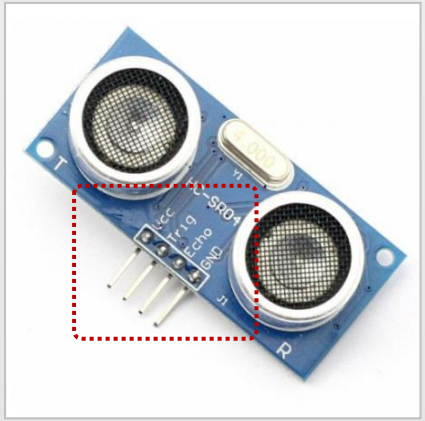
1 초음파 센서

1) 초음파 센서

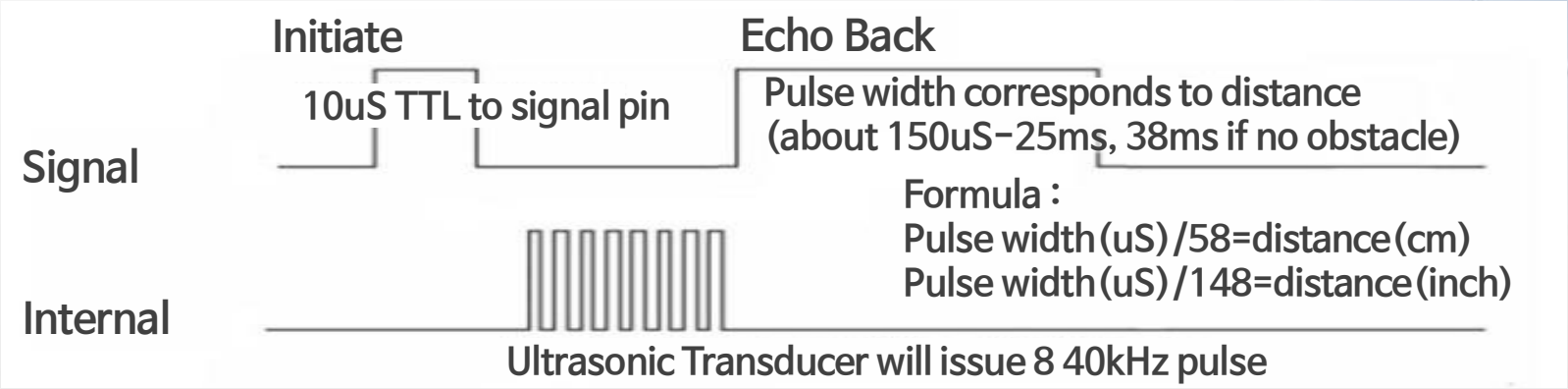
5v 라인에 연결

VCC | Triger | Echo | GND

라즈베리파이의 GPIO 핀에 연결



1) 초음파 센서



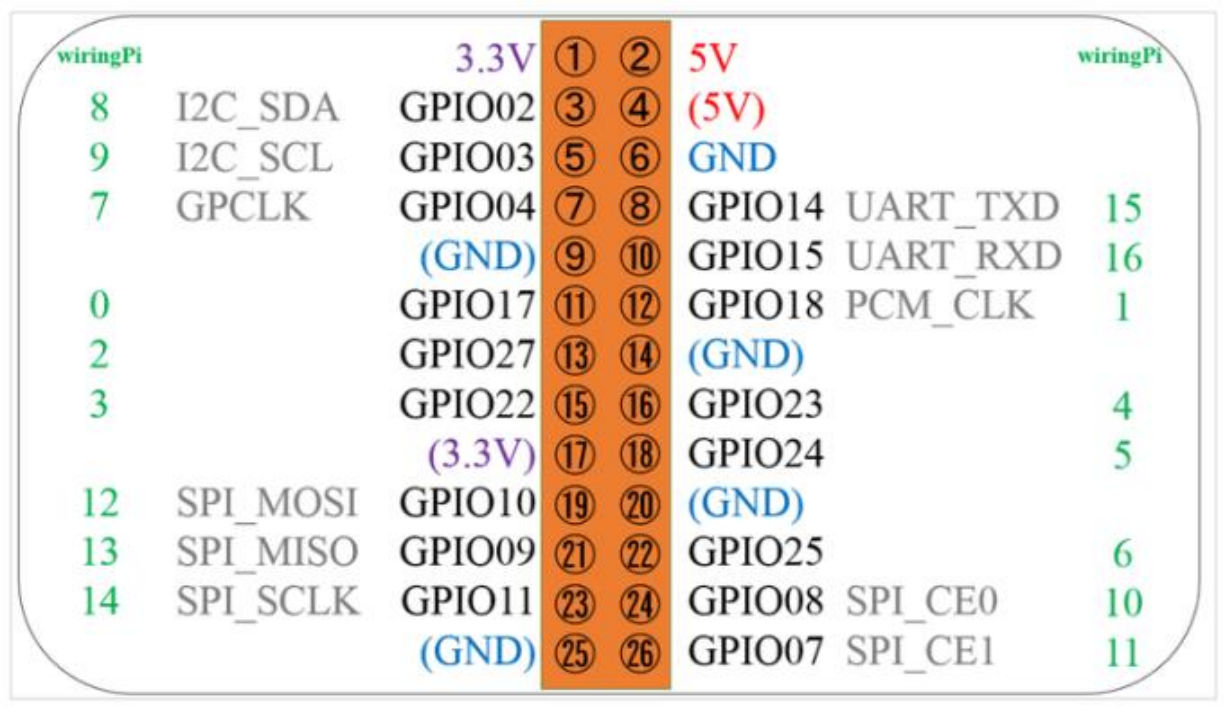
- Triger 핀에 10uS 정도의 High 신호를 주면
  초음파센서는 40kHz 펄스를 자동적으로 8번을 발생함
- 펄스를 발생시킨 직후에 Echo 핀은 High 되고 반사된 초음파가 감지되었을 때
  Echo 핀이 Low가 됨
- Echo 핀이 High였다가 Low가 되는 데 걸리는 시간을 측정하며,
  그 시간을 초음파의 속도/2(즉, 58)로 나누면 거리가 나옴

1) 초음파 센서

```
#include<stdio.h>
#include<wiringPi.h>
#define trigPin 5
#define echoPin 4
int main(void)
{
        int distance=0;
        int pulse = 0;
        if(wiringPiSetup () == -1)
        return 1;

        pinMode (trigPin, OUTPUT);
        pinMode (echoPin, INPUT);
        for(;;)
        {
                digitalWrite (trigPin, LOW);
                usleep(2);
                digitalWrite (trigPin, HIGH);
                usleep(20);
                digitalWrite (trigPin, LOW);
                while(digitalRead(echoPin) == LOW);
                long startTime = micros();
                while(digitalRead(echoPin) == HIGH);
                long travelTime = micros() - startTime;
                int distance = travelTime / 58;
                printf("Distance: %dcm \ n", distance);
                delay(100);
        }
}
```

1) 초음파 센서

Define 부분에 GPIO 23, 24번 사용



wiringPi							wiringPi
		3.3V	1	2	5V		
8	I2C_SDA	GPIO02	3	4	(5V)		
9	I2C_SCL	GPIO03	5	6	GND		
7	GPCLK	GPIO04	7	8	GPIO14	UART_TXD	15
		(GND)	9	10	GPIO15	UART_RXD	16
0		GPIO17	11	12	GPIO18	PCM_CLK	1
2		GPIO27	13	14	(GND)		
3		GPIO22	15	16	GPIO23		4
		(3.3V)	17	18	GPIO24		5
12	SPI_MOSI	GPIO10	19	20	(GND)		
13	SPI_MISO	GPIO09	21	22	GPIO25		6
14	SPI_SCLK	GPIO11	23	24	GPIO08	SPI_CE0	10
		(GND)	25	26	GPIO07	SPI_CE1	11

1) 초음파 센서

컴파일 방법

```
gcc -o ultrasonic ultrasonic.c -lwiringPi
```

2

거리센서 & 가속도 센서

1) 거리 센서

적외선 반사각을 이용한 거리센서로 유용한 센서임

타이머 등을 이용한 초음파 센서보다 거리측정에 많이 사용함

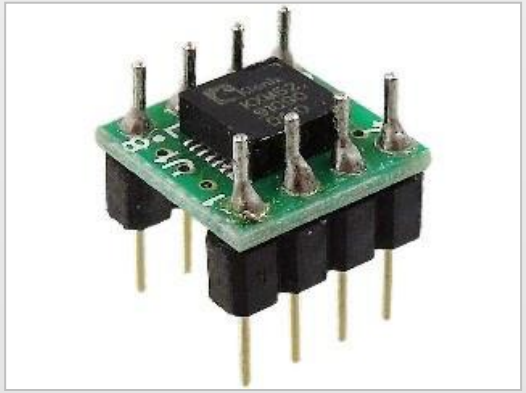
예상가격 : 7천 원~1만 5천 원

2) x, y, z의 3축 가속도 센서

예상가격 : 1만 원~10만 원

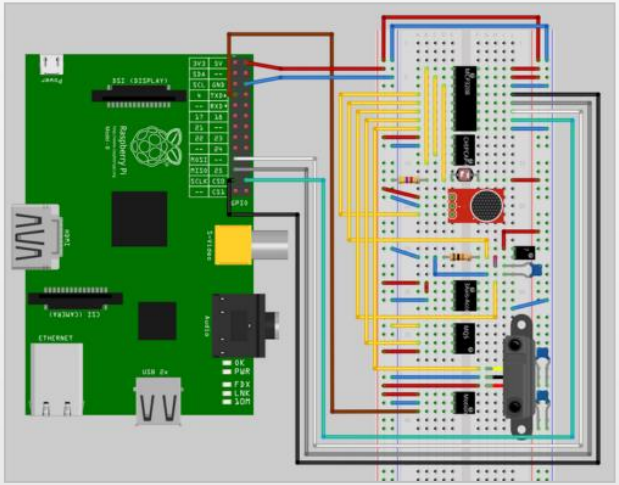
자신의 사용목적에 따라 다양한 선택을 할 수 있음

ADC가 가능한 아날로그 출력도 있으며,
uart 및 i2c, spi 등의 통신 기능을 포함한 제품도 있음



3 인체감지(PIR) 센서

1) 인체감지(PIR) 센서



1) 인체감지(PIR) 센서

인체감지 센서

- 화장실 및 자동 소등 등에 많이 사용되는 인체감지 센서
- 예상가격 : 2천~2만 원
- 참고사항
  - PIR 센서(초전형 적외선 센서)는 적외선의 변화를 감지함
  - 센서 반응이 가능한 범위 내에서의 움직임을 체크함
  - 휴먼모션 센서

1) 인체감지(PIR) 센서

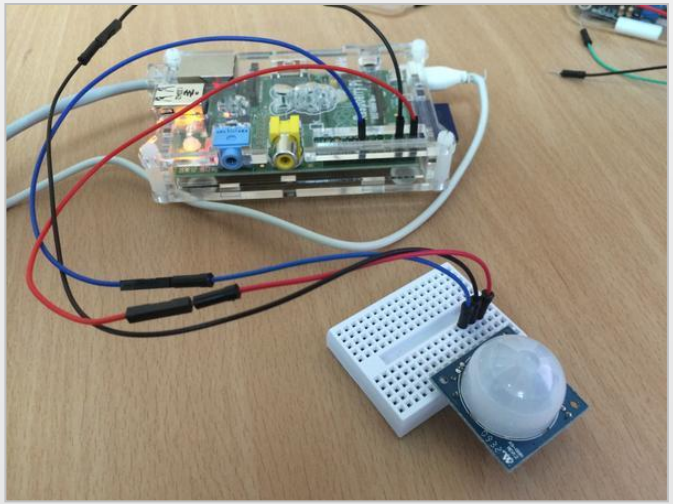
조도 센서

- 주변 환경의 밝기를 측정 가능
- 예상가격 : 200~500원
- 임계값 이상이면 밝은 상태,
  임계값 이하이면 어두운 상태로
  인식함
- 럭스(Lux)와 같은
  절대치를 얻는다는 생각보다는
  상대값으로 처리함

1) 인체감지(PIR) 센서

전원 핀 외 OUT 핀과 CDS가 있으나, CDS만 연결함

배선 : VCC, GND, CDS



1) 인체감지(PIR) 센서

```
import time
import RPi.GPIO as io
io.setmode(io.BCM)

pir_pin = 25

io.setup(pir_pin, io.IN) # activate input with PullUp

while True:
        if io.input(pir_pin):
                print("PIR ALARM!")
        else:
                print("NONE")
        time.sleep(0.5)
```

4

MCP3208를 통한 센서 읽기

1) MCP3208를 통한 센서 읽기


```
#define CS_MCP3208  10      // BCM_GPIO_8
#define SPI_CHANNEL 0
#define SPI_SPEED   1000000   // 1MHz
#define VCC       4.8     // Supply Voltage


int read_mcp3208_adc(unsigned char adcChannel)
{
 unsigned char buff[3];
 int adcValue = 0;

 buff[0] = 0x06 | ((adcChannel & 0x07) >> 2);
 buff[1] = ((adcChannel & 0x07) << 6);
 buff[2] = 0x00;
```

```
 digitalWrite(CS_MCP3208, 0);  // Low : CS Active
 wiringPiSPIDataRW(SPI_CHANNEL, buff, 3);
 buff[1] = 0x0F & buff[1];
 adcValue = ( buff[1] << 8) | buff[2];
 digitalWrite(CS_MCP3208, 1);  // High : CS Inactive

 return adcValue;
}
```

1) MCP3208를 통한 센서 읽기


```
int main (void)
{
 int adcChannel  = 0;
 int adcValue[8] = {0};


 if(wiringPiSetup() == -1)
 {
  fprintf (stdout, "Unable to start wiringPi: %s \ n", strerror(errno));
  return 1 ;
 }

 if(wiringPiSPISetup(SPI_CHANNEL, SPI_SPEED) == -1)
 {
  fprintf (stdout, "wiringPiSPISetup Failed: %s \ n", strerror(errno));
  return 1 ;
 }

 pinMode(CS_MCP3208, OUTPUT);
```

1) MCP3208를 통한 센서 읽기

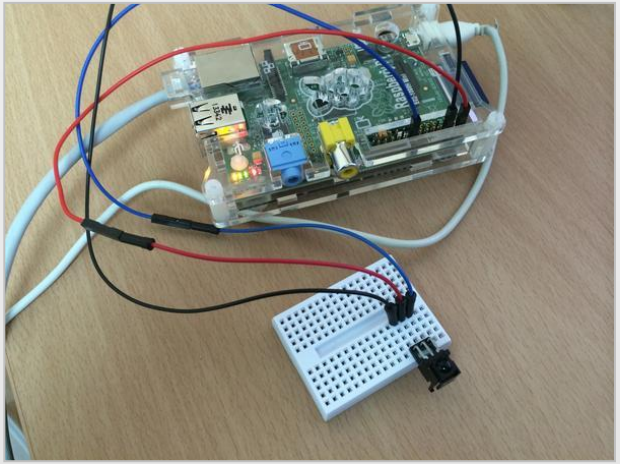

```
while(1)
 {
  adcValue[0] = read_mcp3208_adc(0); // Temperature Sensor
  adcValue[0] = ((adcValue[0] * VCC / 4095 - (0.1 * VCC))/(0.8 * VCC)) * 200 - 50;

  adcValue[1] = read_mcp3208_adc(1); // Humidity Sensor
  adcValue[1] = ((adcValue[1] * VCC / 4095 - (0.1 * VCC))/(0.8 * VCC)) * 100;


 adcValue[2] = read_mcp3208_adc(2); // Illuminance Sensor
  adcValue[3] = read_mcp3208_adc(3); // Mic Sensor
  adcValue[4] = read_mcp3208_adc(4); // Flame Sensor
  adcValue[5] = read_mcp3208_adc(5); // Acceleration Sensor (Z-Axis)
  adcValue[6] = read_mcp3208_adc(6); // Gas Sensor
  adcValue[7] = read_mcp3208_adc(7); // Distace Sensor
  adcValue[7] = 27*pow((double)(adcValue[7]*VCC/4095), -1.10);

  for(int i=0; i<8; i++) printf("%5d  ",adcValue[i]);
  printf("\n");
 }
 return 0;
}
```

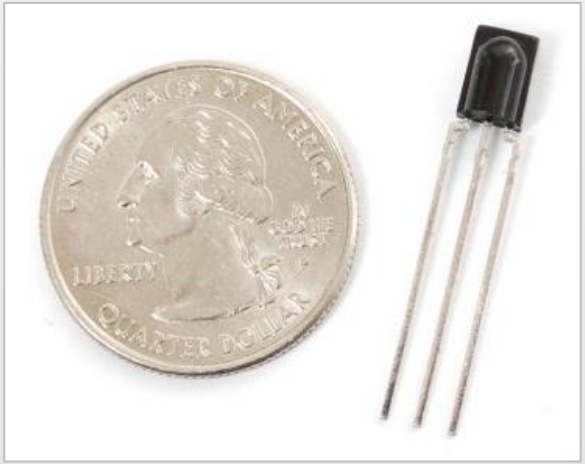
5 IR 센서 활용

1) IR 센서 활용

IR 센서 연결

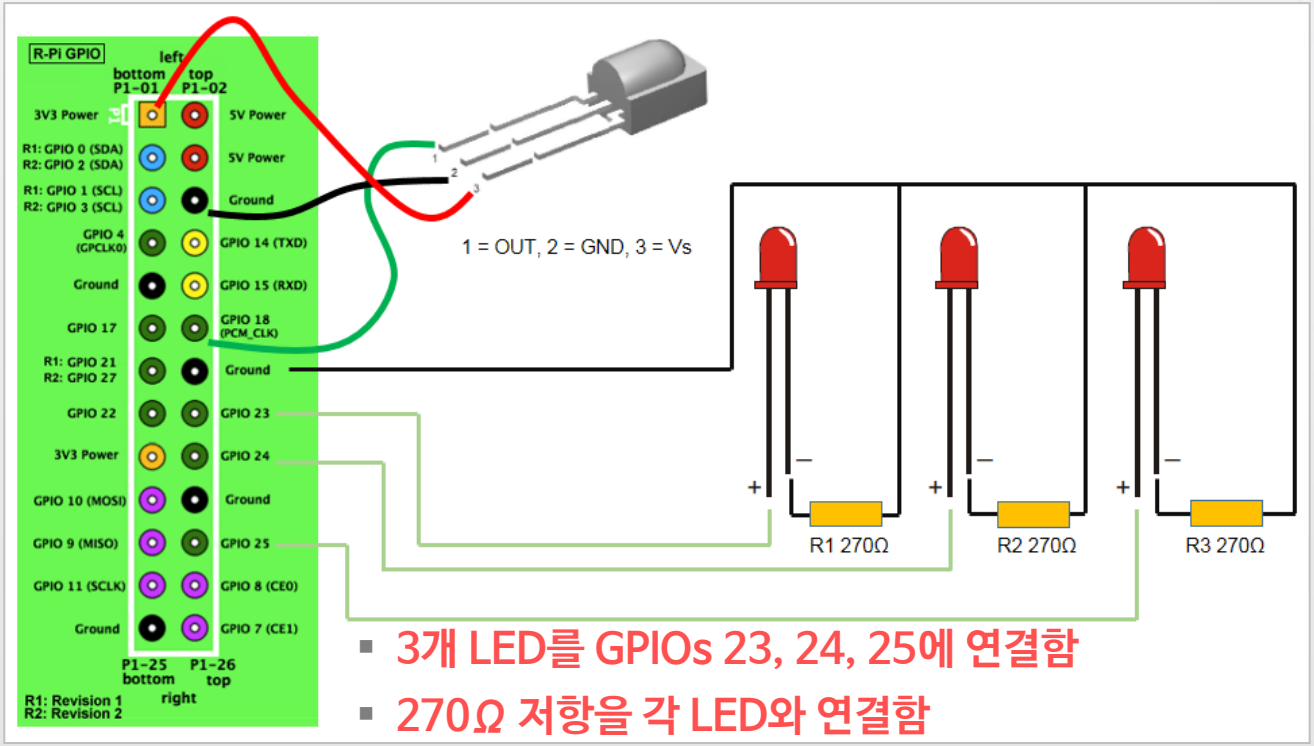
GPIO 18. You can choose another pin, just take note of it as you will need to specify this pin when installing LIRC.





1) IR 센서 활용

IR 센서 연결



- 3개 LED를 GPIOs 23, 24, 25에 연결함
- 270Ω 저항을 각 LED와 연결함

1) IR 센서 활용

Lirc 및 Client Library 설치

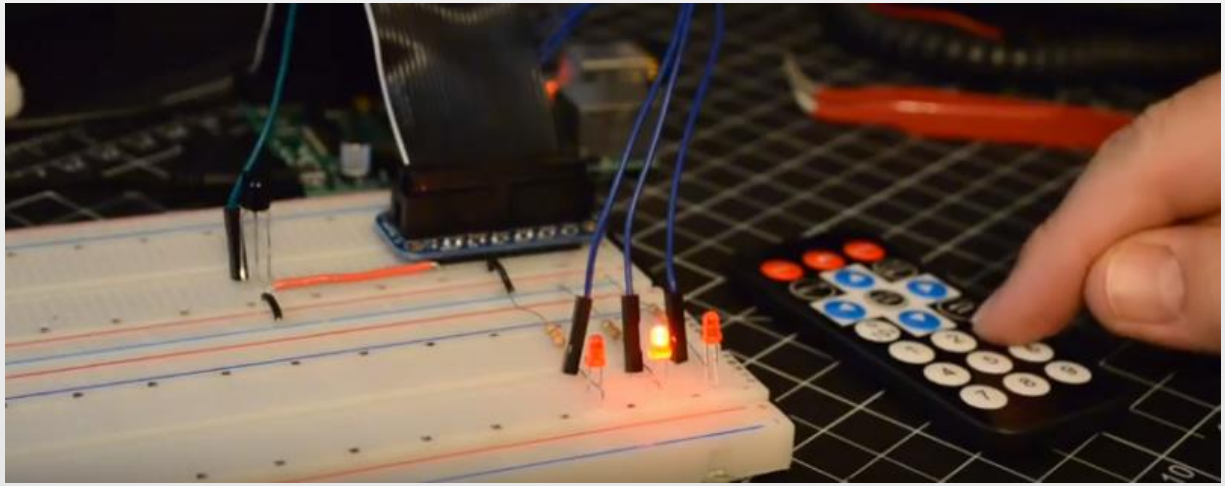
```
pi@raspberrypi ~ $ sudo apt-get install lirc
liblircclient-dev
```

- LIRC(Linux Infrared Remote Control, http://www.lirc.org)는 IR리모컨 신호를 받거나 보내는데 유용한 데몬임

1) IR 센서 활용

/etc/modules 추가

```
lirc_dev
lirc_rpi gpio_in_pin=18
```



출력결과

```
pulse 627
space 514
pulse 624
space 513
pulse 599
space 521
pulse 618
space 1668
pulse 589
space 532
```

6 온도/습도 센서 활용

1) 온도/습도 센서 활용

The GY-80 is tiny, only about 27x17mm in size, and has following I2C sensors:

- HMC5883L (3-Axis Digital Compass), I2C Address 0x1E, datasheet
- ADXL345 (3-Axis Digital Accelerometer), I2C Address 0x53, datasheet
- L3G4200D (3-Axis Angular Rate Sensor/Gyro), Address 0x69, datasheet
- BMP085 (Barometric Pressure/Temperature Sensor), I2C Address 0x77, data sheet

1) 온도/습도 센서 활용

Raspberry Pi I2C Setup

The Raspberry Pi's I2C interface is disabled by default

```
$ sudo vi /etc/modules
```

Add the following lines (and reboot for it to take effect):

```
i2c-bcm2708
i2c-dev
```

1) 온도/습도 센서 활용

Check if the I2C modules have been disabled,
$ more /etc/modprobe.d/raspi-blacklist.conf

```
# blacklist spi and i2c by default (many users don't need them)
blacklist spi-bcm2708
blacklist i2c-bcm2708
```

만약 이와 같이 되어 있다면 두 줄의 Blacklist 라인을 Comment Out함 (#추가)

1) 온도/습도 센서 활용

I2C 툴을 설치함(I2c-tools)

```
$ sudo apt-get install i2c-tools python-smbus
```

This checks the driver side of things - there are two potential I2C Linux devices:

```
sudo i2cdetect -l
i2c-0 i2c       bcm2708_i2c.0           I2C adapter
i2c-1 i2c       bcm2708_i2c.1           I2C adapter
```

1) 온도/습도 센서 활용

Now let's check if there are any I2C devices detected

```
$ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

1) 온도/습도 센서 활용

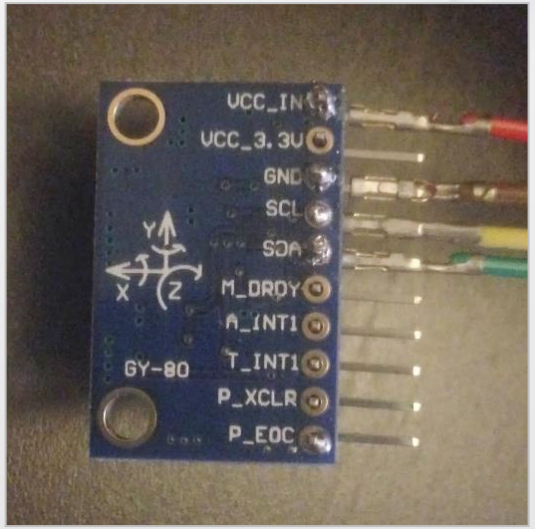
I'm using four jumper cables (red, brown, green and yellow) to connect the GY-80 IMU to the Raspberry Pi GPIO pins (as in the photo above)

VCC_IN ↔ Raspberry Pi GPIO pin 1, 3.3V

GND ↔ Raspberry Pi GPIO pin 6, Ground

SCL ↔ Raspberry Pi GPIO pin 5,
I2C serial clock (SCL)

SDA ↔ Raspberry Pi GPIO pin 3,
I2C serial data (SDA)



1) 온도/습도 센서 활용

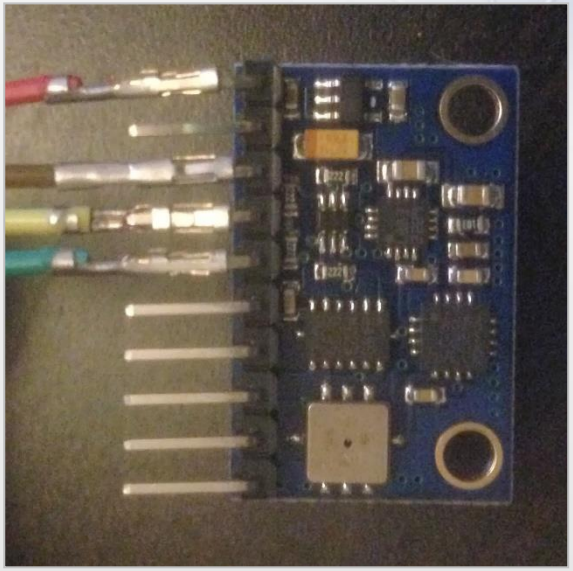
four I2C devices found, with the published hex identifiers

HMC5883L (3-Axis Digital Compass), I2C Address 0x1E

ADXL345 (3-Axis Digital Accelerometer), I2C Address 0×53

L3G4200D (3-Axis Angular Rate Sensor), I2C Address 0×69

BMP085 (Barometric Pressure / Temperature Sensor), I2C Address 0×77



1) 온도/습도 센서 활용

```
$ wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.50.tar.gz
$ tar zxvf bcm2835-1.50.tar.gz
$ cd bcm2835-1.50
$ ./configure
$ make; $ sudo make check; $ sudo make install
$ npm install node-dht-sensor
$ sudo nano dht11.js
```

- Web에서도 접속해서 온습도를 알 수 있도록 Node.js로 구현함
- https://github.com/mqttjs/MQTT.js(Node.js 기반 MQTT 클라이언트 설치)
- npm install mqtt –save 설치 후
- https://github.com/3DKIDS/arduino/blob/master/node_web_dht11

1) 온도/습도 센서 활용

```
var sensorLib = require('node-dht-sensor');
var sensor = {
 initialize: function () {
  return sensorLib.initialize(11, 25); // dht version: 11,
  using 25 pin
 },
 read: function () {
  var readout = sensorLib.read();
  console.log('Temperature: ' + readout.
   temperature.toFixed(2) + 'C, ' + 'humidity: '
   + readout.humidity.toFixed(2) + '%');
  setTimeout(function () {
   sensor.read();
  }, 2000);
 }
};
if (sensor.initialize()) {
 sensor.read();
} else {
 console.warn('Failed to initialize sensor');
}
```

1) 온도/습도 센서 활용

Web 접속화면

1) 온도/습도 센서 활용

콘솔 출력 화면

```
pi@raspberrypi:~/Public $ sudo node dht11.js
Temperature: 0.00C, humidity: 0.00%
Temperature: 26.00C, humidity: 37.00%
Temperature: 26.00C, humidity: 37.00%
Temperature: 26.00C, humidity: 37.00%
Temperature: 26.00C, humidity: 37.00%
Temperature: 26.00C, humidity: 37.00%
```

1) 온도/습도 센서 활용

몽고DB 설치 후 가동제어

```
sudo service mongodb start
```

몽고DB 상태 확인

```
sudo service mongodb status
```

1) 온도/습도 센서 활용

몽고DB 서버 셧다운

```
sudo service mongodb stop
```

몽고DB에 온습도 데이터 넣기

https://github.com/3DKIDS/arduino/blob/master/20160919ModuFarm

1) 온도/습도 센서 활용


```
read: function () {
   var readout = sensorLib.read();
   console.log('Temperature: ' + readout.temperature.toFixed(2) + 'C, ' + 'humidity: ' +
readout.humidity.toFixed(2) + '%');
                Temp =  'Temp: ' + readout.temperature.toFixed(2) ;
                Hum  =  'Hum: ' + readout.humidity.toFixed(2);
                        var url = 'mongodb://localhost:27017/ModuFarm';
                        MongoClient.connect(url, function(err, db) {
                         assert.equal(null, err);
                         console.log("Connected correctly to server");
                         db.collection('Sensor1').insert(Temp, function(err, r) {
                                        assert.equal(null, err);
                                        assert.equal(1, r.insertedCount);
```

1) 온도/습도 센서 활용


```
                    });
                    db.collection('Sensor1').insert(Hum, function(err, r) {
                            assert.equal(null, err);
                            assert.equal(1, r.insertedCount);
                    });
                    db.close();
                });
setTimeout(function () {
  sensor.read();
}, 2000);}
```