**Analog Reader (ADS1115) Interface with Raspberry Pi**

**Introduction:**

The Raspberry Pi is a powerful single-board computer, but it lacks an inbuilt Analog-to-Digital Converter (ADC). This means it cannot directly read analog sensor values like temperature, humidity, or light intensity. To overcome this, an external ADC module like the ADS1115 is used.

The ADS1115 is a 16-bit ADC that allows the Raspberry Pi to interface with analog sensors and read their values digitally. It communicates with the Raspberry Pi via the I2C protocol, making it easy to integrate multiple sensors with minimal wiring.

This manual will guide you through the process of interfacing the ADS1115 ADC module with the Raspberry Pi. By following the steps, you will learn how to set up the hardware connections, understand the working principle, and explore the practical applications of analog-to-digital conversion in embedded systems.

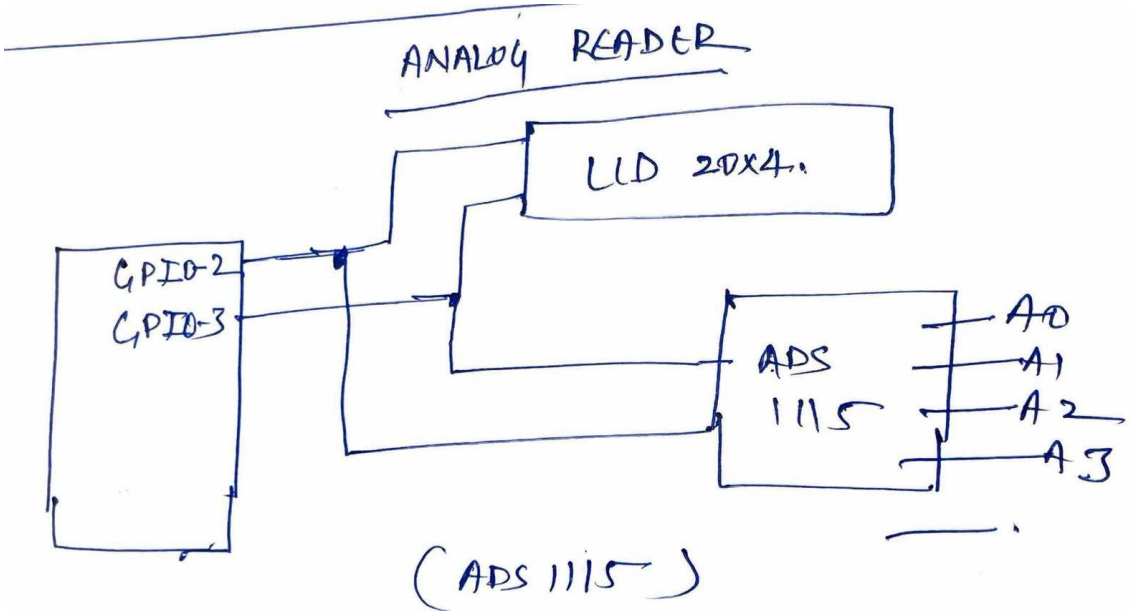
**Components Required:**

- Raspberry Pi (any model with I2C support)
- ADS1115 Analog-to-Digital Converter Module
- Analog Sensor (e.g., potentiometer, temperature sensor)
- Jumper Wires
- Breadboard (optional)


Fig 1. Analog Reader - ADS1115

**Circuit Connection:**

| ADS1115 Pins | Raspberry Pi |
|---|---|
| VCC | 3.3 V |
| GND | GND |
| SCL | GPIO 3 |
| SDA | GPIO 2 |
| A0, A1, A2, A3 | Analog Sensor Output |
| ADDR | GND or VCC |




**Applications:**

- Temperature & Humidity Monitoring – Reading analog sensor values for environmental monitoring.
- Biomedical Applications – Monitoring bio-signals like ECG or EEG.
- IoT & Smart Home Systems – Integrating analog sensors for automation.
- Battery Voltage Monitoring – Checking battery levels in electronic devices.

**Learnings:**

- The importance of Analog-to-Digital Conversion (ADC) in embedded systems
- How to use the ADS1115 ADC module to interface analog sensors with the Raspberry Pi
- How the I2C communication protocol works and how it enables data transfer between devices
- The role of gain settings in adjusting sensor input ranges for accurate readings

**Conclusion:**

Interfacing an ADS1115 with a Raspberry Pi expands its capabilities by allowing it to read analog sensor data, making it useful for a variety of real-world applications like environment monitoring, automation, and IoT projects. This setup is essential for projects that require accurate sensor readings beyond digital inputs.

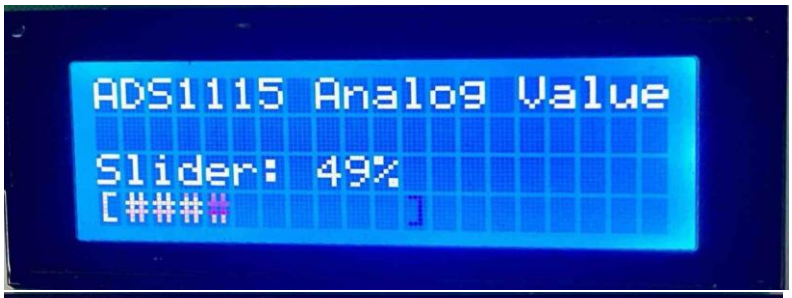
**Outcome:**

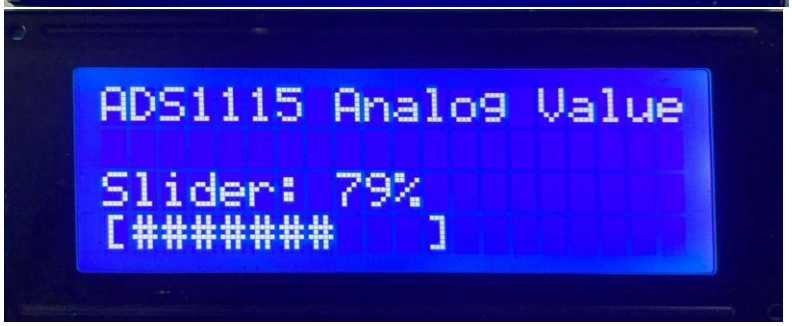

**Program:**

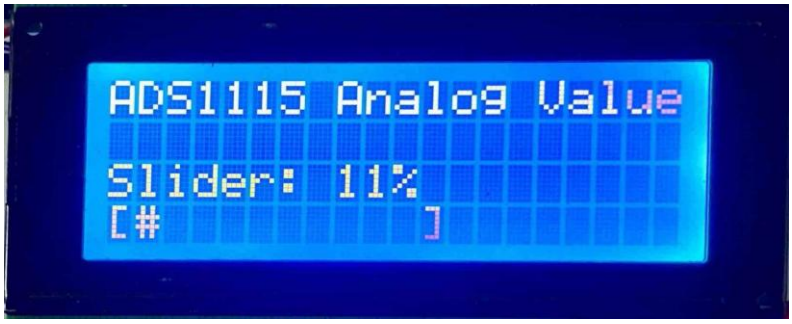
```
import time
import board
import busio
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
from RPLCD.i2c import CharLCD

# Initialize I2C bus and ADS1115
i2c = busio.I2C(board.SCL, board.SDA)
ads = ADS.ADS1115(i2c)

# Initialize LCD
lcd = CharLCD('PCF8574', 0x27)

while True:
    # Read analog values
    value1 = AnalogIn(ads, ADS.P0).value
    value2 = AnalogIn(ads, ADS.P1).value
    value3 = AnalogIn(ads, ADS.P2).value

    # Display on LCD
```

```
lcd.clear()
lcd.cursor_pos = (0, 0)
lcd.write_string("ADS1115 Analog Value\n")
lcd.cursor_pos = (1, 0)
lcd.write_string(f"Value 1 - {value1}\n")
lcd.cursor_pos = (2, 0)
lcd.write_string(f"Value 2 - {value2}\n")
lcd.cursor_pos = (3, 0)
lcd.write_string(f"Value 3 - {value3}")

time.sleep(1)
```

**For slider:**




**Program:**

```
import time
import board
import busio
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
from RPLCD.i2c import CharLCD

# Initialize I2C and ADS1115
i2c = busio.I2C(board.SCL, board.SDA)
ads = ADS.ADS1115(i2c)

# Initialize LCD
lcd = CharLCD('PCF8574', 0x27)
```

```
# ADC maximum value (16-bit ADS1115)
ADC_MAX = 32767

while True:
    # Read analog value from channel 0
    value = AnalogIn(ads, ADS.P0).value

    # Convert to percentage
    percentage = int((value / ADC_MAX) * 100)

    # Generate slider (10 blocks)
    slider_blocks = int((percentage / 10))
    slider = "[" + ("#" * slider_blocks).ljust(10) + "]"

    # Display on LCD
    lcd.clear()
    lcd.cursor_pos = (0, 0)
    lcd.write_string("ADS1115 Analog Value\n")
    lcd.cursor_pos = (2, 0)
    lcd.write_string(f"Slider: {percentage}%\n")
    lcd.cursor_pos = (3, 0)
    lcd.write_string(slider)

    time.sleep(1)
```