensor and peripheral, including every component used in this project.
- **High Computing Power:** The quad-core processor enables not only data acquisition but also on-device data processing. This allows for the implementation of more complex AQI calculation algorithms.

Weaknesses and Mitigation:

- **Lack of Analog Inputs:** The most significant weakness of the Raspberry Pi for electronics projects is its inability to read analog signals directly. This limitation is directly addressed by using an Arduino UNO as an external ADC. The Arduino reads the analog signal from the MQ135, and the Raspberry Pi reads the digitized value from the Arduino over a standard USB serial connection. This is a common and well-documented pattern for interfacing analog sensors with a Raspberry Pi.
- **Higher Power Consumption:** Compared to a microcontroller like the Arduino, the Raspberry Pi is significantly more power-hungry, consuming around 700 mA or more under load. This makes it less suitable for battery-powered, portable applications. For this project, a stationary monitoring node powered by a stable 5V/2.5A DC adapter is assumed, rendering this weakness irrelevant to the primary use case.

Chapter 06

Program Flow Chart

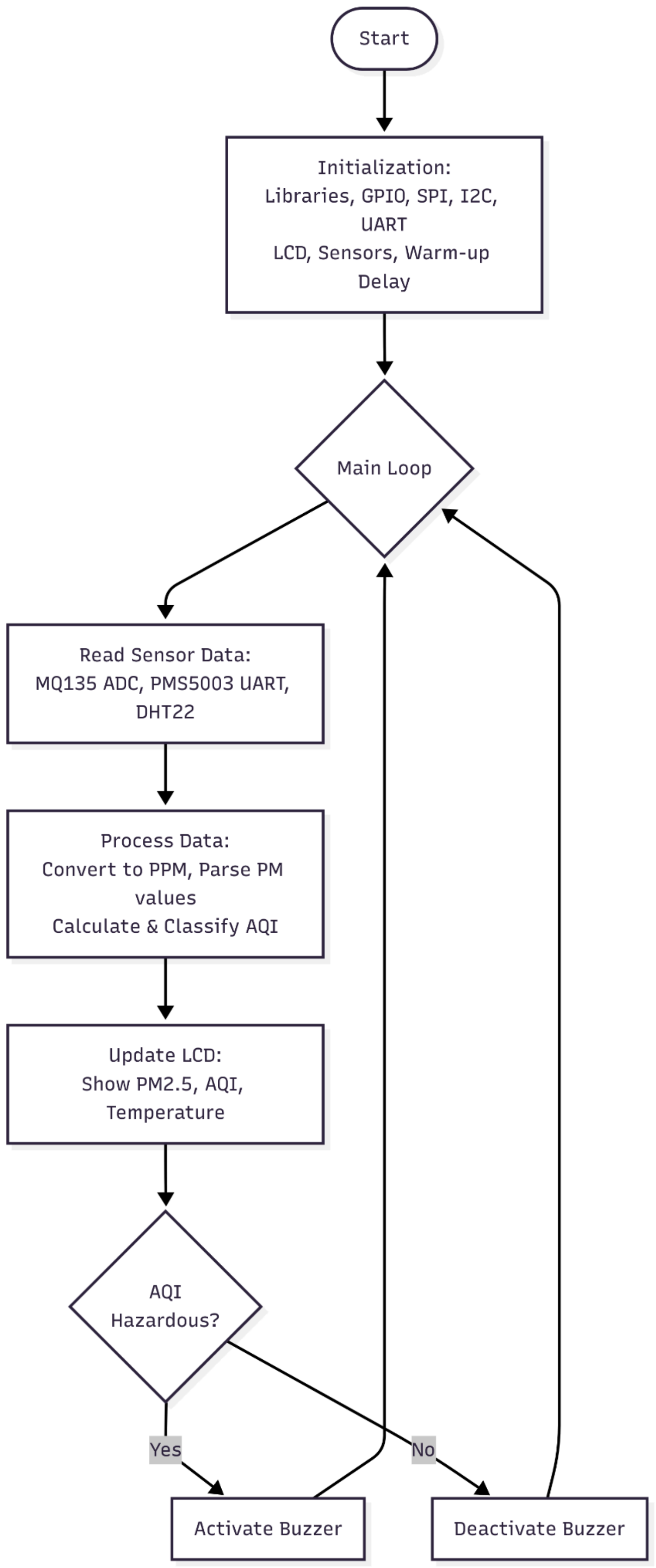
This chapter outlines the logical flow of the software designed to run on the Raspberry Pi. The program is responsible for initializing hardware, continuously reading data from all sensors, processing this data to determine air quality, displaying the results locally, and triggering alerts when necessary.

6.1 High-Level Program Logic Flowchart

The program operates on a continuous loop, orchestrating the various hardware components. A significant aspect of the software architecture is its ability to manage multiple, disparate communication interfaces concurrently—a task well-suited to the Raspberry Pi's multi-threaded Linux environment. The logical sequence of operations is as follows:

1. **Start:** The program execution begins.
2. **Initialization:** This is a critical one-time setup phase.
    - **Import Libraries:** Load necessary Python libraries.
    - **Initialize Interfaces:** Configure the GPIO pins for their respective modes (input/output). This includes setting up the I2C bus for the LCD and the serial (UART) port for the PMS5003. A separate serial object is created for the USB connection to the Arduino.
    - **Initialize Peripherals:** Create objects for each hardware component (e.g., the LCD object, the ADC object).
    - **Display Startup Message:** Write an initial message like "System Initializing..." to the LCD to provide user feedback.
    - **Sensor Warm-up:** Implement a mandatory delay to allow sensors, particularly the MQ135 with its internal heater, to reach a stable operating temperature. This is crucial for accurate initial readings.
3. **Enter Main Loop:** The program enters an infinite `while` loop to perform continuous monitoring.
4. **Read All Sensor Data:** In each iteration of the loop, the program polls each sensor to gather fresh data.
    - Read the serial data line from the Arduino UNO, which contains the 10-bit digitized value from the MQ135 sensor.
    - Read the 32-byte data frame from the PMS5003 via the UART serial buffer and validate the checksum.

5. **Process and Analyze Data:** The raw data from the sensors is converted into meaningful environmental metrics.
    - Convert the raw ADC value received from the Arduino into a voltage and then, using the sensor's characteristic curve from its datasheet, estimate the gas concentration in Parts Per Million (PPM).
    - Parse the PMS5003 data frame to extract the PM2.5 and PM10 concentration values (in µg/m^3).
    - Calculate a composite Air Quality Index (AQI) using a weighted algorithm based on the measured PM2.5 and gas concentration values.
    - Classify the calculated AQI into a human-readable category (e.g., "Good," "Moderate," "Hazardous") based on standard thresholds.
6. **Update Local Display:** The processed information is displayed on the 16x2 I2C LCD.
    - The LCD is cleared.
    - Formatted strings showing the most critical data (e.g., PM2.5 value, AQI category, temperature) are written to the two lines of the display.
7. **Check Alert Condition:** A decision point checks if the air quality has reached a critical level.
    - The program evaluates: `if AQI_category == "Hazardous"`.
8. **Manage Alert System:**
    - **If True (Hazardous):** The program sends a signal to the GPIO pin connected to the active buzzer, causing it to emit an audible alarm.
    - **If False (Not Hazardous):** The program ensures the buzzer's GPIO pin is in a state that keeps the buzzer silent.
9. **Loop:** The execution flow returns to the "Read All Sensor Data" step to begin the next monitoring cycle.


Start
Initialization:
Libraries, GPIO, SPI, I2C, UART
LCD, Sensors, Warm-up Delay
Main Loop
Read Sensor Data:
MQ135 ADC, PMS5003 UART, DHT22
Process Data:
Convert to PPM, Parse PM values
Calculate & Classify AQI
Update LCD:
Show PM2.5, AQI, Temperature
AQI Hazardous?
Yes
No
Activate Buzzer
Deactivate Buzzer

Chapter 07

Sensors

This chapter provides a detailed technical overview of the sensors employed in the air quality monitoring system. For each sensor, the operating principle, physical specifications, power ratings, and method of interfacing with the Raspberry Pi are described.

7.1 MQ135 Air Quality Gas Sensor

The MQ135 is a semiconductor-type gas sensor designed to detect a wide range of gases, making it suitable for general air quality monitoring. It is particularly sensitive to ammonia (NH_3), nitrogen oxides (NO_x), alcohol, benzene, smoke, and carbon dioxide (CO_2).

7.1.1 Operating Principle

The sensing mechanism of the MQ135 is based on a chemoresistive material, specifically Tin Dioxide (SnO_2). The core principle is as follows:

1. **Sensing Material:** The sensor contains a micro-tubular ceramic element coated with a layer of SnO_2. In clean air, oxygen molecules are adsorbed onto the surface of the SnO_2, trapping electrons from the material and creating a potential barrier that results in high electrical resistance.
2. **Gas Detection:** When pollutant gases are introduced, they react with the adsorbed oxygen molecules on the sensor's surface. This reaction releases the trapped electrons back into the SnO_2, lowering the potential barrier and thereby decreasing the material's overall resistance.
3. **Signal Output:** This change in resistance is proportional to the concentration of the target gas. An external load resistor (RL) is used in a voltage divider circuit. As the sensor's resistance (RS) changes, the voltage across the load resistor (VRL) also changes, providing an analog voltage output that corresponds to the gas concentration.
4. **Heating Element:** The sensor includes an internal heating element made of Ni-Cr alloy, which is necessary to bring the SnO_2 material to its optimal operating temperature (typically 100-200°C). This is why the sensor requires a pre-heating period of at least 24 hours for optimal performance and stabilization before accurate measurements can be taken.

7.1.2 Physical Dimensions and Pin Diagram

The MQ135 is commonly available as a module mounted on a small PCB for ease of use.

- **Physical Dimensions:** The module typically measures approximately 35 mm x 22 mm x 23 mm (L x W x H).
- **Pinout:** The module has a 4-pin header for connection:
  - **VCC:** Power supply input (5V).
  - **GND:** Ground connection.
  - **D0 (Digital Output):** Provides a HIGH or LOW signal based on a threshold set by an on-board potentiometer. This pin is not used in this project.
  - **A0 (Analog Output):** Provides the analog voltage signal proportional to gas concentration. This is the primary output used for this project.

7.1.3 Power Ratings and Technical Specifications

1. **Heater Voltage (VH):** 5.0V ± 0.1V AC or DC.
2. **Circuit Voltage (VC):** The sensor module operates at 5V DC.
3. **Heater Consumption (PH):** Approximately ≤950mW.
4. **Load Resistance (RL):** Adjustable via on-board potentiometer, but typically around 10 kΩ.
5. **Detection Range:** 10 – 1000 ppm for gases like ammonia and benzene.

7.1.4 Interfacing with Raspberry Pi

As the Raspberry Pi cannot process analog signals, the A0 pin of the MQ135 cannot be connected directly. An Arduino UNO serves as the necessary intermediary, acting as an Analog-to-Digital Converter (ADC).

- **Circuit:** The MQ135's VCC and GND pins are connected to the Arduino's 5V and GND pins, respectively. The sensor's A0 pin is connected to one of the Arduino's analog input pins (e.g., A0). The Arduino is then connected to the Raspberry Pi via a USB cable. The Arduino reads the analog voltage, converts it into a 10-bit digital value (0-1023), and sends this value over the serial connection to the Raspberry Pi, which can then read and process it.

Chapter 08

Actuators and Displays

This chapter details the output components of the system: the devices that provide feedback to the user. It covers the operating principles, specifications, and interfacing methods for the local display and the audible alert mechanism.

8.1 16x2 I2C LCD Module

The 16x2 I2C LCD module is the primary visual interface for the system, providing real-time, human-readable information about the measured air quality parameters. It displays 16 characters per line across two lines.

8.1.1 Operating Principle

The module is a composite device consisting of two main parts: a standard character LCD and an I2C interface board.

- **Character LCD:** The display itself is typically based on the Hitachi HD44780 controller or a compatible equivalent. This controller requires a parallel interface (either 4-bit or 8-bit) to receive commands (like "clear display" or "move cursor") and data (the ASCII codes for characters to be displayed). Driving this directly would consume at least 6 GPIO pins from the Raspberry Pi.
- **I2C Interface Board:** To simplify wiring, an interface board is "piggy-backed" onto the LCD's 16-pin header. This board is built around a PCF8574 I/O expander chip. The PCF8574 acts as an I2C slave device. The Raspberry Pi, acting as the I2C master, sends commands and data over the two-wire I2C bus (SDA and SCL). The PCF8574 receives these serial commands and translates them into the parallel signals required by the HD44780 controller, managing the LCD's data and control lines. This elegant solution reduces the required GPIO pin count from six or more down to just two, which is a significant advantage in a pin-constrained project.

8.1.2 Physical Dimensions and Pin Diagram

- **Physical Dimensions:** The module's PCB is approximately 80 mm x 36 mm.

- **Pinout:** The I2C interface provides a simple 4-pin header for connection:
    - **GND:** Ground.
    - **VCC:** Power supply (5V).
    - **SDA (Serial Data):** The I2C data line.
    - **SCL (Serial Clock):** The I2C clock line.

8.1.3 Power Ratings and Technical Specifications

- **Operating Voltage:** 5V DC.
- **Interface:** I2C (Inter-Integrated Circuit).
- **I2C Address:** The slave address is typically either 0x27 or 0x3F, depending on the specific I/O expander chip used on the interface board. The default address must be confirmed, often by running the `i2cdetect -y 1` command on the Raspberry Pi's terminal.
- **Display Format:** 16 characters x 2 lines.
- **Backlight:** Blue or green, controllable via software commands sent over I2C.
- **Contrast:** Adjustable via an on-board potentiometer.

8.1.4 Interfacing with Raspberry Pi

Interfacing the module with the Raspberry Pi is straightforward due to the I2C protocol.

- **Circuit:** The four pins of the LCD module (GND, VCC, SDA, SCL) are connected directly to the corresponding pins on the Raspberry Pi's GPIO header (GND, 5V, GPIO2/SDA, GPIO3/SCL). No other components are required for the connection. The Raspberry Pi's I2C interface must be enabled in the `raspi-config` utility.