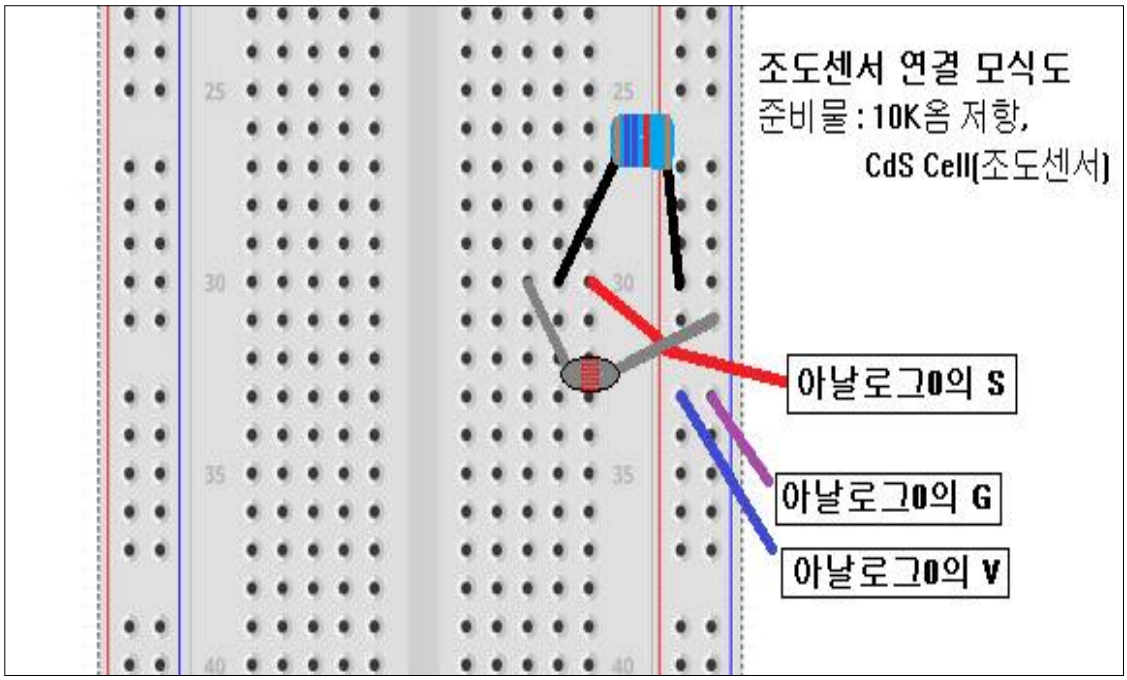
CdS Cell 모듈을 통한 조도 측정

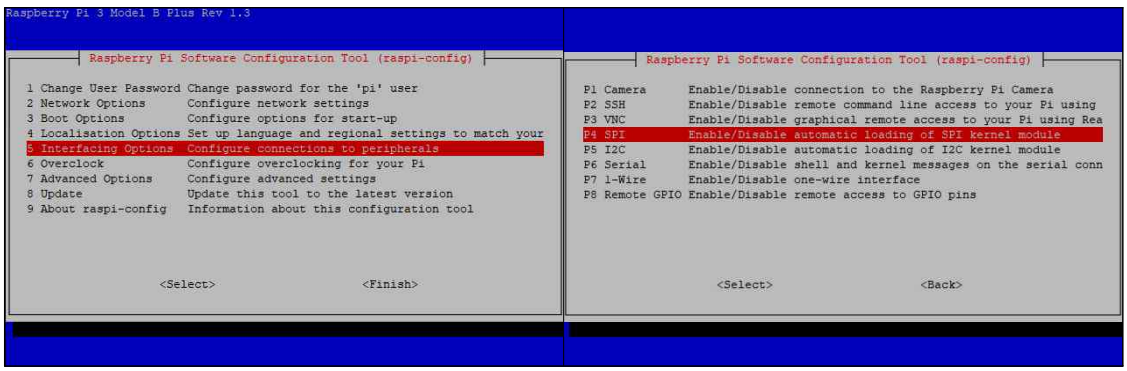
목표. 라즈베리파이와 CdS Cell을 연결하여 실시간 조도 측정 값을 각종 DB에 저장한다.
준비물. 라즈베리파이, 10K옴 저항, CdS Cell(GL5516 등), Octopus 5mm LED Brick.

1. 회로 구성



2. 시스템 준비

가. MYSQL DB 준비, THINGSPEAK DB 준비
나. SPI 통신 활성화
1) sudo raspi-config 입력 후 다음과 같이 적용.



2) sudo nano /etc/modules 입력후 i2c-dev 아랫줄에 spidev 추가
3) 라이브러리 설치
sudo apt-get install python dev
git clone git://github.com/Gadgetoid/py-spidev.git
cd py-spidev
sudo python setup.py install

3. 파이썬 코드작성

가. 아날로그 센서 활용

```python
import spidev
import time
import RPi.GPIO as gpio

#아날로그 센서와 연결을 위해 SPI 통신을 받아옵니다.
spi = spidev.SpiDev()
spi.open(0,0)
spi.max_speed_hz = 1350000

#아날로그 신호로 부터 값을 받아옵니다.
def analog_read(channel):
    r = spi.xfer2([1, (8 + channel) << 4, 0])
    adc_out = ((r[1]&3) << 8) + r[2]
    return adc_out

#참일경우(항상) 0번 슬롯에 연결된 아날로그 센서로부터 값을 받아옵니다.
#try 구문 없이도 작동하지만 try 구문 없이는 except 구문이 작동하지 않습니다.
try:
    while True:
        data = analog_read(0)
        voltage = data * 3.3 / 1024
        print("Data=%d\tVoltage=%f" % (data, voltage))
        time.sleep(1)

except KeyboardInterrupt:
    gpio.cleanup()
```

python3 작성한코드이름.py 로 작동시 다음과 같이 값이 출력된다.

```
pi@raspberrypi:~ $ python3 illumsemi1.py
Data=675        Voltage=2.175293
Data=682        Voltage=2.197852
Data=682        Voltage=2.197852
Data=683        Voltage=2.201074
Data=180        Voltage=0.580078
Data=189        Voltage=0.609082
Data=213        Voltage=0.686426
Data=243        Voltage=0.783105
Data=230        Voltage=0.741211
^Cillumsemi1.py:22: RuntimeWarning: No channels have been set up yet - nothing
o clean up!  Try cleaning up at the end of your program instead!
  gpio.cleanup()
```

표시되는 Reading 값이 0에 가까우면 매우 밝고 매우 어두운 경우 1000에 가깝다.
Cds Cell 버전에 따라 세부적인 차이는 있지만 일반적으로 형광등이 켜진 200~250정도이며 불을 다 끄고 작은 LED 하나를 센서에 가까이 했을 경우 7~800 정도의 값이 출력된다.

나. DB 활용하기

1) MariaDB 활용하기

MariaDB 활용을 위하여 다음과 같은 코드를 추가 한다.

```python
#MariaDB 사용을 위한 기능을 불러온다.
import pymysql

#Maria DB로 값을 전달하기 위한 기능
def insertDB(data):
#자신이 사용하는 DB의 값을 입력 한다.
    conn = pymysql.connect(host='localhost', user='smart', password='123', db='illumdb', charset='utf8')
    with conn.cursor() as cursor:
        sql = 'insert into testtable(illum) values(%s);'
        cnt = cursor.execute(sql, (data))
        r = conn.commit()
        if r == 0:
            print("Failed")
        else:
            print("Save Ok")
    conn.close()

#While True 구문의 아래에 다음 문구를 추가하여
#출력된 data 값이 위에 만든 기능에서 작동하도록 한다.
insertDB(data)
```

1-1) 결과확인

python3 저장한코드명.py 로 작동 시 출력값의 저장 여부가 확인되며, DB에도 출력된 data값이 입력되는 것을 확인 할 수 있다.

```
pi@raspberrypi:~ $ python3 illumsemi2.py
Data=699        Voltage=2.252637
Save Ok
Data=701        Voltage=2.259082
Save Ok
Data=703        Voltage=2.265527
Save Ok
Data=205        Voltage=0.660645
Save Ok
Data=225        Voltage=0.725098
Save Ok
^Cillumsemi2.py:39: RuntimeWarning: No channels h
o clean up!  Try cleaning up at the end of your p
  gpio.cleanup()
```

```
Database changed
MariaDB [illumdb]> select * from testtable;
+-----+-------+---------+
| _id | Illum | Voltage |
+-----+-------+---------+
|   1 | 699   |    NULL |
|   2 | 701   |    NULL |
|   3 | 703   |    NULL |
|   4 | 205   |    NULL |
|   5 | 225   |    NULL |
+-----+-------+---------+
5 rows in set (0.001 sec)
```

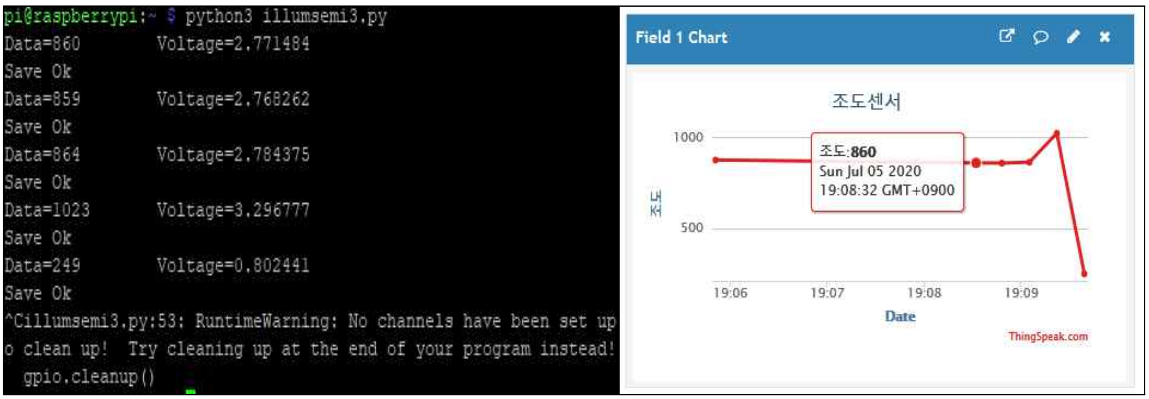
2) THINGSPEAK 활용하기

Thingspeak 활용을 위하여 아래의 코드를 추가한다.

```python
#ThingSpeak 활용을 위해 기능을 불러온다.
import urllib.request

#본인 DB의 api 키 값을 입력한다.
def insertCloud(data)
    api_key = '################'
    url = 'https://api.thingspeak.com/update'
    url = url + '?api_key=%s' % api_key
    url = url + '&field1=%s' % data
    #print(url)
    urllib.request.urlopen(url)

#While True 구문의 아래에 다음 문구를 추가하여
#출력된 data 값이 위에 만든 기능에서 작동하도록 한다.
insertCloud(data)
```

2-1) 결과확인

python3 저장한코드명.py 로 작동 시 출력값의 저장 여부가 확인되며, DB에도 출력된 data값이 입력되는 것을 확인 할 수 있다.

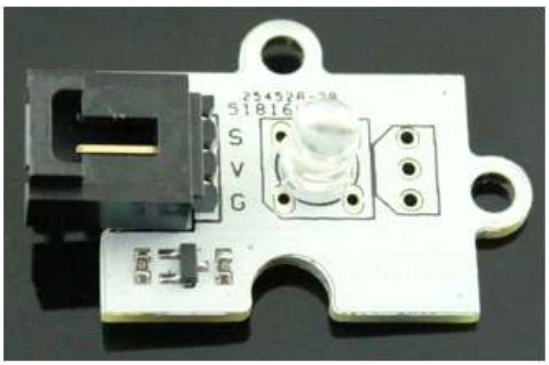


3) 측정된 아날로그 값을 DB로 저장하는 코드 종합

```python
import spidev
import time
import RPi.GPIO as gpio
#MariaDB 사용을 위한 기능을 불러온다.
import pymysql
#Thingspeak 사용을 위한 기능을 불러온다.

import urllib.request
#아날로그 센서와 연결을 위해 SPI 통신을 받아옵니다.
spi = spidev.SpiDev()
spi.open(0,0)
spi.max_speed_hz = 1350000

#mysql DB로 값을 전달하기 위한 기능
def insertDB(data):
#자신이 사용하는 DB의 값을 입력 한다.
    conn = pymysql.connect(host='localhost', user='smart', password='123', db='illumdb', charset='utf8')
    with conn.cursor() as cursor:
        sql = 'insert into testtable(illum) values(%s);'
        cnt = cursor.execute(sql, (data))
        r = conn.commit()
        if r == 0:
            print("Failed")
        else:
            print("Save Ok")
    conn.close()

#Thingspeak로 데이터 값을 전달하기 위한 기능
#본인 DB의 api 키 값을 입력한다.
def insertCloud(data):
    api_key = '자신의Thingspeak api키'
    url = 'https://api.thingspeak.com/update'
    url = url + '?api_key=%s' % api_key
    url = url + '&field1=%s' % data
    #print(url)
    urllib.request.urlopen(url)

#아날로그 신호로 부터 값을 받아옵니다.
def analog_read(channel):
    r = spi.xfer2([1, (8 + channel) << 4, 0])
    adc_out = ((r[1]&3) << 8) + r[2]
    return adc_out

#참일경우(항상) 0번 슬롯에 연결된 아날로그 센서로부터 값을 받아옵니다.
#try 구문 없이도 작동하지만 try 구문 없이는 except 구문이 작동하지 않습니다.
try:
    while True:
        data = analog_read(0)
        voltage = data * 3.3 / 1024
        print("Data=%d\tVoltage=%f" % (data, voltage))
#확인된 값을 MariaDB로 전송한다.
        insertDB(data)
#확인된 값을 Thingspeak로 전송한다.
        insertCloud(data)
#Thingspeak의 데이터 갱신 주기(15초)이상으로 설정한다.
        time.sleep(16)

except KeyboardInterrupt:
    gpio.cleanup()
```

3. 조도의 변화에 따라 LED가 작동하게 하기

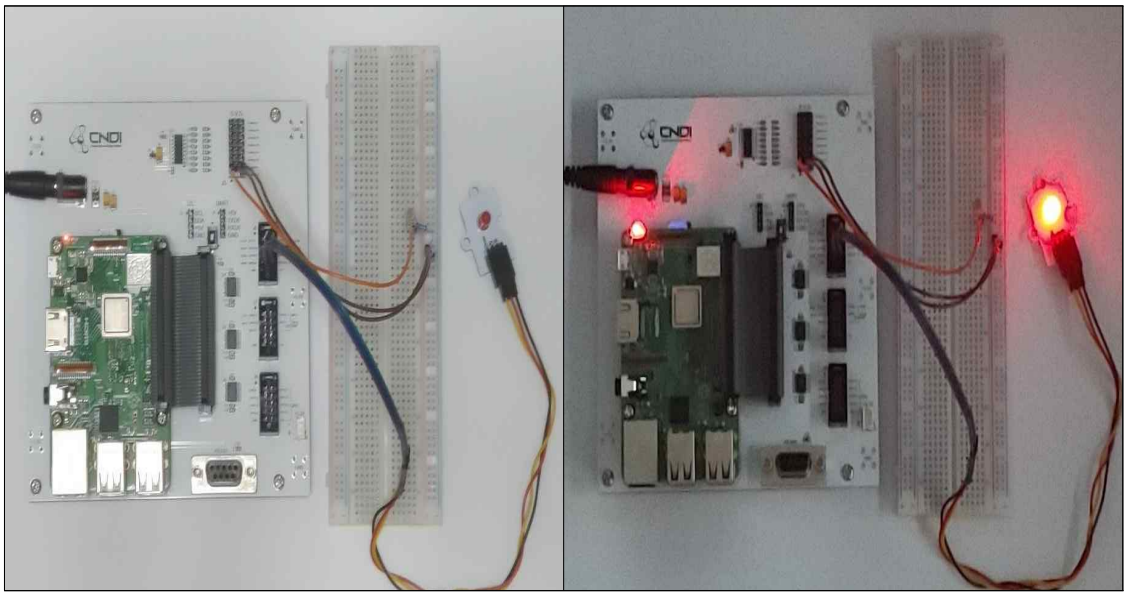
가. LED의 연결

라즈베리파이와 연결 S : GPIO 17 V : VCC 5V G : GND	

나. LED 작동을 위한 코드 추가

```
#LED가 연결된 GPIO 슬롯을 활용하기 위한 기능을 불러온다.
import RPi.GPIO as gpio

#LED를 작동하기 위한 코드
#LED가 장착된 GPIO 슬롯 번호를 입력
LED = 17
gpio.setmode(gpio.BCM)
gpio.setup(LED, gpio.OUT)

#출력된 DATA 값에 따른 LED의 변화를 위한 기능으로
#WhileTrue 구문의 안에 집어넣는다.
#값이 800보다 적은(밝은)경우 LED작동을 정지하게 하는 코드
#어두울때 LED가 작동하게 하는경우 켜진 LED에 의해 LED가 꺼질수 있음.
if data<800:
        gpio.output(LED, gpio.LOW)
else:
        gpio.output(LED, gpio.HIGH)
```

다. 작동확인



위와 같이 주변의 밝기에 따라 LED가 ON/OFF 됨을 확인 할수 있다.

4. 전체 파이썬 코드

```python
import spidev
import time
import RPi.GPIO as gpio
#MariaDB 사용을 위한 기능을 불러온다.
import pymysql
#Thingspeak 사용을 위한 기능을 불러온다.
import urllib.request

#LED를 작동하기 위한 코드
#LED가 장착된 GPIO 슬롯 번호를 입력
LED = 17
gpio.setmode(gpio.BCM)
gpio.setup(LED, gpio.OUT)

#아날로그 센서와 연결을 위해 SPI 통신을 받아옵니다.
spi = spidev.SpiDev()
spi.open(0,0)
spi.max_speed_hz = 1350000

#mysql DB로 값을 전달하기 위한 기능
def insertDB(data):
#자신이 사용하는 DB의 값을 입력 한다.
    conn = pymysql.connect(host='localhost', user='smart', password='123', db='illumdb', charset='utf8')
    with conn.cursor() as cursor:
        sql = 'insert into testtable(illum) values(%s);'
        cnt = cursor.execute(sql, (data))
        r = conn.commit()
        if r == 0:
            print("Failed")
        else:
            print("Save Ok")
    conn.close()

#Thingspeak로 데이터 값을 전달하기 위한 기능
#본인 DB의 api 키 값을 입력한다.
def insertCloud(data):
    api_key = 'I8WGEU0VYPRN5XES'
    url = 'https://api.thingspeak.com/update'
    url = url + '?api_key=%s' % api_key
    url = url + '&field1=%s' % data
    #print(url)
    urllib.request.urlopen(url)

#아날로그 신호로 부터 값을 받아옵니다.
def analog_read(channel):
    r = spi.xfer2([1, (8 + channel) << 4, 0])
    adc_out = ((r[1]&3) << 8) + r[2]
    return adc_out

#참일경우(항상) 0번 슬롯에 연결된 아날로그 센서로부터 값을 받아옵니다.
#try 구문 없이도 작동하지만 try 구문 없이는 except 구문이 작동하지 않습니다.
try:
    while True:
        data = analog_read(0)
        voltage = data * 3.3 / 1024
        print("Data=%d\tVoltage=%f" % (data, voltage))
#값이 800보다 적은(밝은)경우 LED작동을 정지하게 하는 코드
#어두울때 LED가 작동하게 하는경우 켜진 LED에 의해 LED가 꺼질수 있음.
        if data<800:
            gpio.output(LED, gpio.LOW)
        else:
            gpio.output(LED, gpio.HIGH)
#확인된 값을 MariaDB로 전송한다.
        insertDB(data)
#확인된 값을 Thingspeak로 전송한다.
        insertCloud(data)
#Thingspeak의 데이터 갱신 주기(15초)이상으로 설정한다.
        time.sleep(16)

except KeyboardInterrupt:
    gpio.cleanup()

```

5. 참조 사이트

https://m.blog.naver.com/PostView.nhn?blogId=roboholic84&logNo=220367321777&proxyReferer=https:%2F%2Fwww.google.com%2F