SRM Institute of Science and Technology

College of Engineering and Technology

Department of Electronics and Communication Engineering

18ECO109J-Embedded System Design Using Raspberry Pi
2023-2024 (Odd Semester)

Mini Project Report

Name	**: CHINTAN A S**
Register No.	**: RA2111030010135**
Day Order	**: 3**
Venue	**: TP1117**
Project Title	**: "Light-Controlled LED with LDR and Raspberry Pi"**
Lab Supervisor	**: Dr. Kanaparthi / Dr. A. Ruhan Bevi**
Team Members	**: 1) ASHUTOSH DHAWAN (RA2111027010161)**
	2) CHINTAN A S (RA2111030010135)
	3) Sai Varun Guntur (RA2111029010018)

Particulars	**Max. Marks**	**Marks Obtained**
Objective & Description	05	
Algorithm,Flowchart,Program	20	
Demo verification	10	
Viva	10	
Report	05	
Total	**50**	

REPORT VERIFICATION

Date **:**

Staff Name **:**

Light-Controlled LED with LDR and Raspberry Pi

OBJECTIVE:

Create a light-controlled system using an LDR and a Raspberry Pi to control an LED based on ambient light levels.

ABSTRACT:

This project presents an engaging and instructive exploration of light sensing and automation, employing a Raspberry Pi single-board computer in tandem with a Light-Dependent Resistor (LDR) to create a responsive light-controlled LED system. The primary aim is to develop an adaptable setup that empowers an LED to dynamically respond to variations in ambient light, illuminating as the surroundings darken and dimming when light conditions improve. This project underlines the critical role of LDRs in light-sensing applications, elucidating their potential for driving real-world automation.

The hardware configuration is characterized by the seamless integration of an LDR and an LED into the Raspberry Pi ecosystem. A Python script interfaces with the GPIO pins of the Raspberry Pi to capture real-time light data from the LDR. The software-driven intelligence of the system then modulates the brightness of the LED to match the detected light intensity, ensuring that the LED adapts to the ever-changing luminous environment.

The dynamic interplay of the LDR, Raspberry Pi, and LED facilitates various practical applications. It can serve as an educational tool to elucidate the concept of light-dependent resistors and their roles in automation. The project also bears relevance in the context of energy-efficient lighting, where ambient light levels dictate artificial illumination. Moreover, it lays the foundation for more complex light-responsive systems in sectors like home automation, smart lighting, and industrial control, where environmental responsiveness is paramount.

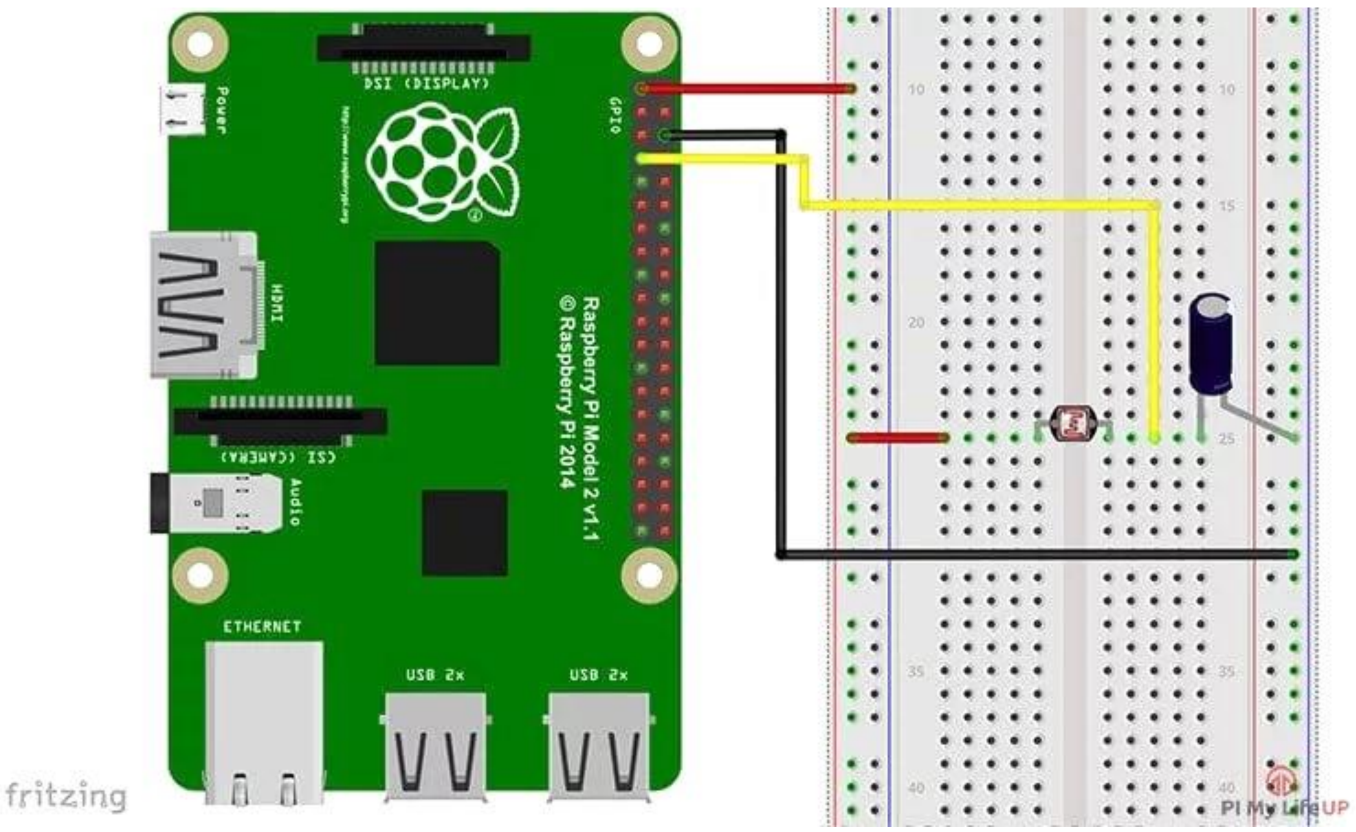
HARDWARE / SOFTWARE REQUIRED:

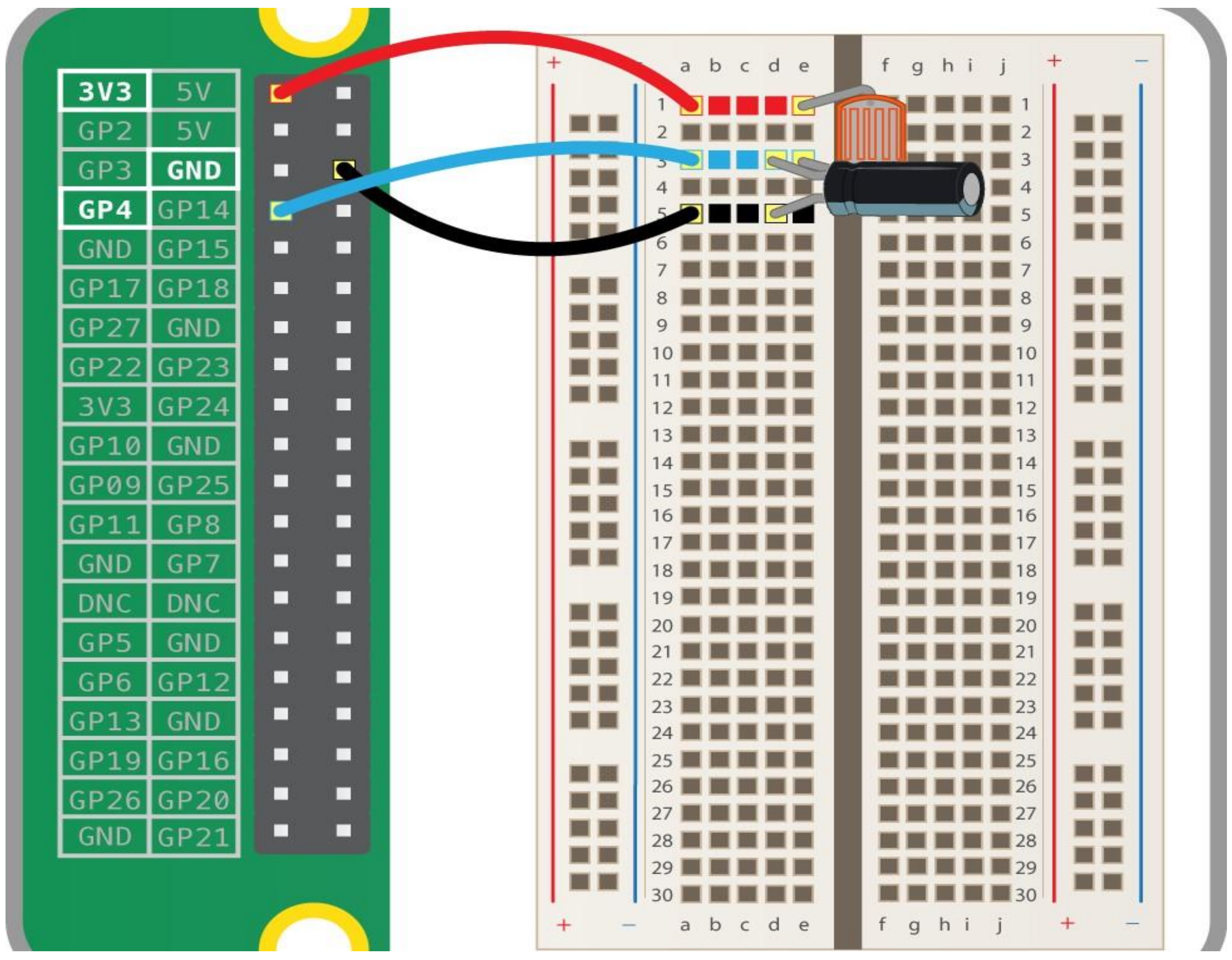
1. **Raspberry Pi:** Utilizing a Raspberry Pi 3 or a newer model, complete with the Raspbian or Raspberry Pi OS.
2. **Light-Dependent Resistor (LDR):** A commonly accessible analog light sensor**.**
3. **LED:** A Light-Emitting Diode, available in a range of colors.
4. **Resistor:** A 220Ω current-limiting resistor, though the exact value may vary based on the LED's specifications.
5. **Jumper Wires**: Employ male-to-female or male-to-male jumper wires for component interconnections.
6. **Breadboard:** A small breadboard may streamline the circuit assembly process**.**
7. **Raspberry Pi Operating System:** we'll need an operating system installed on your Raspberry Pi. Raspbian or Raspberry Pi OS is commonly used. Make sure your Raspberry Pi is up to date with the latest software updates.
8. **RPi.GPIO Library:** This Python library is used for controlling the GPIO pins on the Raspberry Pi.

BLOCK DIAGRAM/CONNECTION DIAGRAM:

- Attach one LDR terminal to a 3.3V Raspberry Pi pin.
- Link the other LDR terminal to a GPIO pin, such as GPIO17, on the Raspberry Pi.
- Connect the shorter (cathode) LED leg to a GND pin on the Raspberry Pi.
- Join the longer (anode) LED leg to a 220Ω current-limiting resistor.
- Complete the circuit by connecting the opposite end of the resistor to a GPIO pin, such as GPIO18, on the Raspberry Pi.

This dynamic system illustrates the principles of light sensing and adaptive lighting control, making it a valuable educational tool, a prototype for sustainable illumination, and a precursor to intricate light-responsive automation systems for various industrial and residential applications.

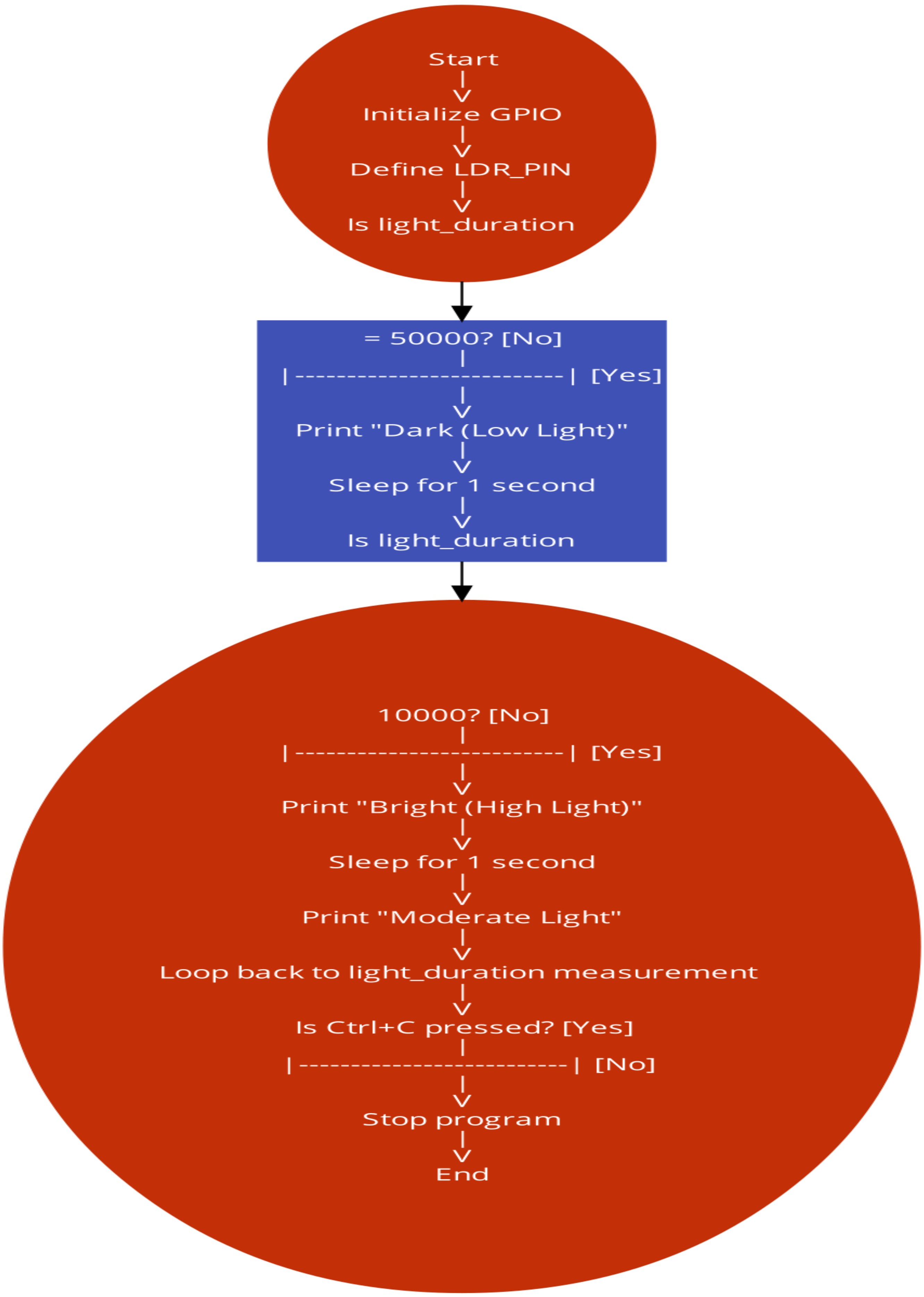

DSI (DISPLAY)
Power
GPIO
HDMI
CSI (CAMERA)
Audio
Raspberry Pi Model 2 v1.1
© Raspberry Pi 2014
ETHERNET
USB 2x
USB 2x
fritzing
PI My LifeUP


3V3 5V
GP2 5V
GP3 GND
GP4 GP14
GND GP15
GP17 GP18
GP27 GND
GP22 GP23
3V3 GP24
GP10 GND
GP09 GP25
GP11 GP8
GND GP7
DNC DNC
GP5 GND
GP6 GP12
GP13 GND
GP19 GP16
GP26 GP20
GND GP21
a b c d e f g h i j
+ −

ALGORITHM:

1. Import the necessary libraries for Raspberry Pi GPIO control and time management.
2. Set up the GPIO pin for the LDR according to the BCM pin numbering.
3. Define a function named **rc_time** that takes one parameter, the GPIO pin connected to the LDR. This function will measure the time taken for the LDR to go from low to high when exposed to light.
4. Inside the **rc_time** function, set the GPIO pin as an input and wait until the pin goes to a low state, indicating that it's dark.
5. Record the current time as the start time.
6. Continuously check the pin's state until it goes high (exposed to light). Record the current time as the end time.
7. Calculate the **light_duration** by subtracting the start time from the end time.
8. Return the **light_duration** to the calling function, representing the time taken for the LDR to go from low to high.
9. In the main program:
   - Call the **rc_time** function, passing the GPIO pin number connected to the LDR.
   - Get the **light_duration** from the function call.
   - Use the **light_duration** to determine the light level:
     - If the **light_duration** is high (e.g., 50K+), it indicates that there is little to no light, and it's dark.
     - If the **light_duration** is low (e.g., lower than 10K), it suggests there is a significant amount of light, and it's bright.
10. Print the light level to the console or perform further actions based on the detected light level.

FLOW CHART:

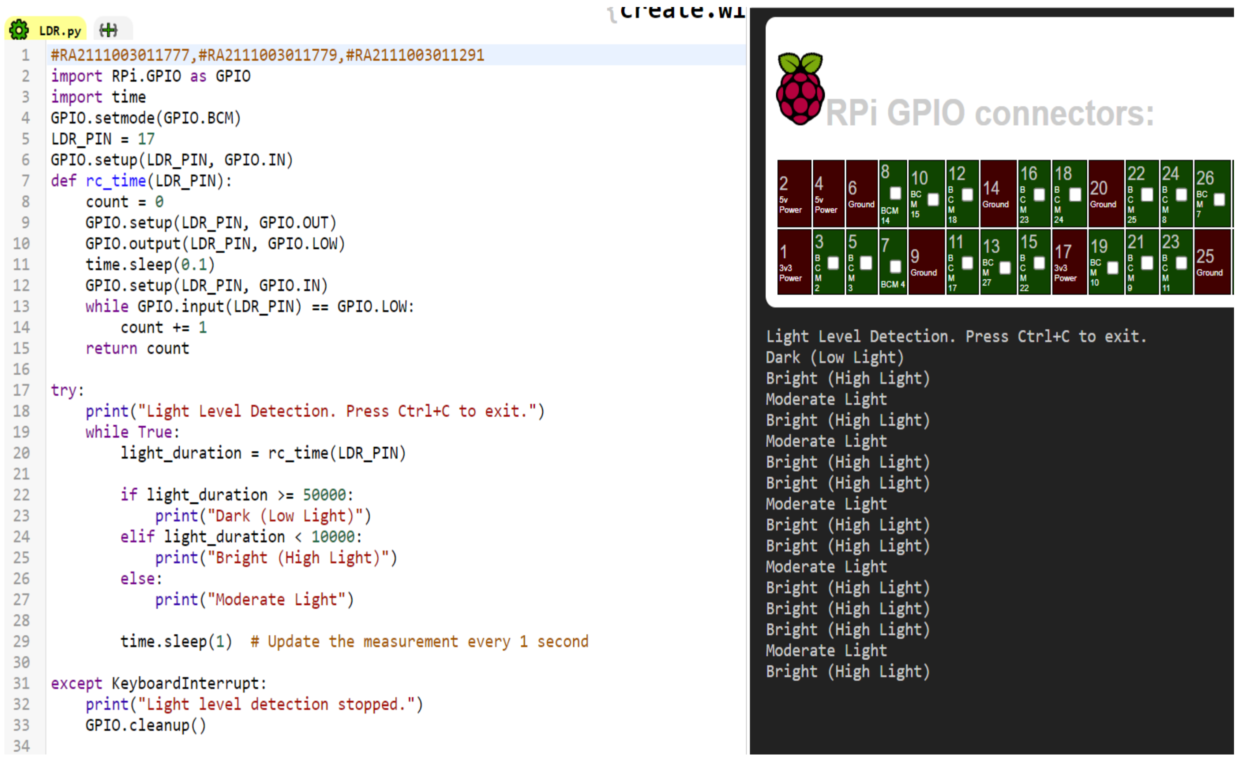


PROGRAM:

```
import RPi.GPIO as GPIO
import time
# Set GPIO mode
GPIO.setmode(GPIO.BCM)
# Define the GPIO pin for the LDR
LDR_PIN = 17  # Change to the appropriate GPIO pin
# Initialize GPIO settings
GPIO.setup(LDR_PIN, GPIO.IN)
# Define a function to measure light levels
def rc_time(LDR_PIN):
    count = 0
    GPIO.setup(LDR_PIN, GPIO.OUT)
    GPIO.output(LDR_PIN, GPIO.LOW)
    time.sleep(0.1)
    GPIO.setup(LDR_PIN, GPIO.IN)
    while GPIO.input(LDR_PIN) == GPIO.LOW:
        count += 1
    return count

try:
    print("Light Level Detection. Press Ctrl+C to exit.")
    while True:
        light_duration = rc_time(LDR_PIN)
        if light_duration >= 50000:
            print("Dark (Low Light)")
        elif light_duration < 10000:
            print("Bright (High Light)")
        else:
            print("Moderate Light")


        time.sleep(1)  # Update the measurement every 1 second
```

```
except KeyboardInterrupt:
    print("Light level detection stopped.")
    GPIO.cleanup()
```

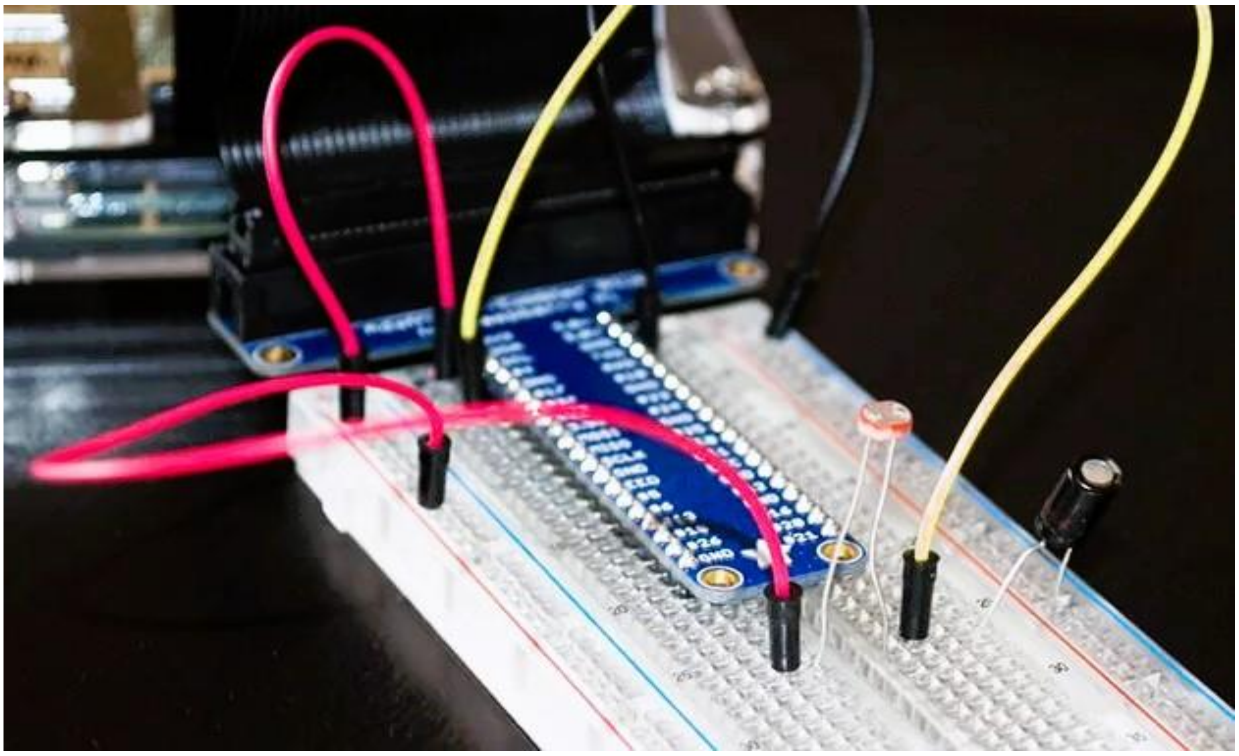
OUTPUT: SOFTWARE:



```
#RA2111003011777,#RA2111003011779,#RA2111003011291
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
LDR_PIN = 17
GPIO.setup(LDR_PIN, GPIO.IN)
def rc_time(LDR_PIN):
    count = 0
    GPIO.setup(LDR_PIN, GPIO.OUT)
    GPIO.output(LDR_PIN, GPIO.LOW)
    time.sleep(0.1)
    GPIO.setup(LDR_PIN, GPIO.IN)
    while GPIO.input(LDR_PIN) == GPIO.LOW:
        count += 1
    return count

try:
    print("Light Level Detection. Press Ctrl+C to exit.")
    while True:
        light_duration = rc_time(LDR_PIN)

        if light_duration >= 50000:
            print("Dark (Low Light)")
        elif light_duration < 10000:
            print("Bright (High Light)")
        else:
            print("Moderate Light")

        time.sleep(1)  # Update the measurement every 1 second

except KeyboardInterrupt:
    print("Light level detection stopped.")
    GPIO.cleanup()
```

```
Light Level Detection. Press Ctrl+C to exit.
Dark (Low Light)
Bright (High Light)
Moderate Light
Bright (High Light)
Moderate Light
Bright (High Light)
Bright (High Light)
Moderate Light
Bright (High Light)
Bright (High Light)
Moderate Light
Bright (High Light)
Bright (High Light)
Bright (High Light)
Moderate Light
Bright (High Light)
```

HARDWARE:



REAL TIME APPLICATION:

Light-Dependent Resistors (LDRs) are commonly used in various real-time applications where the intensity of light needs to be monitored and controlled. Here are some real-time applications for LDRs:

1. **Automatic Outdoor Lighting:** LDRs can be used in outdoor lighting systems to automatically turn on streetlights, garden lights, or security lights when it gets dark and turn them off when there's sufficient natural light. This helps save energy and provides safety.
2. **Photography:** In cameras, LDRs are often used to measure the ambient light and adjust the camera settings such as shutter speed and aperture to capture well-exposed photographs.
3. **Solar Tracking:** In solar panel installations, LDRs are used to track the movement of the sun. Solar panels can be adjusted to face the sun directly, optimizing energy generation.
4. **Weather Stations:** LDRs are used in weather stations to measure the intensity of sunlight. This data is valuable for forecasting weather conditions and solar radiation.

CONCLUSION:

In conclusion, Light-Dependent Resistors (LDRs) are versatile sensors that play a crucial role in real-time applications across various domains. Their ability to detect and respond to changes in ambient light levels makes them invaluable in energy conservation, automation, security, and artistic expression. Whether it's adjusting outdoor lighting, enhancing photography, optimizing solar power generation, or creating interactive art installations, LDRs provide solutions that improve efficiency, safety, and user experience.

References:

Raspberry Pi Foundation (https://www.raspberrypi.org/) , "Practical Electronics for Inventors" by Paul Scherz and Simon Monk,Adafruit Learning System (https://learn.adafruit.com/), "Raspberry Pi Cookbook" by Simon Monk.