

소켓프로그래밍

라즈베리파이로 배우는 소켓 통신 프로그래밍



동양미래대학교
컴퓨터공학부 정석용

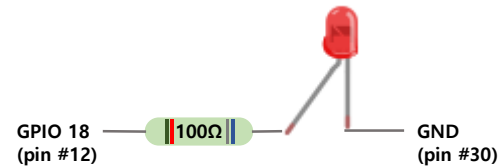
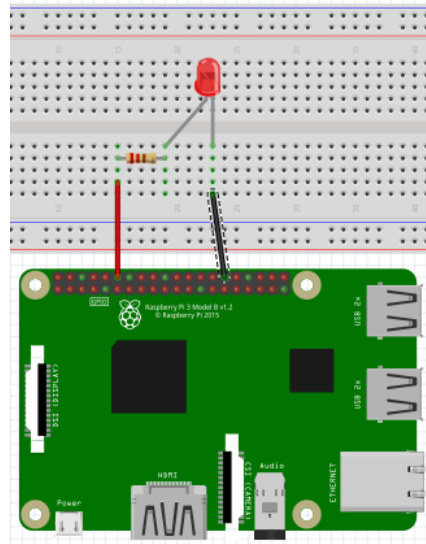


동양미래대학교

PWM 제어

(실습과제 1) gpio 명령을 이용한 LED 점멸

- 회로구성



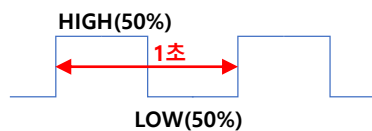
- LED의 Anode(+) 핀은 65 Ω 이상의 저항을 통해 라즈베리파이 GPIO 18(pin #12)에 연결
- LED의 cathod(-) 핀은 라즈베리파이 GND(pin #30)에 연결

PWM(Pulse Width Modulation) Analog Output

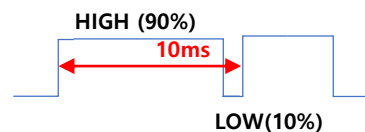
- 디지털 소스를 사용하여 아날로그 신호를 생성하는 방법
- PWM 신호는 주파수와 상하비(듀티 사이클)라는 두 가지 주요 구성 요소로 구성
 - **주파수** : 1초에 반복되는 펄스의 개수, 단위는 Hz
 - **듀티 사이클(상하비)** : 반복하는 펄스에서 HIGH의 비율로, 반복되는 펄스의 총 시간에서 HIGH 상태 시간을 백분율로 나타냄

PWM(Pulse Width Modulation) Analog Output

<u>주파수(Hz)</u>	1초간 반복되는 사각파형의 개수
<u>상하비</u> (듀티 사이클)	High 값과 Low 값의 비율



주파수 : 1Hz
상하비 : (5 : 5)

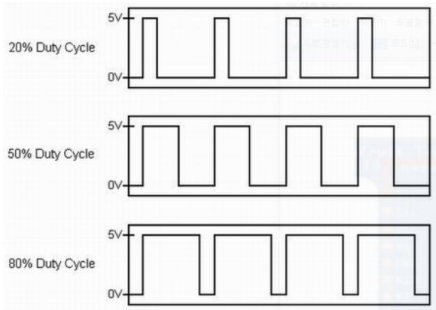


주파수 : 100Hz
상하비 : (9 : 1)

PWM(Pulse Width Modulation) Analog Output

디지털 신호를 빠른 속도로 일정한 상하비(듀티 사이클)를 통해 켜다가 켜면

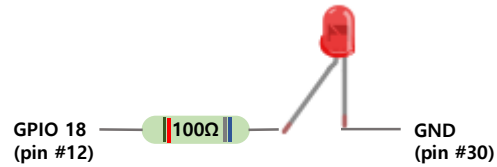
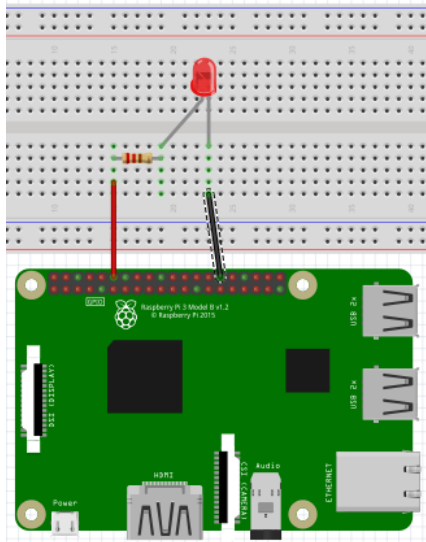
출력이 정전압 아날로그 신호처럼 작동하는 것처럼 보일 수 있음

입력전압	상하비 (듀티 사이클)	출력 전압	비고	비고
5V	2:8	1V	$5V \times 0.2 = 1.0V$	
	5:5	2.5V	$5V \times 0.5 = 2.5V$	
	8:2	4V	$5V \times 0.8 = 4.0V$	

PWM(Pulse Width Modulation) Analog Output

```
pwm = GPIO.PWM(pin, Hz)          # 주파수 설정
pwm.start(duty_cycle)             # 0 ~ 100%, 1주기에 High 비율
pwm.ChangeDutyCycle(duty_cycle)
pwm.stop()
```

[과제] PWM을 이용하여 LED의 밝기가 점점 밝아졌다가 어두어지도록 프로그램



- LED의 Anode(+) 핀은 65 Ω 이상의 저항을 통해 라즈베리파이 GPIO 18(pin #12)에 연결
- LED의 Cathod(-) 핀은 라즈베리파이 GND(pin #30)에 연결


```
import RPi.GPIO as GPIO
import time

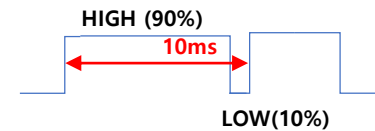
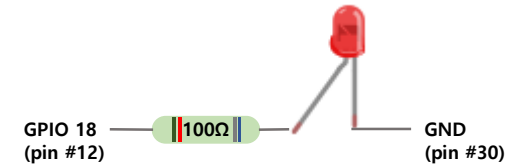
try:
    pin = 18
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(pin, GPIO.OUT)

    pwm = GPIO.PWM(pin, 100) # 주파수 100Hz
    pwm.start(0)             # duty ratio : 0%

    while True:
        for i in range(0, 101):
            pwm.ChangeDutyCycle(i)
            time.sleep(0.05)

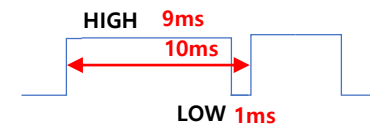
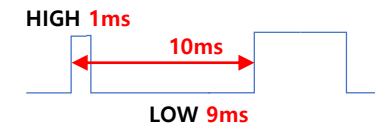
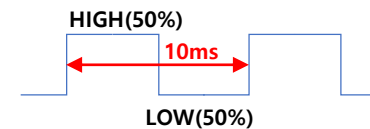
        for i in range(100, -1, -1):
            pwm.ChangeDutyCycle(i)
            time.sleep(0.05)

finally:
    pwm.stop()
    GPIO.cleanup()
```



주파수 : 100Hz (주기 10ms)

상하비 : (9 : 1) / Duty Cycle 90%



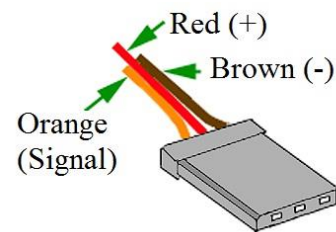
sg90 서버 모터 제어

SG90 서보(servo) 모터

- 특정위치로 이동하거나 특정한 수치(속도) 만큼 가동시킬 때, 모터로 부터의 피드백을 통해 정확히 제어할 수 있는 구조를 갖춘 모터
- 서보모터는 무한정 회전이 가능한 DC 모터와 달리 회전반경이 정해져 있음
- 일반적으로 소보 모터는 0 ~ 180도 범위에서 작동
- 대부분의 서보 모터는 PWM 펄스 주기와 듀티비(상하비)를 이용하여 회전 각도 조절

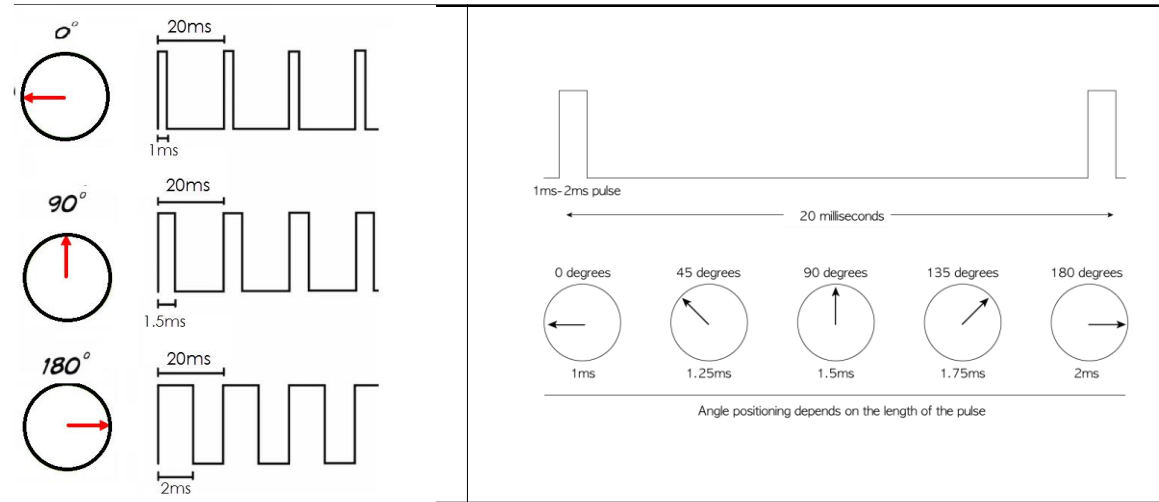


SG90 서보(servo) 모터

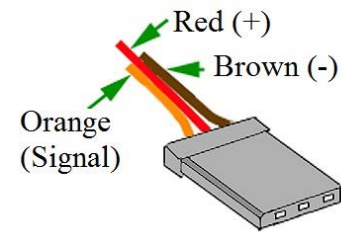
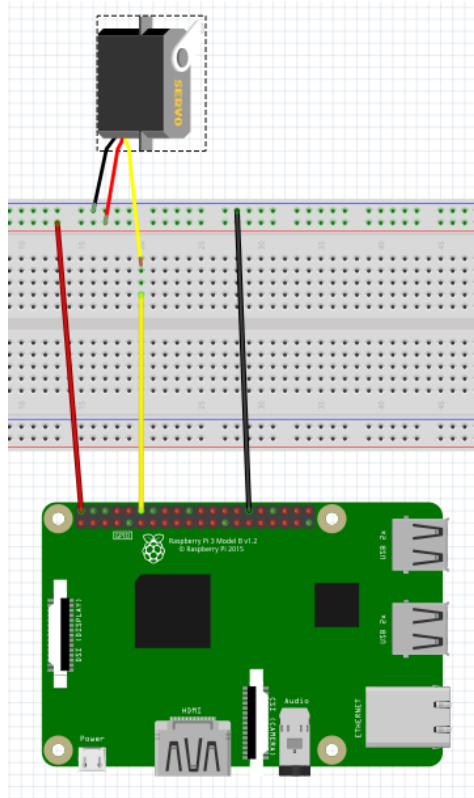


- 붉은 색과 검정색은 전원(5V), 노란색은 동작제어 신호
- 180도 각도 별 회전 제어 가능
- PWM 펄스 주기와 듀티비(상하비)를 이용하여 회전 각도 조절
 - PWM 주파수 : 50Hz (20ms)
 - 1ms(-90도), 1.5ms(0도), 2ms(+90도)

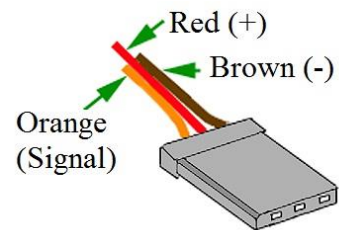
SG90 서보(servo) 모터



- 서보모터의 파형 주기는 **20ms (50Hz)**이며, 입력 파형의 High 값에 따라 움직이는 각도가 달라짐
 - ※ PCM 파형 주파수는 50 Hz : 1초(1000ms)에 50회 (1000ms / 50 = 20ms)
 - ※ 실습에 사용하는 sg90서버 모터는, **0.6ms**와 **2.5ms** HIGH 값을 주어야 0도에서 180도 범위를 움직임



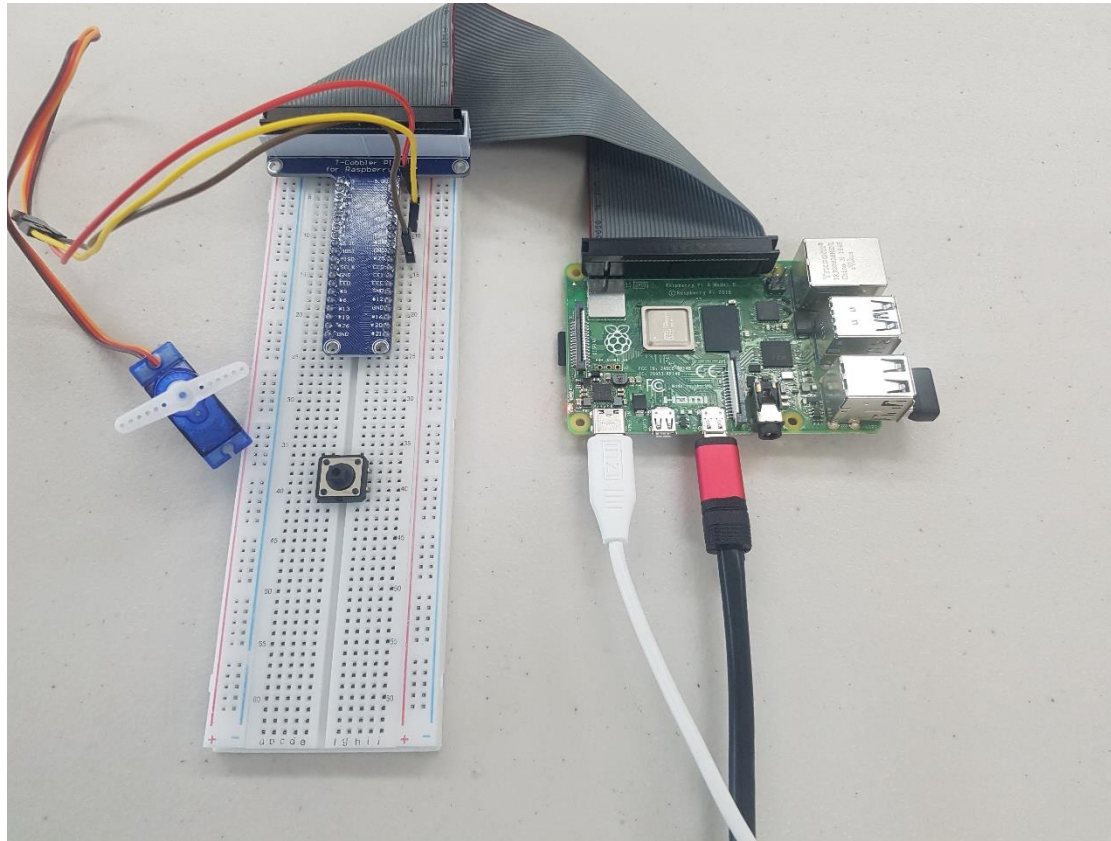
- Red(+) : 5.0V
- Brown(-) : GND
- Orange(PWM 시그널) : GPIO18



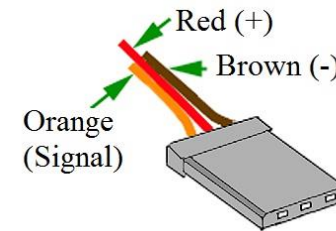
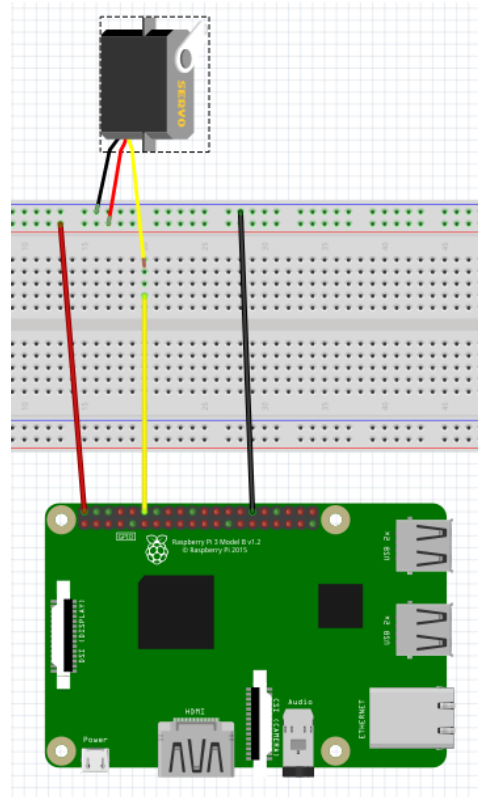
- Red(+) : 5.0V
- Brown(-) : GND
- Orange(PWM 시그널) : GPIO18

전기사용량 주의

- 모터 구동을 위해서는 많은 전기 사용
- 전기 사용량이 많은 디바이스 연결시
 - 전기부족으로 시스템 다운
 - 전지 등의 오동작 가능
- 모터를 활용한 프로젝트시 외부 전원 공급장치 활용



[과제] 사용자 입력 값이 0이면 0도 위치로, 1이면 90도 위치로, 2이면 180도 위치로
SG90 모터가 이동하도록 제어하시오.



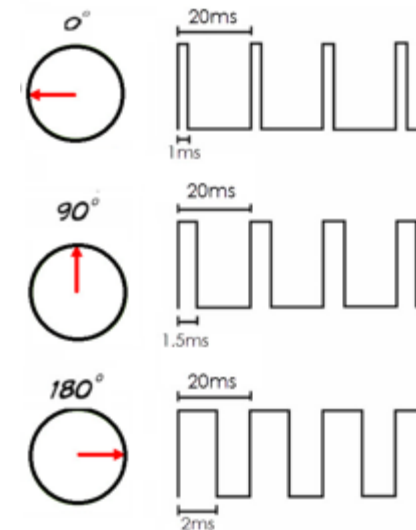
- Red(+) : 5.0V
- Brown(-) : GND
- Orange(PWM 시그널) : GPIO18

```
import RPi.GPIO as GPIO
import time
try:
    pin = 18
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(pin, GPIO.OUT)

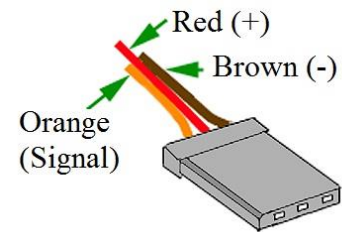
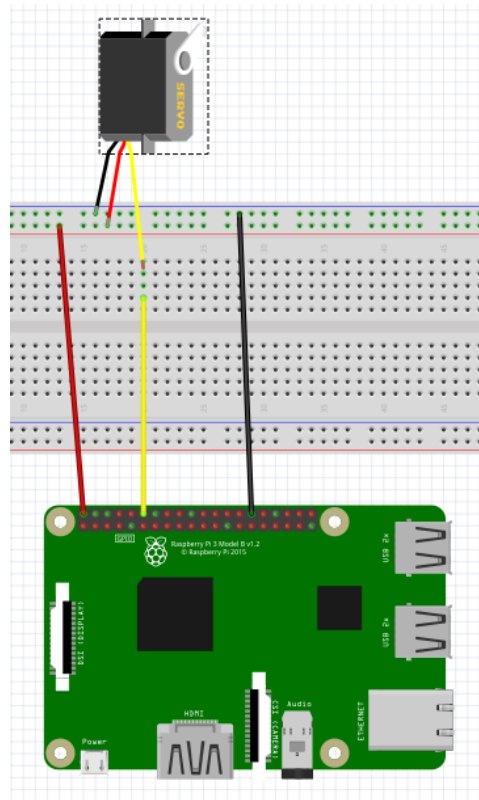
    pwm = GPIO.PWM(pin, 50) # 주파수 50Hz
    pwm.start(5)            # duty ratio : 5%

    while True:
        val = input('1: -90, 2: 0, 3: +90 > ')
        if val == '1':
            pwm.ChangeDutyCycle(2.5) # -90 degree, 20ms * 2.5% = 0.5ms
        elif val == '2':
            pwm.ChangeDutyCycle(7.5) # 0 degree, 20ms * 7.5% = 1.5ms
        else:
            pwm.ChangeDutyCycle(12.5) # +90 degree, 20ms * 12.5% = 2.5ms

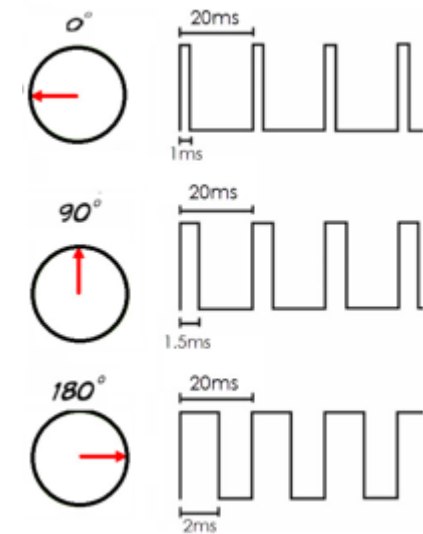
finally:
    pwm.stop()
    GPIO.cleanup()
```



[과제] SG90 모터가 0도에서 180도까지 회전한 후, 다시 180도에서 0도까지 회전하도록 제어하시오.



- Red(+) : 5.0V
- Brown(-) : GND
- Orange(PWM 시그널) : GPIO18



초음파 센서 제어

HC-SR04 초음파센서



4개 핀으로 구성

VCC	Trig	Echo	GND
5V	초음파 발사 요청	장애물 왕복시간	GND

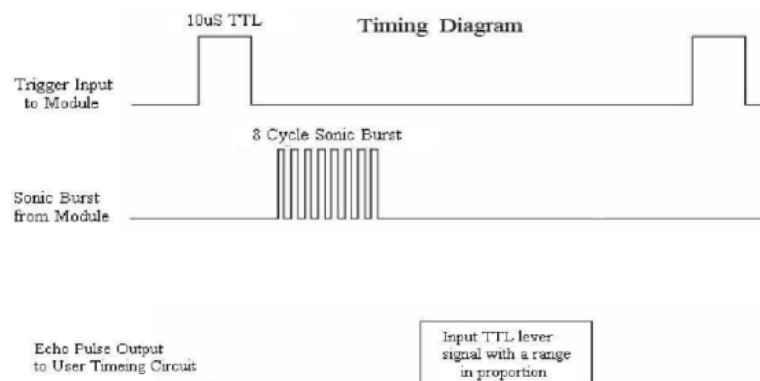
- ① Trig 핀에 10us 동안 HIGH 신호를 주어 HC-SR04가 초음파 발사를 요청
- ② HC-SR04는 8개 사이클의 초음파 발사
- ③ HC-SR04는 초음파가 장애물에 부딪쳐서 돌아오는 시간을 측정하고, 측정시간만큼 echo 핀에 HIGH 신호 유지
- ④ Echo 핀의 HIGH 신호 유지시간(왕복시간)을 측정하여 거리 계산

HC-SR04 초음파센서



4개 핀으로 구성

VCC	Trig	Echo	GND
5V	초음파 발사 요청	장애물 왕복시간	GND



HC-SR04 초음파센서



거리측정

VCC	Trig	Echo	GND
5V	초음파 발사 요청	장애물 왕복시간	GND

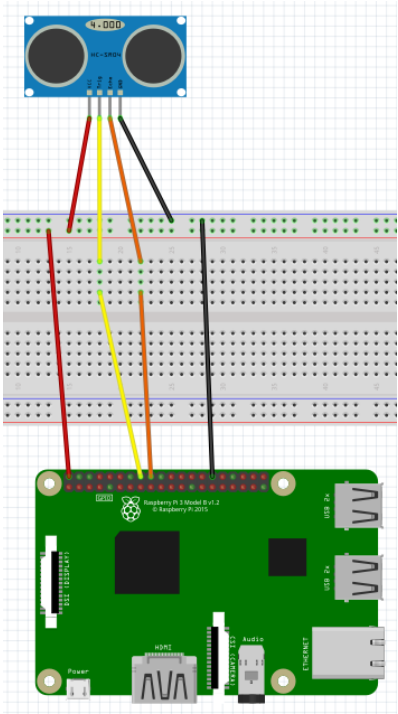
① 초음파 속도는 1초에, 약 340m (34,000cm) 이동

② 즉, 초음파는 1초 동안 34,000cm 이동



물체와의 거리(cm) = (echo_측정시간 * 34,000 / 2)
= echo_측정시간 * 17000

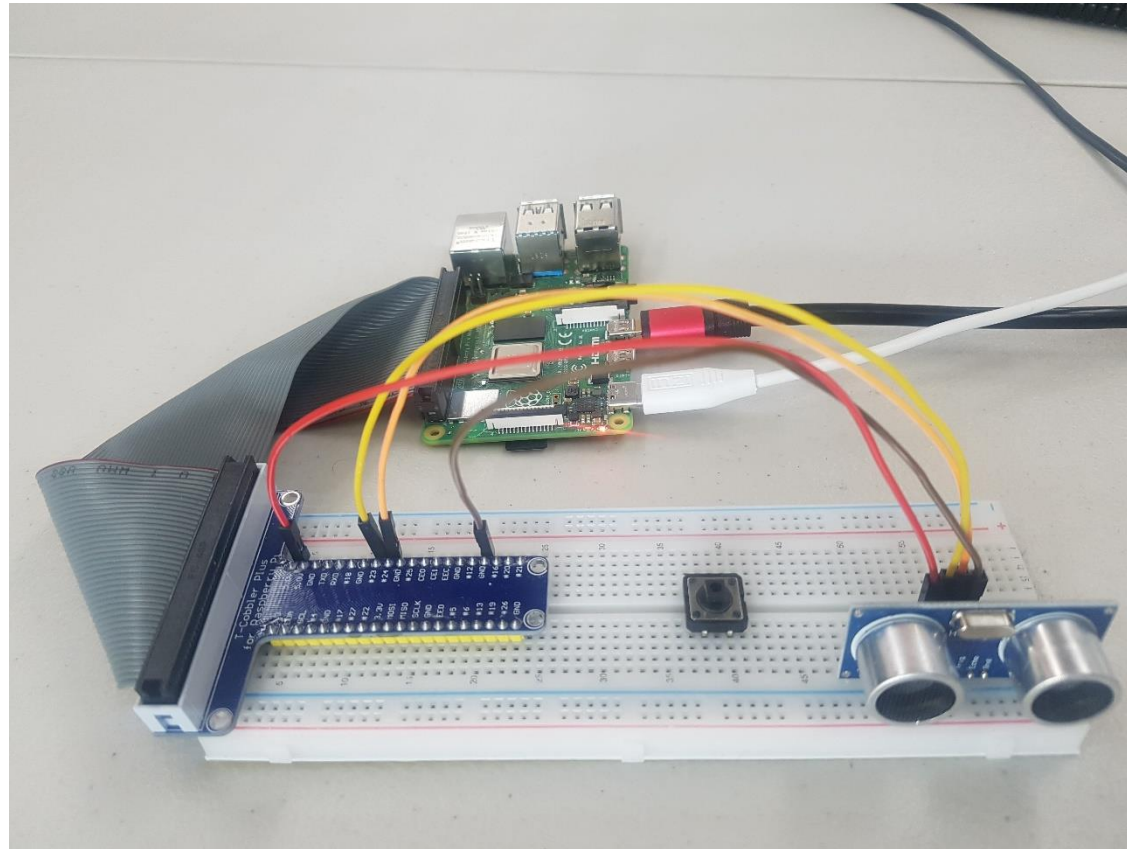
HC-SR04
초음파센서



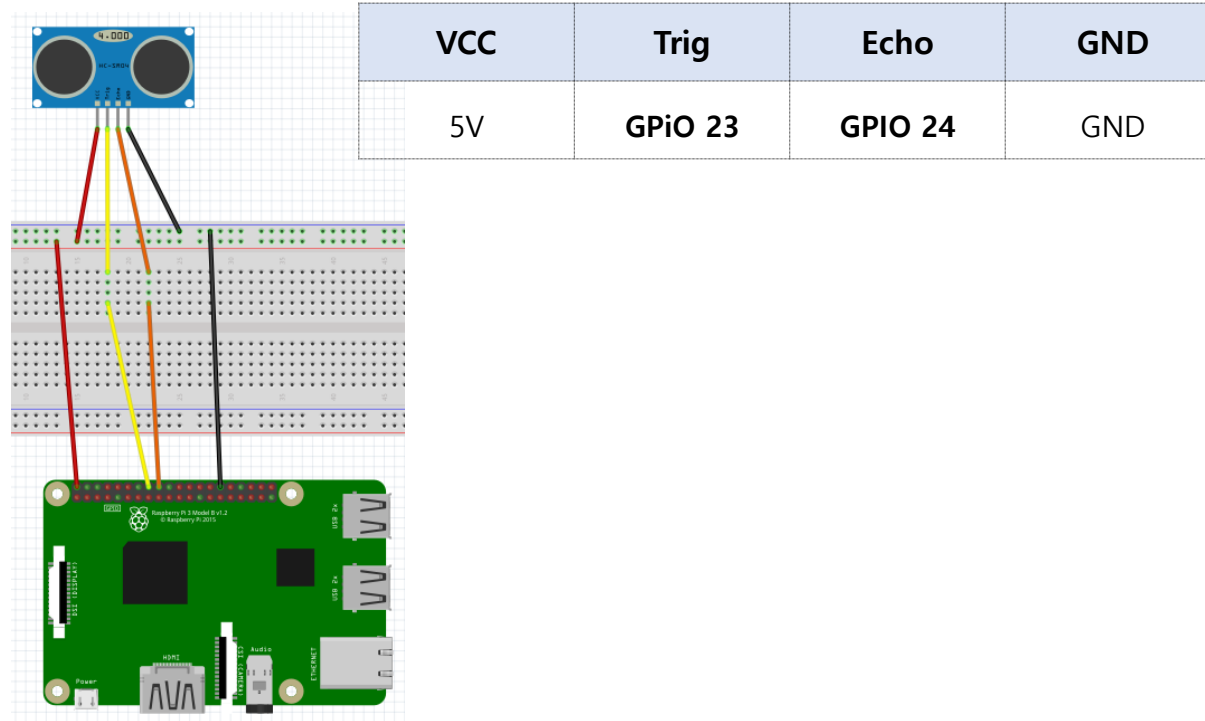
회로 구성

VCC	Trig	Echo	GND
5V	GPIO 23	GPIO 24	GND





[과제] 초음파 센서를 이용하여 물체와의 거리를 측정하시오.



```
import RPi.GPIO as GPIO
import time
trig_pin = 18
echo_pin = 23
distance = 0
start_time = 0
end_time = 0
```

try :

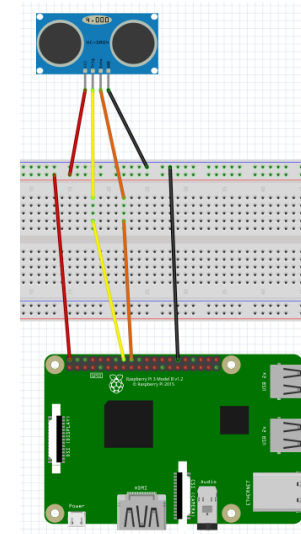
```
GPIO.setmode(GPIO.BCM)
GPIO.setup(trig_pin, GPIO.OUT)
GPIO.setup(echo_pin, GPIO.IN)

while True :
    GPIO.output(trig_pin, False)
    time.sleep(0.2)
    GPIO.output(trig_pin, True)
    time.sleep(0.00001) # set HIGH for 10us
    GPIO.output(trig_pin, False)
    while GPIO.input(echo_pin) == 0 :
        start_time = time.time()
    while GPIO.input(echo_pin) == 1 :
        end_time = time.time()
    travel_time = end_time - start_time
    distance = travel_time * 17000
    distance = round(distance, 2)
    print 'Distance: %d cm' % distance
```

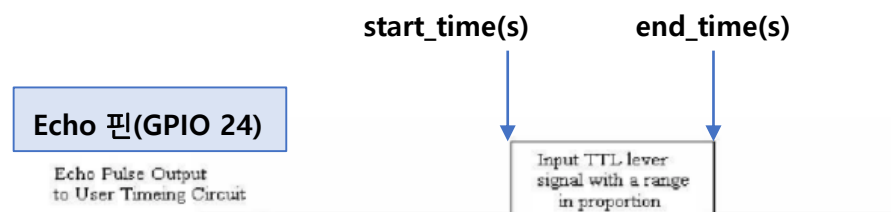
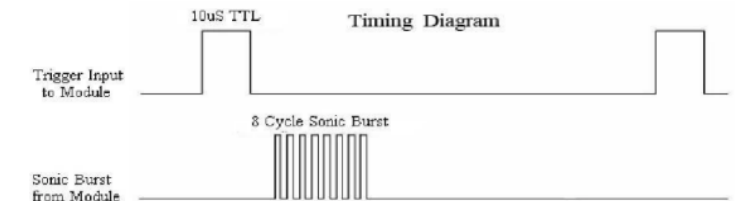
finally:

```
print 'Clean up'
GPIO.cleanup()
```

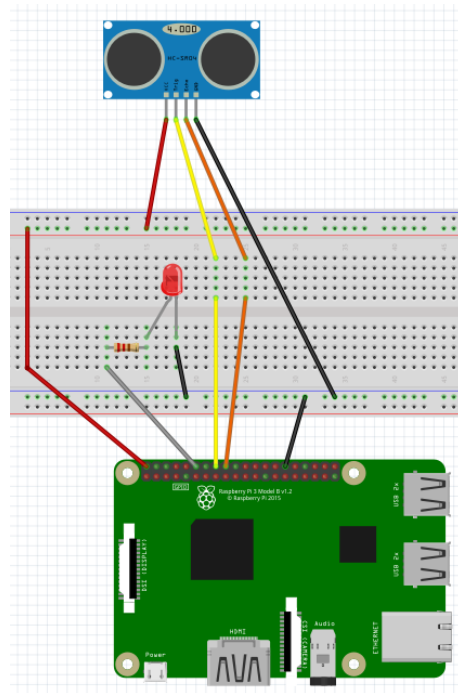
$$\begin{aligned} \text{물체와의 거리(cm)} &= (\text{echo_측정시간} * 34,000 / 2) \\ &= \text{echo_측정시간} * 17000 \end{aligned}$$



VCC	Trig	Echo	GND
5V	GPIO 18	GPIO 23	GND



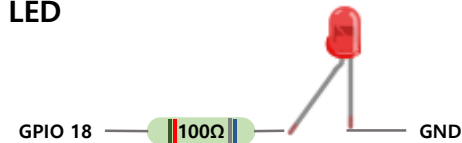
[과제] 물체가 10cm 이내로 접근하며 LED에 불이 들어오도록 프로그램하시오.

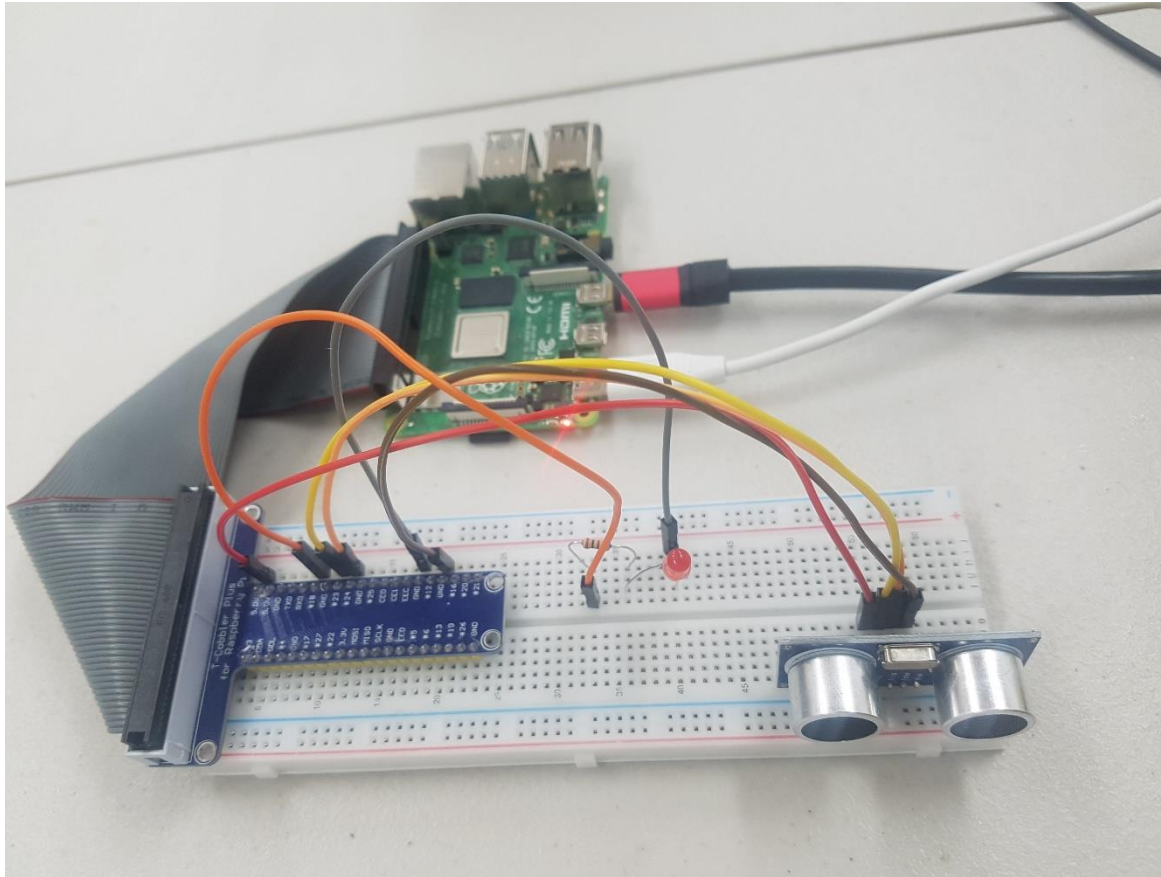


HC-SR04

VCC	Trig	Echo	GND
5V	GPIO 23	GPIO 24	GND

LED





VCC	Trig	Echo	GND
5V	GPIO 23	GPIO 24	GND

출력용 핀

입력용 핀



출력용 핀

GPIO 18

100Ω

GND



distance

distance

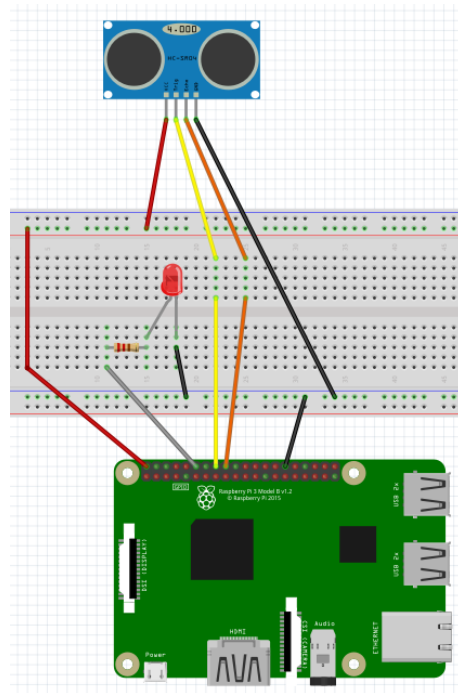
< 10cm

>= 10cm

GPIO.output(led_pin, True)

GPIO.output(led_pin, False)

[과제] 2명이 짝이 되어 물체가 10cm 이내로 접근하며 원격지 LED에 불이 들어오도록 프로그램하시오.



HC-SR04

VCC	Trig	Echo	GND
5V	GPIO 23	GPIO 24	GND

LED

