IoT 특론

5차시

AI첨단기술학과

이의혁

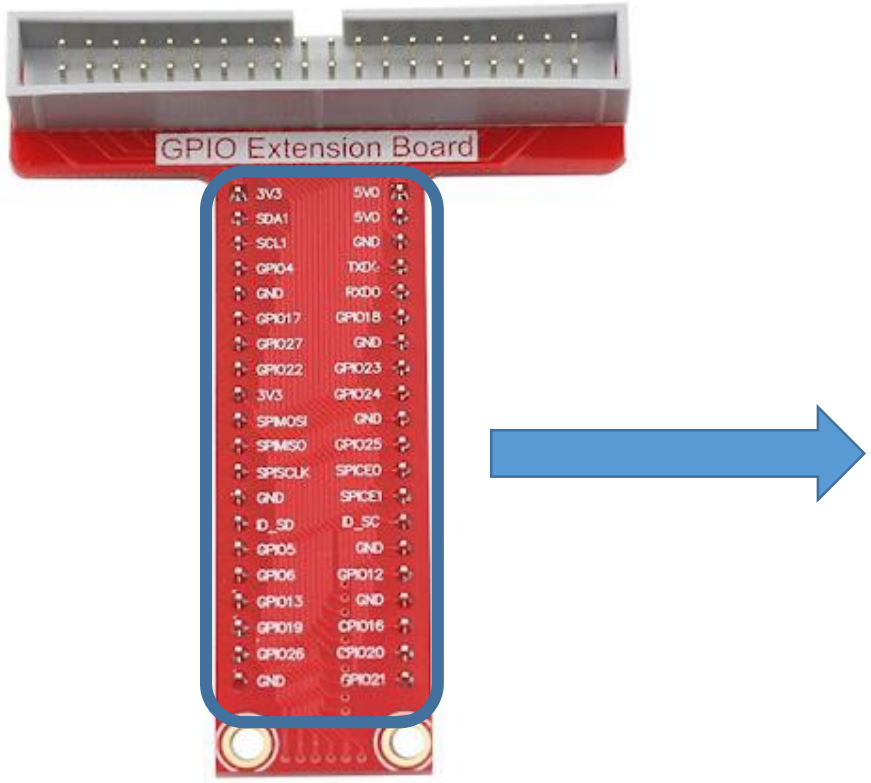
2. 사물 인터넷 디바이스

2-3. IoT 디바이스 프로그래밍

2) IoT 디바이스 프로그래밍

Raspberry Pi GPIO

- 라즈베리 파이에서 센서를 제어하고 센서 데이터를 읽거나 액추에이터를 제어하기 위해서 GPIO를 이용
- 이를 위해서 GPIO 프로그래밍 필요



Raspberry Pi 3 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

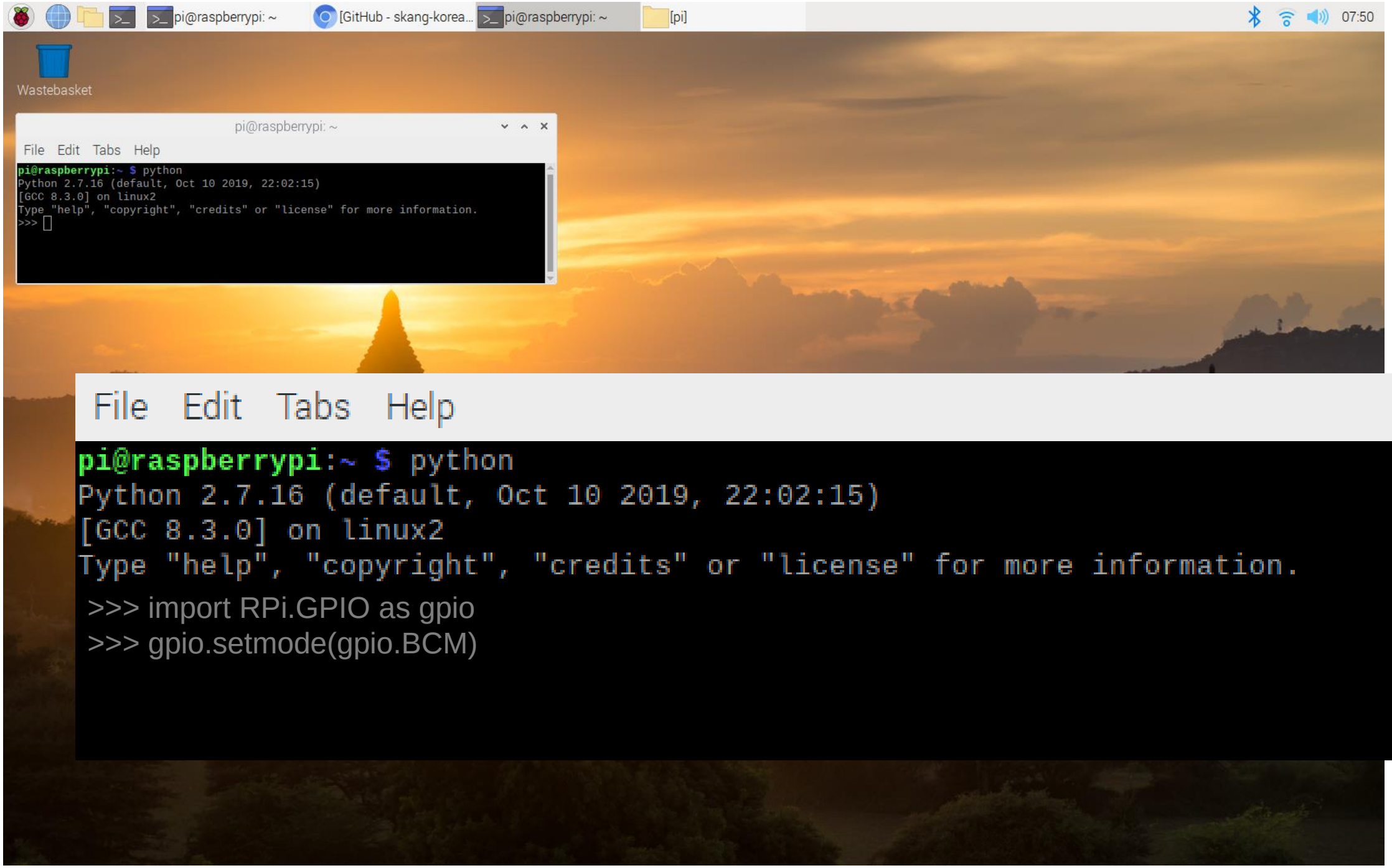
Rev. 2
29/02/2016

www.element14.com/RaspberryPi

RPi.GPIO 모듈

- RPi.GPIO
  - GPIO (General Purpose Input Output) 핀을 사용하기 위한 Python 모듈
  - 이를 이용하여 센서/액추에이터를 이용하는 프로그램 구현
  - http://sourceforge.net/p/raspberry-gpio-python/wiki/BasicUsage/

  - Raspberry Pi OS에 기본 설치되어 있음
    - 확인 : Command line 터미널에 python 입력 후 모듈을 import 해 본다.
      에러가 발생하지 않으면 이용 가능한 것임


pi@raspberrypi: ~
[GitHub - skang-korea...
pi@raspberrypi: ~
[pi]
07:50
Wastebasket
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ python
Python 2.7.16 (default, Oct 10 2019, 22:02:15)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
File Edit Tabs Help
pi@raspberrypi:~ $ python
Python 2.7.16 (default, Oct 10 2019, 22:02:15)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import RPi.GPIO as gpio
>>> gpio.setmode(gpio.BCM)

RPi.GPIO 모듈 기본

- 모듈 import
  - import RPi.GPIO as gpio
    - 모듈 이름에 대한 별칭을 지정하는 것
    - RPi.GPIO를 쓰지 않고 gpio를 쓰면 됨
- 초기 설정
  - gpio.setmode(gpio.BCM)
    - gpio.BOARD: Rasperry Pi Board 핀 번호 (1부터 40까지 순서대로 부여된 번호)
    - gpio.BCM: Broadcom SoC channel로 Broadcom에서 지정한 핀 번호
  - gpio.setup(channel, gpio.IN)
    - 여기서 channel은 사용하려고 하는 핀 번호
      - 여러 개의 핀 번호 설정: 리스트 이용
    - 두번째 파라미터는 입력용/출력용을 설정
    - 출력용으로 사용하고자 하는 경우, gpio.OUT

❖ Arduino pinMode() 함수

RPi.GPIO 모듈 기본

- 입력/출력
  - gpio.input(channel)
    - channel 변수로 명시된 핀에서 값을 읽음
    - 0 / gpio.LOW / False 또는 1 / gpio.HIGH / True를 반환
  - gpio.output(channel, state)
    - channel 변수로 명시된 핀에 state에 해당하는 값을 출력
    - state는 0 / gpio.LOW / False 또는 1 / gpio.HIGH / True 가 될 수 있음
- 종료 시
  - gpio.cleanup()

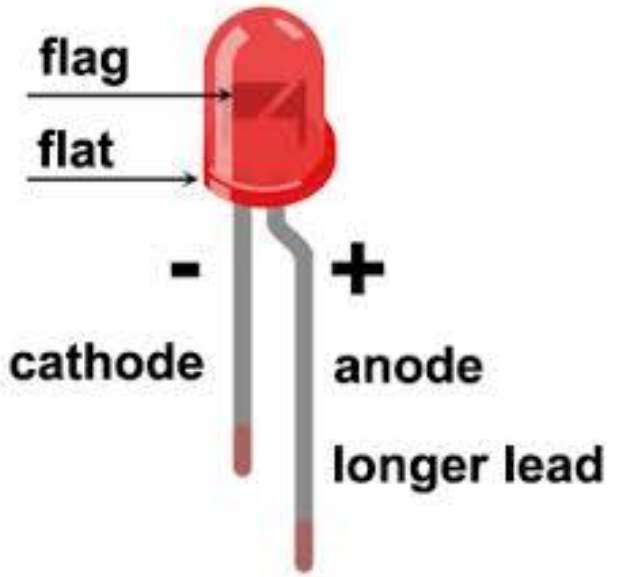
❖ Arduino digitalRead() 함수

❖ Arduino digitalWrite() 함수

LED 제어하기 – GPIO 출력

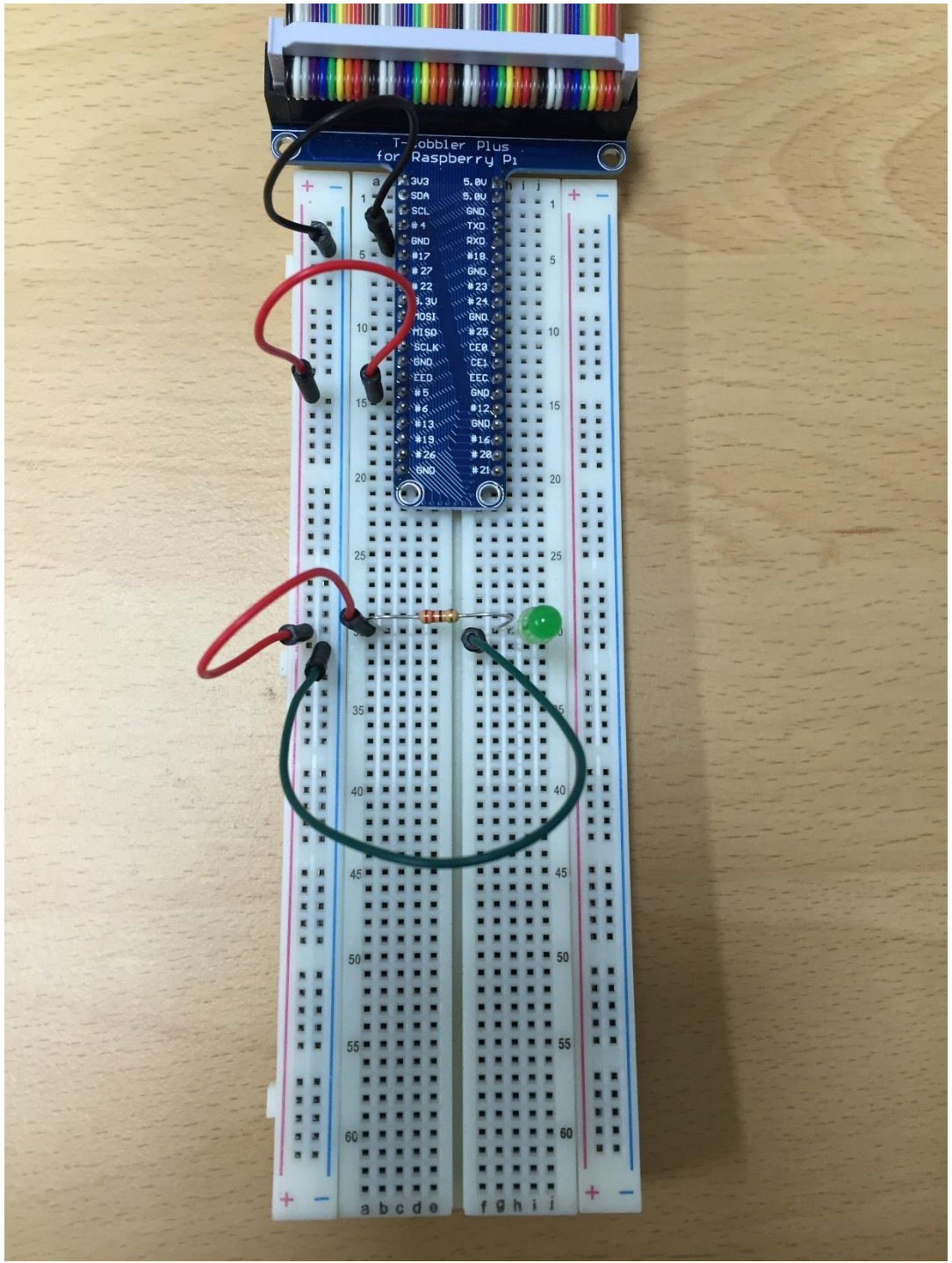
- Raspberry Pi GPIO 핀을 통해 LED를 켜고 끄는 것을 제어

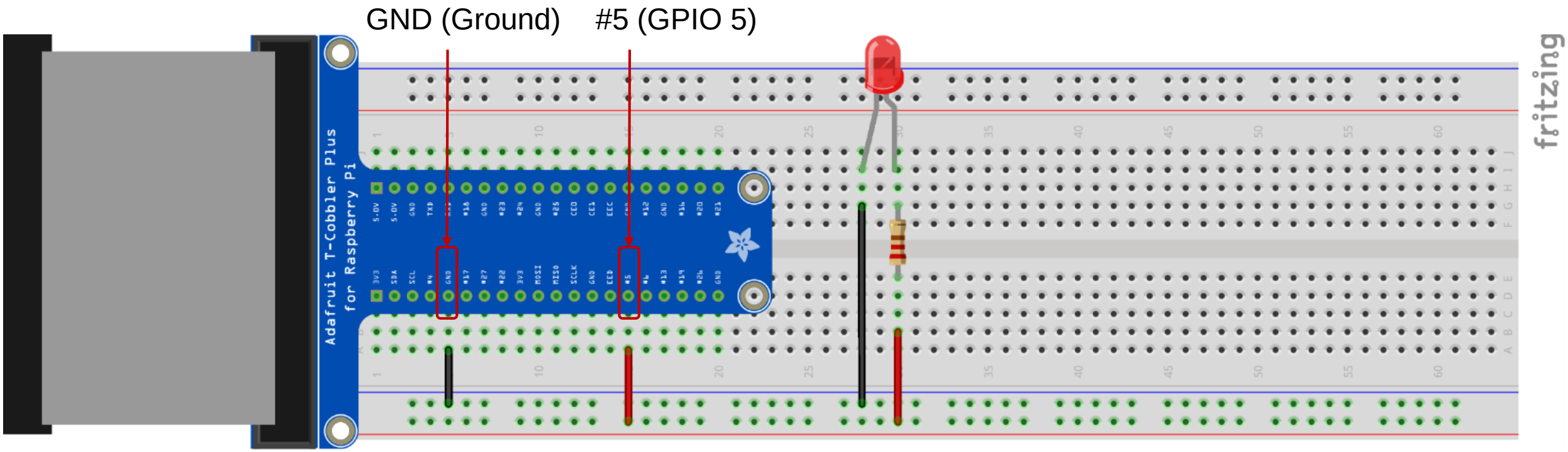
- 준비
  - Raspberry Pi와 GPIO 케이블로 연결된 코블러 브레이크아웃 보드와 브레드보드
  - LED
  - 저항
  - 점퍼 와이어



http://www.mobilefish.com/developer/arduino/arduino.html

- 회로 구성
  - 전원 공급용 GPIO 핀 연결
    - 이 예제에서는 5번 핀 사용
    - 코블러 브레이크아웃 보드에서 GPIO 5번 핀을 브레드보드의 빨간선으로 표시된 (+) 홀에 와이어로 연결
  - 그라운드 연결
    - 코블러 브레이크아웃 보드의 그라운드(GND) 핀을 와이어로 브레드보드의 파란선 (–) 홀에 연결
  - LED 장착
    - LED의 긴 전극이 양극, 짧은 전극이 음극이므로 음극을 브레이크아웃 보드의 GND 핀에 연결된 파란선 (–) 홀과 와이어로 연결
  - 저항 연결
    - LED의 양극과 브레드보드 빨간선 (+) 홀 사이에 작은 저항 하나를 연결




GND (Ground)
#5 (GPIO 5)
Adafruit T-Cobbler Plus
for Raspberry Pi
fritzing

```
import RPi.GPIO as gpio
import time
led_pin = 5
gpio.setmode(gpio.BCM)
gpio.setup(led_pin, gpio.OUT)

gpio.output(led_pin, True)
time.sleep(0.5)
gpio.output(led_pin, False)
time.sleep(0.5)
gpio.output(led_pin, True)
time.sleep(0.5)
gpio.output(led_pin, False)
time.sleep(0.5)

print(“Blink Finished”)
gpio.cleanup()
```

- 예제 코드: /actuator_led/simpleLedBlink.py
- 0.5초 간격으로 LED를 켰다 껐다 2번 반복하는 예제 프로그램
- 5번 GPIO 핀을 출력핀으로 사용하여 LED 제어
  - 5번 핀이 아닌 다른 핀을 사용하는 경우 3번 라인에서 해당 번호에 맞게 숫자를 변경해주어야 함
- 10번 켰다 껐다 반복하도록 프로그램을 변경한다면?

예제 코드: /actuator_led/blinkLed.py

```
import RPi.GPIO as gpio
import time
led_pin = 5
gpio.setmode(gpio.BCM)
gpio.setup(led_pin, gpio.OUT)

def blinkLED(numTimes, speed):
    for i in range(0, numTimes):
        print("Iteration " + str(i+1))
        gpio.output(led_pin, True)
        time.sleep(speed)
        gpio.output(led_pin, False)
        time.sleep(speed)
    print(“Blink Finished”)
    gpio.cleanup()
```

```
try:
    iterations = input("Enter total number of times to blink: ")
    speed = input("Enter length of each blink(seconds): ")
    blinkLED(int(iterations), float(speed))
except KeyboardInterrupt:
    gpio.cleanup()
```

- try, except: 프로그램 실행 중 발생하는 오류(예외)를 처리하는 데 사용
  - 예외 처리를 하지 않으면 오류가 나서 프로그램이 그냥 중단되어 버리는데 try, except 문을 사용하여 매끄럽게 종료되도록 하거나 종료되지 않고 계속 실행되도록 할 수 있음
- KeyboardInterrupt: 프로그램 실행 중 사용자가 Ctrl+C 키를 누를 때 발생하는 예외

실습 해보기

- 사용자에게 입력 받은 값에 따라 LED 제어
  - 세 가지 색의 LED를 제어하는 프로그램 구현
    - 예: 빨간색, 노란색, 녹색

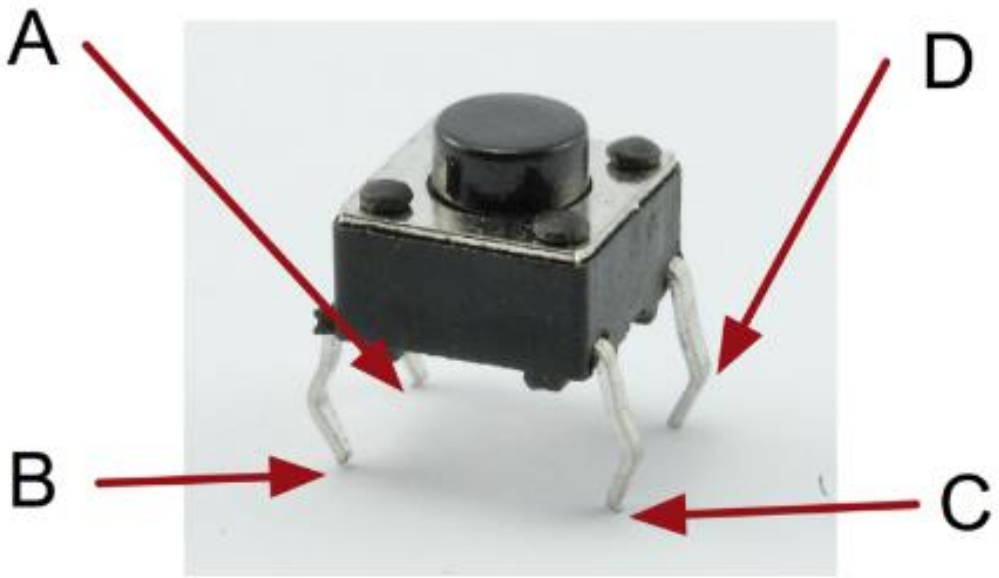
    - 1 입력: 빨간색 LED 만 켜기
    - 2 입력: 노란색 LED 만 켜기
    - 3 입력: 녹색 LED 만 켜기

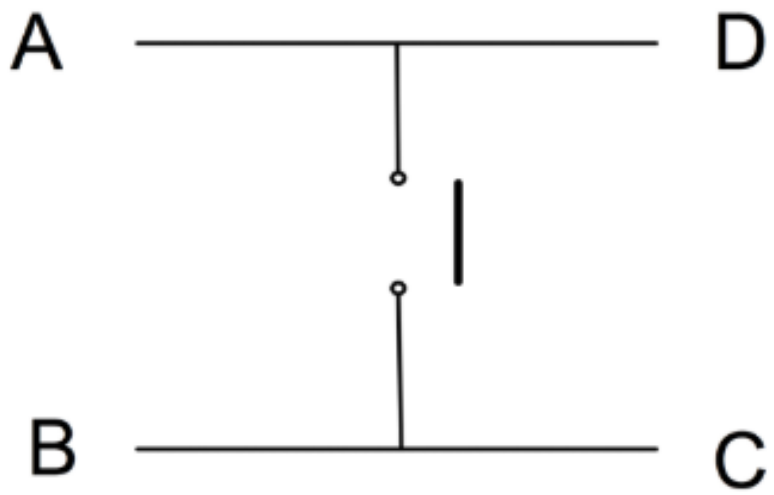
스위치 버튼 이용하기 – GPIO 입력

- GPIO 핀을 통해 사용자가 Switch 버튼을 누르는 신호 입력 받기

- 준비
  - Raspberry Pi와 GPIO 케이블로 연결된 코블러 브레이크아웃 보드와 브레드보드
  - 4핀 푸시 스위치
  - 점퍼 와이어

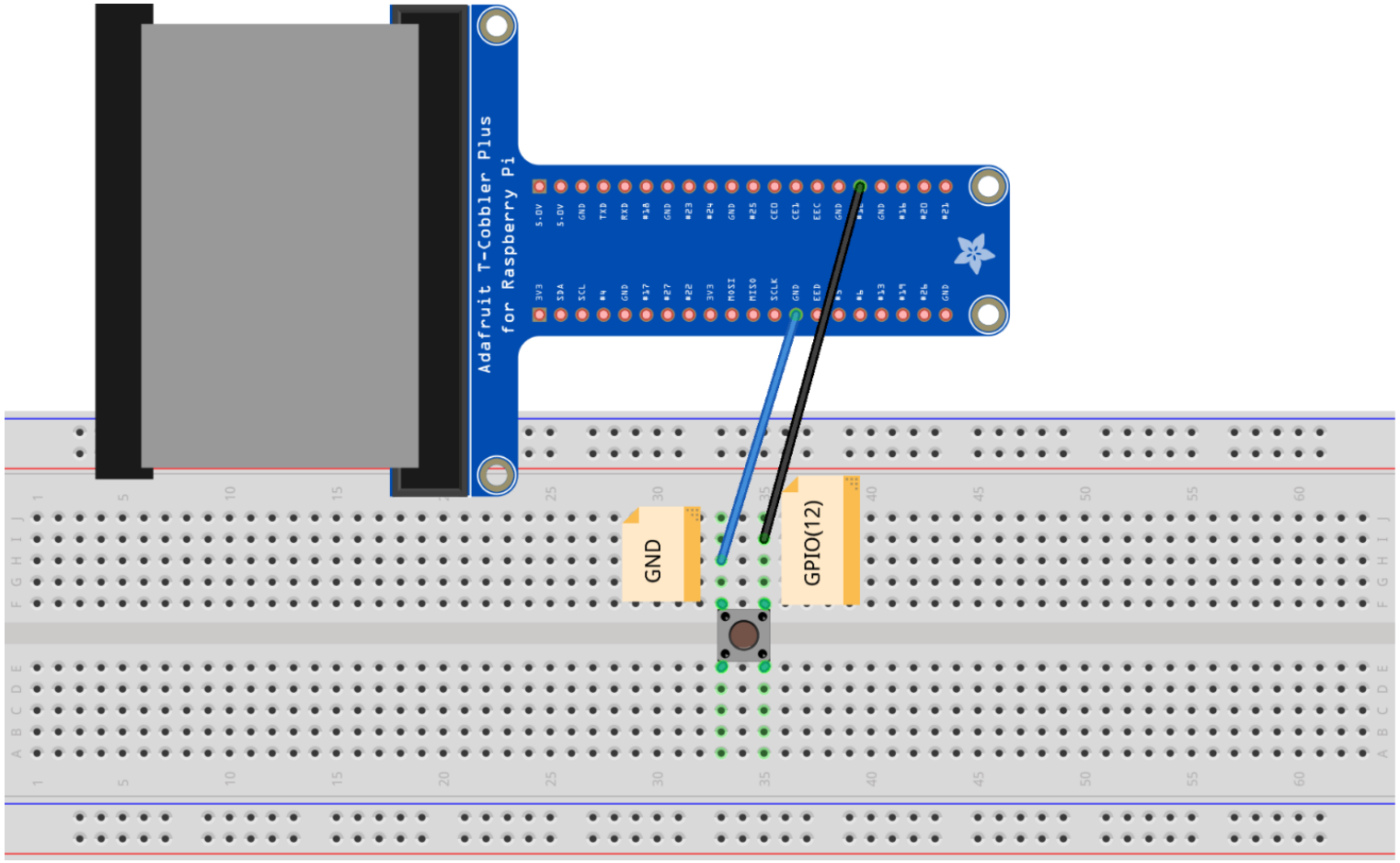
- 4핀 푸시 스위치 구조





- 핀A와 핀D, 핀B와 핀C는 서로 연결되어 있음
- 스위치를 누르면 모든 핀이 연결되는 구조
- 스위치 입력을 사용하기 위해서 A와 B 혹은 D와 C를 사용 (A와 C, D와 B를 사용해도 됨)

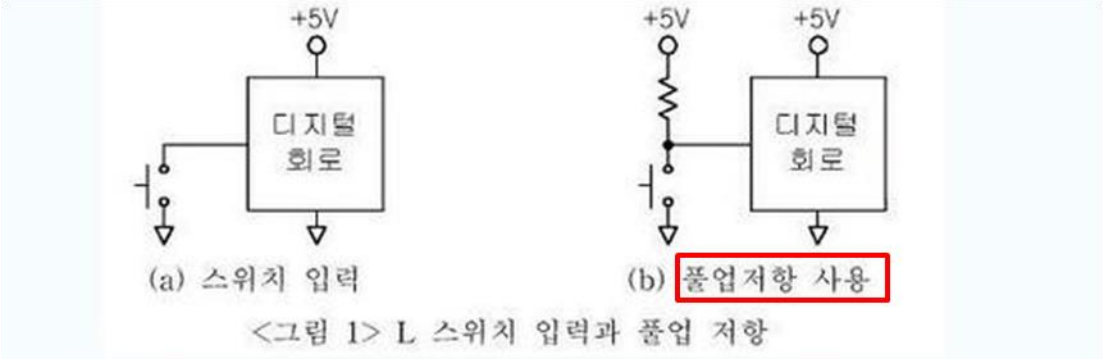
- 회로 구성



- 버튼을 누르면 GPIO 12번 핀에 연결된 스위치 핀이 GND와 연결되게 됨 → 따라서 12번 핀에서 입력을 읽으면 0
- 버튼을 누르지 않으면 핀이 floating 상태가 되게 되므로 풀업 혹은 풀다운 저항으로 연결되어 있어야 함
- Raspberry Pi의 GPIO 핀은 소프트웨어적으로 설정 가능한 풀업/풀다운 저항을 가지고 있음
  - RPi.GPIO 모듈의 setup 함수에서 설정 가능

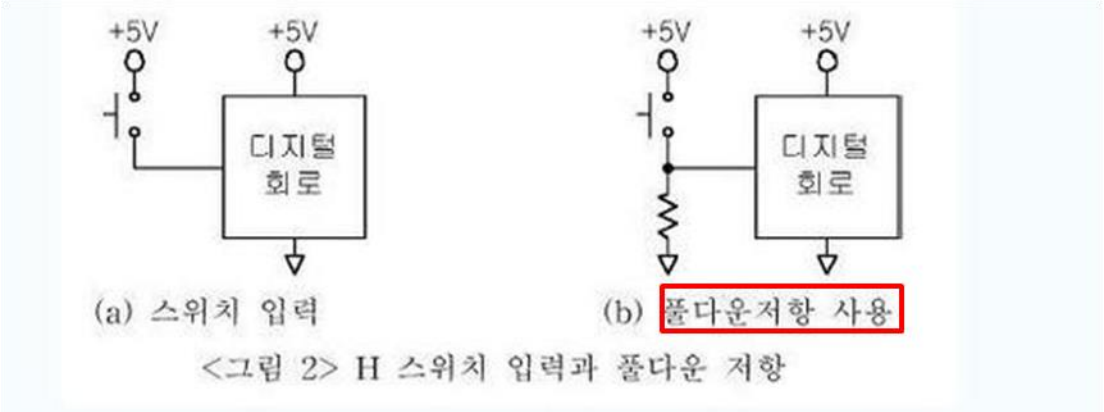
풀업/풀다운 저항

- 풀업/풀다운 저항
  - 디지털 회로가 floating 상태로 있는 것을 방지하기 위해 사용하는 저항
- Floating 상태
  - 스위치가 열려 있을 때 low, high 둘 중 어떤 상태도 아닌 상태
- 풀업 저항
  - 스위치 off 시 high 상태로 만들어 주기 위해 사용하는 저항
  - 저항을 Vcc 쪽으로 연결
- 풀다운 저항
  - 스위치 off 시 low 상태로 만들어주기 위해 사용하는 저항
  - 저항을 GND 쪽으로 연결



<그림 1> L 스위치 입력과 풀업 저항

스위치	ON	OFF
(a)그림	0V(Low)	Floating
(b)그림	0V(Low)	+5V(High)



<그림 2> H 스위치 입력과 풀다운 저항

스위치	ON	OFF
(a)그림	+5V(High)	Floating
(b)그림	+5V(High)	0V(Low)

http://aboutmadlife.blogspot.kr/2014/09/pull-up-pull-down.html

예제 코드: /gpio_switch/switch.py

- 스위치의 신호 입력을 받는 GPIO 핀이 12번이라 가정

gpio.setup(switch_pin, gpio.IN, gpio.PUD_UP)

- 여기서 세번째 파라미터가 풀업 저항을 활성화하는 용도로 사용됨
  - 풀다운 저항은 gpio.PUD_DOWN
- 즉 12번 핀을 입력 핀으로 설정하면서, 풀업 저항을 활성화하여 스위치 버튼을 누르지 않은 상황에서는 입력이 1이 되도록 함

```
import RPi.GPIO as gpio
import time
switch_pin = 12
gpio.setmode(gpio.BCM)
gpio.setup(switch_pin, gpio.IN, gpio.PUD_UP)

def button():
    isClicked = False
    if gpio.input(switch_pin) == 0:
        isClicked = True
    return isClicked
try:
    while True:
        if button() == True:
            print(“Switch ON”)
            time.sleep(0.5)
        else:
            print(“Switch OFF”)
            time.sleep(0.5)
except KeyboardInterrupt:
    gpio.cleanup()
```

실습 해보기

- 스위치 입력을 이용한 LED 제어
  - 스위치를 누르면 빨간색 LED를 켜고 스위치를 떼면 노란색 LED를 켜는 프로그램 구현

초음파 센서로 거리 측정하기 – GPIO 입력

- 초음파 센서를 이용하여 초음파 센서 앞에 있는 물체와의 거리를 측정
- 준비
  - Raspberry Pi와 GPIO 케이블로 연결된 코블러 브레이크아웃 보드와 브레드보드
  - 초음파 센서(HC-SR04) 1개
  - 저항 2개
  - 점퍼 와이어

http://microcontrollerelectronics.com/distance-sensing

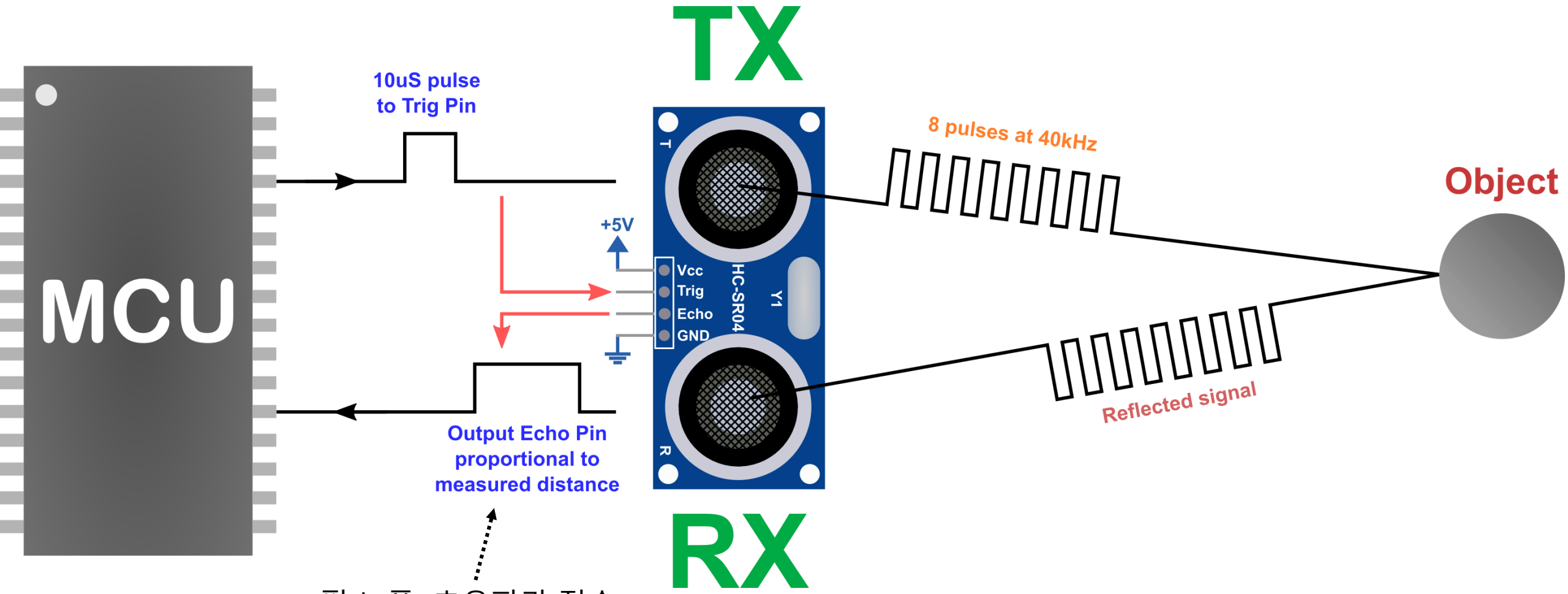
HC-SR04 초음파 센서

- 2~400cm 거리 감지
- 30도 정도 각도 범위 이내 물체 감지
- 4개 핀
  - VCC: 5V 전원을 공급하는 핀
  - Trig: 트리거 신호 입력 핀 (Pi의 신호 출력이 센서로 입력)
  - Echo: Trig 핀과 반대 역할을 하는 핀으로서, 센서의 출력 신호 핀
  - GND: 그라운드에 연결하는 핀
    - VCC 핀으로 전원을 공급하기 전에 그라운드 핀을 그라운드에 연결

HC-SR04 초음파 센서

- Trig: 트리거 신호 입력 핀
  - 라즈베리 파이나 아두이노와 같은 마이크로컨트롤러 유닛에서 트리거 신호를 발생하여 이 핀을 통해 센서로 입력된다.
  - 트리거 신호가 입력되면 HC-SR04 센서의 좌측의 초음파 발생부에서 초음파 신호가 전송된다.
- Echo: 센서의 출력 신호 핀
  - HC-SR04에서 초음파가 발생되어 전송됐을 때, 물체에 반사되어 돌아오는 초음파 신호가 센서의 우측 초음파 수신부에서 수신되고 그 신호가 에코 핀으로 전달된다.
  - 이를 핀을 통해 에코 신호가 마이크로컨트롤러 유닛에 전송된다.

HC-SR04 동작 원리

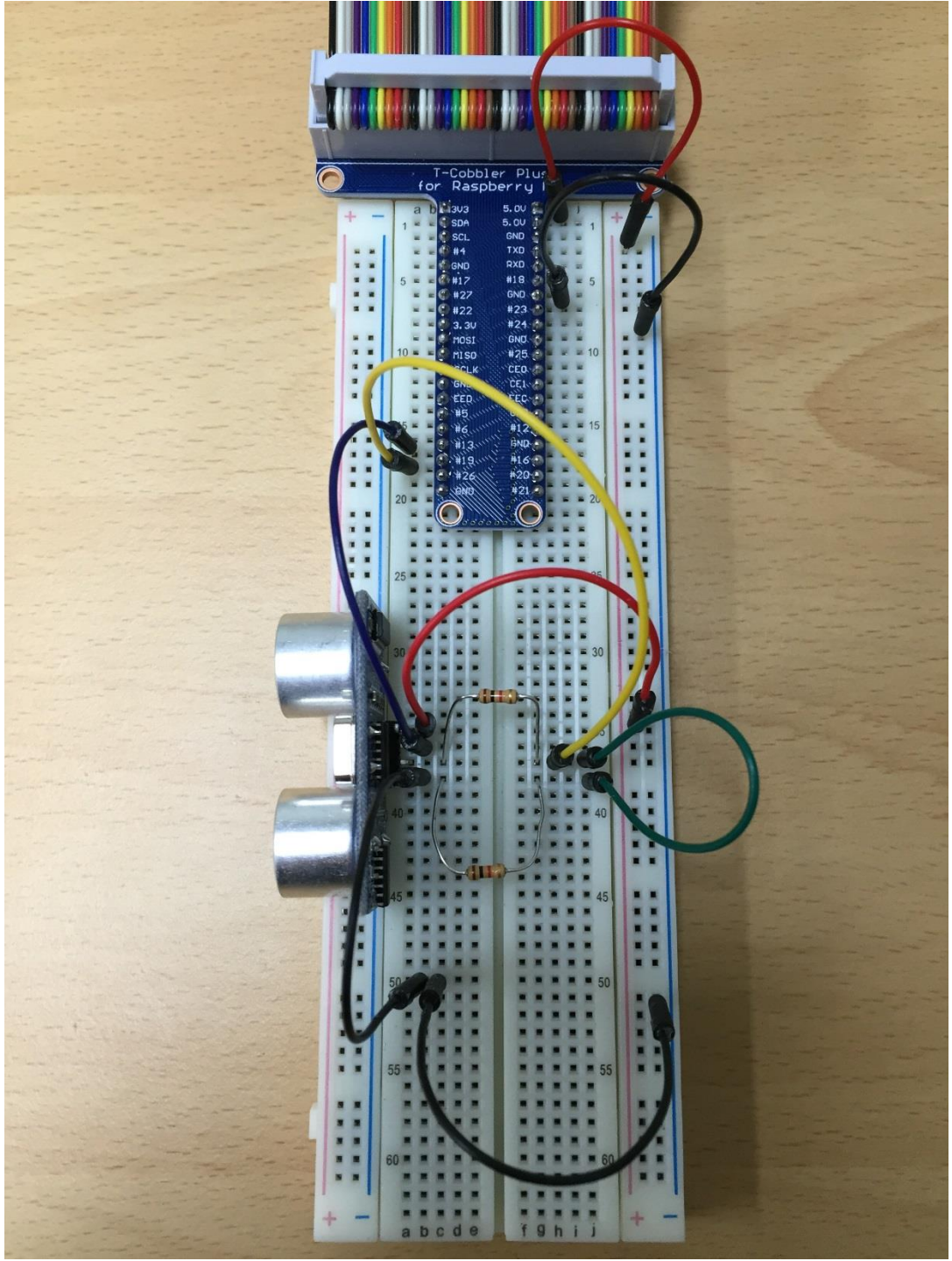


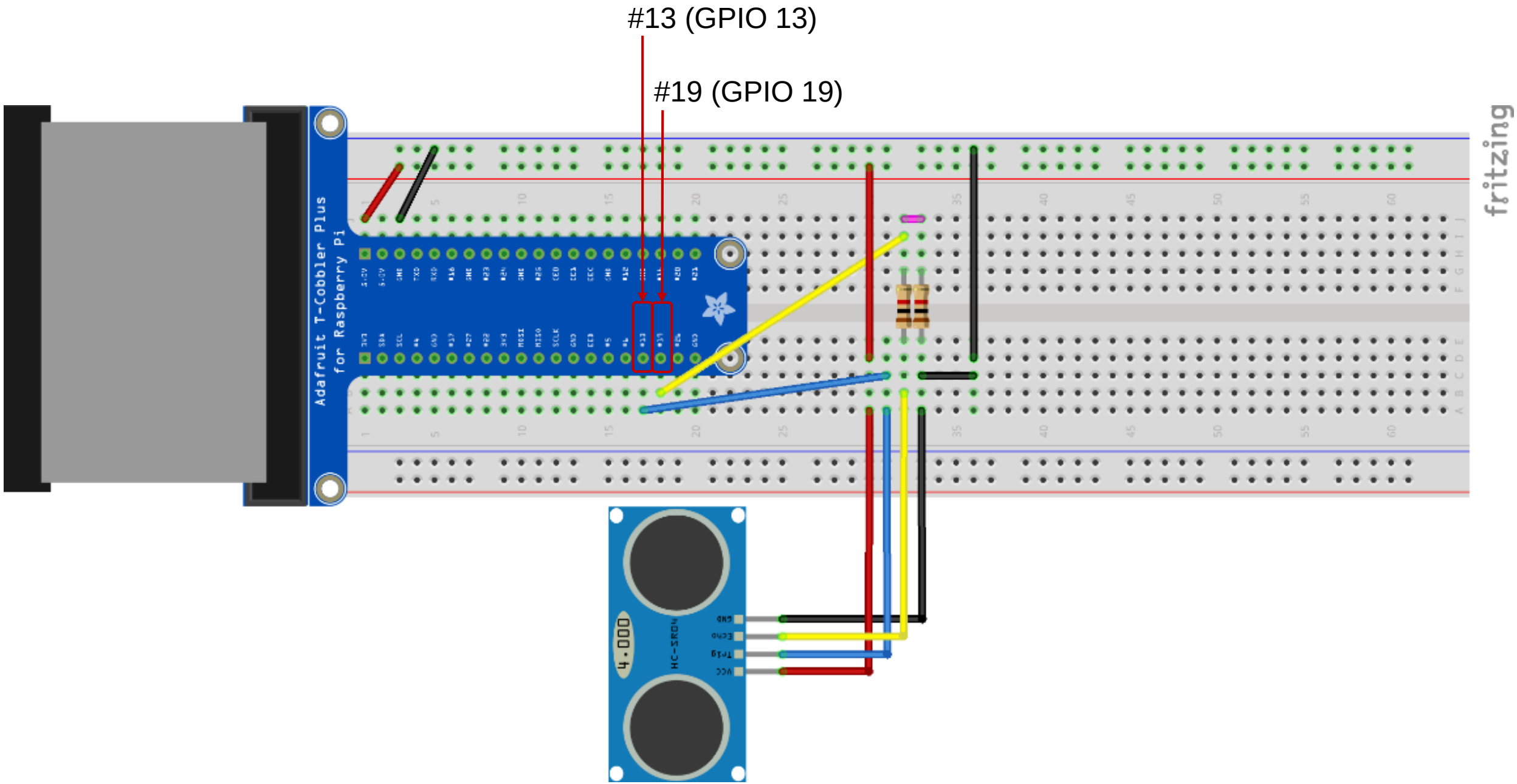
펄스 폭: 초음파가 전송되어 반사파가 돌아올 때까지 걸린 시간

https://trionprojects.org/ultrasonic-sensor-hc-sr04-with-pic-microcontroller/

- 회로 구성
  - 전원 연결
    - 코블러 브레이크아웃 보드의 5V 출력 핀 ←→ 브레드보드 빨간 선 (+) 홀
  - 그라운드 연결
    - 코블러 브레이크아웃 보드의 그라운드(GND) 핀 ←→ 브레드보드 파란선 (–) 홀
  - 초음파 센서 장착 및 전원, GND 연결
    - 초음파 센서 GND 핀 ←→ (-) 홀
    - 초음파 센서 VCC 핀 ←→ (+) 홀
  - Trig, Echo 핀 연결
    - Trig 핀 → GPIO 13번 핀
    - Echo 핀 → 1K옴 저항 → GPIO 19번 핀
    - GND 핀 → 1K옴 저항 → GPIO 19번 핀에 연결한 지점

Trig, Echo 핀 용 GPIO 핀은 다른 것을 사용해도 무방




#13 (GPIO 13)
#19 (GPIO 19)
Adafruit T-Cobbler Plus
for Raspberry Pi
HC-SR04
4.000
GND
Echo
Trig
VCC
fritzing

예제 코드: /sensor_ultrasonic/ultra.py

```
import RPi.GPIO as gpio
import time

trig_pin = 13
echo_pin = 19

gpio.setmode(gpio.BCM)
gpio.setup(trig_pin, gpio.OUT)
gpio.setup(echo_pin, gpio.IN)
```

```
try:
    while True:
        gpio.output(trig_pin, False)
        time.sleep(1)

        gpio.output(trig_pin, True)
        time.sleep(0.00001)
        gpio.output(trig_pin, False)

        while gpio.input(echo_pin) == 0:
            pulse_start = time.time()

        while gpio.input(echo_pin) == 1:
            pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start
        distance = pulse_duration * 34000 / 2
        distance = round(distance, 2)

        print("Distance : ", distance, “cm”)
except KeyboardInterrupt:
    gpio.cleanup()
```

- 10us 펄스 신호를 트리거 핀에 발생
- 에코 신호의 펄스 폭을 계산

- time.time() 함수: UTC(GMT+0) 기준으로 1970년 1월 1일 0시 0분 0초부터 경과 시간
  - 예: 1642325278.9895427
  - 정수부 – 초 단위
- round() 함수: 반올림 기능
  - round(3.1415, 2) → 3.14

실습 해보기

- 초음파 센서를 이용한 LED 제어
  - 초음파 센서에서 측정된 거리 값을 이용하여 LED를 제어하는 프로그램 구현
    - 40cm 이상: 녹색 LED on
    - 20-40cm: 노란색 LED on
    - 20cm 이하: 빨간색 LED on
    - LED를 켰다 껐다 하는 것 외에 화면에 해당 내용을 출력하도록 함