LDR Using Raspberry Pi

Introduction:

A Light Dependent Resistor (LDR) is a special type of resistor that changes its resistance based on the amount of light falling on it. When exposed to bright light, its resistance decreases, and in darkness, its resistance increases. This property makes LDRs useful for applications such as automatic night lamps, light intensity monitoring, and smart lighting systems.

In this manual, we will learn how to interface an LDR with a Raspberry Pi, which is a small, affordable, and powerful single-board computer. Since the Raspberry Pi does not have an inbuilt Analog-to-Digital Converter (ADC), we will use an MCP3008 ADC to read the analog values from the LDR and convert them into digital values that the Raspberry Pi can process.

By the end of this guide, you will be able to build a simple LDR-based light intensity monitoring system using a Raspberry Pi. This project will help you understand the basics of sensor interfacing, ADC communication, and Python programming on Raspberry Pi.

Components Required:

- Raspberry Pi (any model with GPIO support)
- LDR (Light Dependent Resistor)
- 10kΩ Resistor
- MCP3008 (ADC Converter)
- Breadboard
- Jumper Wires

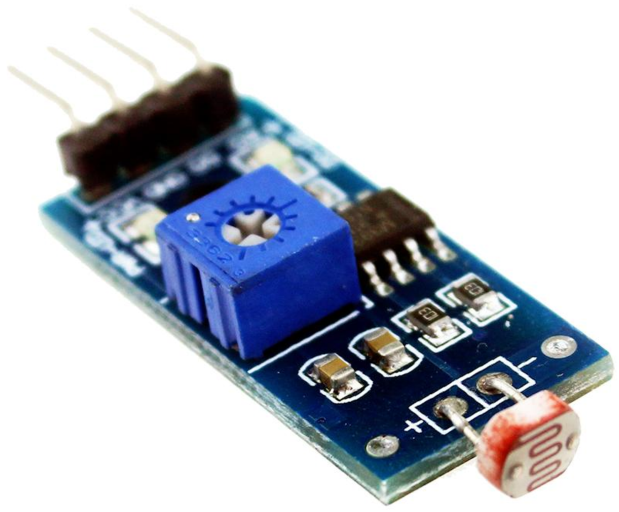
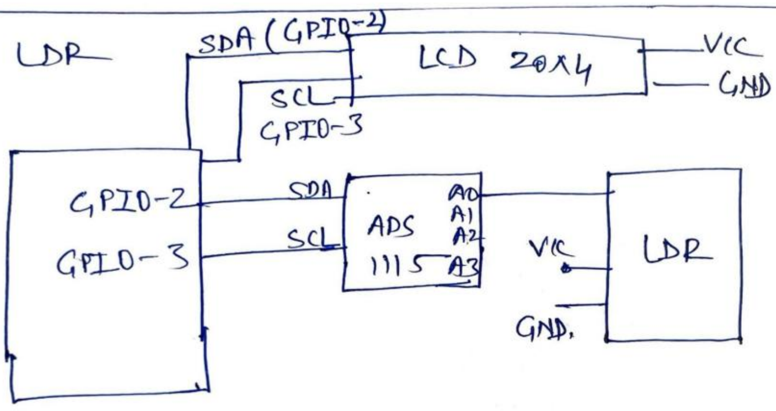


Fig 1. LDR with Raspberry Pi

Circuit Connection:

1. Connect one leg of the LDR to **3.3V (GPIO Pin 1)**.
2. Connect the other leg to **CH0 of MCP3008** and to a **10kΩ resistor**.
3. Connect the other end of the resistor to **GND**.
4. Connect the **MCP3008 to the Raspberry Pi**

MCP3008 Pins	**Raspberry Pi**
VDD	3.3 V
VREF	3.3 V
AGND	GND
CLK	GPIO 18
DOUT	GPIO 23
DIN	GPIO 24
CS	GPIO 25
DGND	GND



LDR

Applications:

- Automatic Street Lights – Turns lights ON/OFF based on ambient light.
- Smart Home Automation – Controls indoor lighting and curtains.
- Solar Tracking Systems – Adjusts solar panels for maximum sunlight.
- Weather Monitoring – Measures daylight intensity for climate analysis.

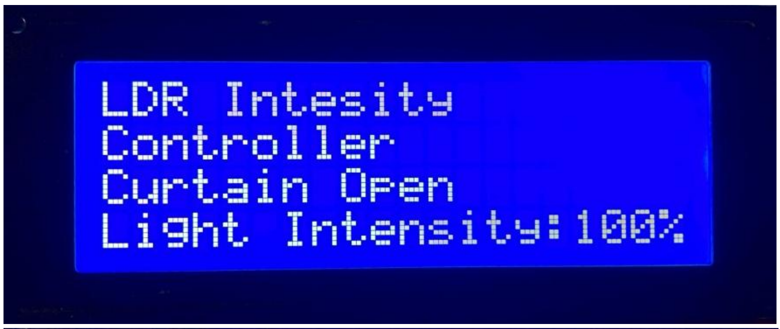
Learnings:

- How an LDR (Light Dependent Resistor) works and its real-world applications.
- The concept of voltage dividers for reading analog sensor values.
- How to use an MCP3008 ADC to interface analog sensors with the Raspberry Pi.
- Basics of SPI communication and how to read analog sensor data in Python.
- How to write a Python program to collect and process sensor data.

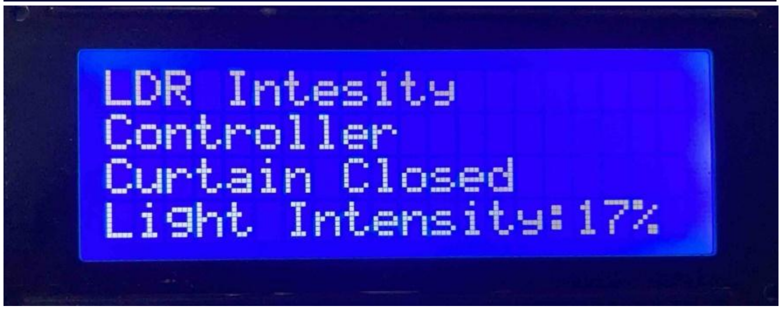
Conclusion:

This project provides a fundamental understanding of using sensors with Raspberry Pi. By learning how to interface an LDR using an MCP3008 ADC, you have gained practical knowledge of sensor integration, SPI communication, and Python scripting. You can extend this project further by adding features like automated light control, data visualization using Matplotlib, or integrating it with IoT platforms for remote monitoring.

Outcome:



```
LDR Intesity
Controller
Curtain Open
Light Intensity:100%
```



```
LDR Intesity
Controller
Curtain Closed
Light Intensity:17%
```

Program:

```
import time

import board

import busio

import adafruit_ads1x15.ads1115 as ADS

from adafruit_ads1x15.analog_in import AnalogIn

import RPi.GPIO as GPIO

from RPLCD.i2c import CharLCD

lcd = CharLCD('PCF8574', 0x27)


GPIO.setwarnings(False)


LED_Pin = 26


GPIO.setmode(GPIO.BCM)

GPIO.setup(LED_Pin, GPIO.OUT)
```

```
# Initialize I2C
i2c = busio.I2C(board.SCL, board.SDA)

# Initialize ADS1115
ads = ADS.ADS1115(i2c)

# Select channel A0 (where LDR is connected)
ldr_channel = AnalogIn(ads, ADS.P0)

# Define reference voltage (ADS1115 default is 4.096V)
V_REF = 4.096

try:
    while True:
        lcd.clear()
        lcd.cursor_pos = (0, 0)  # First row
        lcd.write_string(f'LDR Intesity')
        lcd.cursor_pos = (1, 0)  # First row
        lcd.write_string(f'Controller')

        voltage = ldr_channel.voltage  # Read voltage
        intensity = (voltage / V_REF) * 100  # Convert to percentage

        print(f"Light Intensity: {100 - intensity:.0f}%")
        lcd.cursor_pos = (3, 0)  # First row
        lcd.write_string(f"Light Intensity:{100 - intensity:.0f}%")
        if (100 - intensity) > 50:
            lcd.cursor_pos = (2, 0)  # First row
            lcd.write_string(f"Curtain Open")
            GPIO.output(LED_Pin, GPIO.HIGH)
        else:
            lcd.cursor_pos = (2, 0)  # First row
            lcd.write_string(f"Curtain Closed")
            GPIO.output(LED_Pin, GPIO.LOW)
        time.sleep(0.5)  # Small delay

except KeyboardInterrupt:
    print("\nExiting...")
```