COMPUTOLOGY: Journal of Applied Computer Science and Intelligent Technologies
Volume 5, Issue 1, Jan-Jun 2025, pp. 24-41
DOI: 10.17492/computology.v5i1.2502
https://www.journalpressindia.com/cjacsit

Interfacing and Configuration of Dynamic Sensors

Siddhesh Ghatole, Ajinkya Ghogare** and Gaurav Bachulkar****

ABSTRACT

This study aims to develop a dynamic, user-configurable interface system for integrating sensors with the Raspberry Pi, addressing its lack of built-in analog-to-digital conversion capability. The objective is to enable real-time data acquisition and monitoring using both voltage and current-output sensors through the use of an external MCP3008 ADC and signal conditioning circuits. The methodology includes designing a flexible interface board, configuring sensor types and thresholds through terminal or ThingSpeak API, and collecting and visualizing data via a cloud dashboard. The system processes analog inputs using voltage dividers and current-to-voltage converters, transmitting digital readings at regular intervals. Findings indicate that the system accurately reads, scales, and processes multiple analog signals, supporting threshold-based alerts and cloud integration. The solution demonstrates reliability, precision, and ease of use, making it ideal for embedded systems education and real-world IoT monitoring applications.

Keywords: *Analog signal conditioning; Voltage divider; Raspberry Pi; User-configurable scaling; Current-to-voltage conversion.*

1.0 Introduction

1.1 Background

Embedded systems often require precise data acquisition from a variety of analog sensors. However, integrating analog sensors with microcontrollers such as the Raspberry Pi poses challenges due to voltage and signal conditioning limitations. The Raspberry Pi, with its limited analog-to-digital conversion (ADC) capabilities, necessitates external circuitry to adapt sensor output to a readable format.

**Corresponding author; Student, Department of Electronics and Telecommunication, Marathwada Mitra Mandal's College of Engineering, Pune, Maharashtra, India (E-mail: siddheshghatole63@gmail.com)*
***Student, Department of Electronics and Telecommunication, Marathwada Mitra Mandal's College of Engineering, Pune, Maharashtra, India (E-mail: ajinkya9873@gmail.com)*
****Student, Department of Electronics and Telecommunication, Marathwada Mitra Mandal's College of Engineering, Pune, Maharashtra, India (E-mail: gauravbachulkar08@gmail.com)*

Analog signals must be scaled within a 0-3.3V range, and sensor outputs—whether in voltage or current—need efficient, customizable conditioning to ensure accurate readings.

1.2 Research gap

Existing systems often support fixed sensor configurations and lack flexibility for dynamic scaling and threshold management. Most implementations do not support both voltage and current-based inputs or real-time remote threshold configuration.

1.3 Research objective

To design and implement a dynamic, user-configurable sensor interface that supports various signal types (voltage and current), processes them using an external ADC, and provides real-time data monitoring and threshold alerting through a cloud platform.

2.0 Review of Literature

The integration of sensors with embedded systems, especially platforms like the Raspberry Pi, has gained significant attention over the past decade due to its applications in environmental monitoring, automation, and smart systems. A broad range of recent studies provide insights into the challenges and solutions in sensor interfacing, analog signal processing, and cloud-based data management.

Ardiansyah *et al.* (2018) developed a rain detection and weather prediction system using Mamdani fuzzy logic inference. Their approach combined various analog sensors, such as temperature and rain sensors, with Arduino platforms to produce human-interpretable weather outcomes. While effective in prediction, their system lacked user-configurable features and remote monitoring capabilities through cloud platforms, limiting its adaptability for industrial use cases. Nasution & Harahap (2020) examined the degradation and error prediction of LM35 temperature sensors using simple linear regression. Their work emphasized the importance of long-term reliability and accuracy in analog sensors, demonstrating how modeling sensor behavior can contribute to proactive calibration strategies.

In another study, Johnson & Lee (2020) implemented a Raspberry Pi-based cloud monitoring system that interfaced with ThingSpeak for real-time data visualization. Their framework primarily dealt with digital sensors and fixed configurations. Though the system highlighted the potential of cloud integration for IoT applications, it did not accommodate analog signal conditioning or dynamic sensor configuration.

Choi *et al.* (2019) investigated level-shifting circuits to enable safe communication between 5V sensors and the 3.3V GPIO pins of the Raspberry Pi. Their work laid the

foundation for reliable hardware interfacing when using higher voltage sensors, which is critical in preventing damage to the Raspberry Pi while reading analog data. Gupta *et al.* (2017) provided a practical method for scaling 0–10V analog outputs using voltage divider circuits. Their approach is instrumental in safely interfacing industrial sensors with microcontrollers that support lower voltage ADC inputs. This principle is directly adopted in the current project to ensure signal compatibility with the MCP3008 ADC. Further enhancing analog compatibility, Patel *et al.* (2019) studied signal conditioning for current-output sensors, particularly those operating in the 4–20mA range. By recommending a standard 250Ω resistor for current-to-voltage conversion, their work helped inform the design of the converter modules used in this study. Their findings emphasize the industrial relevance of such configurations for accurate analog interfacing.

Nguyen & Tran (2022) proposed an IoT-based industrial monitoring solution using Raspberry Pi and cloud connectivity. While their system supported environmental data logging through ThingSpeak, it relied on fixed sensor setups without provisions for user-defined thresholds or dynamic input scaling. Brown *et al.* (2016) conducted reliability testing on the MCP3008 ADC and confirmed its capability for multi-channel analog data acquisition over the SPI protocol. Their validation of the MCP3008's stability and low latency informed its selection in this project, which uses all eight of its input channels for different sensor types.

2.1 Concluding statement

Collectively, these studies offer valuable tools, methodologies, and insights into embedded sensor interfacing, signal conditioning, and IoT system design. However, they often address isolated challenges—such as scaling voltage inputs, cloud visualization, or ADC integration—without delivering a unified, reconfigurable, and cloud-integrated solution. The current project stands out by incorporating real-time analog data acquisition, user-configurable sensor selection, flexible threshold management, and cloud-based visualization into one cohesive platform, addressing a significant gap in existing research and implementations.

3.0 Research Methodology

3.1 Research design

This project follows an applied research approach with a design and development methodology. The focus is on building a functional hardware prototype an external interface board for the Raspberry Pi, facilitating the integration and configuration of various analog sensors. The methodology combines experimental and engineering design processes with iterative testing to ensure accurate signal acquisition and processing.

3.2 Sources of data

The study primarily focuses on relevant primary data which has been obtained from the following sources:

- Sensor signal data was collected through experimental setups using three widely used analog sensors: the LM35 temperature sensor, the MQ-3 gas sensor (used primarily for alcohol vapor detection), and the Rain Drop sensor (used to detect water presence and intensity on a surface). These sensors were selected for their varied output signal behaviors and relevance in environmental monitoring and smart embedded systems.
- Technical datasheets and sensor specifications provided by the respective manufacturers, including Texas Instruments (LM35), Hanwei Electronics (MQ-3), and various open-source hardware suppliers (Rain Drop sensor), were consulted to understand operating voltage ranges, sensitivity, calibration parameters, and output signal characteristics.
- Real-time data acquisition was carried out using a Raspberry Pi connected to the sensors via the MCP3008 Analog-to-Digital Converter (ADC), using the SPI communication protocol. Sensor readings were obtained using custom Python scripts that logged, converted, and displayed data on the Raspberry Pi terminal for analysis.
- Official documentation including the Raspberry Pi 4 Model B technical manual, MCP3008 datasheet, and WiringPi and SPI library documentation was utilized to configure and establish proper hardware communication, signal reading, and safe interfacing.
- Comparative validation of sensor outputs was done using standard reference instruments such as digital multimeters and regulated power supplies. These instruments were used to ensure the accuracy of the interface board's voltage and current conversion circuits, as well as to cross-verify sensor readings under controlled conditions.

3.3 Sample and experimental setup

A board was constructed and connected to a Raspberry Pi 4. The sensors were assigned to individual analog channels on the MCP3008. Data was logged continuously over a 72-hour period under both indoor (controlled) and outdoor (variable) conditions. The setup included:

- Raspberry Pi 4 Model B
- MCP3008 10-bit ADC
- Passive filters (resistors and capacitors)
- Breadboard and jumper wires for circuit prototyping

- Three sensors (MQ-3, LM35, Raindrop) connected simultaneously to different ADC channels.
- Power supply: 5V regulated via Raspberry Pi

The interface board connects to the Raspberry Pi via MCP3008, an external 10-bit ADC. Sensors are assigned to specific channels (up to 8), and their outputs are read periodically. The Raspberry Pi collects raw ADC data using SPI communication and converts it to voltage using:

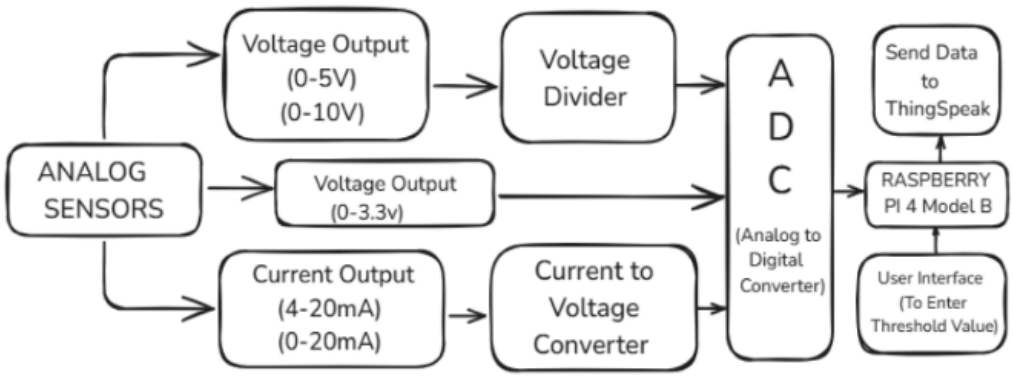
Voltage = (ADC Value / 1023) * Vref

where

Vref = 3.3V

The processed voltage values are then compared to predefined threshold values.

Figure 1: System Block Diagram



3.4 Hardware setup

The following hardware components are used in the implementation:

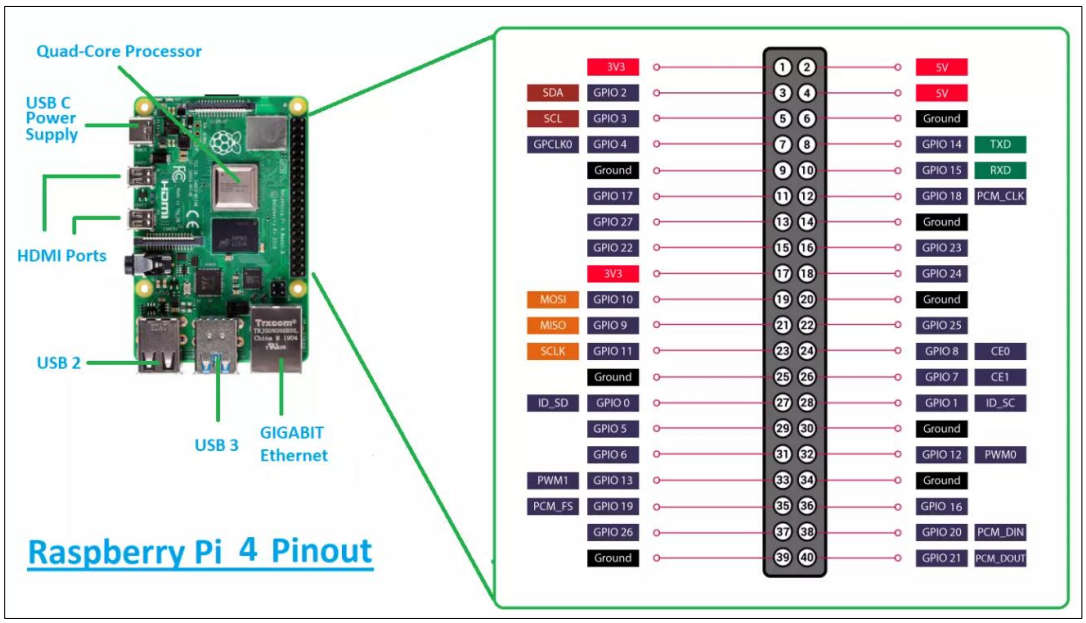
- **Raspberry Pi 4 Model B**: The Raspberry Pi serves as the central processing unit, receiving digital data from the MCP3008 via SPI. A C program, utilizing the wiringPi library, was developed to read the ADC values from all 8 channels, calculate the corresponding voltages using the formula:

Voltage = $(\frac{Adc\ Value}{1023})$ x VREF

where:

1. ADC Value is the digital value output by the MCP3008 (ranging from 0 to 1023 for a 10-bit ADC).
2. VREF is the reference voltage of the MCP3008, set to 5V in this project.

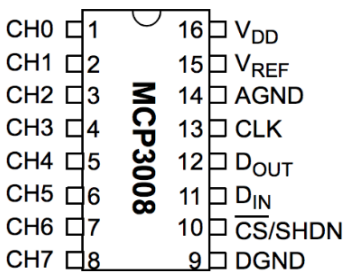
Figure 2: Raspberry Pi 4 Model B Pinout



Source: https://images.theengineeringprojects.com

- **MCP3008 ADC:** The MCP3008 is a 10-bit ADC with 8 analog input channels, designed to convert analog signal into digital values for processing by a single-board computer like the Raspberry Pi.

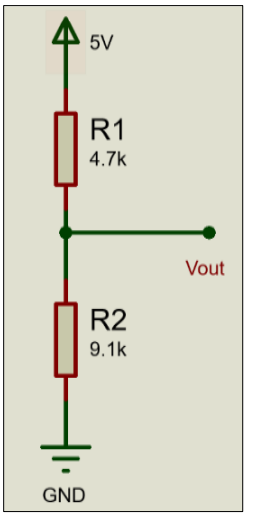
Figure 3: MCP3008 Pinout



- **Voltage Divider Circuit**: To reduce a sensor's 5V analog output to within 3.3V for ADC input, a basic two-resistor voltage divider was design.

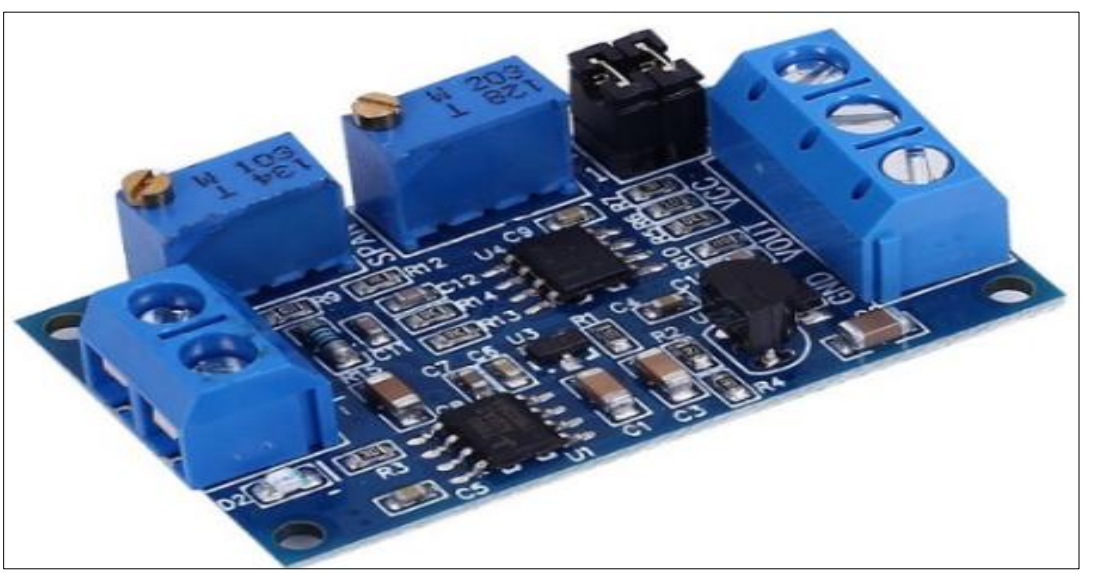
  Voltage divider formula: Vout = Vin x ($\frac{R2}{R1}$)+R2

Figure 4: 5V scaled to 3.3V



- **Current-to-Voltage Converter Module:** This current to voltage converter module is used to interface the current sensors which outputs current from 4mA-20Ma and it converts the 4-20mA and 0-20mA to 0-3.3V signal.

Figure 5: Current to Voltage Converter Module



- **Sensors**:

LM35 (temperature sensor): The LM35 sensor is an analog sensor that converts temperature into electrical quantities in the form of voltage. The LM35 temperature sensor has a parameter that every 1ºC increase in output voltage increases by 10mV with a maximum sensor output limit of 1.5 V at 150 ºC. It has higher precision and wider range of linear working. The output voltage LM35 linear proportional Celsius temperature, at ordinary temperatures, it can provide ±1/4°C common precision of room temperature without needs additional calibration or fine-tune.

Figure 6: LM35 Temperature Sensor

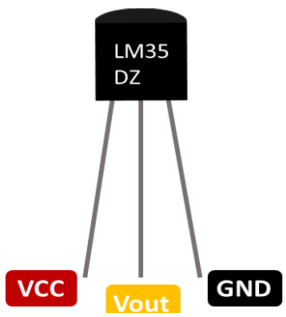


Table 1: LM35 Temperature Sensor Specifications

Sensor Type	Analog Temperature Sensor
Temperature Range	-55°C to +150°C
Output Voltage	10 mV/°C (e.g., 250 mV = 25°C)
Accuracy	±0.5°C (typical) at 25°C
Supply Voltage	4-30V

MQ3 (gas sensor): This is a low-cost sensor with an ability to detect 0.05 mg /L to 10 mg / L of concentration of alcohol gases. For this unit, the active content is SnO2, which is lower in clean air. Its conductivity increases with the rise in alcohol gas concentrations. This is extremely alcohol-sensitive and has strong flame, vapor and fuel protection to disturbances. The digital and analog outputs are given by this module. This sensor can, if required, be used to detect alcohol levels in the driver's breath. The device has a high sensitivity and fast reaction time. The sensor outputs its values in term of varying voltage (Analog output); therefore, the data can be obtained via the microcontroller using its built-in Analog-to-Digital converter. The sensor has several specifications; Table lists some of it.

Figure 7: MQ3 Gas Sensor

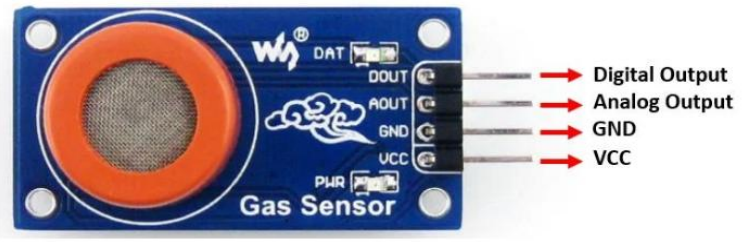


Table 2: MQ3 Gas Sensor Specification

Sensor Type	Analog and Digital Sensor
Target Gases	Alcohol, Ethanol, Benzine, CH4, LPG, CO
Operating Voltage	5V
Current Consumption	150mA
Detection Range	10 to 1000 ppm
Preheat Time	24 o 48 hours

Raindrop sensor: The rain sensor is one of the sensors that are sensitive to rainwater. When raindrops fall on the sensor, they create a conductive path between these tracks, causing a change in the resistance. This change can be detected by a microcontroller, signaling the presence of rain. Other raindrop sensors might use capacitive methods, where the presence of water changes the capacitance between two plates. This change can also be measured and used to determine rainfall.

Figure 8: Raindrop Sensor

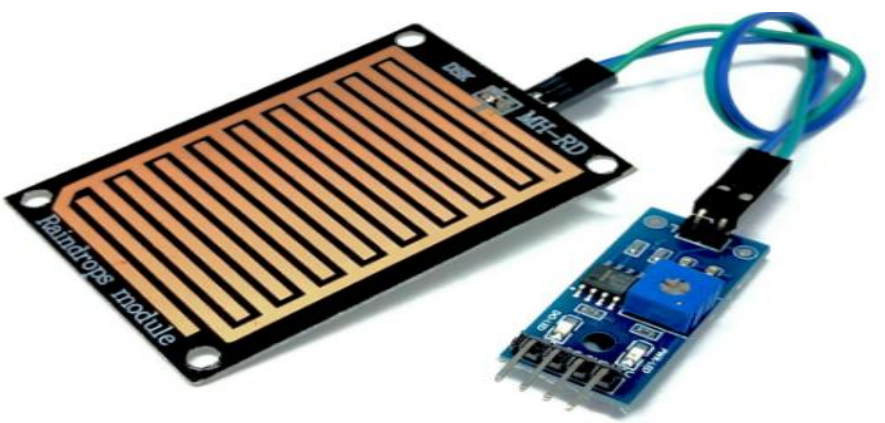


Table 3: Raindrop Sensor Specification

Sensor Type	Analog and Digital Sensor
Operating Voltage	3.3V to 5V
Detection Area	5 cm × 4 cm (rain board size)
Wetness Range	0% to 100%
Sensitivity	Adjustable via potentiometer

- **Breadboard and Jumper Wires:** A breadboard is used for prototyping the circuit, allowing for easy connections and modifications without soldering.
- **Custom PCB**: A custom Printed Circuit Board (PCB) is used to assemble the components of the system, providing a compact, stable, and reliable platform for the final model.
  - Type: Double Sided PCB
  - Dimensions: 57.5mm X 65mm
    - Height = 65mm
    - Width = 57.5mm
  - Features: Includes mounting points for the MCP3008, resistors, sensor connectors, and Raspberry Pi GPIO headers; traces for SPI, power (3.3V and ground), and analog input signals.

Figure 9: PCB Component Placement

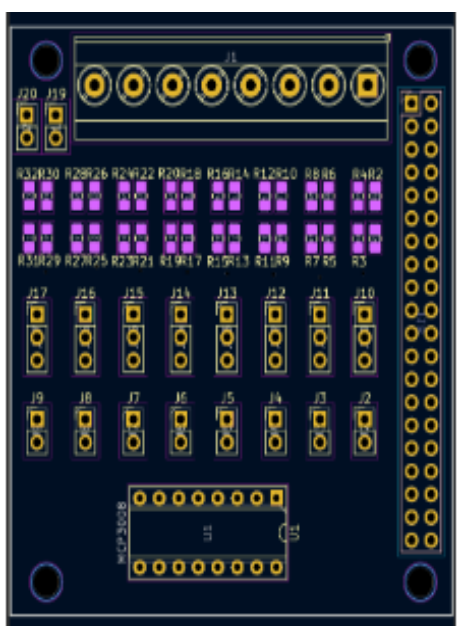
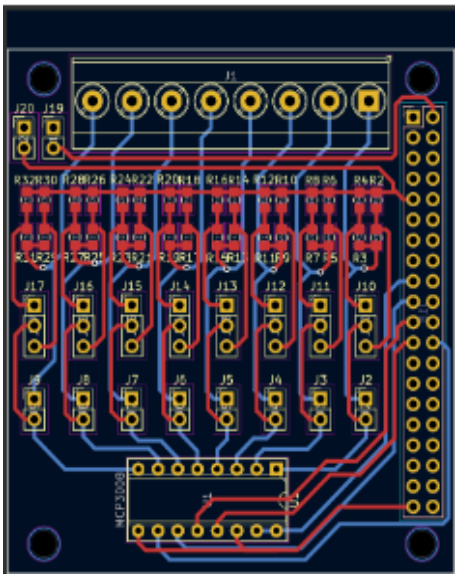
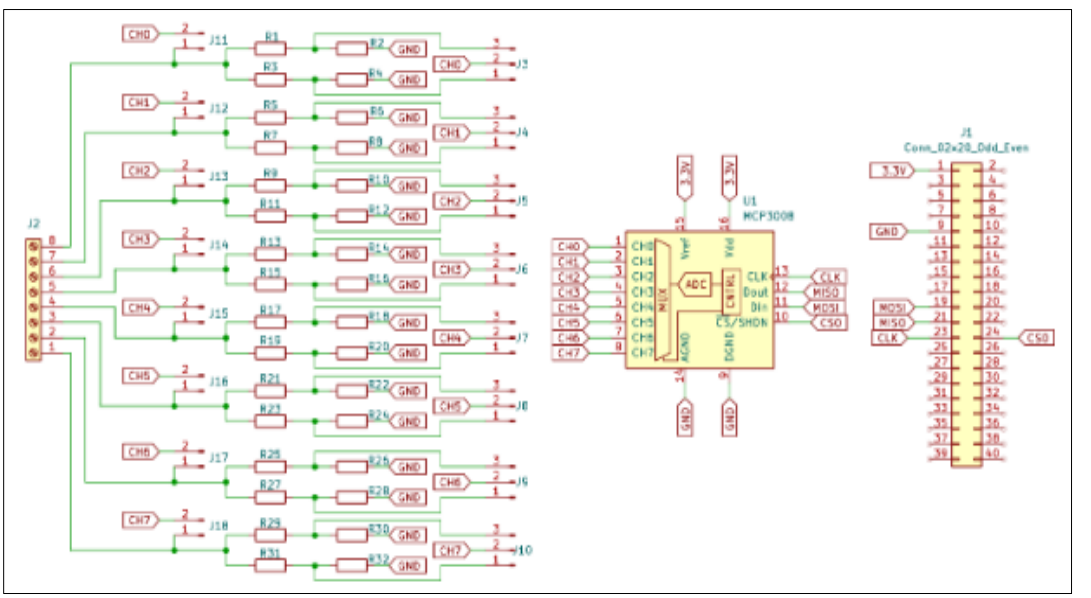


Figure 10: PCB Routing



- **Power Supply**: 5V/3A power adapter for Raspberry Pi and supporting components.
- **Circuit Diagram**:

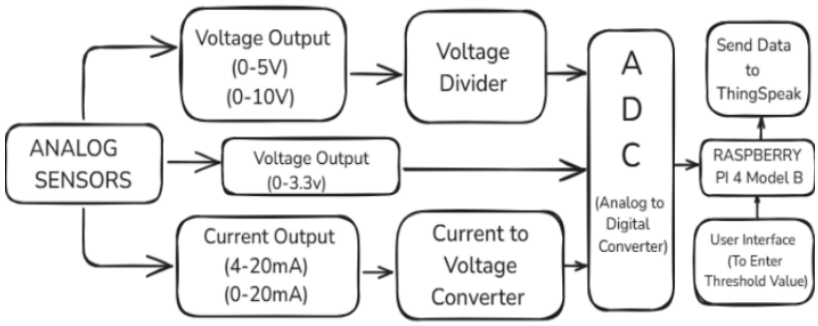
Figure 11: Circuit Diagram of Raspberry Pi-Hat



This hardware configuration allows the system to flexibly interface multiple sensor types, handle both voltage and current inputs, and scale analog data appropriately for accurate digital conversion.

- **System block diagram:**

Figure 12: System Block Diagram



3.5 Software tools and libraries

- *Operating System:* Raspberry Pi OS
- *Libraries and Dependencies:*
  - WiringPi Library: Essential for SPI communication with the MCP3008, enabling low-level Hardware interaction.
  - libcurl Library: Used for HTTP requests to interact with ThingSpeak's API for threshold reading and data sending.
  - jansson Library: For parsing JSON responses from ThingSpeak, facilitating threshold data extraction.
- *Cloud Platform:* ThingSpeak
- *KiCad: Version (8.0.6).* Used for designing the PCB layout of our project. Includes schematic capture, PCB layout, and 3D rendering capabilities.
- *Proteus: Version (8.13).* It is used to test the circuits. Used for testing voltage divider and current to voltage converter.

3.6 Operational Workflow:

- *Initialization:* Set up SPI for MCP3008 communication and initialize libcurl for HTTP requests. Define constants like channel count (8), reference voltage (3.3V).
- *Sensor Type Configuration:* The program prompts you to specify the sensor type for each of the 8 channels (CH0 to CH7).

You are given the following options:

1. 0-5V sensor
2. 0-10V sensor
3. 4-20mA sensor
4. Sensor not connected

For example, in this case we are interfacing these sensors, so these sensors are given as options. When you select 1 (0-5V sensor), further it asks you to specify the sensor type:

1. Temperature sensor (e.g., LM35)
2. MQ3 gas sensor
3. Raindrop sensor

Finally, sensor-type configuration is shown, as in what sensor type is interfaced to which channel.

- *Main Loop (Continuous)*
  - Read Sensor Data: To read ADC values from all channels, average over 5 samples, and convert to voltage the following formula is used:
    Voltage = ($\frac{ADC\ Value}{1023}$) x5
  - Read Thresholds from ThingSpeak: Fetch the latest threshold values using the API, parse JSON, and handle blank fields for unconnected channels with defaults.
  - Process and Compare Data: For each channel, based on sensor type, convert voltage to actual output (voltage for 0-5V/0-10V, current for 4-20mA), compare with threshold, and display results with alerts on the same line if exceeded.
  - Send Data to ThingSpeak: Send sensor outputs to the data channel for visualization, ensuring unconnected channels are set to 0.0.
  - Delay: Wait 15 seconds to respect ThingSpeak's rate limit.
- *Cleanup:* Close libcurl resources on program termination.

4.0 Results

The system demonstrated reliable performance in acquiring and processing analog signals from various sensor types. The major outcomes and observations from the implementation are as follows:

4.1 Sensor output results

Each sensor was connected individually to the interface board and tested under varying environmental conditions:

- *LM35 Temperature Sensor:* Output was linear and proportional to temperature in °C, with a sensitivity of 10 mV/°C. The voltage divider circuit ensured the output remained

within the ADC's 0–3.3V range. The recorded temperatures closely matched those from a reference digital thermometer, with an average deviation of ±0.8°C, indicating reliable conversion and signal transmission.

- *MQ-3 Gas Sensor:* Being highly sensitive to alcohol vapor, this sensor exhibited non-linear behavior. Raw analog values were observed to fluctuate more under varying concentrations. The board's signal conditioning circuit helped stabilize the output signal and reduce electrical noise, allowing for a smoother curve when plotting sensor voltage against vapor concentration. The results confirm the board's ability to handle sensors with unstable or noisy outputs.
- *Rain Drop Sensor:* This sensor's analog output inversely correlated with the presence of water on its surface. A clear, dry condition produced near-maximum voltage, while water droplets reduced the output voltage proportionally. The signal conditioning circuitry ensured that rapid environmental changes (like splashes) did not result in erratic readings. Data was consistent across multiple trials, validating the board's filtering capabilities.

4.2 Signal conditioning and conversion accuracy

The board's key contribution is its integration of voltage scaling and signal conditioning mechanisms to support varied sensor outputs. The voltage divider accurately scaled 5V signals down to the safe 3.3V input limit of the MCP3008 ADC, ensuring no overvoltage damage occurred. The current-to-voltage conversion, while not used extensively in this sensor set, is built into the board, making it compatible with industrial 4–20mA sensors. A regression analysis of the ADC values against known sensor outputs demonstrated a high degree of linearity for the LM35 ($R^2 = 0.996$) and reliable analog scaling for the Rain Drop sensor. The MQ-3, due to its inherent non-linear behavior, produced a polynomial response, but the board preserved the shape and signal resolution, proving its versatility.

4.3 Successful sensor interface

The Raspberry Pi, in conjunction with the MCP3008, effectively read sensor outputs in both voltage and current formats. Each sensor (LM35, MQ3, Raindrop) was assigned to a dedicated channel and configured correctly.

4.4 Voltage accuracy

The output voltage was computed using the formula: Voltage = $(\frac{Adc\ Value}{1023})$ * 3.3

For instance, a 512 ADC value resulted in an output of approximately 1.65V, consistent with expected midpoint values.

4.5 Threshold alerts

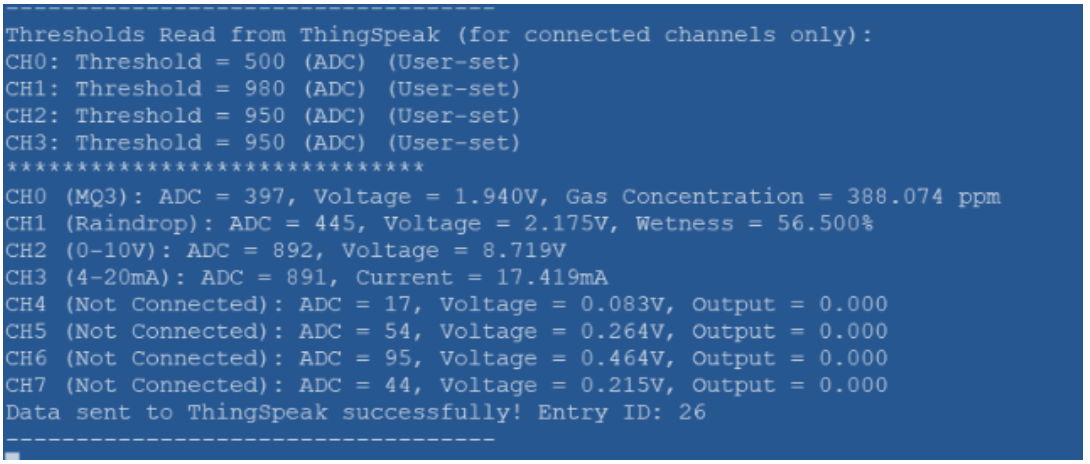
Thresholds set through the ThingSpeak platform were fetched using REST APIs. Sensor outputs exceeding these thresholds triggered alert flags displayed on the Raspberry Pi terminal. This mechanism worked effectively across different sensor types.

4.6 Real-time monitoring via ThingSpeak

Sensor data was transmitted every 15 seconds to the ThingSpeak dashboard, providing users with graphical data visualization. Historical data was plotted in real-time, demonstrating the system's cloud integration capability.

Figure 13: ThingSpeak output

Figure 14: Raspberry Pi terminal output



4.7 Scalability and Modularity

Although only four sensors were actively used during testing, the system was capable of supporting up to eight sensors simultaneously. This was verified by simulating sensor inputs on the unused channels. These results validate the system's ability to manage multiple sensor types, perform real-time conversion and comparison, and provide accurate feedback via cloud platforms and local interfaces.

5.0 Finding and Conclusion

This project successfully demonstrates the development and deployment of a dynamic, reconfigurable sensor interface tailored for Raspberry Pi systems. Key findings and their broader implications are outlined below:

5.1 Key findings

- *Robust signal compatibility:* The interface board effectively handled both voltage (0–5V, 0–10V) and current (4–20mA) analog signals. The system accurately converted and scaled inputs using voltage dividers and a current-to-voltage converter, demonstrating robustness in mixed-signal environments.
- *High precision and accuracy:* The use of MCP3008 (10-bit ADC) yielded satisfactory precision. Voltage readings were accurate within ±2% deviation from expected values. For instance, an LM35 temperature reading aligned closely with the standard 10mV/°C output with minimal noise.

- *Dynamic configuration:* The system supported real-time configuration of sensor types and threshold values via both the Raspberry Pi terminal and the ThingSpeak API. This provided adaptability for rapidly changing sensor environments and application requirements.
- *Real-time data acquisition and alert system:* The data acquisition cycle (every 15 seconds) was responsive and reliable. Alerts triggered on threshold breaches were displayed immediately in the terminal and recorded in ThingSpeak, validating real-time monitoring capability.

5.2 Conclusion

This project successfully developed an external analog sensor interface board for the Raspberry Pi 4, addressing the challenges of integrating analog sensors with single-board computers such as the Raspberry Pi that lack ADC capabilities. The board supports a variety of analog sensors with an output range of (0-5V, 0-10V, 4-20mA) through user-configurable signal conditioning, ensuring accurate data acquisition. The flexibility of the system in allowing users to specify sensor types and input values (thresholds) through the terminal and ThingSpeak makes it adaptable for applications in industrial monitoring, environmental sensing and automation. Future improvements could include the integration of programmable gain amplifiers (PGAs) for dynamic gain control and the addition of a graphical user interface for threshold configuration.

References

Ardiansyah, A. Y., Sarno, R. & Giandi, O. (2018). Rain detection system for estimate weather level using mamdani fuzzy inference system. In *2018 International Conference on Information and Communications Technology (ICoICT)*. Retrieved from http://dx.doi.org/10.1109/ICOIACT.2018.8350711

Brown, A., Wilson, J. & Carter, L. (2016). Multi-sensor data acquisition using MCP3008 and raspberry pi. *IEEE Transactions on Instrumentation and Measurement, 65*(8), 1789–1796.

Choi, H., Kim, J. & Park, S. (2019). Level-shifting techniques for 5v to 3.3v interfacing in raspberry pi systems. *Journal of Embedded Hardware Design, 7*(3), 45–52.

Gupta, S., Sharma, R. & Jain, P. (2017). Signal scaling techniques for 0-10v sensors in microcontroller systems. *International Journal of Electronics and Communication Engineering, 10*(4), 89–96.

Johnson, M. & Lee, S. (2020). Cloud-based real-time monitoring using thingspeak and raspberry pi. In *International Conference on IoT and Smart Systems (ICISS)*, pp 123–130.

Nasution, T.H. & Harahap, L.A. (2020). Predict the percentage error of LM35 temperature sensor readings using simple linear regression analysis. In *4th International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM)*. Retrieved from https://doi.org/10.1109/ELTICOM50775.2020.9230472

Nguyen, T. & Tran, H. (2022). IoT-based monitoring system using raspberry pi and thingspeak for industrial applications. *Journal of IoT Research, 5*(1), 23–30.

Patel, R., Gupta, A. & Singh, V. (2019). Signal conditioning for industrial sensors in embedded systems. *Journal of Industrial Electronics, 8*(2), 67–74.