버튼

버튼

버튼은 눌림(1, HIGH) / 안 눌림(0, LOW) 두 상태만 가짐

라즈베리파이는 이를 GPIO 핀을 통해 감지

풀업(Pull-up) / 풀다운(Pull-down) 저항
- 풀다운: 평소 0V(LOW), 누를 때 3.3V(HIGH)
- 풀업: 평소 3.3V(HIGH), 누를 때 0V(LOW)

버튼

```
import RPi.GPIO as GPIO
import time

# 사용할 GPIO 핀 번호 설정 (BCM 기준)
button_pin = 15

# GPIO 기본 설정
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)  # BCM 번호 모드 사용

# 버튼 핀을 입력으로 설정하고, 내부 풀다운 저항 사용
GPIO.setup(button_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

print("버튼 입력 감지를 시작합니다. (종료: Ctrl + C)")

try:
    while True:
        # 버튼이 눌렸는지 확인 (HIGH 상태이면 눌림)
        if GPIO.input(button_pin) == GPIO.HIGH:
            print("Button pushed!")
            time.sleep(0.2)  # 채터링 방지용 딜레이
        time.sleep(0.1)

except KeyboardInterrupt:
    print("₩n프로그램을 종료합니다.")
    GPIO.cleanup()  # GPIO 설정 초기화
```

버튼(이벤트기반)

```
import RPi.GPIO as GPIO
import time

# 버튼이 눌릴 때 실행될 콜백 함수 정의
def button_callback(channel):
    print("Button pressed!")

# --- GPIO 기본 설정 ---
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)  # BCM 핀 번호 방식 사용

# 사용할 버튼 핀 설정
button_pin = 15
GPIO.setup(button_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  # 풀다운 입력 설정
```

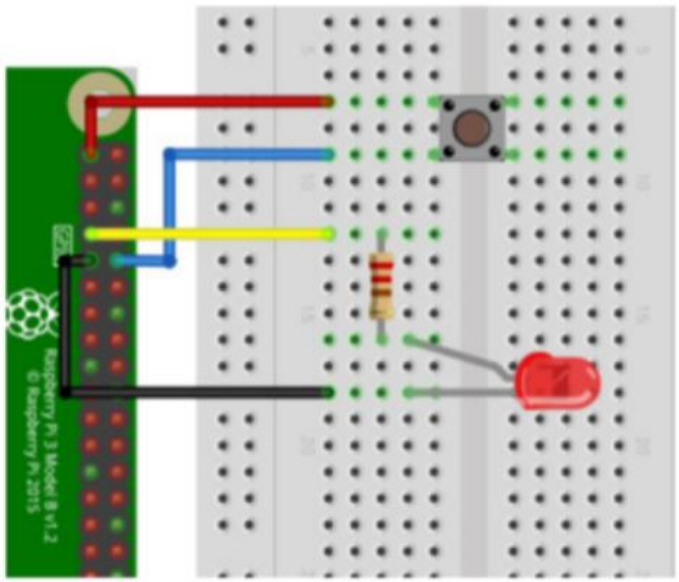
버튼(이벤트기반)

```
# --- 이벤트 감지 방식 (인터럽트 기반) ---
GPIO.add_event_detect(button_pin, GPIO.RISING, callback=button_callback, bouncetime=200)

print("버튼 이벤트 감지를 시작합니다. (종료: Ctrl + C)")

# --- 메인 루프 ---
try:
    while True:
        time.sleep(0.1)  # 프로그램 유지용 (CPU 점유율 방지)
except KeyboardInterrupt:
    print("₩n프로그램을 종료합니다.")
    GPIO.cleanup()  # GPIO 설정 초기화
```

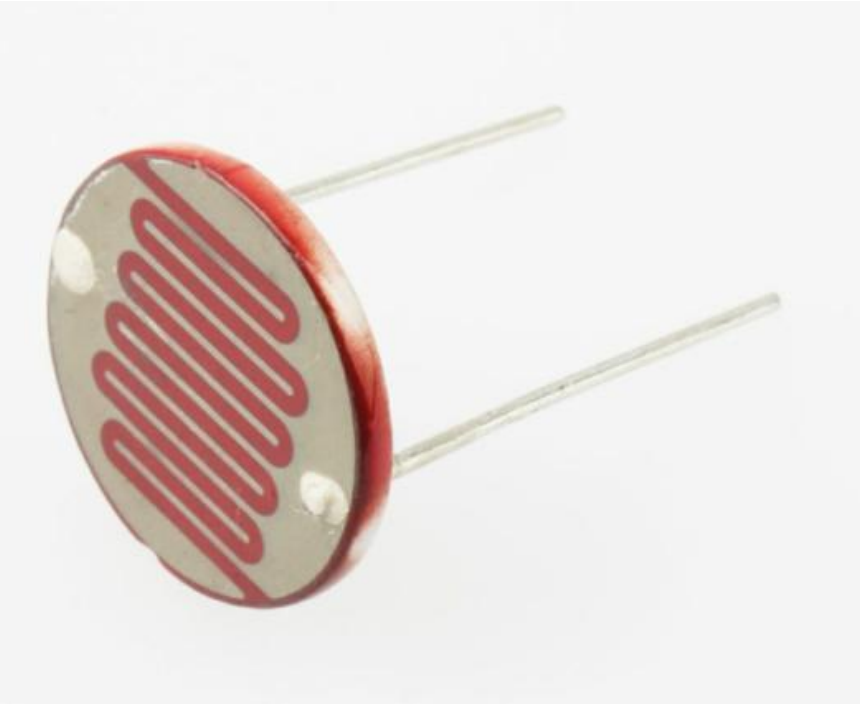
버튼



라즈베리 파이	스위치
GPIO 15	스위치 연결
VCC(3.3V)	스위치 연결
GPIO 4	저항
GND	LED –

조도센서

조도센서



빛의 양을 측정하는 센서도 빛센서라고 부르기도 함
조도센서는 주변 밝기를 측정하여 입력 값으로 보내는 센서
조도센서는 조도에 따라 저항의 값이 변화
빛의 밝기(세기)에 따라 저항값이 변화
빛이 밝으면 저항값이 감소하고 어두우면 저항이 증가
별도의 극성이 없음

헤 드 라 이 트

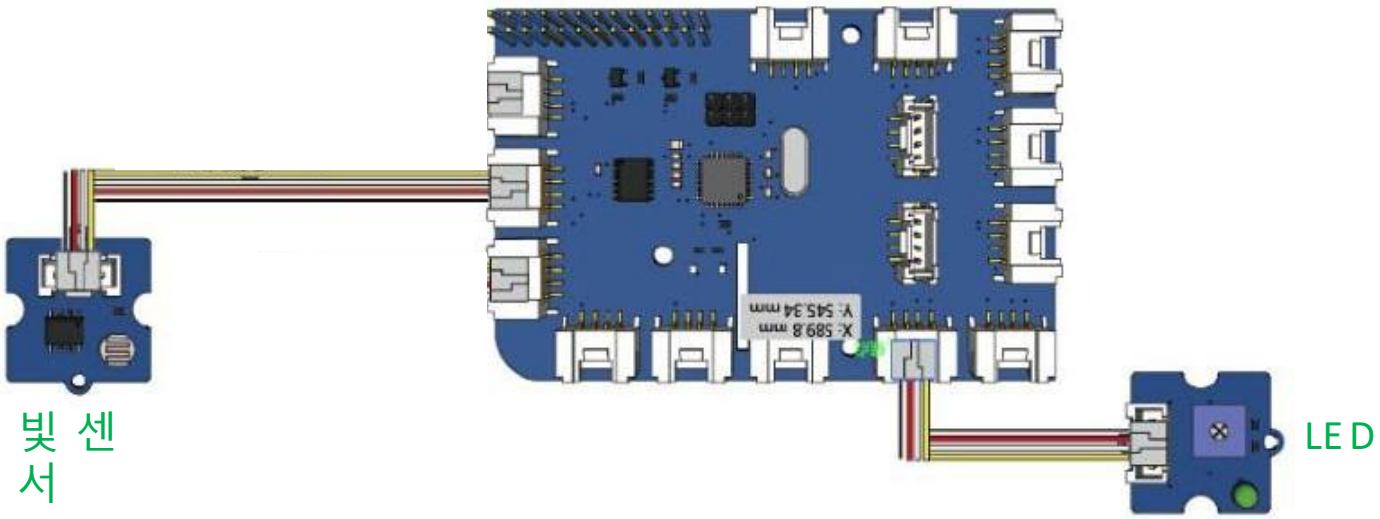
- 센서가 빛의 양을 감지하여 3단계로 구분하여 작동
- 밝은날 : 꺼짐
- 저무는 시간 : 미등
- 어두운 시간 : 전조등

- 휴대폰 상단의 주변 조명 센서에서 감지된 주변 조명 조건에 따라 화면 밝기가 자동 변경

가로등

- 가로등 자동 점멸

조도센서



소스코드-PMW LED

```
import time
import grovepi

led = 5

# Digital ports that support Pulse Width Modulation (PWM)
# D3, D5, D6

# Digital ports that do not support PWM
# D2, D4, D7, D8

grovepi.pinMode(led,"OUTPUT")
time.sleep(1)
i = 0
```

소스코드-PMW LED

```
while True:
    try:
        # Reset
        if i > 255:
            i = 0

        print (i)
        grovepi.analogWrite(led,i)

        # Increment brightness for next iteration
        i = i + 20
        time.sleep(.5)

    except KeyboardInterrupt:
        grovepi.analogWrite(led,0)
        break
    except IOError:
        print ("Error")
```

소스코드 - 조도센서

```
import time
import grovepi

light_sensor = 0

led = 4

threshold = 10

grovepi.pinMode(light_sensor,"INPUT")
grovepi.pinMode(led,"OUTPUT")
```

소스코드 - 조도센서

```
while True:
    try:
        # Get sensor value
        sensor_value = grovepi.analogRead(light_sensor)

        # Calculate resistance of sensor in K
        resistance = (float)(1023 - sensor_value) * 10 / sensor_value

        if resistance > threshold:
            grovepi.digitalWrite(led,1)
        else:
            grovepi.digitalWrite(led,0)

        print("sensor_value = %d resistance = %.2f" %(sensor_value,  resistance))
        time.sleep(.5)

    except IOError:
        print ("Error")
```