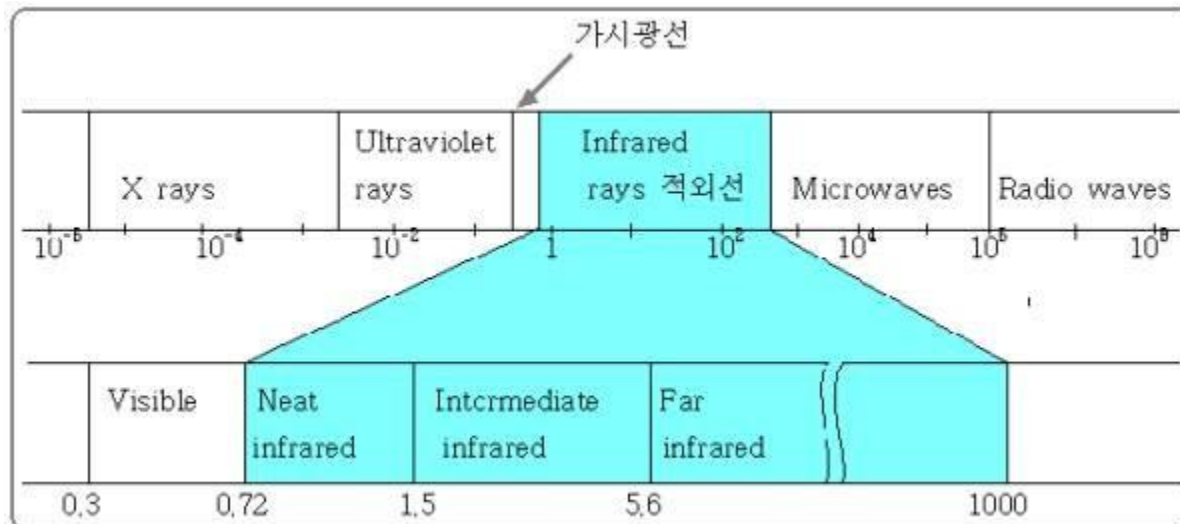


CH. 3. 라즈베리파이 센서 제어

Section 01 적외선 센서

□ 적외선이란?

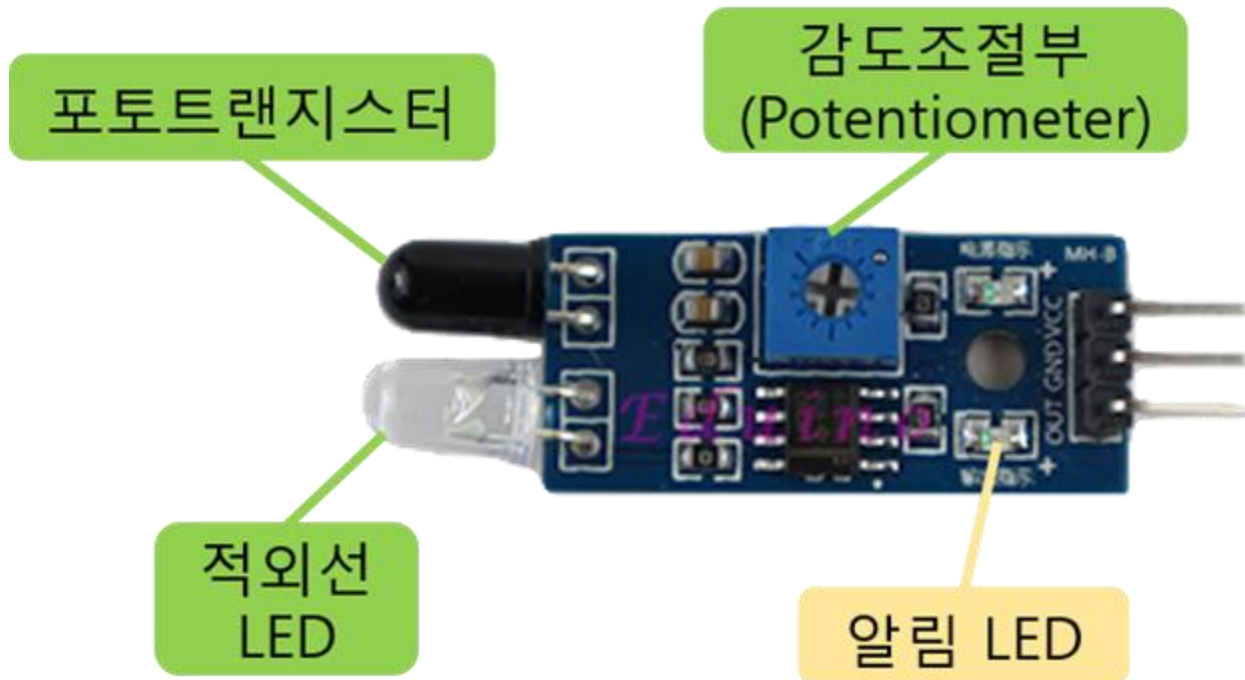
- 적외선은 가시광선이 빨,주,노,초,파,남,보로 프리즘을 통해 나타날 때, 빨강색 바깥쪽에 나타난다고 해서 적외선이라고 부르게 되었다.
- 태양이나 물체가 내는 복사열의 대부분은 이 적외선으로 이루어져있으므로 적외선을 열선이라고 한다.
- 적외선의 파장범위는 가시광선의 장파장 끝의 $0.76\sim 0.8\mu\text{m}$ 를 하단으로 하고, 상단은 1mm정도까지이다.



Section 01 적외선 센서

□ 적외선센서 사용방법

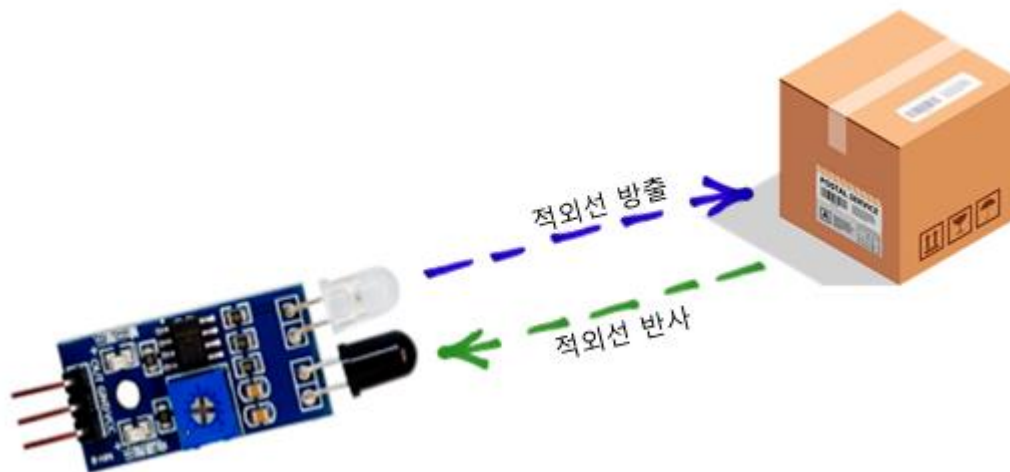
- 적외선도 빛의 한 종류이기 때문에 빛의 성질을 가지고 있다.
- 적외선은 직진하며 물체에 닿으면 반사하는 성질을 가지고 있다.
- 적외선 장애물 감지 센서는 적외선을 보내는 부분과 반사된 적외선을 감지하는 감지 부분으로 구성되어 있다.



Section 01 적외선 센서

□ 적외선센서 사용방법

- 적외선 LED에서 적외선을 보내게 되고, 물체에 닿아 반사되는 빛은 포토 트랜지스터에서 감지하게 된다.
- 적외선 장애물 감지 센서는 앞에 장애물이 놓이게 되면 센서에서 보낸 적외선이 장애물에 반사되어 수신부에 들어가게 되고, 이를 통해 센서가 장애물을 인식하게 된다.
- 10cm부근에서 최대의 전압값을 갖다가 거리가 멀어질수록 다시 전압이 감소하는 형태를 띄고 있다.



Section 01 적외선 센서

□ 적외선센서 수광부

- 적외선 센서의 수광부의 역할은 적외선을 인식하는 것이다.
- 우선 적외선을 인식하기 위해서는 포토트랜지스터(Phototransister)라는 것이 필요하다.
- 포토트랜지스터는 적외선을 인식하면 전류를 흐르게 하는 특징을 가지고 있다.
- 그러니까 적외선 빛이 베이스(Base)가 되고 긴다리가 콜렉터(Collector), 짧은 다리가 에미터(Emitter)가 되는 트랜지스터이다.



Section 01 적외선 센서

□ 적외선센서 발광부

- 적외선 LED를 켜는데 저항이 필요한데, 통상 라즈베리파이로 LED를 켜는데 220옴을 쓰지만 IR LED는 220옴을 이용해서 켜면 너무 약하게 켜진다.
- 물론 적외선 이기 때문에 우리 눈으로는 안보이는데, 핸드폰 카메라로 보면 불빛을 눈으로 볼 수 있다.
- 자, 그럼 저항을 몇 옴을 써야 할까?

Absolute maximum ratings

Characteristic	Symbol	Ratings	Unit
Power Dissipation	P_D	150	mW
Forward Current	I_F	100	mA
*1Peak Forward Current	I_{FP}	1	A
Reverse Voltage	V_R	4	V
Operating Temperature	T_{opr}	-25~85	°C
Storage Temperature	T_{stg}	-30~100	°C
*2Soldering Temperature	T_{sol}	260°C for 5 seconds	

*1.Duty ratio = 1/16, Pulse width = 0.1ms

*2.Keep the distance more than 2.0mm from PCB to the bottom of IRED package

Electrical Characteristics

Characteristic	Symbol	Test Condition	Min.	Typ.	Max.	Unit
Forward Voltage	V_F	$I_F = 50\text{mA}$	-	1.3	1.7	V
Radiant Intensity	I_E	$I_F = 50\text{mA}$	30	70	-	mW/Sr



Section 01 적외선 센서

□ 적외선센서 발광부

- 위 표에서 Forward Current(100mA)가 IR LED의 사용 전류이고, Forward Voltage(1.3V~1.7V)가 사용 전압이다.
- 그럼 $V = I * R$ 이라는 옴의 법칙에 따라 필요한 저항 값을 찾을 수 있다.
- 전류가 100mA(= 0.1A)이고 전압은 1.3V~ 1.7V이니까 그 중간 정도 인 1.5V로 보고 계산을 하면, 라즈베리파이가 3.3V가 나오니까 3.3V에서 IR LED가 쓰는 1.5V를 빼면, 1.8V가 나온다.
- 그것을 사용 전류 0.1(단위를 맞춰야 하니까 100mA를 0.1A로 바꾼다.)로 나누면 된다.
- 식으로 다시 쓰면 :

$$R = \frac{3.3V - 1.5V}{0.1A} = 18\Omega$$

Section 01 적외선 센서

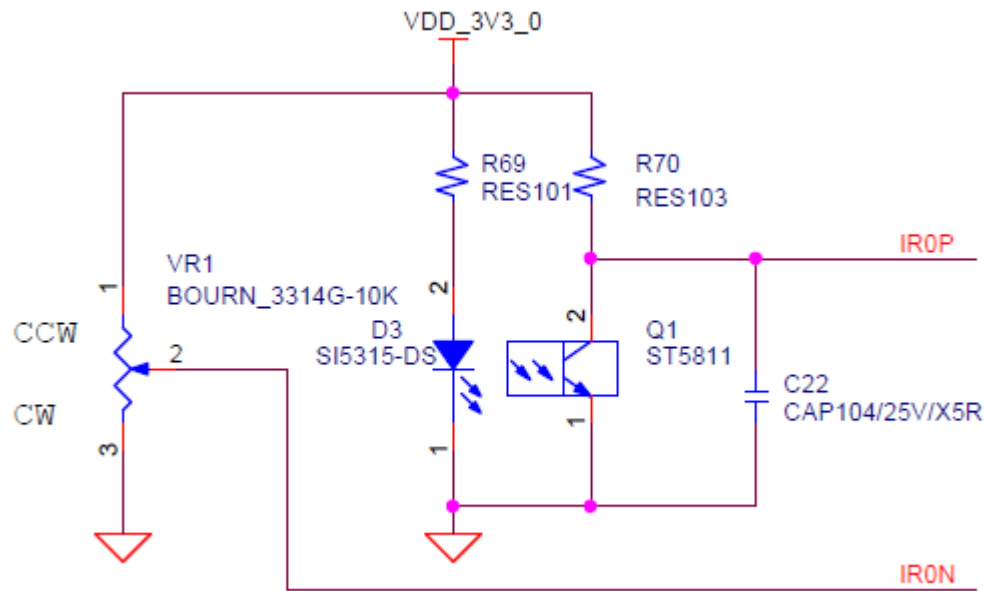
□ 적외선센서 발광부

- 결국 18옴을 쓰면 가장 밝게 IR LED를 켤 수 있다.
- 만약 여기서 조금만 전류나 전압이 세지면 LED는 바로 망가지는 버랑끝 수치이니까 약간 안전하게 수치를 낮출 필요가 있다.
- 그래서 30옴에서 100옴 사이에 가지고 있는 저항 중 적당한 것을 골라서 쓰면 된다.
- 18옴 보다 작은 저항을 쓰면 IR LED가 타버려서 못쓰게 되고 100옴 보다 큰 저항을 쓰면 빛이 너무 약해서 실험이 잘 안된다.

Section 01 적외선 센서

□ 적외선센서 회로도

- 적외선 센서부의 회로도는 다음과 같다.



Section 01 적외선 센서

□ 파이썬 코드

- 다음과 같은 코드를 작성하여 적외선 센서의 측정의 값을 입력받아 보자.

```
1  from ctypes import *
2  import os
3  import time
4
5  WiringPi = CDLL("/home/pi/WiringPi/wiringPi/libwiringPi.so.2.70", mode=RTLD_G
  LOBAL)
6  swcar = cdll.LoadLibrary('home/pi/swcar/libswcar.so')
7
8  swcar.SIO_Init(1)
9
10 print('press ctrl + c to terminate program')
11
```

Section 01 적외선 센서

□ 파이썬 코드

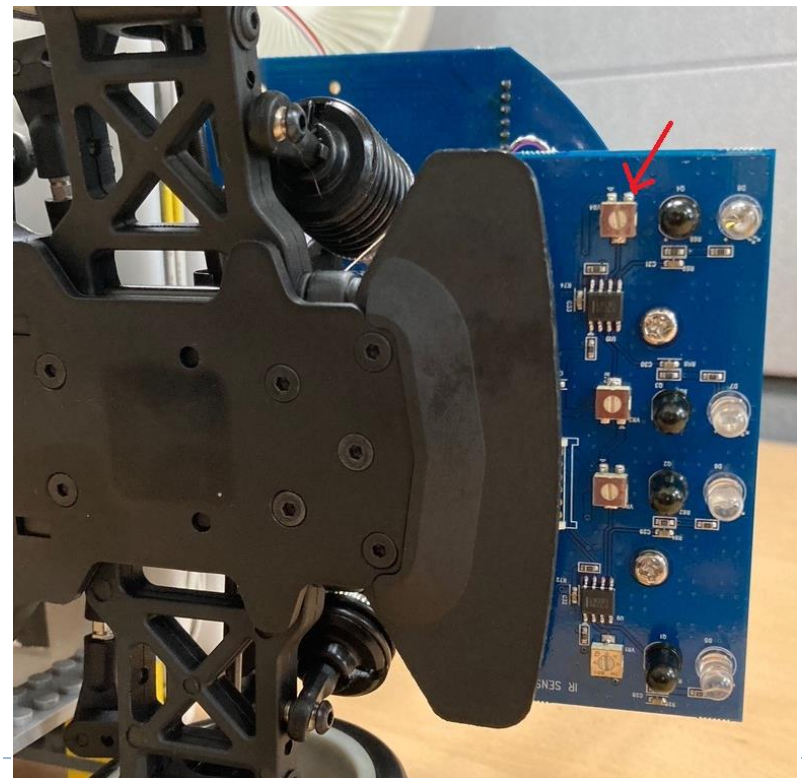
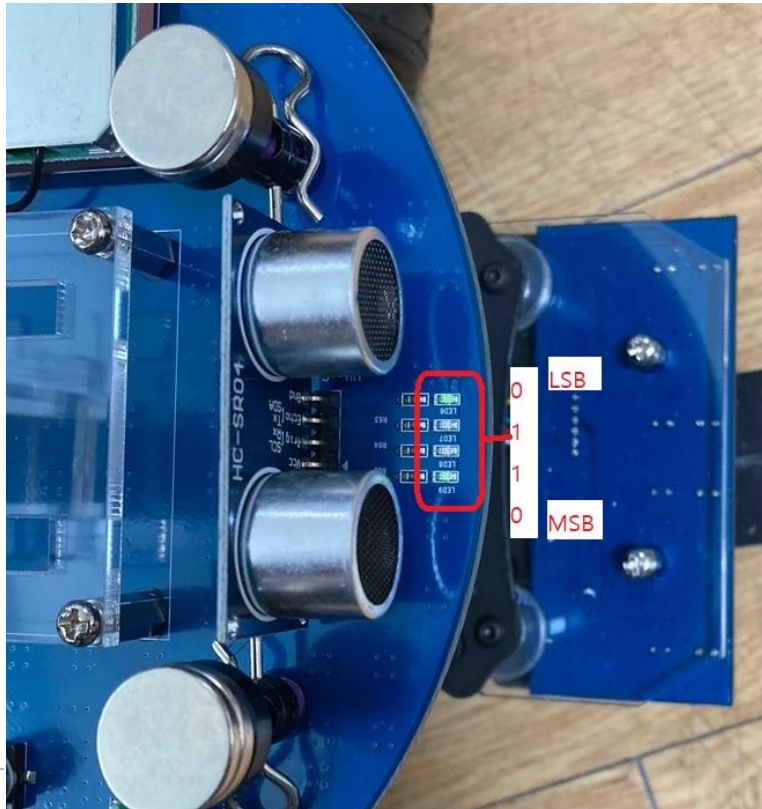
- 다음과 같은 코드를 작성하여 적외선 센서의 측정의 값을 입력받아 보자.

```
11
12  try:
13      while(True):
14          ir = swcar.SIO_ReadIR()
15          print("ir = ", bin(ir))
16          time.sleep(1)
17
18  except KeyboardInterrupt:
19      pass
```

Section 01 적외선 센서

□ 파이썬 코드

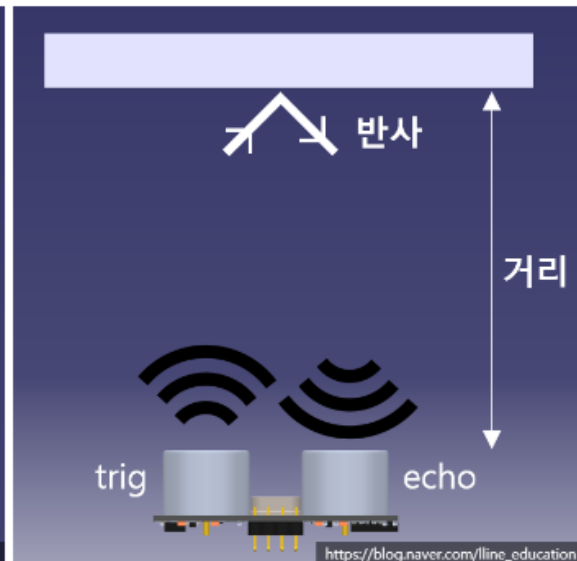
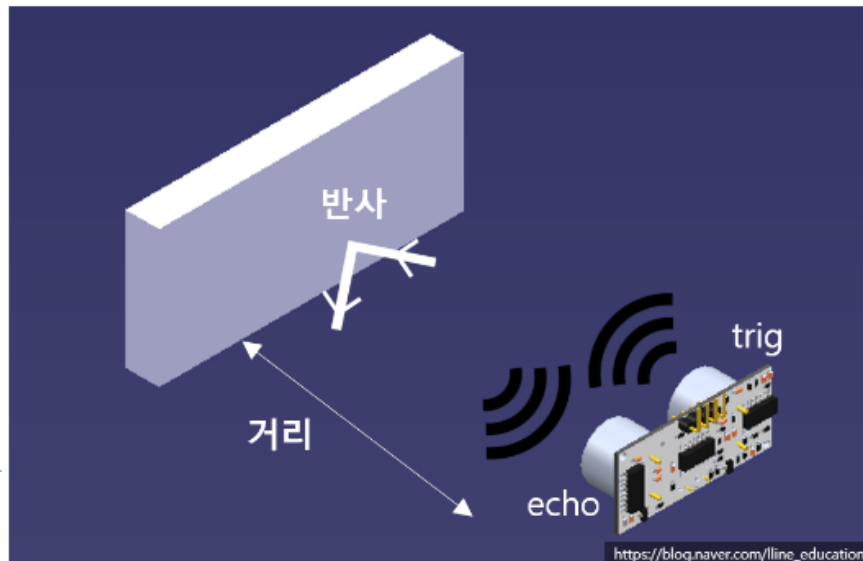
- 코드를 실행하면 화면에 00이 출력된다.
- 적외선 센서는 자동차 키트 전면부에 장착되어 있는데, 그 밑에 색이 있는 물체를 집어넣으면 출력되는 숫자가 바뀌는 것을 확인할 수 있다.



Section 02 초음파 센서

□ 초음파 센서 US-015

- SmartCar에 사용되는 US-015는 거리 측정용으로 많이 사용하는 HC-SR04와 모양도 사용방법도 동일하지만 보다 나은 성능을 가진 고정밀 초음파 거리센서이다.
- US-015 센서는 로봇의 눈처럼 생겼다.
- 2개의 초음파 장치 중 하나에서는 40kHz의 초음파를 발사하고 나머지 하나의 센서에서는 반사되어 되돌아오는 초음파를 감지한다.
- 이 시간차를 이용해 거리를 측정하는 것이다.



Section 02 초음파 센서

□ 초음파 센서 US-015

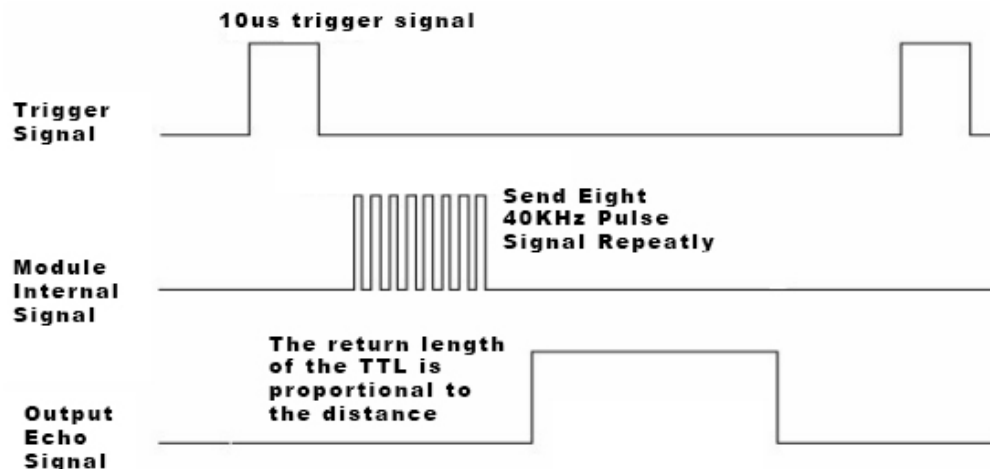
- 초음파 센서는 송신부와 수신부로 이루어져 있으며 송신부는 Transmitter 초음파를 발생하는 것이고, 수신부는 Receiver 초음파가 발생한 것이 물체에 부딪혀 다시 돌아오는 것이다.
- 이에 따라 초음파가 돌아오는 시간에 따라 물체와 초음파 사이에 거리를 알 수 있다.
- 공식으로 보자면 다음과 같다.

$$D(\text{거리}) = \frac{T(\text{시간})}{2} \times V_s(\text{속도})$$

Section 02 초음파 센서

□ 초음파 센서 US-015

- 측정은 다음과 같은 흐름으로 이루어진다.
 - 트리거 핀으로 10us 시간 동안 ON 신호를 주면 센서가 8개 펄스의 초음파를 발사한다.
 - 물체에서 반사된 초음파(echo)를 감지해 에코 핀을 통해 신호를 전달한다.
 - 초음파가 발사되는 순간 에코 핀은 ON 상태가 되고 반사파를 감지하는 순간 OFF로 변하는데, 이 시간차를 측정해서 해당 값을 초음파의 거리 속도 계산식에 넣어 거리를 구한다.



Section 02 초음파 센서

□ 파이썬 코딩

- 다음과 같은 코드를 작성하여 초음파 센서의 측정의 값을 입력받아 보자.

```
1  from ctypes import *
2  import os
3  import time
4
5  WiringPi = CDLL("/home/pi/WiringPi/wiringPi/libwiringPi.so.2.70", mode=RTLD_GLO
  OBAL)
6  swcar = cdll.LoadLibrary('home/pi/swcar/libswcar.so')
7
8  swcar.SIO_Init(0)
9
10 print('press ctrl + c to terminate program')
11
```


Section 02 초음파 센서

□ 파이썬 코딩

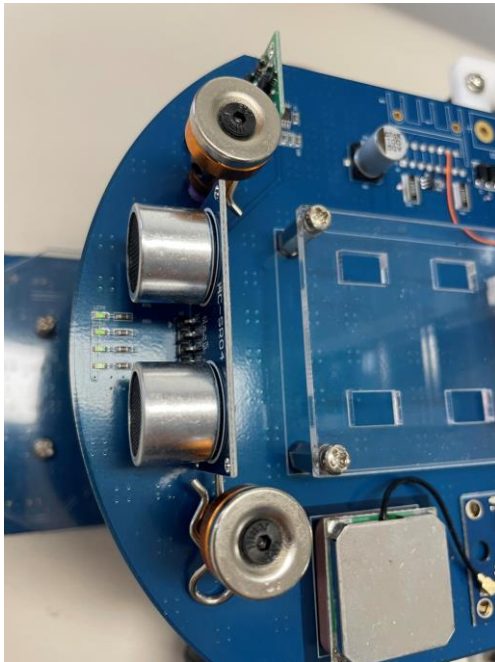
- 다음과 같은 코드를 작성하여 초음파 센서의 측정의 값을 입력받아 보자.

```
11
12 try:
13     while(True):
14         us_front = swcar.SIO_ReadDistUS(1)
15         us_rear = swcar.SIO_ReadDistUS(0)
16         print("front = ", us_front)
17         print("rear = ", us_rear)
18         time.sleep(1)
19
20 except KeyboardInterrupt:
21     pass
```

Section 02 초음파 센서

□ 파이썬 코딩

- 코드를 실행하면 콘솔창에 센서 측정값이 출력된다.
- 초음파 센서에 장애물을 배치하여 값의 변화를 확인한다.



전면부 초음파 센서



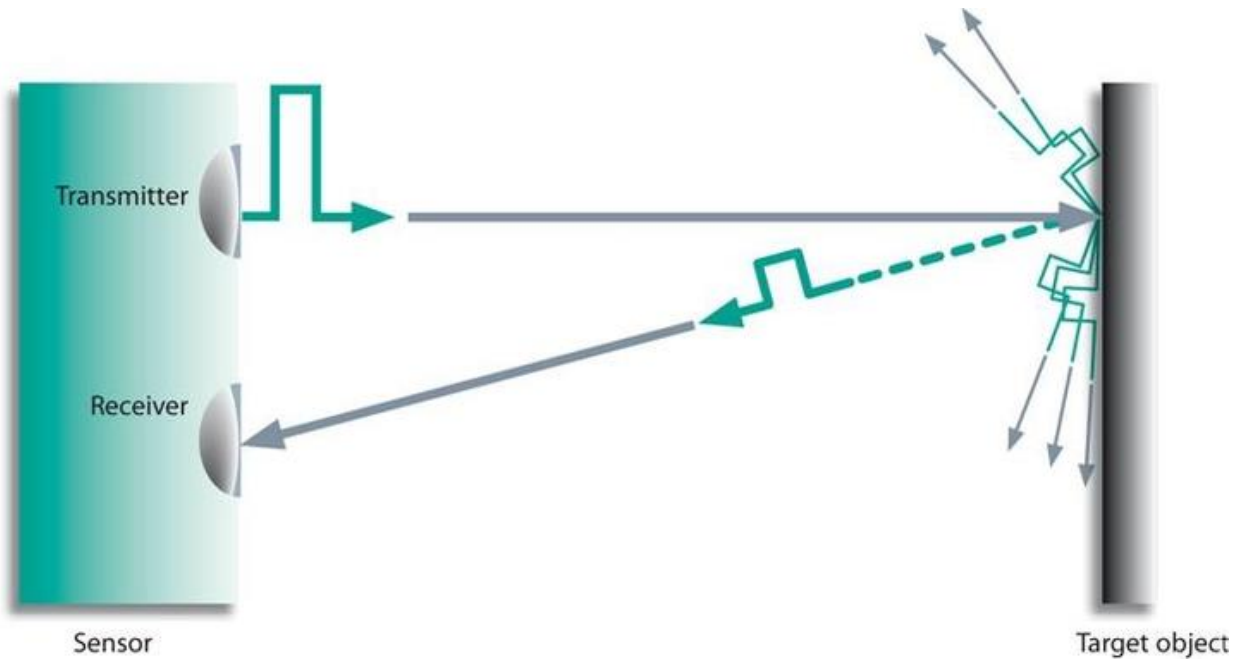
후면부 초음파 센서

```
Shell
front = 9999
rear = 657
front = 9999
rear = 657
front = 9999
rear = 657
front = 9999
rear = 661
front = 9999
rear = 657
front = 9999
rear = 661
```

콘솔 출력 결과

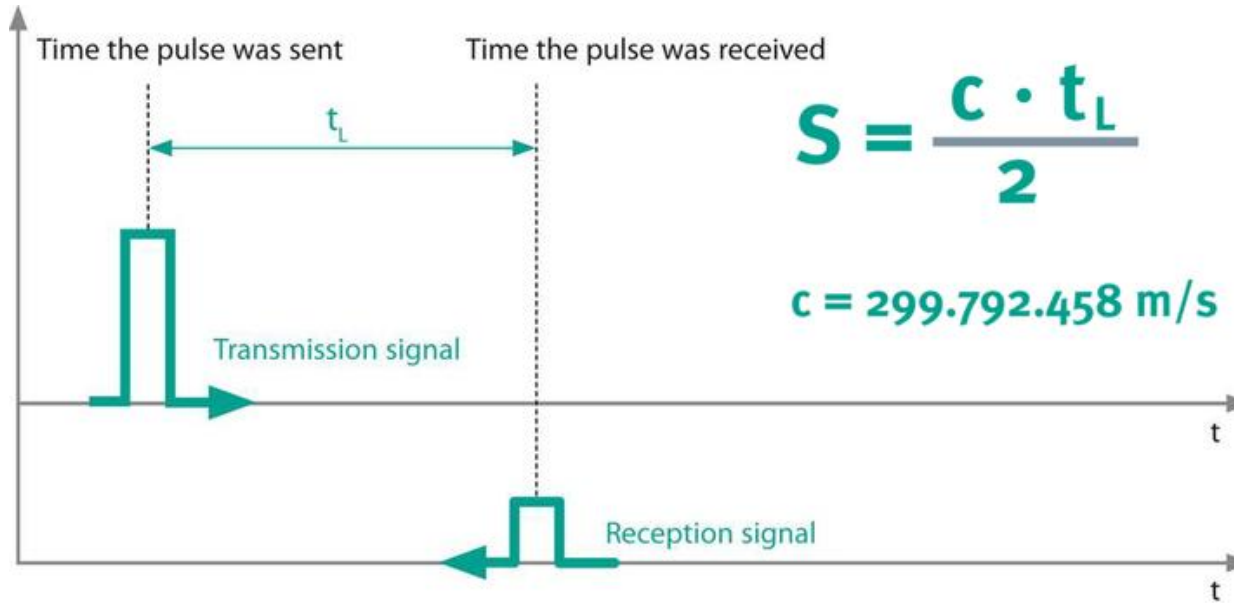
Section 03 레이저 센서

□ 레이저 센서의 거리 측정



Section 03 레이저 센서

□ 레이저 센서의 거리 측정



Section 03 레이저 센서

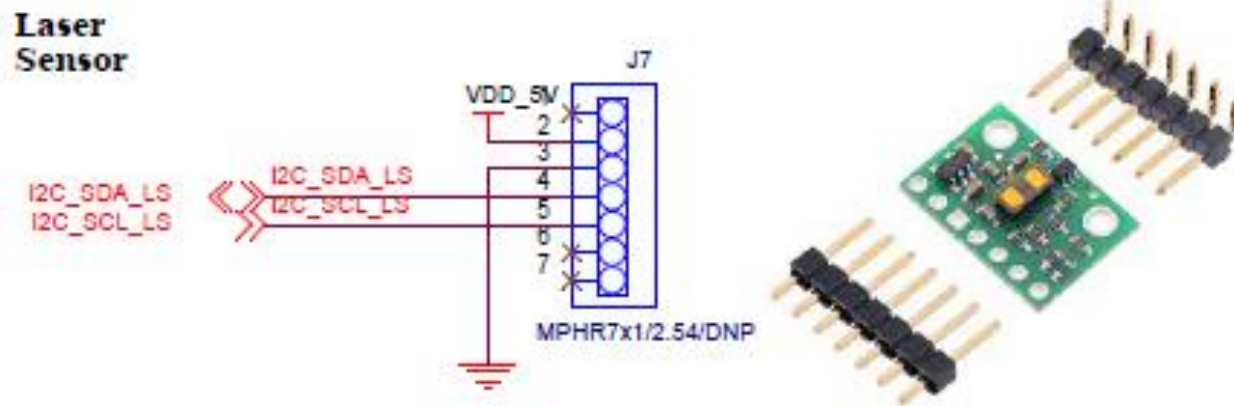
□ 레이저 센서의 거리 측정

- 레이저 센서와 타겟 사이의 거리는 $D = ct / 2$ 로 주어지며, 여기서 c 는 빛의 속도와 같고 t 는 미터와 타겟 사이의 왕복 시간에 해당한다.
- 레이저는 초점이 맞추어진 강렬한 광선, 보통 단일 주파수의 광선이다.
- 그들은 대기를 통해 상당히 일정한 속도로 이동하고 발산하기 전에 더 먼 거리를 이동하기 때문에 거리 측정에 매우 유용하다.
- 빛의 약화와 퍼짐이 거리 측정 센서의 효능을 감소시키지만, 레이저 광은 백색광과 같이 분산되기 어렵기 때문에 레이저 광이 강도를 잃지 않고 훨씬 더 먼 거리를 이동할 수 있다.
- 일반적인 백색광과 비교할 때 레이저 펄스는 대상에서 반사될 때 원래의 강도의 대부분을 유지한다.
- 이는 대상까지의 거리를 계산할 때 매우 중요하다.

Section 03 레이저 센서

□ 레이저 센서 모듈의 회로도

- 레이저 센서 모듈의 회로도는 다음과 같다.



Section 03 레이저 센서

□ 파이썬 코드

- 다음과 같은 코드를 작성하여 레이저 센서의 측정의 값을 입력받아 보자.

```
1  from ctypes import *
2  import os
3  import time
4
5  WiringPi = CDLL("/home/pi/WiringPi/wiringPi/libwiringPi.so.2.70", mode=RTLD_GLO
  BAL)
6  swcar = cdll.LoadLibrary('home/pi/swcar/libswcar.so')
7
8  swcar.SIO_Init(0)
9
10 print('press ctrl + c to terminate program')
11
```

Section 03 레이저 센서

□ 파이썬 코드

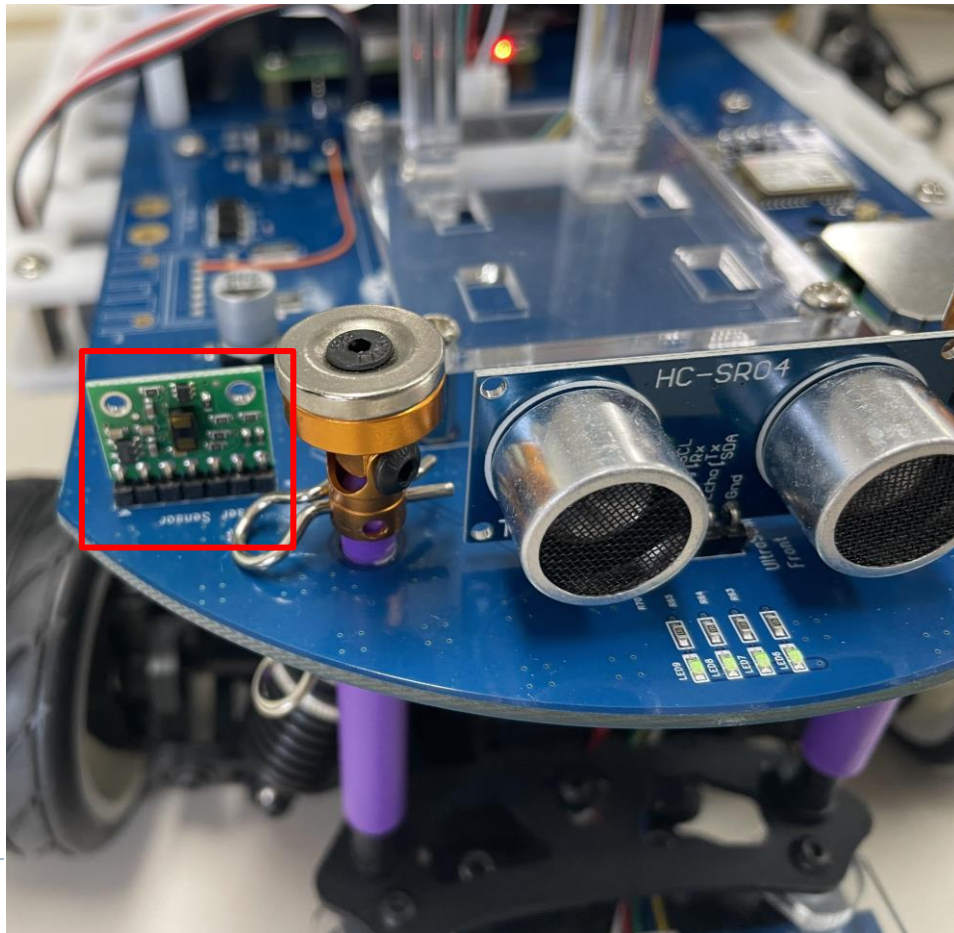
- 다음과 같은 코드를 작성하여 레이저 센서의 측정의 값을 입력받아 보자.

```
11
12 try:
13     while(True):
14         laser = swcar.SIO_ReadDistLS()
15         print("laser_dist = ", laser)
16         time.sleep(1)
17
18 except KeyboardInterrupt:
19     pass
```


Section 03 레이저 센서

□ 파이썬 코드

- 코드를 실행한 후 레이저 센서의 앞에 장애물을 배치하여 위치를 조절하며 출력값의 변경을 확인한다.

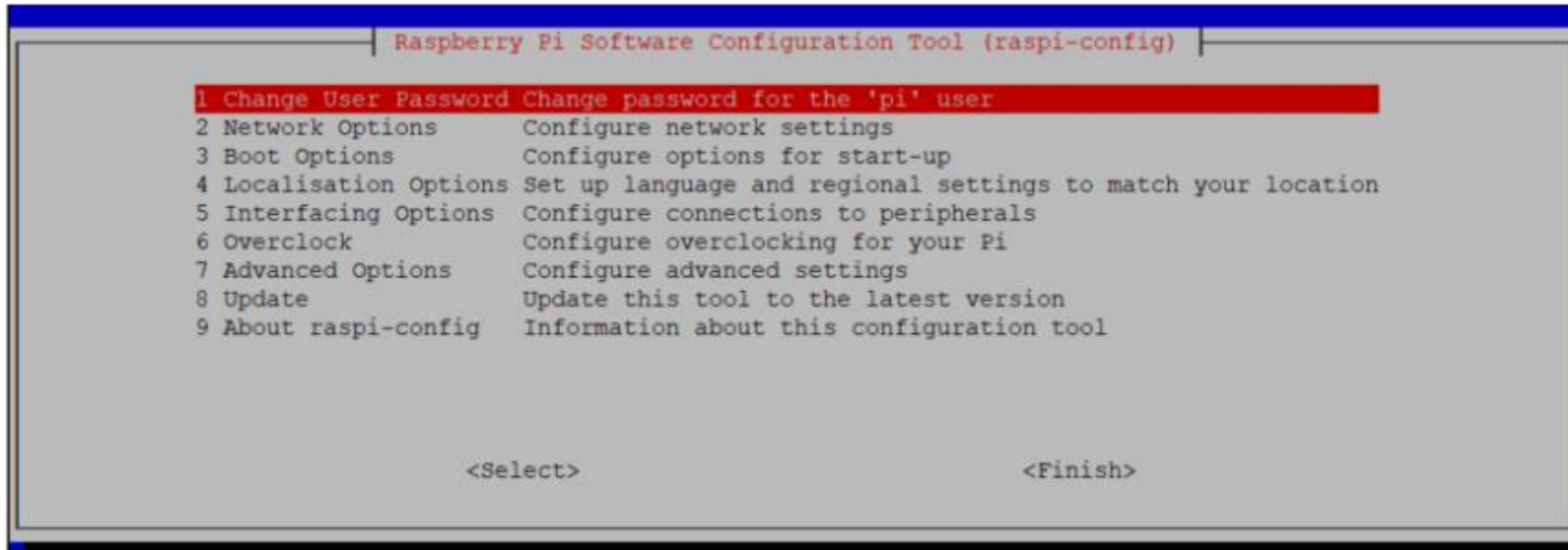


Shell	
laser_dist	= 9999
laser_dist	= 9999
laser_dist	= 140
laser_dist	= 155
laser_dist	= 156
laser_dist	= 199
laser_dist	= 243
laser_dist	= 264
laser_dist	= 9999
laser_dist	= 9999
laser_dist	= 9999
laser_dist	= 9999
laser_dist	= 9999

Section 04 가속도 센서

□ I2C 활성화

- 가속도 센서인 mpu6050을 사용하기 위해서는 우선적으로 라즈베리파이의 I2C를 활성화해야 한다.
- 터미널에서 `sudo raspi-config` 명령어를 실행



```
Raspberry Pi Software Configuration Tool (raspi-config)

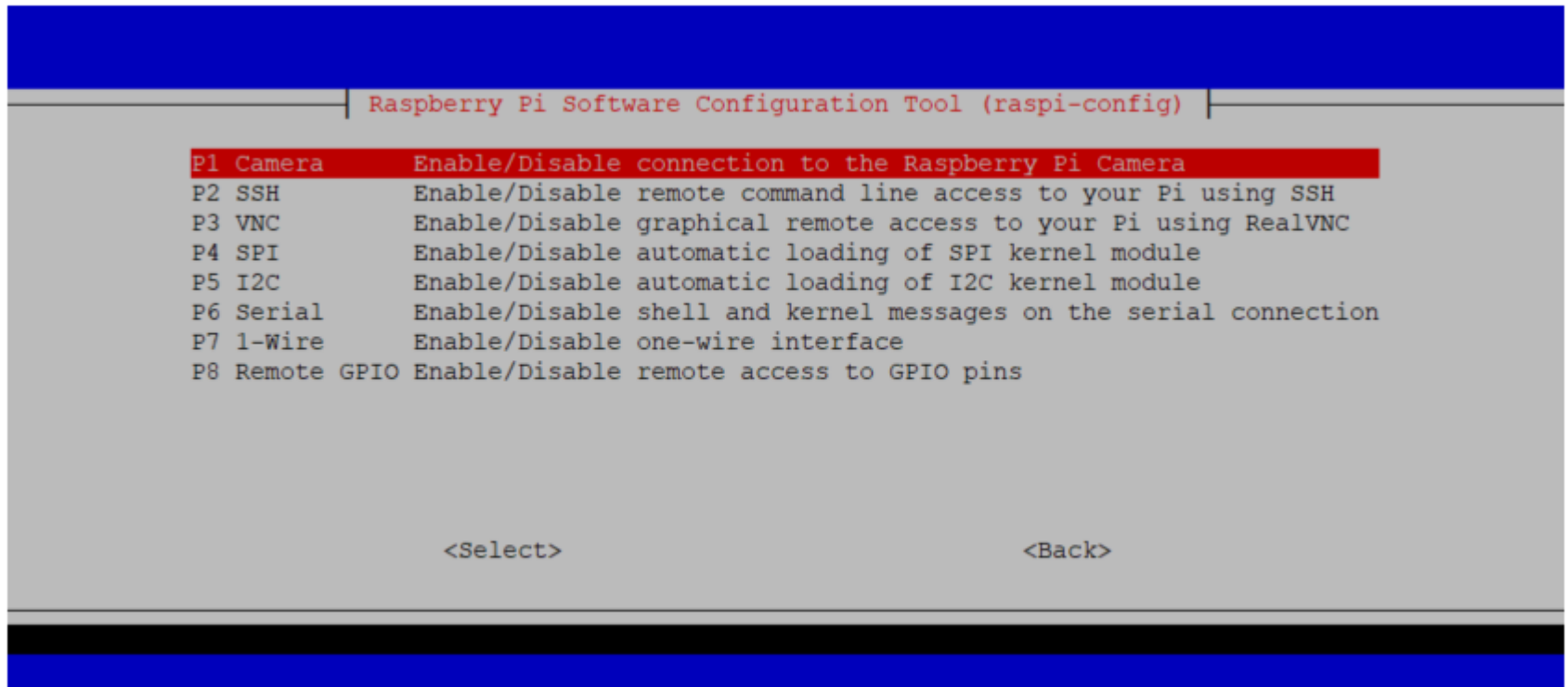
1 Change User Password Change password for the 'pi' user
2 Network Options       Configure network settings
3 Boot Options          Configure options for start-up
4 Localisation Options  Set up language and regional settings to match your location
5 Interfacing Options   Configure connections to peripherals
6 Overclock             Configure overclocking for your Pi
7 Advanced Options      Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config    Information about this configuration tool

<Select>                                <Finish>
```

Section 04 가속도 센서

□ I2C 활성화

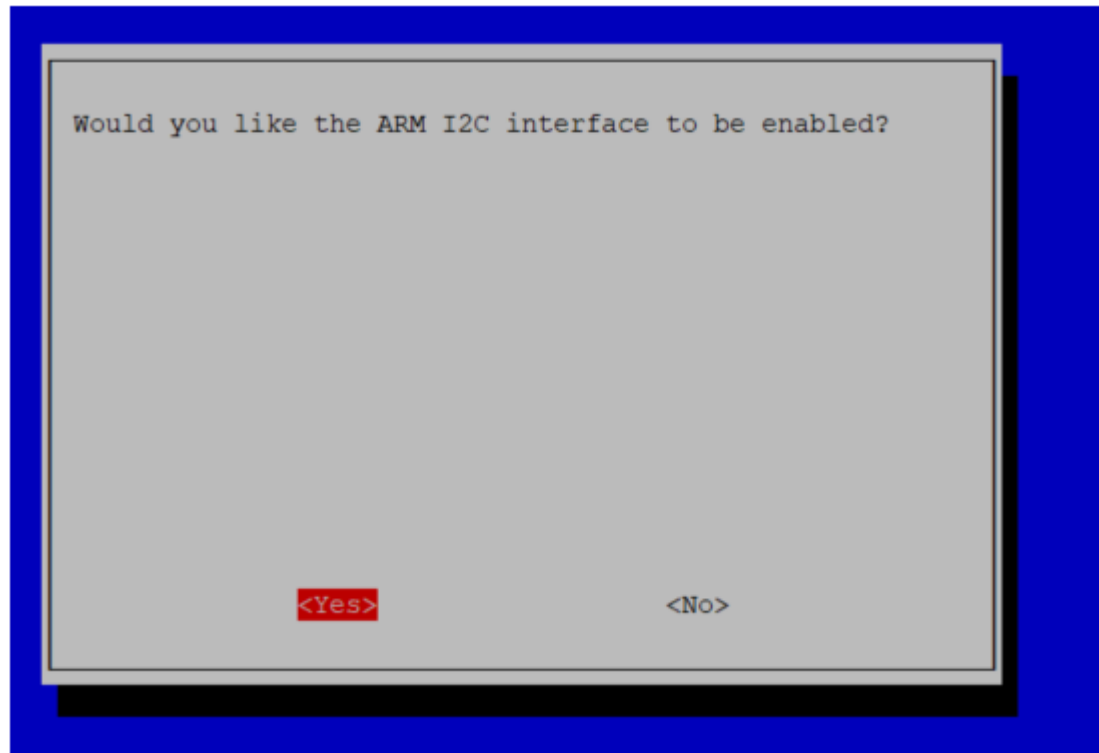
- Interfacing Options으로 이동하여 엔터를 입력



Section 04 가속도 센서

□ I2C 활성화

- I2C로 이동하여 엔터를 입력



Section 04 가속도 센서

□ I2C 활성화

- Yes를 선택후 엔터를 입력
- Esc를 눌러서 환경설정 창을 종료
- 터미널에서 `reboot` 명령어를 이용하여 라즈베리파이를 재부팅

Section 04 가속도 센서

□ I2C 활성화 확인

- 터미널에서 다음과 같이 입력

```
$ lsmod | grep i2c
```

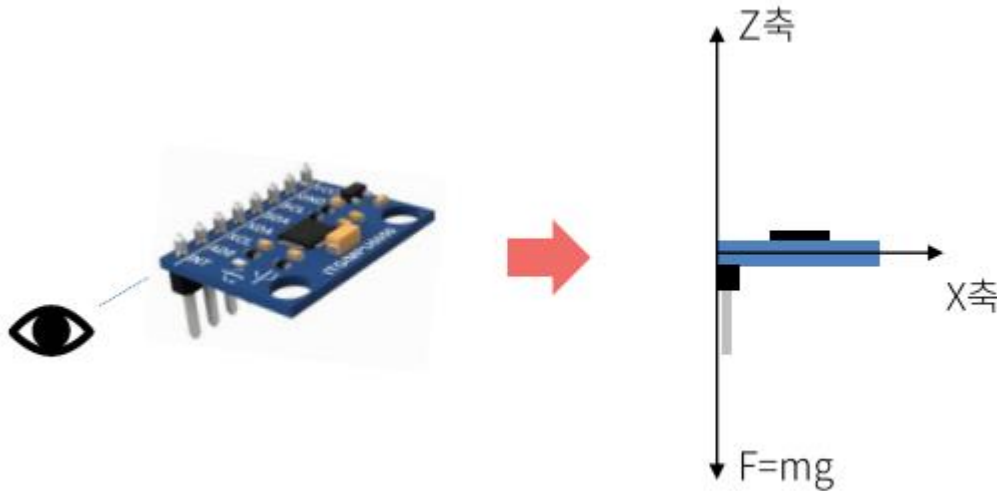
```
pi@raspberrypi:~ $ lsmod | grep i2c
i2c_bcm2835      16384  0
i2c_dev         20480  0
```

- 입력 결과로 i2c 모듈이 나오게 된다면 정상적으로 설정이 된 것!!

Section 04 가속도 센서

□ 가속도 센서를 이용한 각도 계산

- 각도계산을 위해 위의 사진처럼 특정 시점에서 본 벡터를 계산한다.

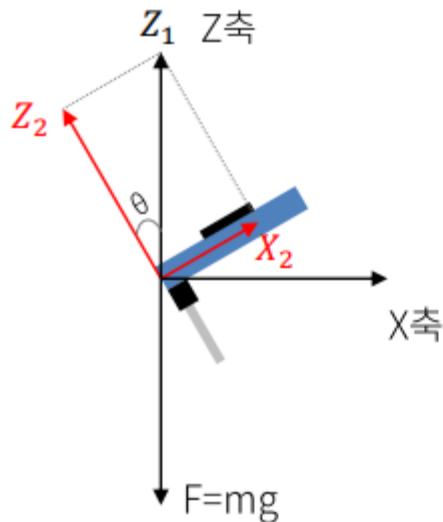


Z축 방향 각도 계산

Section 04 가속도 센서

□ 가속도 센서를 이용한 각도 계산

- 우선 Z축을 각도 θ 만큼 기울였을 때 계산하는 과정을 살펴보면 아래와 같다.
- 삼각함수를 이용하여 기울어진 각도 θ 에 대해 다음과 같은 식을 얻을 수 있다.



$$Z_1 = -mg$$

$$\cos \theta = Z_2 / Z_1$$

$$Z_2 = Z_1 \times \cos \theta$$

$$\cos(90 - \theta) = X_2 / Z_1$$

$$X_2 = Z_1 \times \cos(90 - \theta)$$

$$X_2 = Z_1 \times \sin \theta$$



$$Z_1 = -mg$$

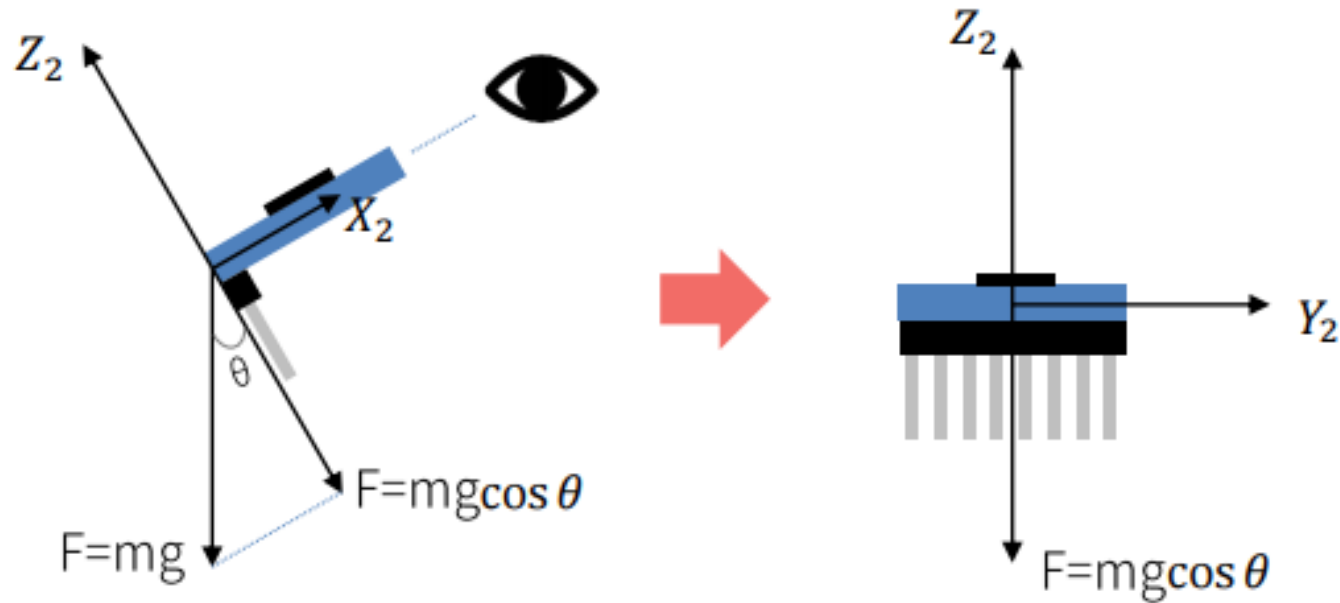
$$Z_2 = Z_1 \times \cos \theta$$

$$X_2 = Z_1 \times \sin \theta$$

Section 04 가속도 센서

□ 가속도 센서를 이용한 각도 계산

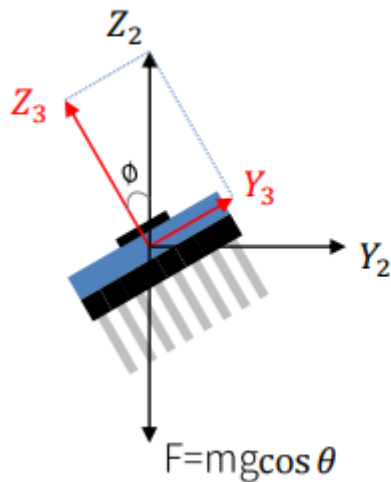
- 이어서 다음의 시점에서 본 벡터를 계산한다.



Section 04 가속도 센서

□ 가속도 센서를 이용한 각도 계산

- Y축을 각도 θ 만큼 기울어져 있을 때를 계산하면 아래와 같은 과정을 통해 최종적으로 식을 구할 수 있다.



$$\cos \phi = Z_3 / Z_2$$

$$Z_3 = Z_2 \times \cos \phi$$

$$Z_3 = Z_1 \cos \theta \cos \phi$$

$$\cos(90 - \phi) = Y_3 / Z_2$$

$$Y_3 = Z_2 \times \cos(90 - \phi)$$

$$Y_3 = Z_2 \times \sin \phi$$

$$Y_3 = Z_1 \cos \theta \sin \phi$$

$$X_3 = X_2$$



$$Z_3 = Z_1 \cos \theta \cos \phi$$

$$Y_3 = Z_1 \cos \theta \sin \phi$$

$$X_3 = X_2 = Z_1 \times \sin \theta$$

$$Z_1 = -mg$$

Section 04 가속도 센서

□ 가속도 센서를 이용한 각도 계산

- 위의 식을 전개하면 다음과 같은 결과를 얻을 수 있다.

$$Z_3^2 + Y_3^2 = m^2 g^2 \cos^2 \theta \cos^2 \phi + m^2 g^2 \cos^2 \theta \sin^2 \phi$$

$$Z_3^2 + Y_3^2 = m^2 g^2 \cos^2 \theta (\cos^2 \phi + \sin^2 \phi)$$

$$Z_3^2 + Y_3^2 = m^2 g^2 \cos^2 \theta$$

$$\sqrt{Z_3^2 + Y_3^2} = mg \cos \theta$$

$$X_3 = -mg \sin \theta$$

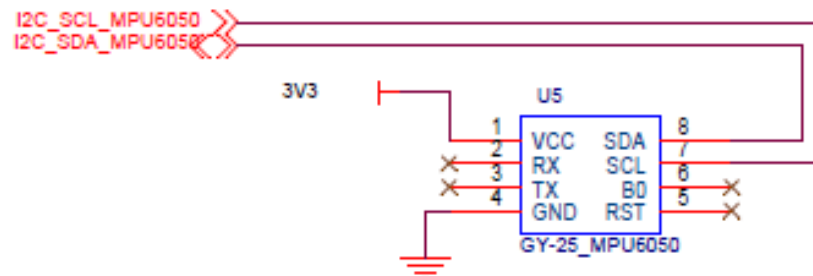
$$\frac{\sin \theta}{\cos \theta} = \tan \theta = \frac{-X_3}{\sqrt{Z_3^2 + Y_3^2}}$$

$$\therefore \theta = \tan^{-1} \frac{-X_3}{\sqrt{Z_3^2 + Y_3^2}}$$

Section 04 가속도 센서

□ 가속도 센서를 이용한 각도 계산

- 이러한 가속도센서를 이용한 각도계산은, 힘이 일정하다면 정확한 각도 값을 얻어낼 수 있지만 진동과 같은 떨림에 약하다는 단점이 있다.
- 다음 그림은 자이로 센서의 회로도이다.



- 여기까지 가속도센서를 통해 기울어진 각도를 계산해보았는데, 다음으로 이 식이 적용된 파이썬 코드를 작성하여 MPU6050센서를 기울였을 때 각도가 계산되는지 확인하는 실습을 해보자.

Section 04 가속도 센서

□ 파이썬 코드

```
1  from ctypes import *
2  import os
3  import time
4
5  WiringPi = CDLL("/home/pi/WiringPi/wiringPi/libwiringPi.so.2.70", mode=RTLD_GLOBAL)
6  swcar = cdll.LoadLibrary('home/pi/swcar/libswcar.so')
7
8  swcar.SIO_Init(0)
9
10 print('press ctrl + c to terminate program')
11
12 swcar.SIO_ReadGyroAccel.restype = c_char_p
13 swcar.SIO_ReadGyroAccel.argtype = [POINTER(c_int), POINTER(c_int),
14                                     POINTER(c_int)]
15 swcar.SIO_ReadGyroRotate.restype = c_char_p
```

Section 04 가속도 센서

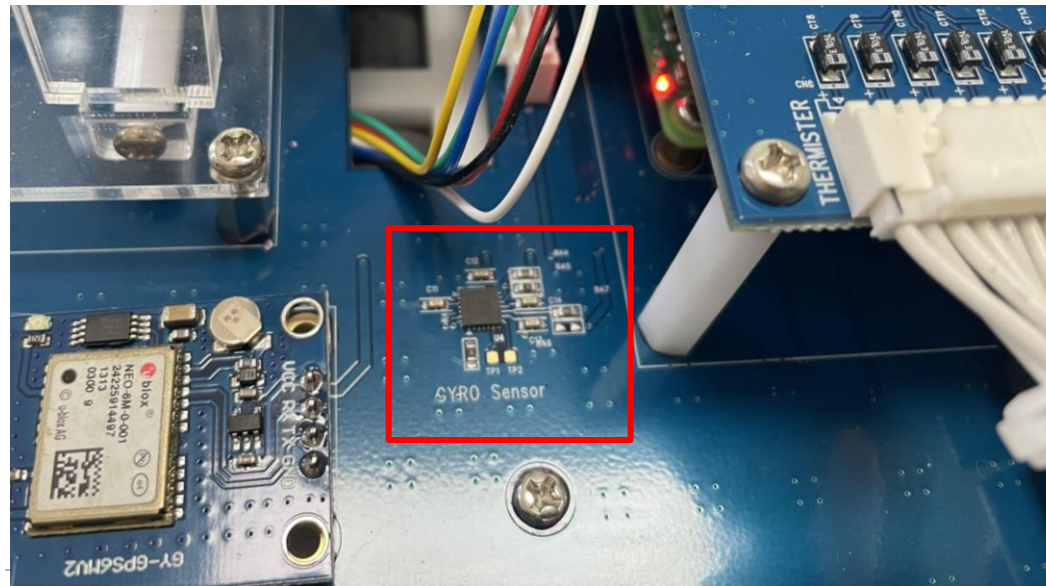
□ 파이썬 코드

```
15 swcar.SIO_ReadGyroRotate.argtype = [POINTER(c_int), POINTER(c_int),  
    POINTER(c_int)]  
16  
17 acc_x = pointer(c_int(0))  
18 acc_y = pointer(c_int(0))  
19 acc_z = pointer(c_int(0))  
20  
21 rot_x = pointer(c_int(0))  
22 rot_y = pointer(c_int(0))  
23 rot_z = pointer(c_int(0))  
24  
25 try:  
26     while True:  
27         swcar.SIO_ReadGyroAccel(acc_x, acc_y, acc_z)  
28         swcar.SIO_ReadGyroRotate(rot_x, rot_y, rot_z)
```

Section 04 가속도 센서

□ 파이썬 코드

```
29  
30     print("acc = ", acc_x[0], acc_y[0], acc_z[0])  
31     print("rot = ", rot_x[0], rot_y[0], rot_z[0])  
32  
33     time.sleep(1)
```



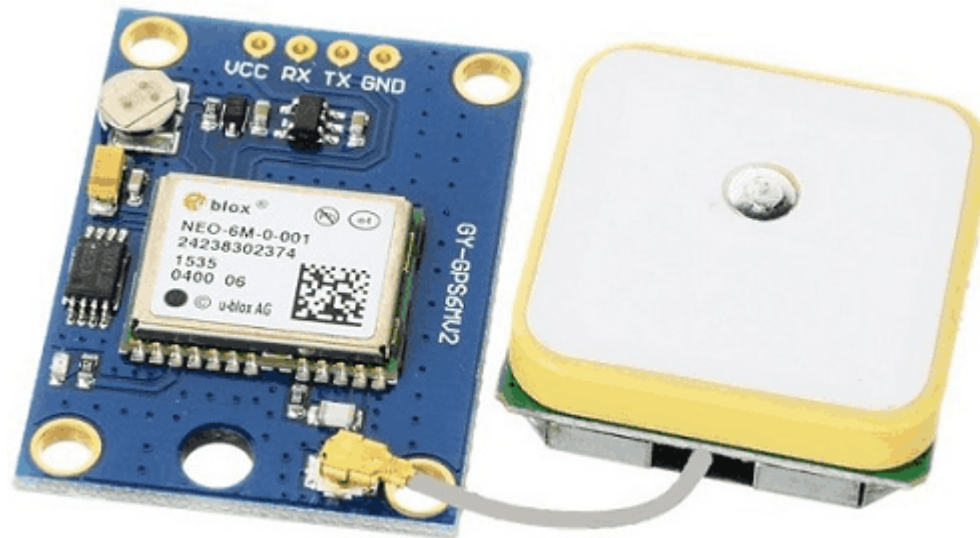
Section 05 GPS

□ GPS 소개

- Global Positioning System(GPS)은 지구 상에 정확하게 그 위치를 결정하기 위해 지구 공간 및 지상 스테이션에서 위성들에 의해 전송된 신호를 사용한다.
- 위성 및 지상국에서 전송 된 무선 주파수 신호는 GPS에 의해 수신된다.
- GPS는 이러한 신호를 사용하여 정확한 위치를 결정한다. GPS 자체는 정보를 전송할 필요가 없다.
- 위성 및 지상국에서 수신 된 신호에는 신호가 전송된 시간의 타임스탬프가 포함된다. 신호가 전송된 시간과 신호가 수신된 시간의 차이를 계산한다. 신호의 속도를 사용하여 위성과 GPS 수신기 사이의 거리는 속도와 시간을 사용한 거리에 대한 간단한 공식을 사용하여 결정할 수 있다.
- 3 개 이상의 위성 정보를 사용하여 GPS의 정확한 위치를 삼각 측량 할 수 있다.
- GPS 수신기 모듈은 UART 통신을 사용하여 컨트롤러 또는 PC 터미널과 통신한다. Raspberry Pi에서 UART를 사용하기 전에 이를 구성하고
▶ 40 활성화해야 한다.

Section 05 GPS

□ NEO-6M 제품 사양과 회로연결



Supply Voltage: 3.3~5V

Module Size: 35 x 26 x 3mm

Antenna Size: 50 x 25mm

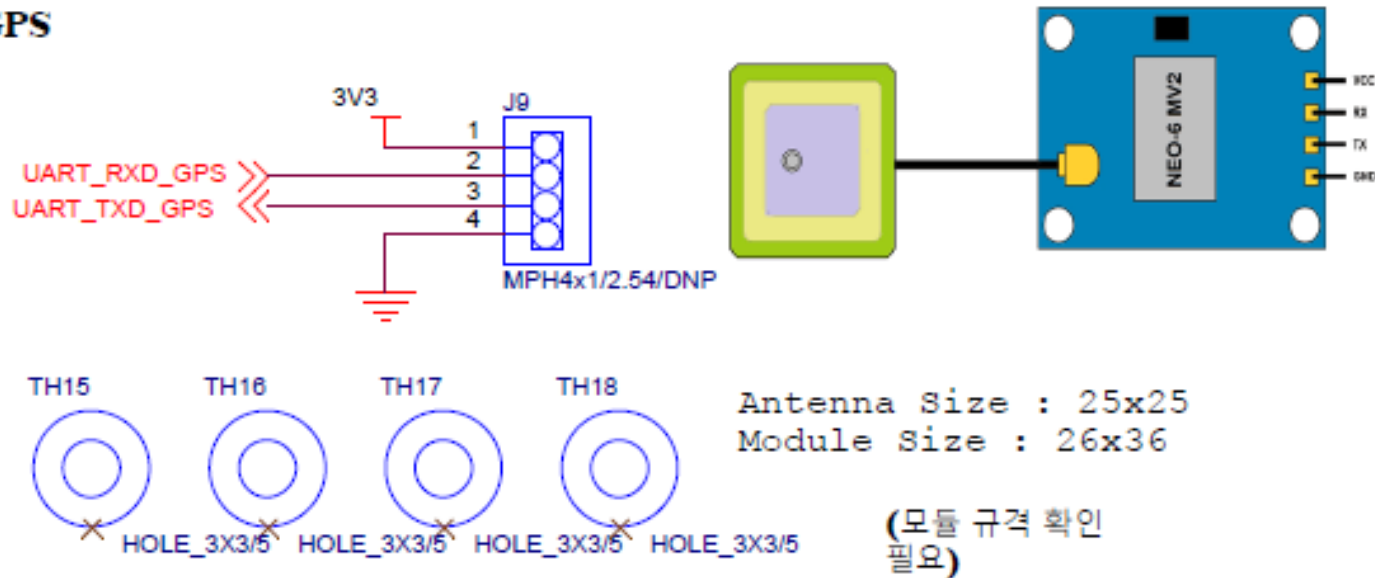
UART Interface

Baud Rate : 9600bps

Section 05 GPS

□ NEO-6M 제품 사양과 회로연결

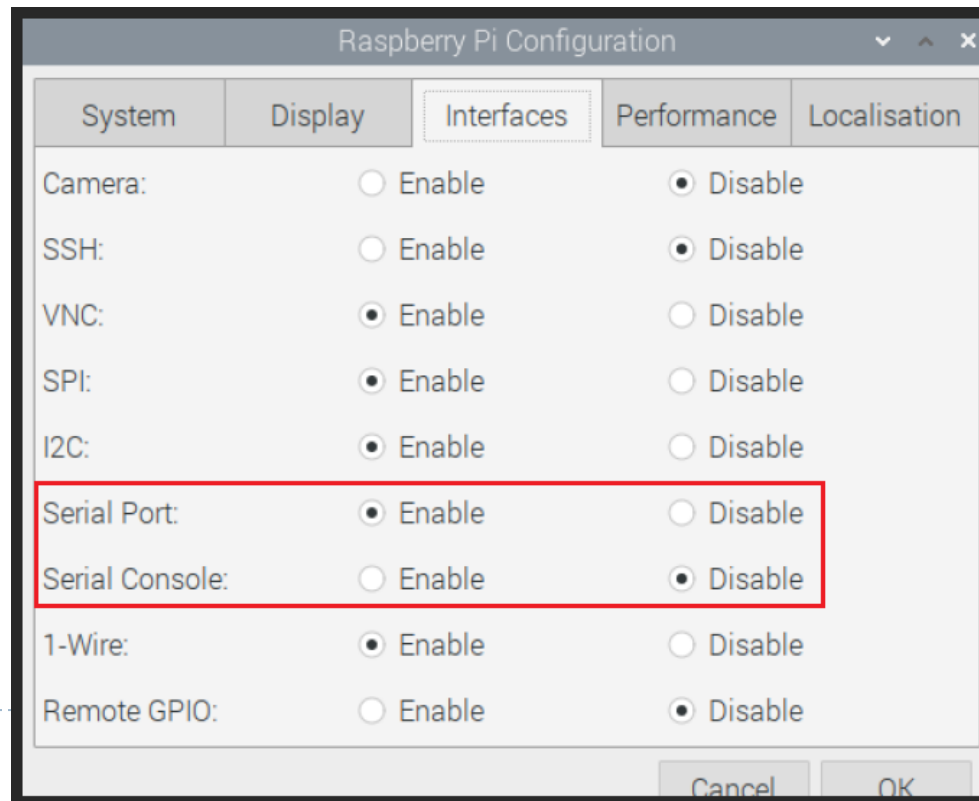
GPS



Section 05 GPS

□ 라즈베리파이 UART 기본 설정

- UART 설정의 목적은 "외부 장치(GPS 모듈)를 연결하는데 다른 UART 연결 장치(콘솔, 블루투스 등)와의 충돌을 없애기 위함"이다.
- 라즈베리파이에서 "Preferences" -> "Raspberry Pi Configuration" -> "Interfaces" "Serial Port"를 Enable, "Serial Console"을 Disable 설정



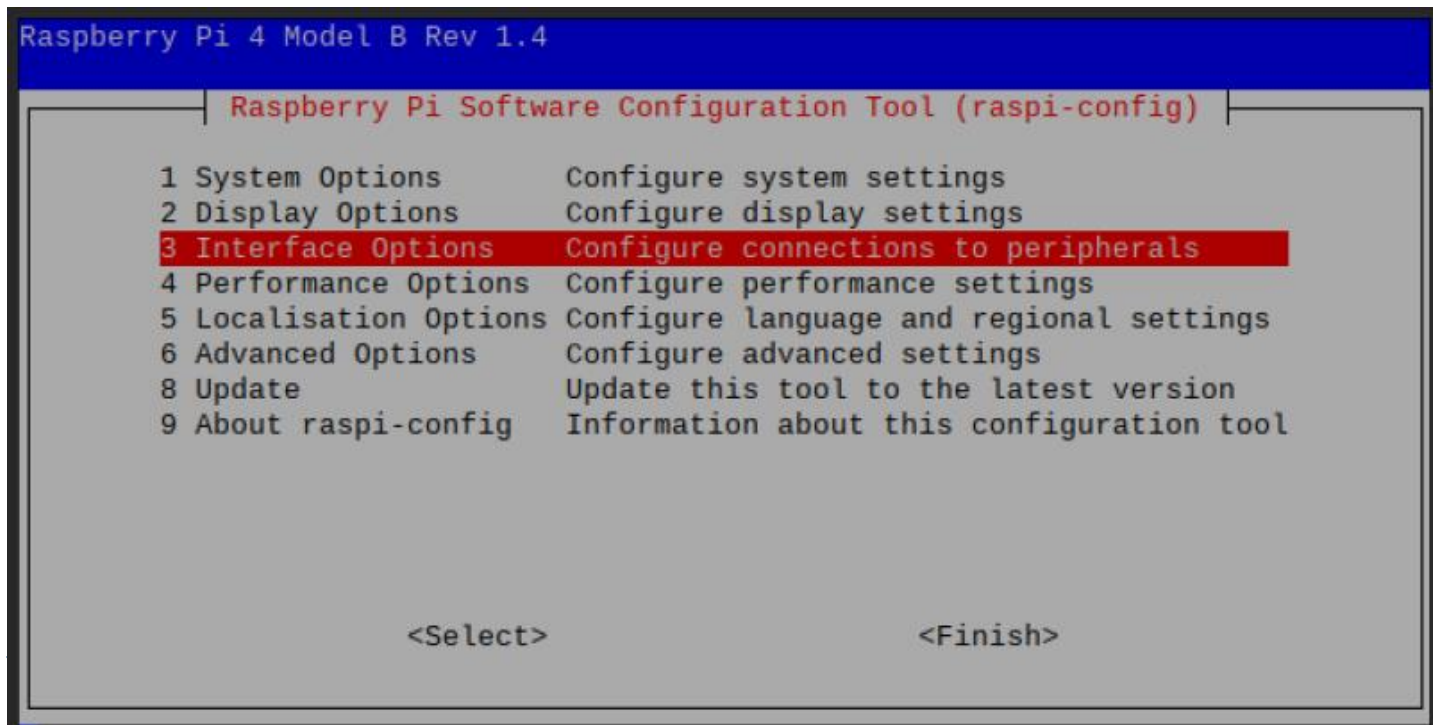
Section 05 GPS

□ 라즈베리파이 UART 기본 설정

- 터미널 창에서 "sudo raspi-config" 명령을 통해 환경 설정 도구를 실행

\$ sudo raspi-config

- 다음으로 "Interface Options" -> "Serial Port Enable/Disable..." 항목을 선택



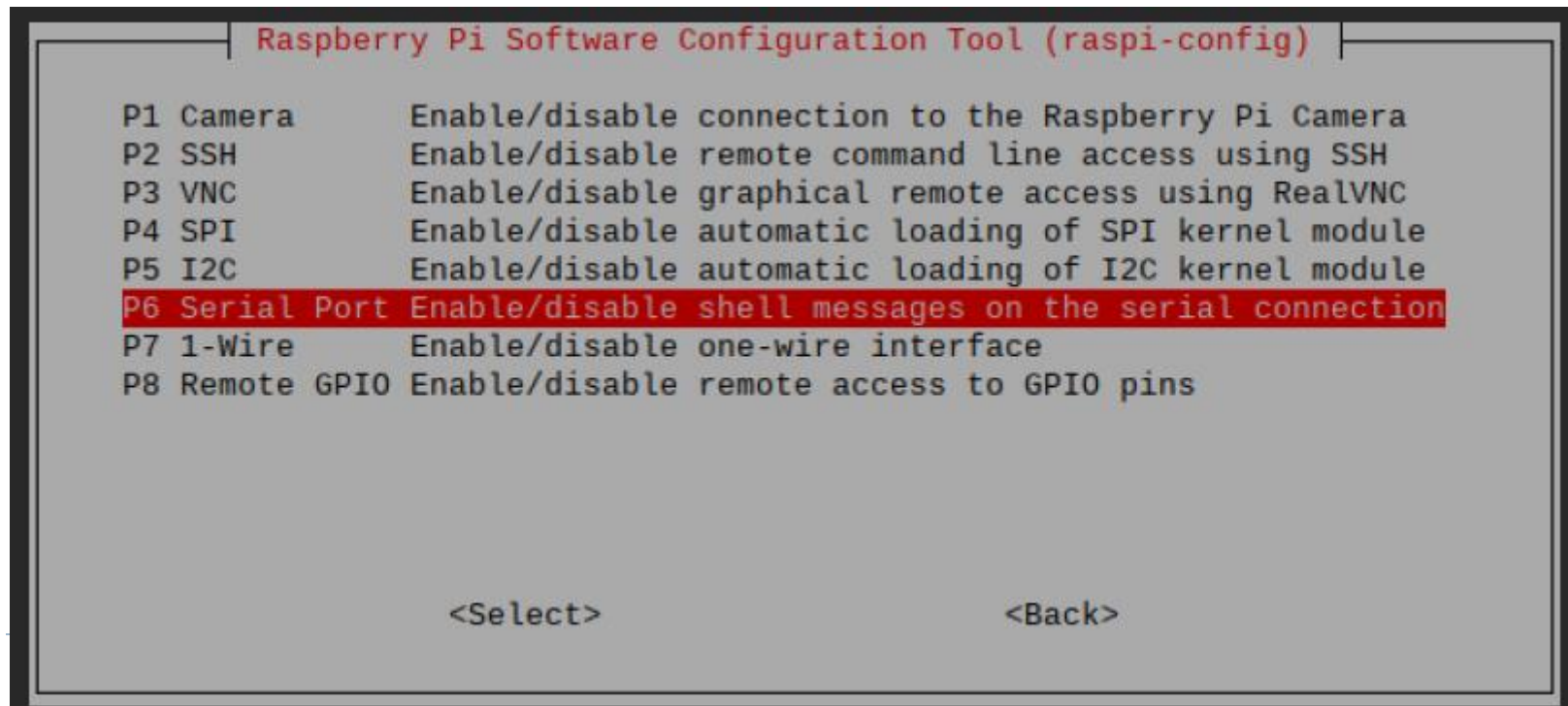
Section 05 GPS

□ 라즈베리파이 UART 기본 설정

- 터미널 창에서 "sudo raspi-config" 명령을 통해 환경 설정 도구를 실행

\$ sudo raspi-config

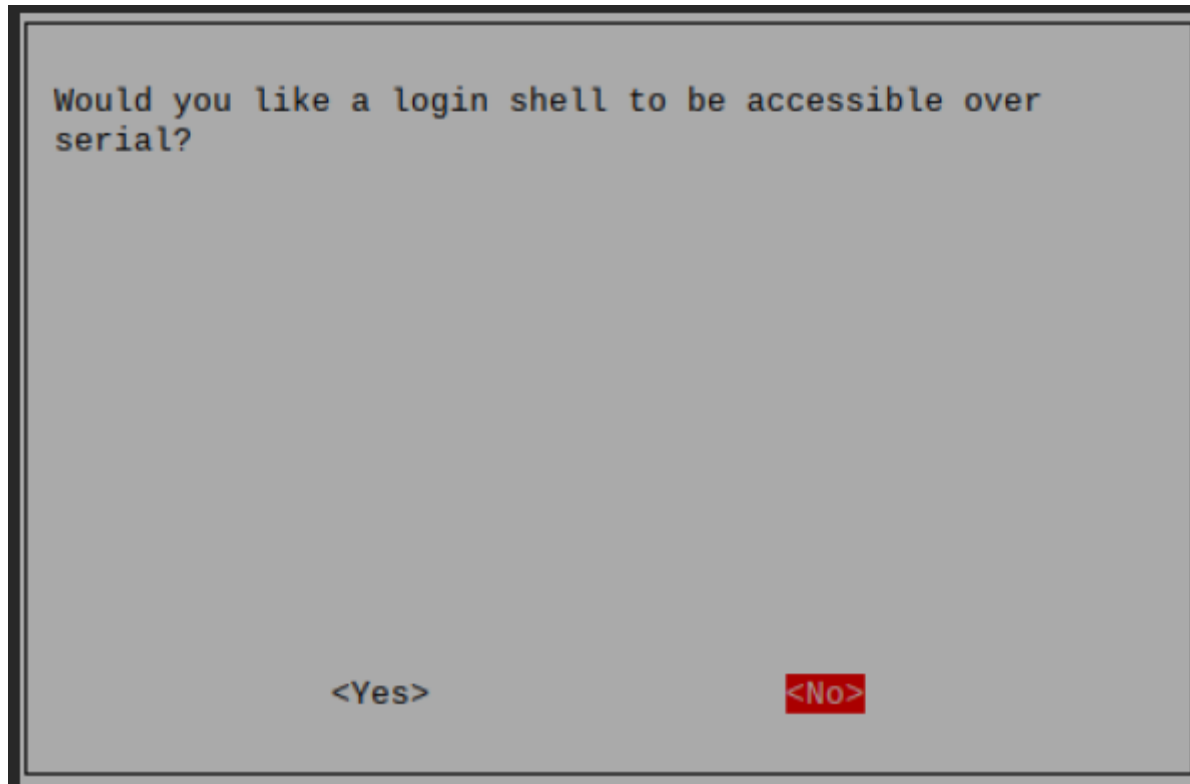
- 다음으로 "Interface Options" -> "Serial Port Enable/Disable..." 항목을 선택



Section 05 GPS

□ 라즈베리파이 UART 기본 설정

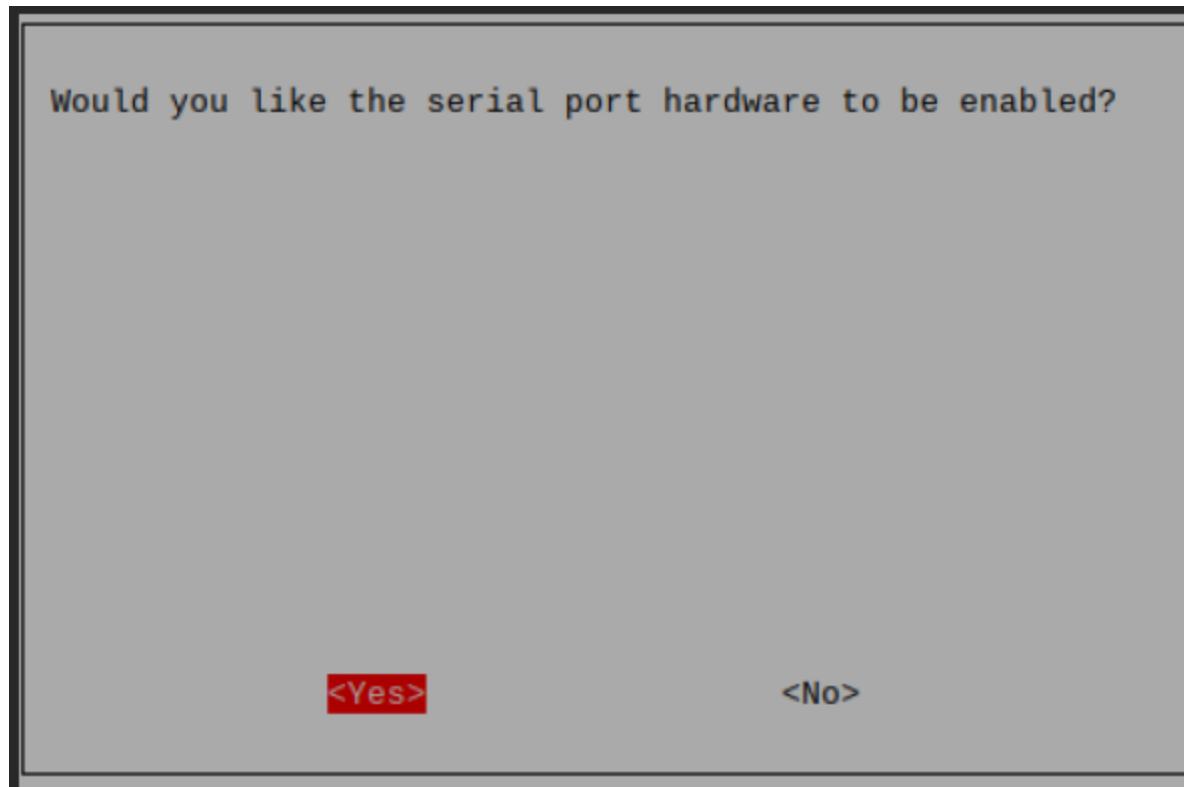
- 로그인 셸(Login shell) 액세스 관련 설정 -> <No>



Section 05 GPS

□ 라즈베리파이 UART 기본 설정

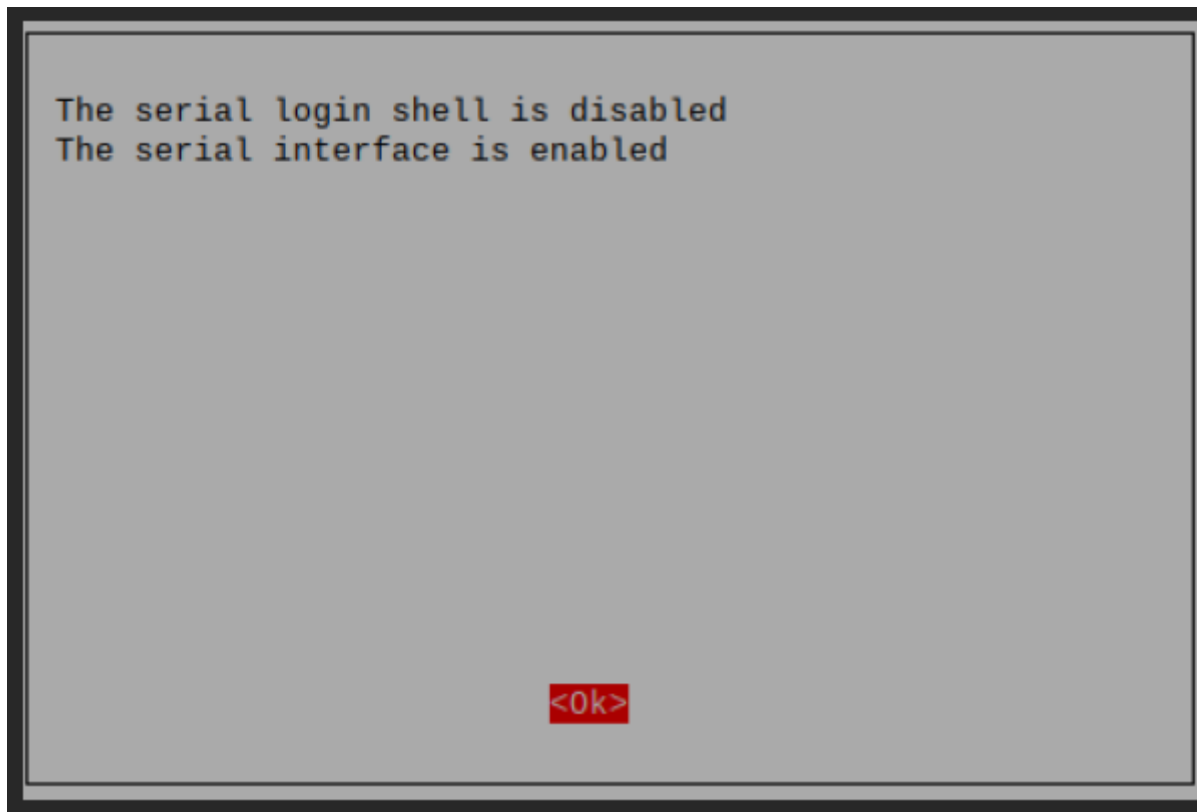
- 시리얼 포트 하드웨어(Serial Port Hardware) 허용 관련 설정 -> <Yes>



Section 05 GPS

□ 라즈베리파이 UART 기본 설정

- 시리얼 포트 하드웨어(Serial Port Hardware) 허용 관련 설정 -> <Yes>



Section 05 GPS

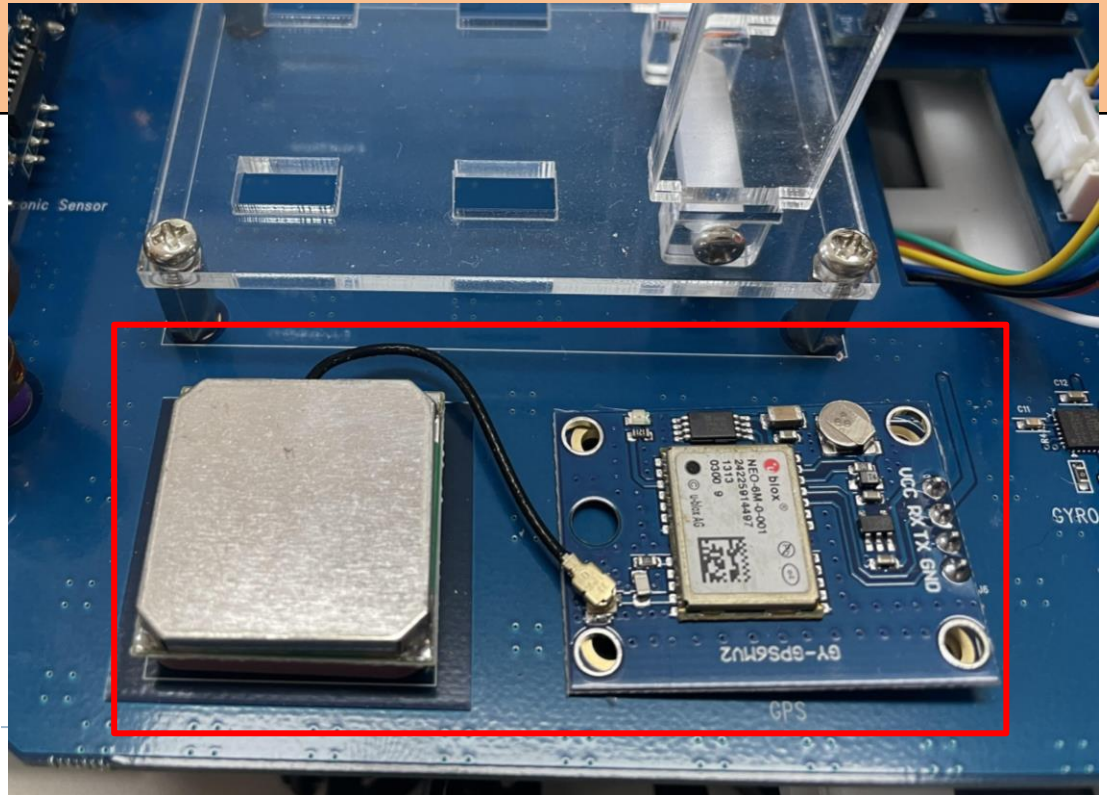
□ GPS 실습 코드

```
1  from ctypes import *
2  import os
3  import time
4
5  WiringPi = CDLL("/home/pi/WiringPi/wiringPi/libwiringPi.so.2.70", mode=RTLD_GLO
  BAL)
6  swcar = cdll.LoadLibrary('home/pi/swcar/libswcar.so')
7
8  swcar.SIO_Init(0)
9
10 print('press ctrl + c to terminate program')
11
12 swcar.SIO_ReadGPS.restype = c_char_p
13 swcar.SIO_ReadGPS.argtypes = None
14
```

Section 05 GPS

□ GPS 실습 코드

```
15 try:
16     while True:
17         gps = swcar.SIO_ReadGPS()
18         print("gps = ", gps)
19         time.sleep(1)
```



Q&A

