임베디드 보드 실습과 응용 프로그램

Chapter 3.

– GPIO 입 · 출력, PWM 모듈, callback

최영근
010-5898-3202

GPIO

- **General Purpose Input Output**
  - 일반적인 용도의 입력, 출력
  - 입력 : 버튼, 각종 센서 등
  - 출력 : LED, 각종 모터 등

  - 사용하는 핀을 입력으로 사용할지, 출력으로 사용할지 C나 파이썬 언어 등 소프트웨어에 의해 설정

  - I2C, SPI, UART 등 통신 용도로 설정해서 사용 가능한 핀도 있으며, 이런 핀들은 하드웨어적으로 이미 설정되어 있음

  - 라즈베리 파이의 GPIO 입출력 전압은 3.3V로서 보호 회로가 따로 없기 때문에 5V가 인가되지 않도록 주의해야 함

GPIO

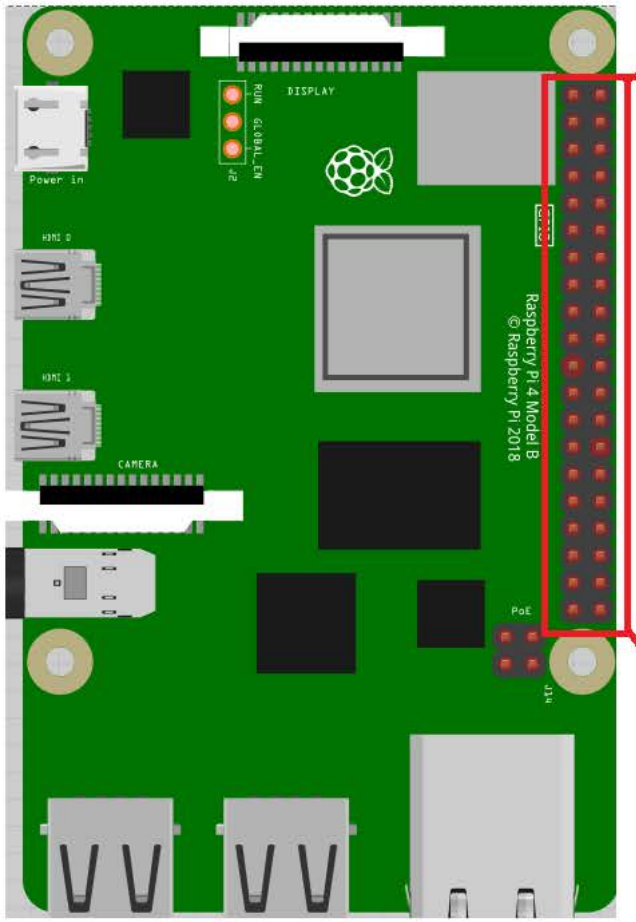
•RPi.GPIO 설치

- 라즈베리 파이의 GPIO 제어를 위해 사용되는 라이브러리는 다양함
- 가장 많이 이용되는 라이브러리는 파이썬용 라이브러리인 RPi.GPIO, C언어용 라이브러리인 wiringPi

- 라즈베리 파이 3 / 4의 GPIO 핀은 총 26개.
- 27번, 28번 핀은 reserved pin으로서 I2C-EEPROM 간의 통신에 사용

- 라즈베리 파이 OS 설치 시 RPi.GPIO 도 설치됨

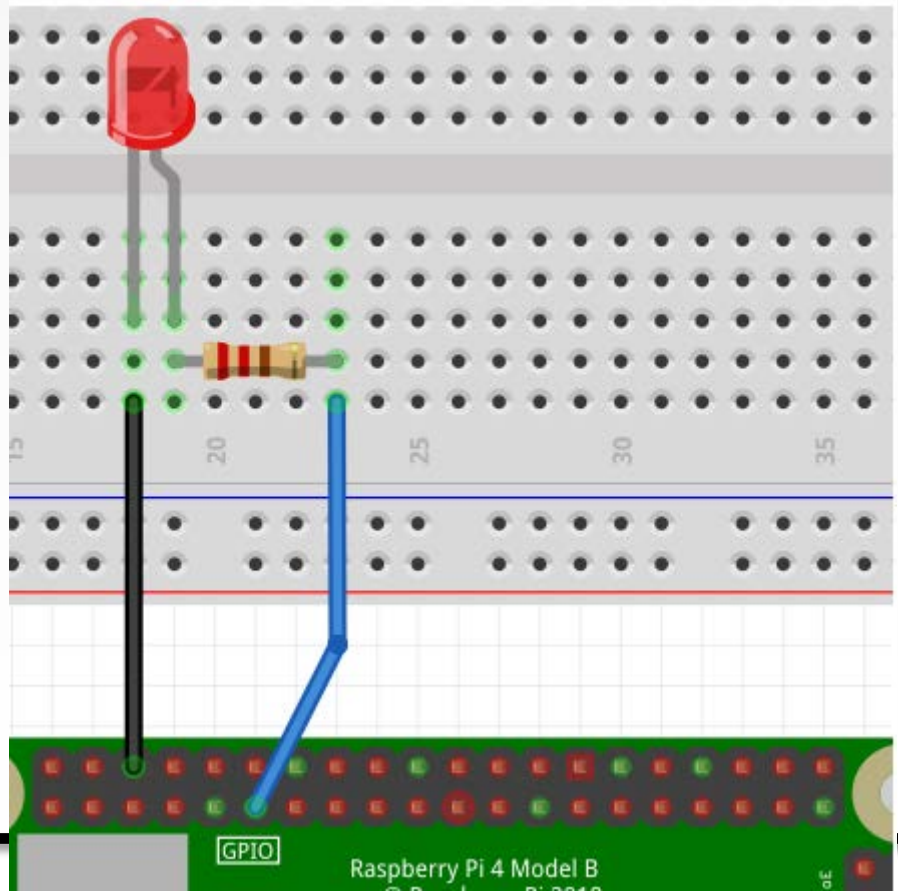
GPIO



wPi	BCM	Pin	No	No	Pin	BCM	wPi
		3.3V	1	2	5V		
GPIO 08	GPIO 02	SDA1	3	4	5V		
GPIO 09	GPIO 03	SCL1	5	6	GND		
GPIO 07	GPIO 04		7	8	TXD	GPIO 14	GPIO 15
		GND	9	10	RXD	GPIO 15	GPIO 16
GPIO 00	GPIO 17		11	12		GPIO 18	GPIO 01
GPIO 02	GPIO 27		13	14	GND		
GPIO 03	GPIO 22		15	16		GPIO 23	GPIO 04
		3.3V	17	18		GPIO 24	GPIO 05
GPIO 12	GPIO 10	MOSI	19	20	GND		
GPIO 13	GPIO 09	MISO	21	22		GPIO 25	GPIO 06
GPIO 14	GPIO 11	SCLK	23	24	CE0	GPIO 08	GPIO 10
		GND	25	26	CE1	GPIO 07	GPIO 11
GPIO 30	GPIO 00	SDA0	27	28	SCL0	GPIO 01	GPIO 31
GPIO 21	GPIO 05		29	30	GND		
GPIO 22	GPIO 06		31	32		GPIO 12	GPIO 26
GPIO 23	GPIO 13		33	34	GND		
GPIO 24	GPIO 19		35	36		GPIO 16	GPIO 27
GPIO 25	GPIO 26		37	38		GPIO 20	GPIO 28
		GND	39	40		GPIO 21	GPIO 29

LED

- **결선**
  - BCM 17 번 핀 – 220옴 저항 – LED - GND



LED 제어

- LED on

```
import RPi.GPIO as gpio          # RPi.GPIO 모듈을 gpio라는 이름으로 불러옴
led_pin =17                      # BCM GPIO 17번에 연결
gpio.setmode(gpio.BCM)           # GPIO.setmode 함수 호출 - BCM 핀 번호 사용
gpio.setup(led_pin, gpio.OUT)    # BCM GPIO 17번 핀을 출력으로 설정
gpio.output(led_pin, True)       # LED on
try:
        while True:              # 평상시에는
                pass             # 아무것도 안 함 == LED on 유지
except KeyboardInterrupt:        # Ctrl + C 누르면 while문 종료. 키보드 인터럽트
        pass
gpio.cleanup()                   # GPIO 핀의 상태를 초기화
```

LED 제어

- LED off

```
import RPi.GPIO as gpio
import time                    # sleep 함수 사용을 위해 time 모듈을 import

led_pin =17

gpio.setmode(gpio.BCM)

gpio.setup(led_pin, gpio.OUT)

gpio.output(led_pin, True)
time.sleep(2.0)                # LED on한 다음 2초 delay
gpio.output(led_pin, False) # LED off

gpio.cleanup()
```

LED 제어

- **LED flicker**
  - 주파수 1Hz로 점멸

```
import RPi.GPIO as gpio
import time

led_pin =17

gpio.setmode(gpio.BCM)

gpio.setup(led_pin, gpio.OUT)

try:
        while True:
                gpio.output(led_pin, True)
                time.sleep(0.5)                 # LED on 후 0.5sec delay
                gpio.output(led_pin, False)
                time.sleep(0.5)                 # LED off 후 0.5sec delay
except KeyboardInterrupt:
        pass

gpio.cleanup()
```

LED 제어

- **LED flicker**
  - 주파수 10Hz로 점멸

```
import RPi.GPIO as gpio
import time

led_pin =17

gpio.setmode(gpio.BCM)

gpio.setup(led_pin, gpio.OUT)

try:
        while True:
                gpio.output(led_pin, True)
                time.sleep(0.05)                    # LED on 후 0.05sec delay
                gpio.output(led_pin, False)
                time.sleep(0.05)                    # LED off 후 0.05sec delay
except KeyboardInterrupt:
        pass

gpio.cleanup()
```

LED 제어

•LED 밝기 조절

- 주파수 100Hz로 점멸
- 일반적으로 43Hz 이상의 주파수로 LED가 점멸되면 눈으로 인식하기 어려움

```
import RPi.GPIO as gpio
import time

led_pin =17

gpio.setmode(gpio.BCM)

gpio.setup(led_pin, gpio.OUT)

try:
        while True:
                gpio.output(led_pin, True)
                time.sleep(0.005)                # LED on 후 0.005sec delay
                gpio.output(led_pin, False)      # LED off 후 0.005sec delay
                time.sleep(0.005)
except KeyboardInterrupt:
        pass

gpio.cleanup()
```

LED 제어

- **LED 밝기 조절**
  - 주파수 100Hz로 점멸
  - (HIGH:LOW)의 비가 1:9일 경우 LED의 밝기는?

```
import RPi.GPIO as gpio
import time

led_pin =17

gpio.setmode(gpio.BCM)

gpio.setup(led_pin, gpio.OUT)

try:
        while True:
                gpio.output(led_pin, True)
                time.sleep(0.001)                    # LED on 후 0.001sec delay
                gpio.output(led_pin, False)
                time.sleep(0.009)                    # LED off 후 0.009sec delay
except KeyboardInterrupt:
        pass

gpio.cleanup()
```

LED 제어

•LED 밝기 조절

- 주파수 100Hz로 점멸
- 밝기 변화를 눈으로 확인하기 어려움

```
import RPi.GPIO as gpio
import time

led_pin =17

gpio.setmode(gpio.BCM)

gpio.setup(led_pin, gpio.OUT)

try:
        while True:
                for t_high in range(0,11):                  # 0~10까지 1씩 증가
                        gpio.output(led_pin, True)
                        time.sleep(t_high*0.005)            # 0~0.05까지 0.005씩 증가
                        gpio.output(led_pin, False)
                        time.sleep((10-t_high)*0.005)       # 0.05~0까지 0.005씩 감소
except KeyboardInterrupt:
        pass

gpio.cleanup()
```

LED 제어

- LED 밝기 조절

```
import RPi.GPIO as gpio
import time

led_pin =17

gpio.setmode(gpio.BCM)

gpio.setup(led_pin, gpio.OUT)

try:
        while True:
                for t_high in range(0,11):      # t_high 값이 유지되는 시간을
                        cnt =0                  # 늘려주기 위함
                        while True:
                                gpio.output(led_pin, True)
                                time.sleep(t_high*0.005)
                                gpio.output(led_pin, False)
                                time.sleep((10-t_high)*0.005)

                                cnt +=1
                                if cnt==10: break
except KeyboardInterrupt:
        pass

gpio.cleanup()
```

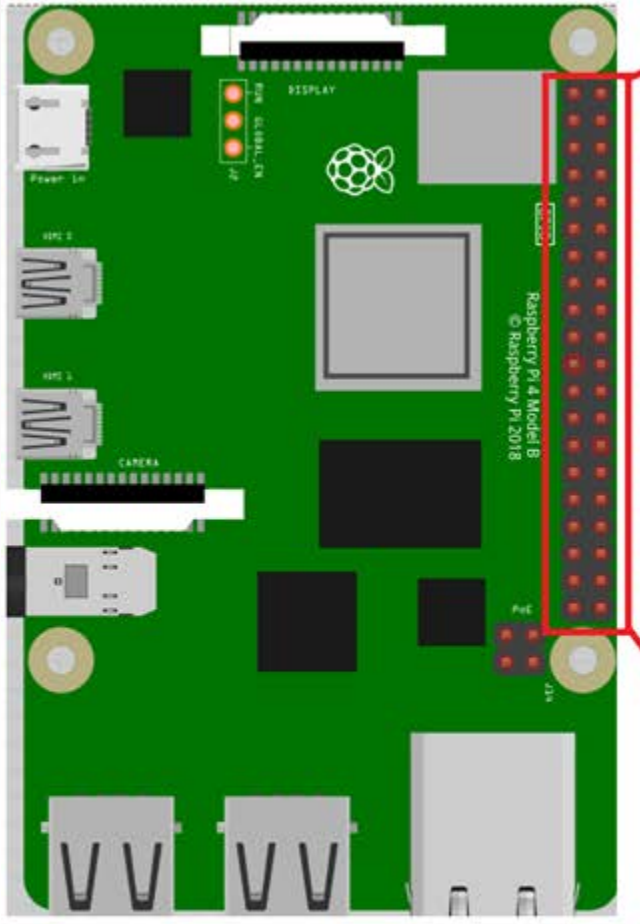
LED 제어

- LED 밝기 조절

```
import RPi.GPIO as gpio
import time

led_pin =17

gpio.setmode(gpio.BCM)

gpio.setup(led_pin, gpio.OUT)

try:
        while True:
                for t_high in range(0,11):
                        cnt =0
                        while True:
                                gpio.output(led_pin, True)
                                time.sleep(t_high*0.001)
                                gpio.output(led_pin, False)
                                time.sleep((10-t_high)*0.001)

                                cnt +=1
                                if cnt==10: break
                for t_high in range(10,-1,-1):
                        cnt =0
                        while True:
                                gpio.output(led_pin, True)
                                time.sleep(t_high*0.001)
                                gpio.output(led_pin, False)
                                time.sleep((10-t_high)*0.001)

                                cnt +=1
                                if cnt==10: break
except KeyboardInterrupt:
        pass
```

소프트웨어 PWM

- PWM 핀



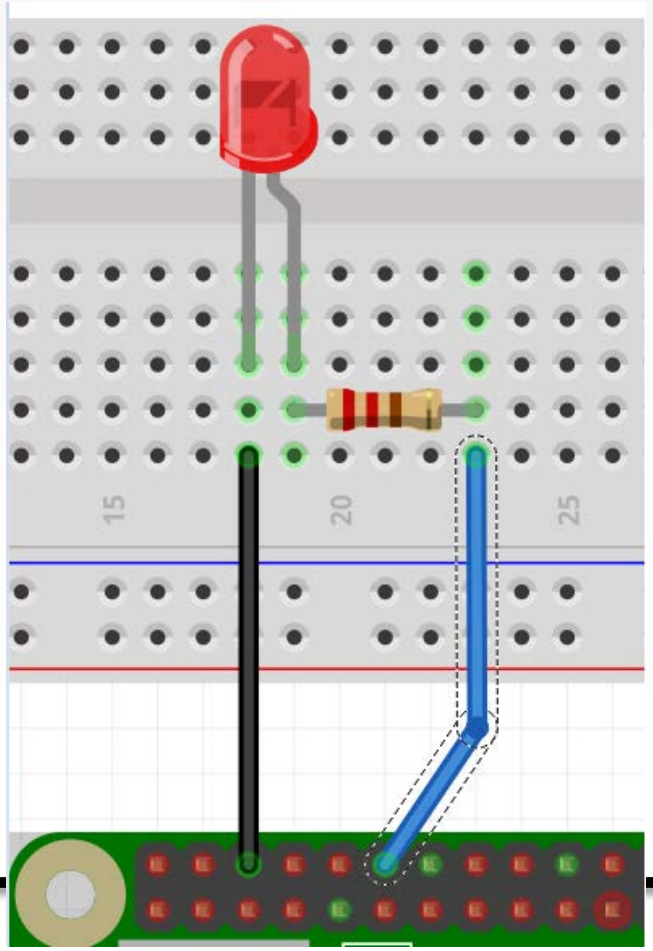
wPi	BCM	Pin	No	No	Pin	BCM	wPi
		3.3V	1	2	5V		
GPIO 08	GPIO 02	SDA1	3	4	5V		
GPIO 09	GPIO 03	SCL1	5	6	GND		
GPIO 07	GPIO 04		7	8	TXD	GPIO 14	GPIO 15
		GND	9	10	RXD	GPIO 15	GPIO 16
GPIO 00	GPIO 17		11	12	PWM0	GPIO 18	GPIO 01
GPIO 02	GPIO 27		13	14	GND		
GPIO 03	GPIO 22		15	16		GPIO 23	GPIO 04
		3.3V	17	18		GPIO 24	GPIO 05
GPIO 12	GPIO 10	MOSI	19	20	GND		
GPIO 13	GPIO 09	MISO	21	22		GPIO 25	GPIO 06
GPIO 14	GPIO 11	SCLK	23	24	CE0	GPIO 08	GPIO 10
		GND	25	26	CE1	GPIO 07	GPIO 11
GPIO 30	GPIO 00	SDA0	27	28	SCL0	GPIO 01	GPIO 31
GPIO 21	GPIO 05		29	30	GND		
GPIO 22	GPIO 06		31	32	PWM0	GPIO 12	GPIO 26
GPIO 23	GPIO 13	PWM1	33	34	GND		
GPIO 24	GPIO 19	PWM1	35	36		GPIO 16	GPIO 27
GPIO 25	GPIO 26		37	38		GPIO 20	GPIO 28
		GND	39	40		GPIO 21	GPIO 29

소프트웨어 PWM

•RPi.GPIO.PWM 모듈

- (High:Low) 비를 0.0% ~ 100.0% 범위에서 조절할 수 있음

```
import RPi.GPIO as gpio

led_pin =18      # 18번 핀으로 변경

gpio.setmode(gpio.BCM)

gpio.setup(led_pin, gpio.OUT)

pwm = gpio.PWM(led_pin, 1.0) # 1.0Hz
pwm.start(50.0) # 0.0~100.0

try:
        while True:
                pass
except KeyboardInterrupt:
        pass

pwm.stop()
gpio.cleanup()
```



소프트웨어 PWM

- **RPi.GPIO.PWM 모듈**
  - 주파수를 10.0Hz 로 변경

```
import RPi.GPIO as gpio

led_pin =18

gpio.setmode(gpio.BCM)

gpio.setup(led_pin, gpio.OUT)

pwm = gpio.PWM(led_pin, 10.0) # 10.0Hz
pwm.start(50.0) # 0.0~100.0

try:
        while True:
                pass
except KeyboardInterrupt:
        pass

pwm.stop()
gpio.cleanup()
```

소프트웨어 PWM

- **RPi.GPIO.PWM 모듈**
  - 주파수를 1000.0Hz 로 변경. HIGH:LOW 비를 1:9로 변경.

```
import RPi.GPIO as gpio

led_pin =18

gpio.setmode(gpio.BCM)

gpio.setup(led_pin, gpio.OUT)

pwm = gpio.PWM(led_pin, 1000.0) # 주파수 1000.0Hz
pwm.start(10.0) # (HIGH:LOW) 비를 10:90 으로 변경

try:
        while True:
                pass
except KeyboardInterrupt:
        pass

pwm.stop()
gpio.cleanup()
```

소프트웨어 PWM

- **RPi.GPIO.PWM 모듈**
  - 0.1초 간격으로 LED 밝기가 0% ~ 100% 로 1%씩 증가 후 1%씩 감소해서 0%가 됨

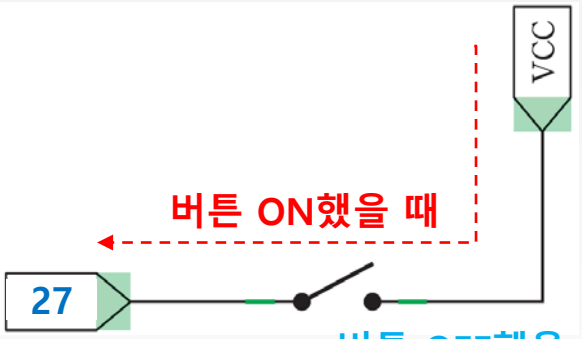
```
import RPi.GPIO as gpio
import time

led_pin =18

gpio.setmode(gpio.BCM)

gpio.setup(led_pin, gpio.OUT)

pwm = gpio.PWM(led_pin, 1000.0) # 1000.0Hz
pwm.start(0.0) # 0.0% 밝기로 시작

try:
        while True:
                for t_high in range(0,101):
                        pwm.ChangeDutyCycle(t_high)
                        time.sleep(0.1)
                for t_high in range(100,-1,-1):
                        pwm.ChangeDutyCycle(t_high)
                        time.sleep(0.1)
except KeyboardInterrupt:
        pass

pwm.stop()
gpio.cleanup()
```

소프트웨어 PWM

- **RPi.GPIO.PWM 모듈**
  - 0.1초 간격으로 LED 밝기가 0% ~ 100% 로 1%씩 증가 후 1%씩 감소해서 0%가 됨

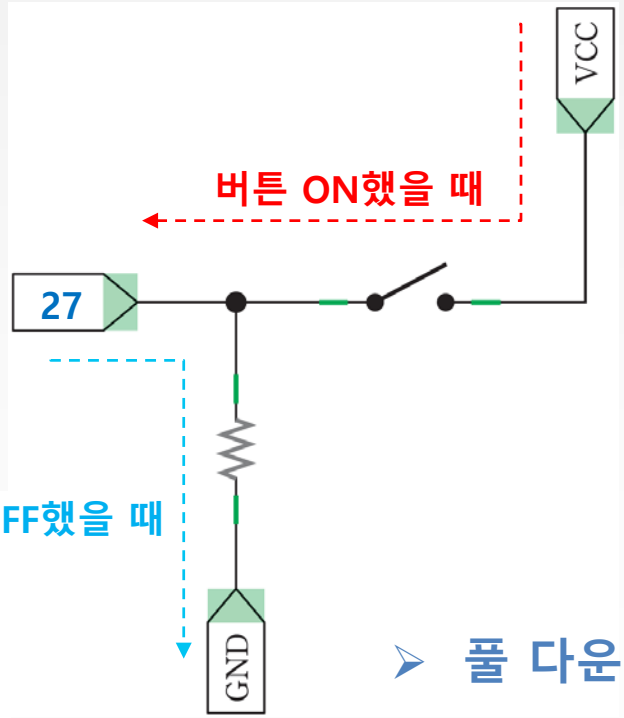
```
import RPi.GPIO as gpio
import time

led_pin =18

gpio.setmode(gpio.BCM)

gpio.setup(led_pin, gpio.OUT)

pwm = gpio.PWM(led_pin, 1000.0) # 1000.0Hz
pwm.start(0.0) # 0.0% 밝기로 시작

try:
        while True:
                for t_high in range(0,101):
                        pwm.ChangeDutyCycle(t_high)
                        time.sleep(0.1)
                for t_high in range(100,-1,-1):
                        pwm.ChangeDutyCycle(t_high)
                        time.sleep(0.1)
except KeyboardInterrupt:
        pass

pwm.stop()
gpio.cleanup()
```

풀 다운/풀 업 저항

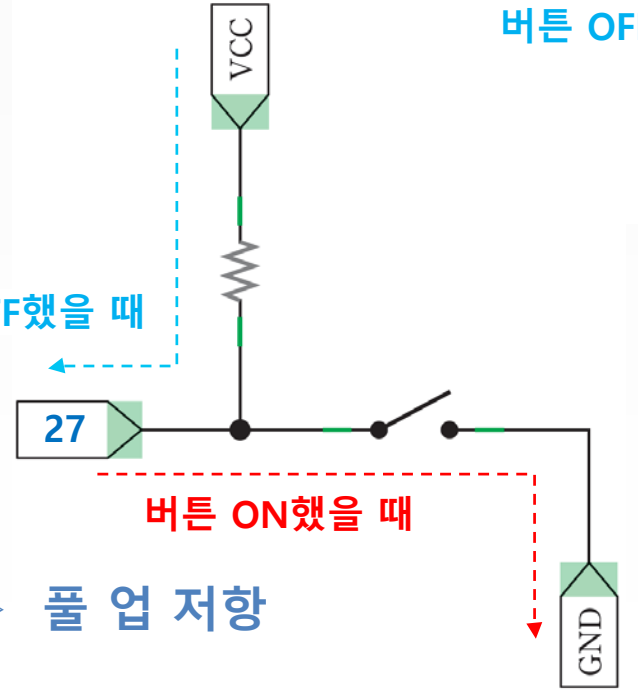


버튼 OFF했을 때는 9번 핀이 VCC, GND 어디에도 연결되어 있지 않음

➢ 기본 회로



➢ 풀 다운 저항



➢ 풀 업 저항

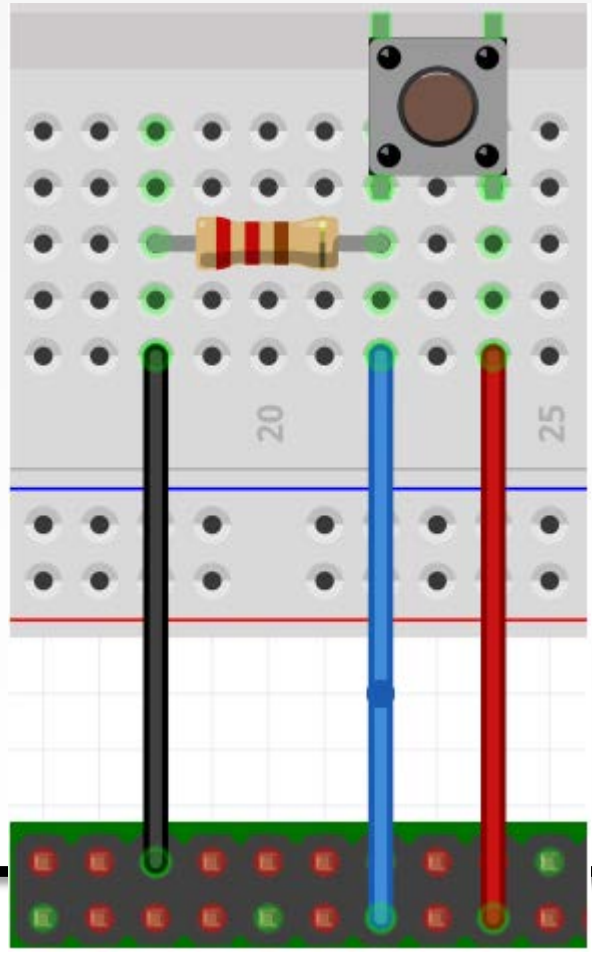
풀 다운/풀 업 저항

	버튼 누르지 않음	버튼 누름
사용 안함	플로팅 (1이나 0이 아닌 미결정 상태)	1
풀다운 저항 사용	0	1
풀업 저항 사용	1	0

라즈베리 파이 입력

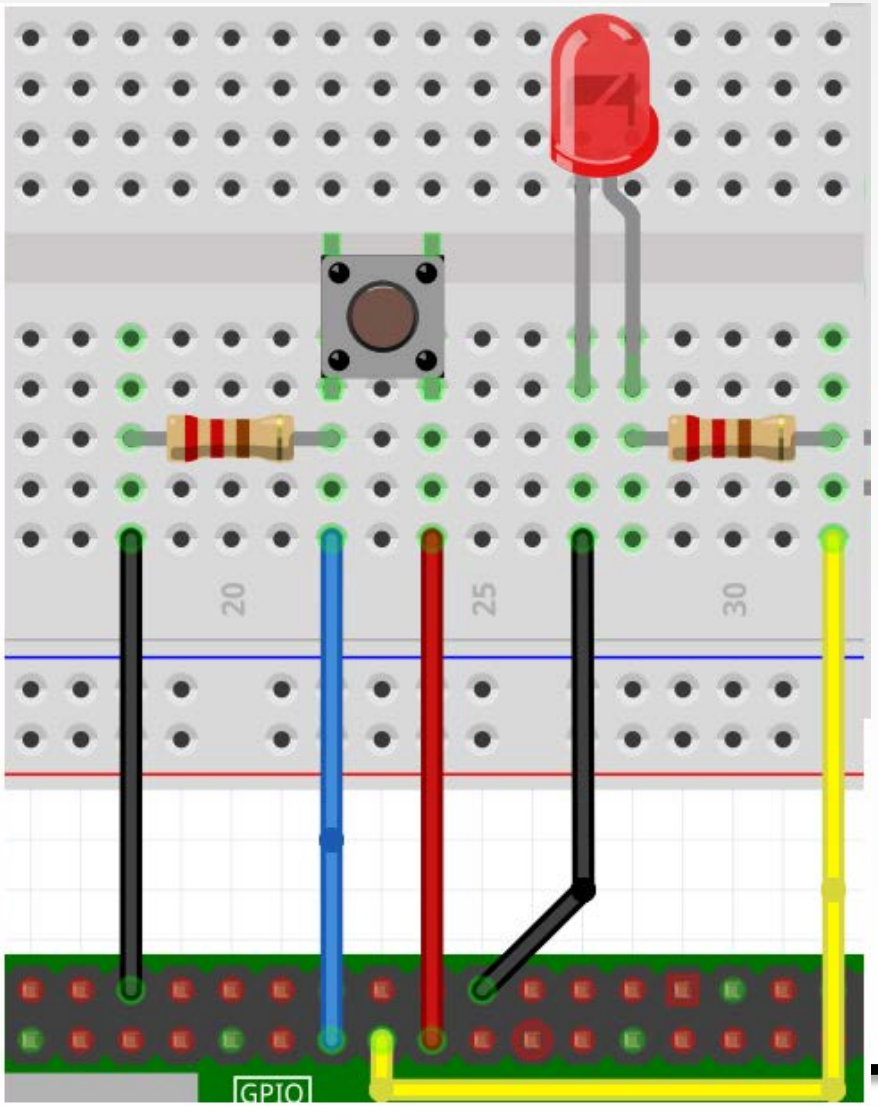
- **RPi.GPIO.input**
  - GPIO 27번 핀에 버튼 연결. 가급적 큰 저항값을 가진 저항 사용.
  - 0 / 1 읽어보기

```
import RPi.GPIO as gpio

button_pin =27

gpio.setmode(gpio.BCM)

gpio.setup(button_pin, gpio.IN)

try:
        while True:
                buttonInput = gpio.input(button_pin)
                print(buttonInput)

except KeyboardInterrupt:
        pass

gpio.cleanup()
```



라즈베리 파이 입력

- **RPi.GPIO.input**
  - GPIO 27번 핀에 버튼 연결
  - GPIO 22번 핀에 LED 연결
  - 버튼의 0 / 1 값에 따라 LED on/off



라즈베리 파이 입력

- **RPi.GPIO.input**

```
import RPi.GPIO as gpio

button_pin =27
led_pin =22

gpio.setmode(gpio.BCM)

gpio.setup(button_pin, gpio.IN)
gpio.setup(led_pin, gpio.OUT)

try:
        while True:
                buttonInput = gpio.input(button_pin)
                gpio.output(led_pin, buttonInput)
                # 버튼과 LED의 상태를 동일하게
except KeyboardInterrupt:
        pass

gpio.cleanup()
```

라즈베리 파이 입력

- **RPi.GPIO.input**
  - 버튼 토글

```
import RPi.GPIO as gpio

button_pin =27
led_pin =22

gpio.setmode(gpio.BCM)

gpio.setup(button_pin, gpio.IN)
gpio.setup(led_pin, gpio.OUT)

buttonInputPrev = False # 직전에 gpio.input() 함수가 호출되었을 때 버튼의 상태
ledOn = False

try:
        while True:
                buttonInput = gpio.input(button_pin)

                if not buttonInputPrev and buttonInput:
                # 직전 버튼의 상태가 0이었고 현재 버튼의 상태가 1이라면
                        print("rising edge")
                        ledOn = True if not ledOn else False
                        # LED 상태가 0이라면 1로, 1이라면 0으로 변경
                        gpio.output(led_pin, ledOn)
                elif buttonInputPrev and not buttonInput:
                # 직전 버튼의 상태가 1이었고 현재 버튼의 상태가 0이라면
                        print("falling edge")
                else: pass

                buttonInputPrev = buttonInput
                # 현재 버튼의 상태를 직전 버튼의 상태로 지정

except KeyboardInterrupt:
        pass

gpio.cleanup()
```

callback

•외부 인터럽트

- 이전 페이지의 버튼 토글 코드는 while True에 의해 button_pin의 상태를 계속해서 확인하므로 비효율적

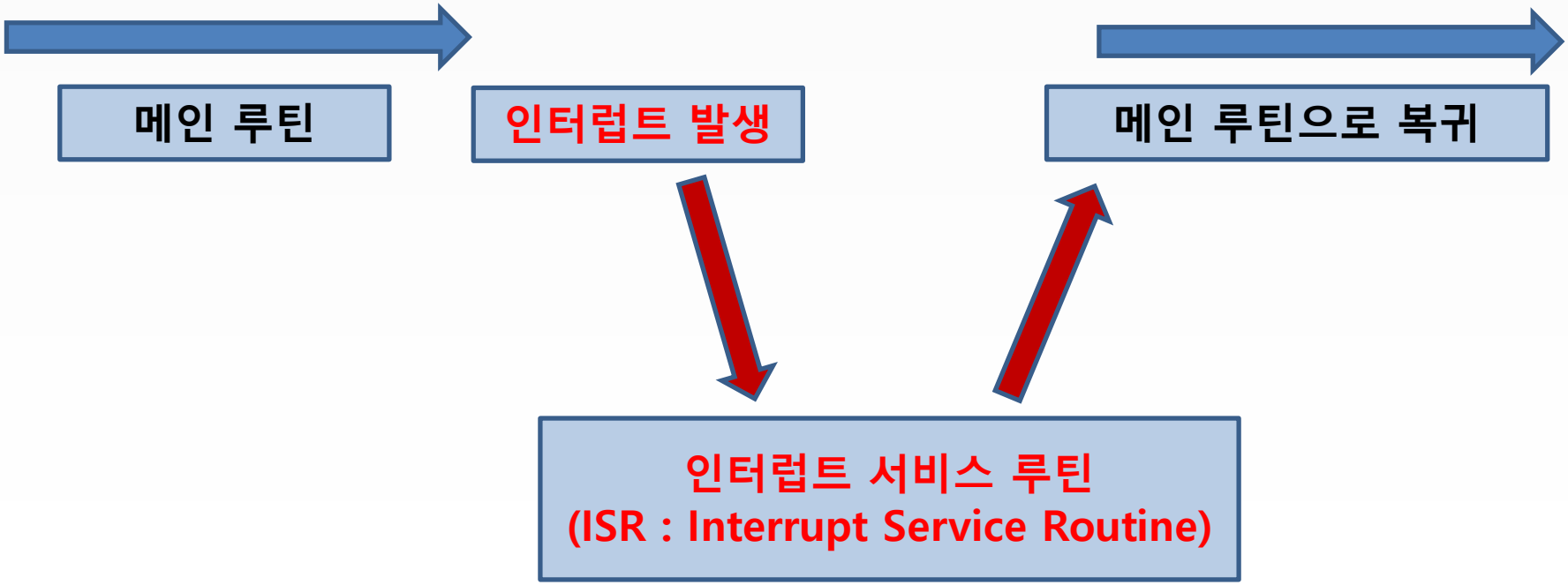
- 버튼 토글을 외부 인터럽트로 구현하는 것이 대안이 될 수 있음

```
while True:
        buttonInput = gpio.input(button_pin)

        if not buttonInputPrev and buttonInput:
        # 직전 버튼의 상태가 0이었고 현재 버튼의 상태가 1이라면
            print("rising edge")
            ledOn = True if not ledOn else False
            # LED 상태가 0이라면 1로, 1이라면 0으로 변경
            gpio.output(led_pin, ledOn)
        elif buttonInputPrev and not buttonInput:
        # 직전 버튼의 상태가 1이었고 현재 버튼의 상태가 0이라면
            print("falling edge")
        else: pass

        buttonInputPrev = buttonInput
        # 현재 버튼의 상태를 직전 버튼의 상태로 지정
```

callback

•외부 인터럽트

- 일반적으로 프로그램은 순차적으로 처리됨
- CPU에서 메인 루틴 처리 도중에 타이머, 카운터, 기타 외부의 신호에 의해 인터럽트가 발생하면 해당 인터럽트를 요청한 인터럽트 서비스 루틴(ISR)으로 실행 제어권을 넘겨 처리
- ISR 실행이 완료되면 메인 루틴으로 복귀할 수 있도록 기존 프로그램의 정보들(레지스터 값)을 백업해놓고 ISR이 완료되면 백업 정보들을 다시 복구



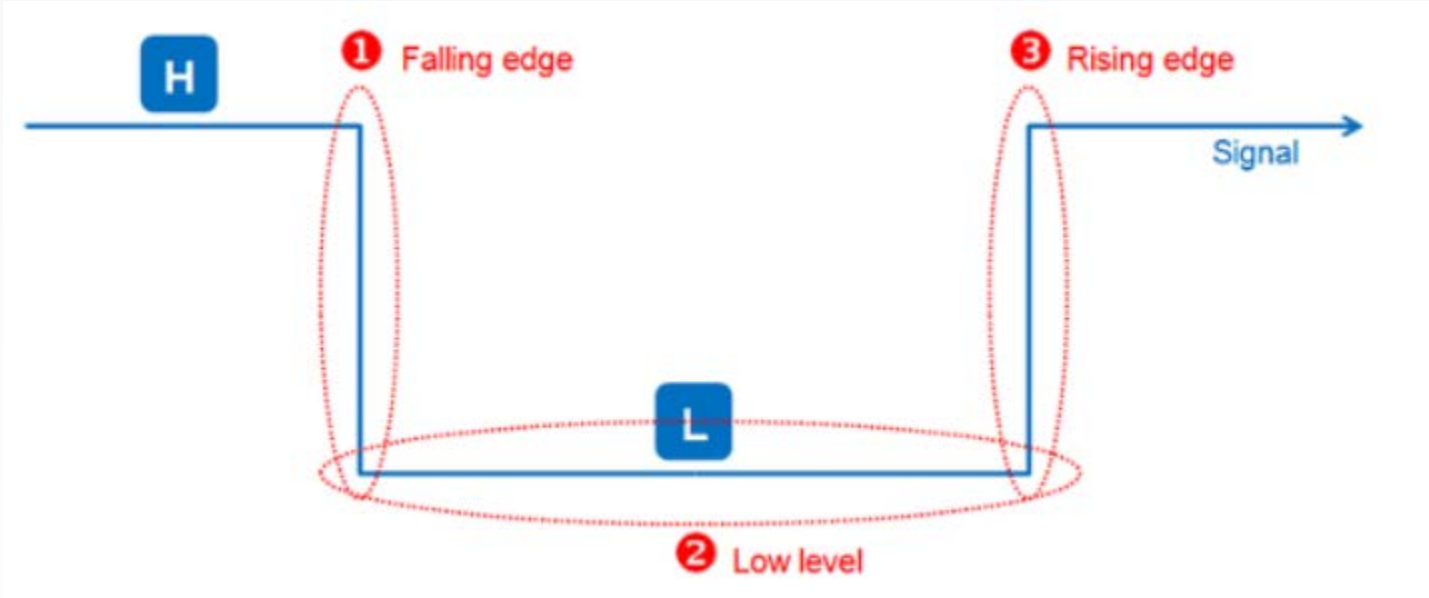
callback

- RPi.GPIO.add_event_detect 함수의 파라미터

RPi.GPIO.add_event_detect(channel, edge, callback, bounce_time)	
channel	핀 번호
edge	RPi.GPIO.Rising / RPi.GPIO.Falling
callback	ISR 처리로 사용할 함수
bounce_time	msec 단위로 설정

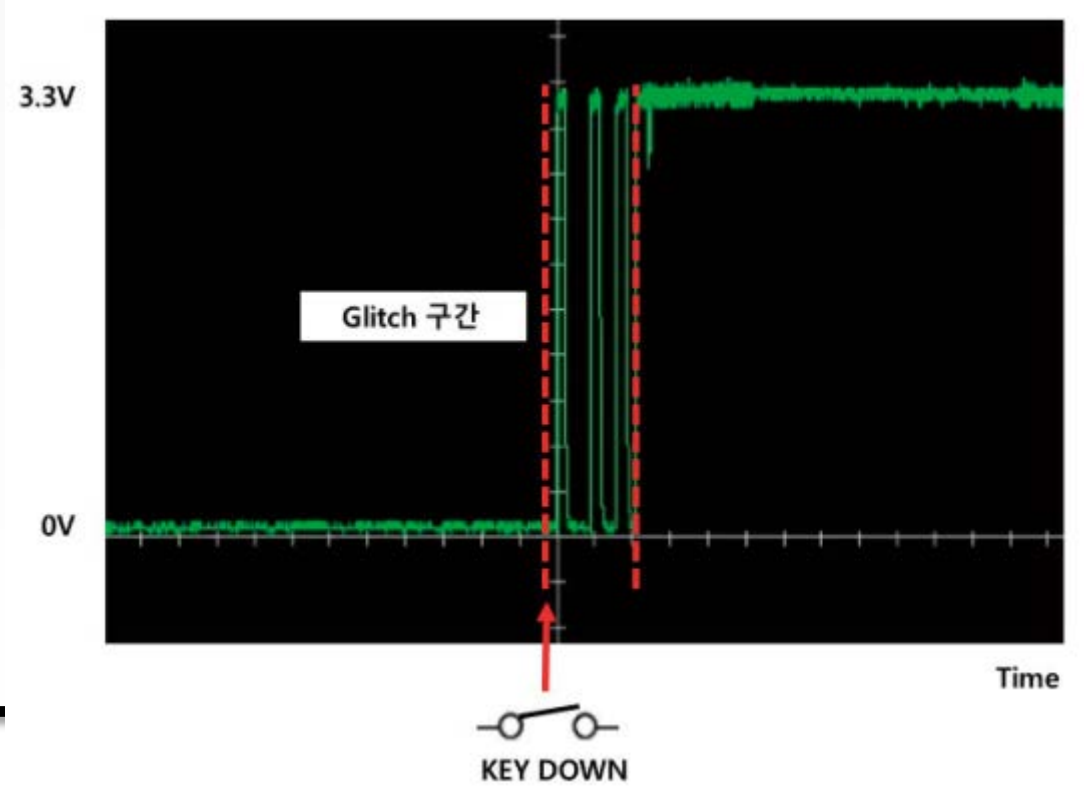
callback

- Falling edge / Rising edge



callback

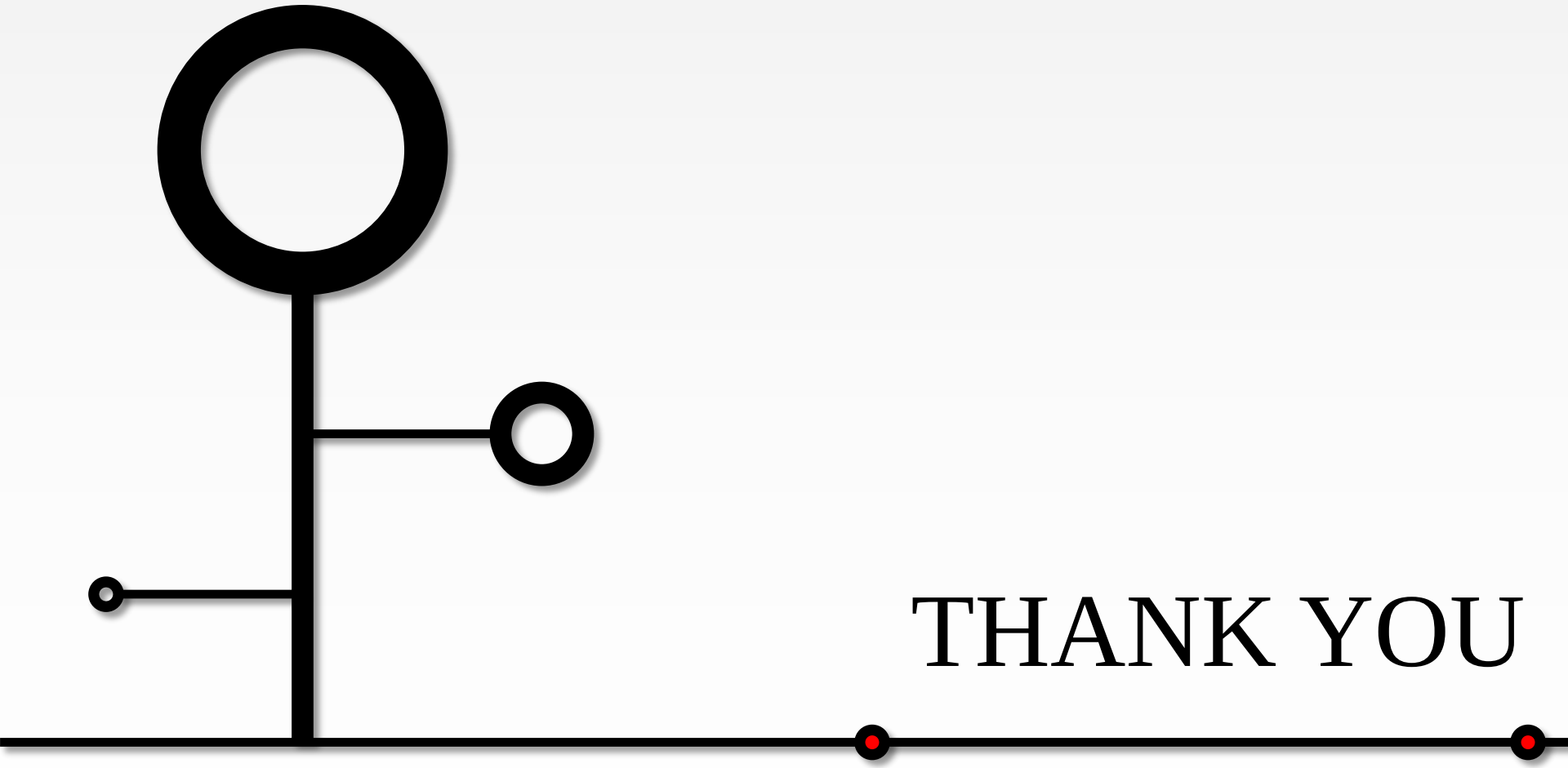
- **bounce time**
  - 버튼을 눌렀을 때 즉시 0에서 1로 변경되는 것이 아니라 아래와 같이 노이즈가 발생 – glitch
  - glitch가 발생하는 시간을 무시하고 인터럽트 처리



callback

- RPi.GPIO.add_event_detect

```
import RPi.GPIO as gpio
import time

button_pin =27
led_pin =22

gpio.setmode(gpio.BCM)

gpio.setup(led_pin, gpio.OUT)
gpio.setup(button_pin, gpio.IN)

def isr_event(pin):
    print("Button is pressed [%d]" %pin)
    if gpio.input(led_pin) == True:
        gpio.output(led_pin, False)
    elif gpio.input(led_pin) == False:
        gpio.output(led_pin, True)

gpio.add_event_detect(button_pin, gpio.FALLING, callback=isr_event, bouncetime=300)
     # RPi.GPIO.add_event_detect(channel, edge, callback, bouncetime)
     #                           channel : 핀 번호
     #                                  edge : FALLING/RISING 중 선택
     #                                         callback : 외부 인터럽트 발생시 실행할 함수
     #                                                    bouncetime 설정
sec = 0
try:
    while True:
        print("sec = %d" %sec)
        sec = sec + 1   # sec += 1로 대체 가능
        time.sleep(1)   # 1초 간격으로 print
except KeyboardInterrupt:
      pass

finally:
      gpio.cleanup()
```



THANK YOU