IoT 개론 및 실습

2018 1학기

강승우

2. 사물 인터넷 디바이스

2-3. 라즈베리 파이를 이용한 IoT 디바이스 프로그래밍

학습 목표

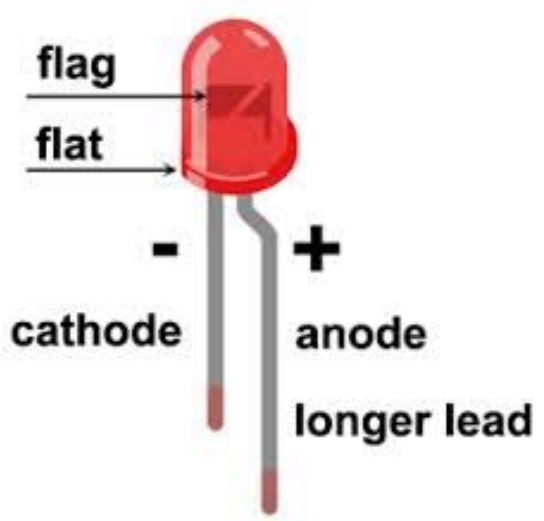
- 라즈베리 파이를 이용하기 위한 기본 지식을 이해할 수 있다
- 파이썬을 이용한 라즈베리 파이 GPIO 프로그래밍을 할 수 있다

2) IoT 디바이스 프로그래밍

1. LED 제어하기

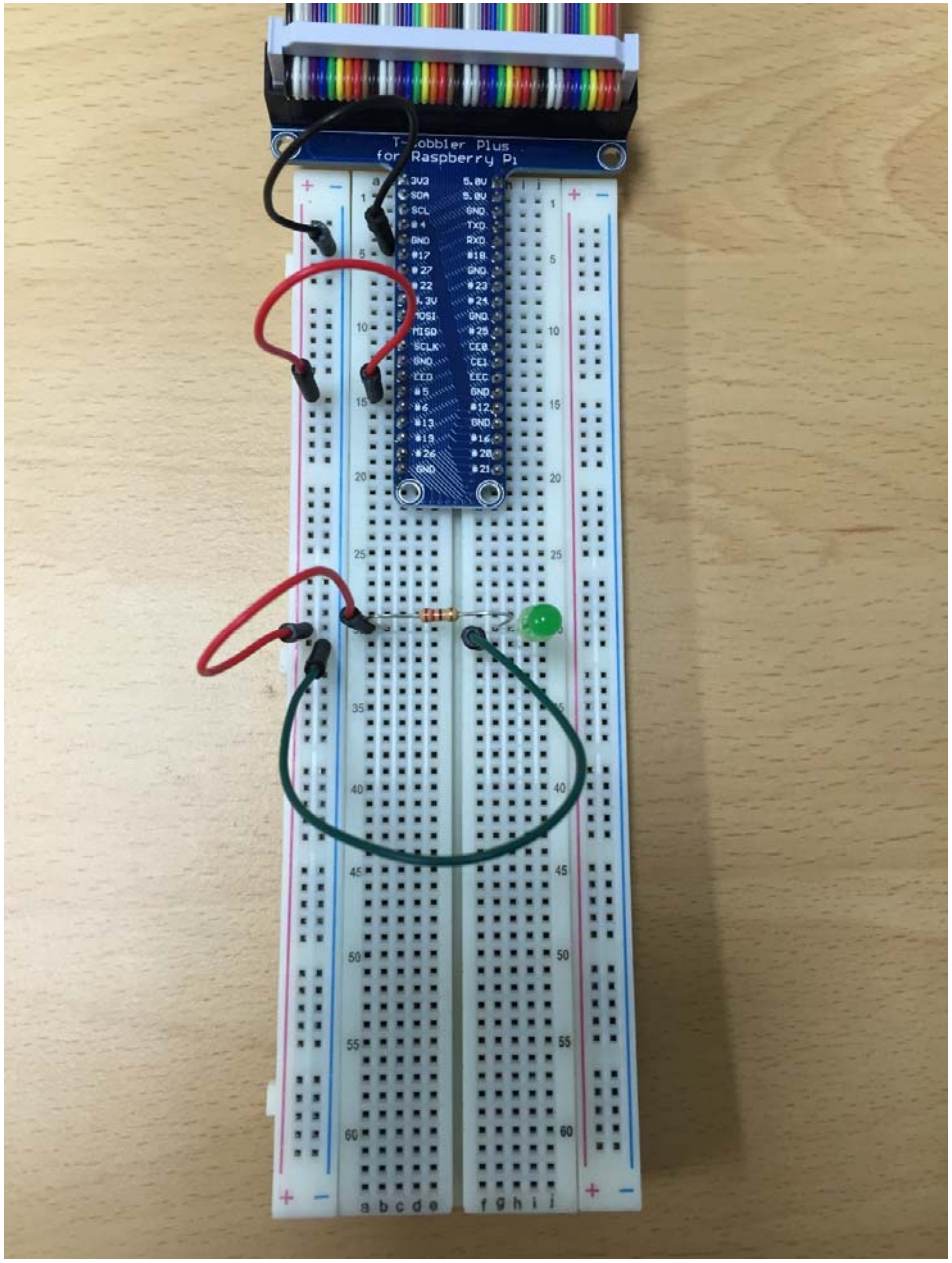
- 라즈베리 파이의 GPIO 핀을 통해 LED를 켜고 끄는 것을 제어

- 준비
  - 라즈베리 파이와 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



```
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
- 5번 GPIO 핀을 출력핀으로 사용하여 LED 제어
  - 5번 핀이 아닌 다른 핀을 사용하는 경우 3번 라인에서 숫자를 변경해주어야 함

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()
```

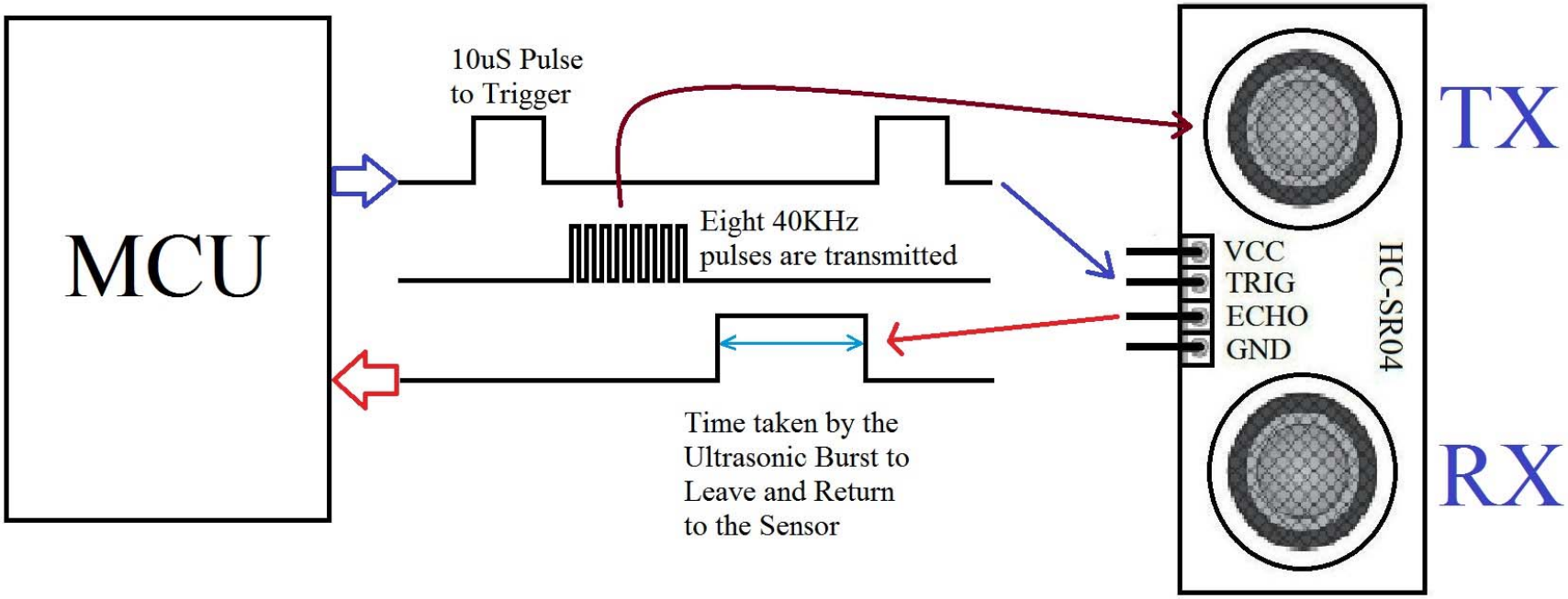
2. 초음파 센서로 거리 측정하기

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

http://microcontrollerelectronics.com/distance-sensing

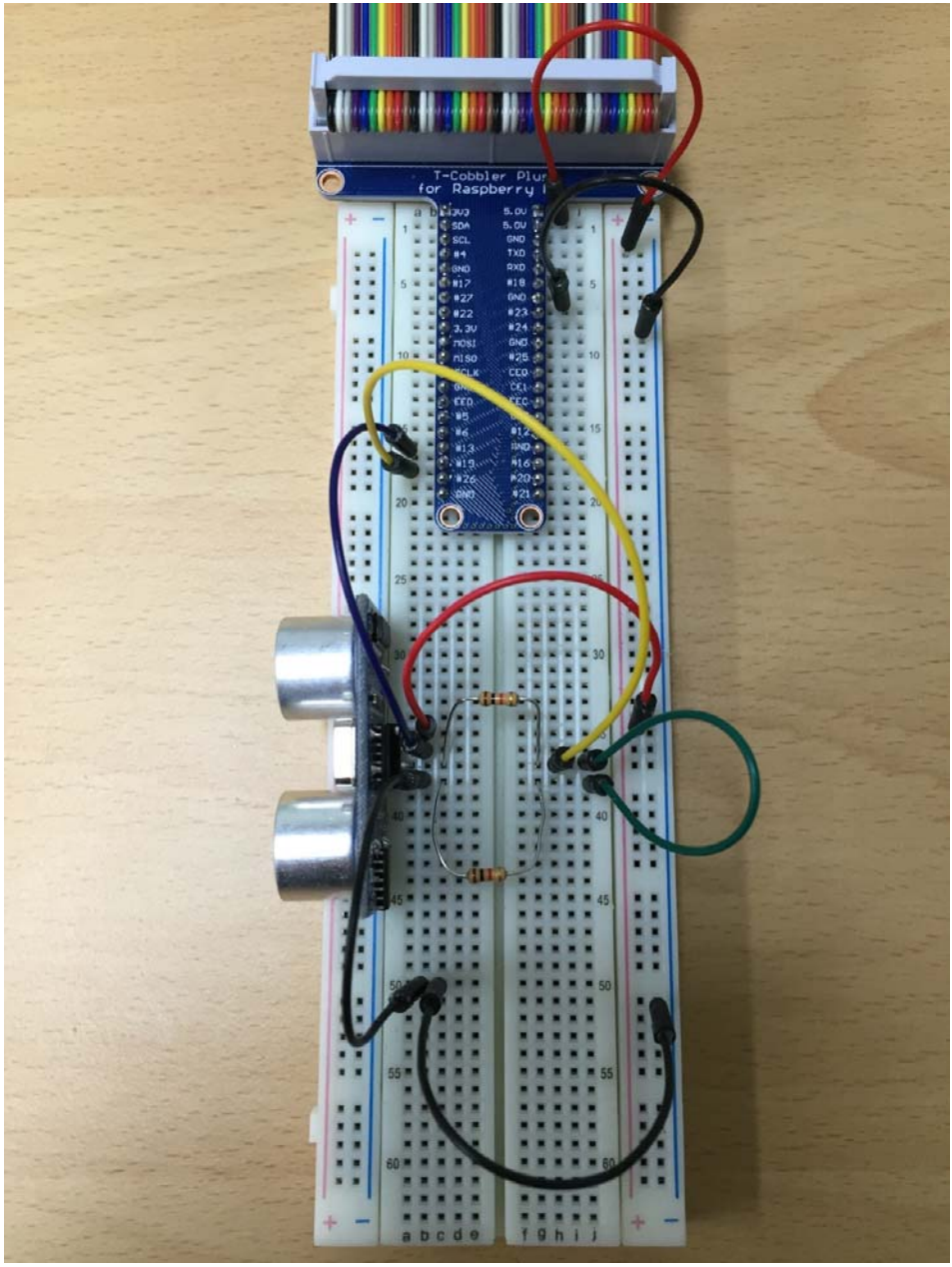
- HC-SR04 초음파 센서
  - 2~400cm 거리 감지
  - 해상도 0.3cm
  - 30도 정도 각도 범위 이내 물체 감지
- 4개 핀
  - VCC: 5V 전원을 공급하는 핀
  - Trig: 트리거 신호 입력 핀 트리거신호를 주면 음파가 발생한다
    - 라즈베리 파이나 아두이노와 같은 마이크로컨트롤러 유닛에서 트리거 신호를 발생하여 이 핀을 통해 센서로 입력된다. 트리거 신호가 입력되면 HC-SR04 센서의 좌측의 초음파 발생부에서 초음파 신호가 전송된다.
  - Echo: Trig 핀과 반대 역할을 하는 핀으로서, 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. 그리고 이를 핀을 통해 에코 신호가 마이크로컨트롤러 유닛에 전송된다.
  - GND: 그라운드에 연결하는 핀
    - VCC 핀으로 전원을 공급하기 전에 그라운드 핀을 그라운드에 연결해야 한다.

- 동작 원리



- 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 13번 핀 줄력으로 할거임
    - 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



예제 코드: /sensor_ultrasonic/ultra.py

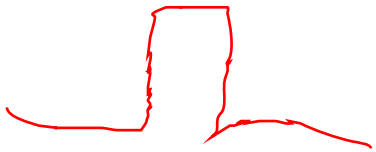
```
import RPi.GPIO as gpio
import time

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()
```

1이되는 순간 그 직전의 시간이 pulse_start에 기록이 될 것임

pulse Duraition 계산

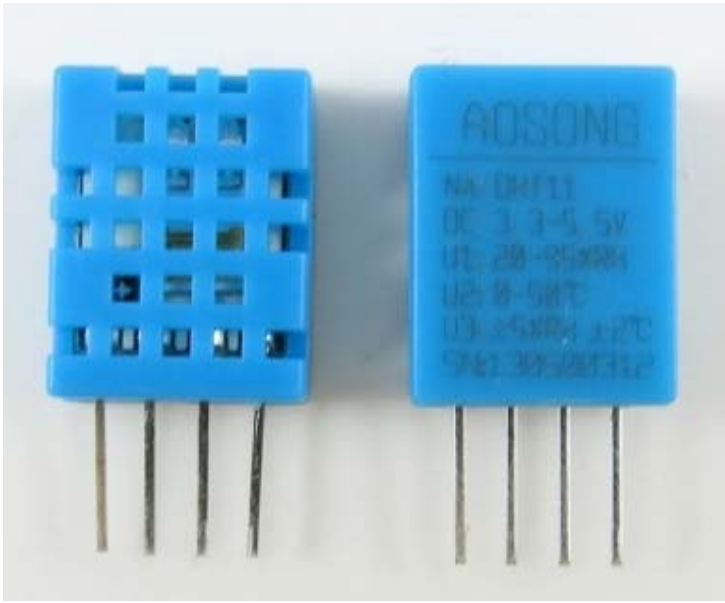
false로 있다가 true로 탁 해줄 거임

소숫점 2자리까지



3. 온도/습도 측정하기

- 온/습도 센서를 이용하여 온도와 습도를 측정
- 준비
  - 라즈베리 파이와 GPIO 케이블로 연결된 코블러 브레이크아웃 보드와 브레드보드
  - DHT11 온/습도 센서 1개
  - 저항 1개
  - 점퍼 와이어

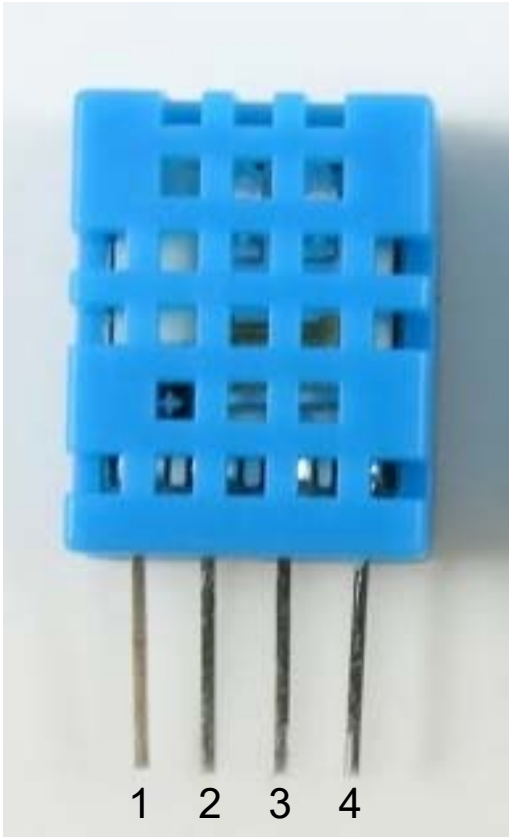


http://rayshobby.net/cart/dht11

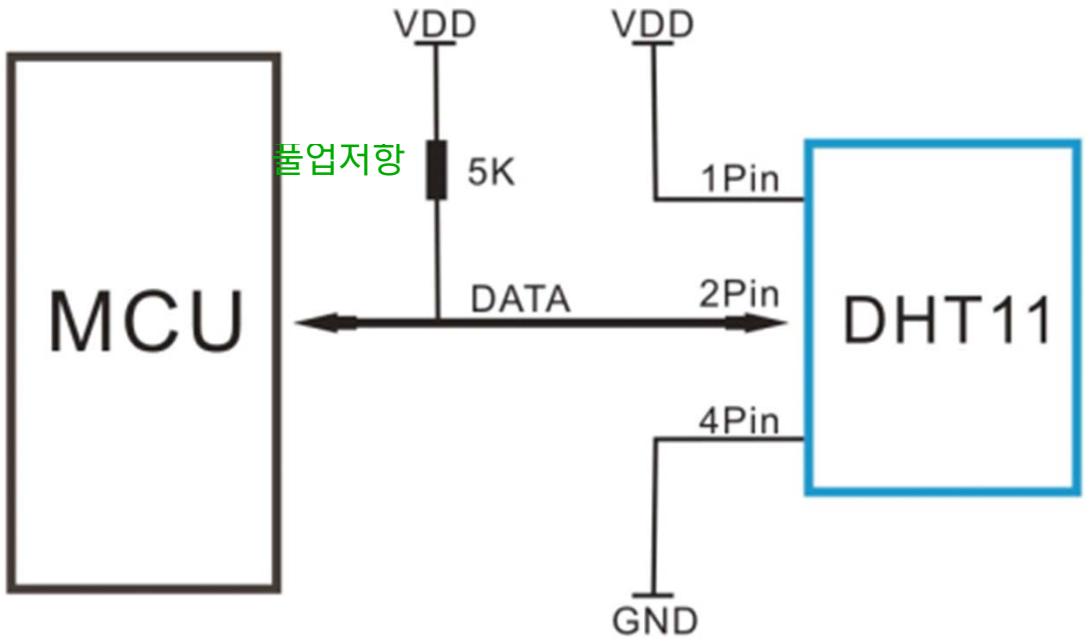
- DHT11 센서
  - 온도와 습도를 모두 측정
  - 습도
    - 상대 습도를 나타내며, 16 비트로 표현
    - 20%에서 90% 범위의 상대 습도를 측정할 수 있으며, 해상도는 1%
    - 약 5%의 오차가 있을 수 있는 정확도
  - 온도
    - 16 비트로 표현되며 섭씨 0도에서 50도 범위의 온도를 측정
    - 해상도는 섭씨 1도
    - 약 섭씨 2도의 오차 가능

그렇게 좋은센서는 아니다

- DHT11 센서 핀
  - 정면 좌측에서 우측으로 1번 - 4번 핀
  - 1번 핀: 전원 공급 핀
    - 3.5-5.5V 직류 전압을 인가한다.
  - 2번 핀: 데이터 핀 2번핀을 통해 온도와습도가 들어온다.
    - DHT11 센서는 하나의 데이터 버스를 이용하여 통신
    - 이 핀을 통해 MCU와 신호를 주고받음
  - 3번 핀: 사용하지 않는 핀
  - 4번 핀: 그라운드(GND) 핀



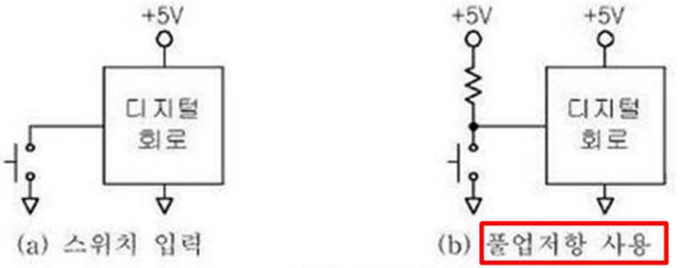
- DHT11 센서의 일반적인 핀 연결
  - 2번 핀의 연결
    - 마이크로컨트롤러 유닛과 연결
  - 풀업 저항 연결
    - 20미터 이내의 케이블을 이용하는 경우 5K 옴 정도의 저항 사용 권장



http://www.micropik.com/PDF/dht11.pdf

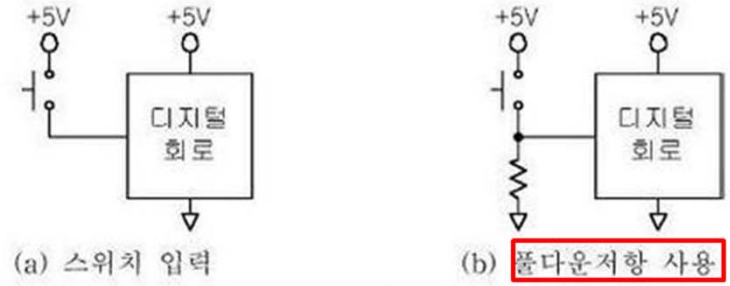
- 풀업/풀다운 저항
  - 아래 링크의 튜토리얼 참고
    - https://www.kocoafab.cc/tutorial/view/526

UP상태가 기본인게 풀업



<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

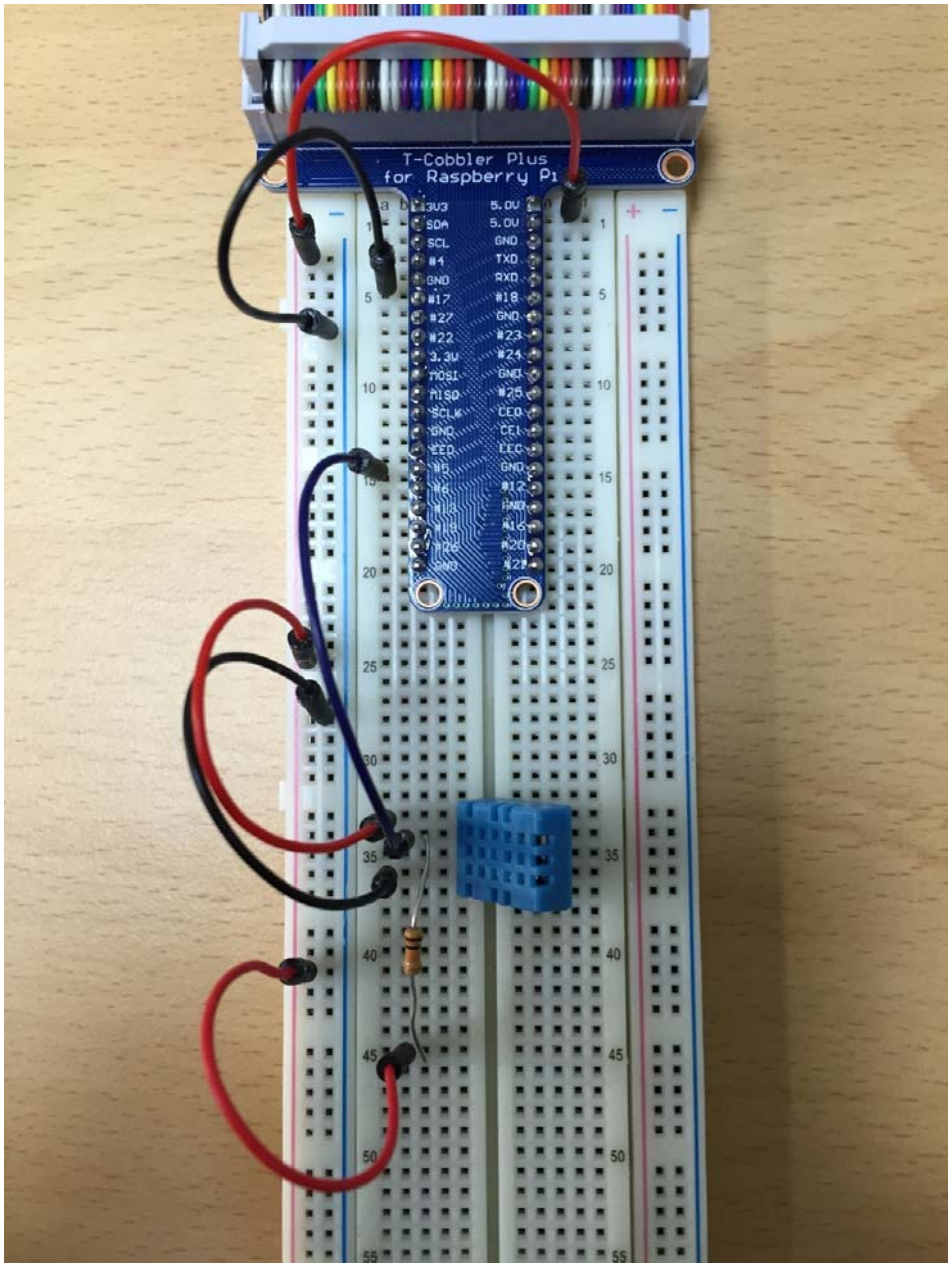
DOWN상태가 기본인게 풀다운

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

- DHT11 데이터 통신 2번핀을 통해 주고받음
  - 2번 데이터 핀을 통해서 DHT11 센서와 MCU 사이에 통신 및 동기화가 이루어짐
  - MCU에서 먼저 트리거 신호를 보내면 이에 따라 온도/습도 데이터가 담긴 응답 신호를 센서에서 MCU로 전송
- DHT11 센서 데이터 포맷 하나의핀을 가지고 신호를 보내고 받는게 이루어진다.
  - 40비트 일단 데이터는 40비트 데이터를 전송하고, 받음
    - 16비트: 습도, 16비트: 온도, 8비트: 패리티 비트 (에러 확인용)
    - 16비트 온/습도: 상위 8비트 (정수부), 하위 8비트 (소수점 이하)
    - 4개의 8비트 데이터를 모두 합한 값이 8비트 패리티 비트와 일치해야만 수신된 데이터가 정확하다는 의미
  - 예: 0011 0101 0000 0000 0001 1000 0000 0000 0100 1101
    - 습도 정수부: 0011 0101, 소수점 이하: 0000 0000
    - 온도 정수부: 0001 1000, 소수점 이하: 0000 0000
    - 모두 더하면: 0100 1101 → 마지막 8비트 패리티 비트와 동일 동일하면 정상적인데이터
    - 습도 값: 32 + 16 + 4 + 1 = 53 → 상대 습도 53%
    - 온도 값: 16 + 8 = 24 → 섭씨 24도

- 회로 구성
  - 전원 연결
    - 5V 전원
  - 그라운드 연결
  - DHT11 센서 장착 및 전원, GND 연결
    - 4번 GND 핀 ←→ (-) 홀
    - 1번 VCC 핀 ←→ (+) 홀
  - 데이터 핀(2번) 연결
    - DHT11 센서 2번 핀 ←→ GPIO 5번 핀
    - DHT11 센서 2번 핀 ←→ 저항 한쪽 끝. 저항 다른 쪽 끝 ←→ (+) 홀



40비트 나누어서 계산하는 건
직접 할 필요 없고 알아서 해주는 모듈이 있다.

예제 코드: /sensor_DHT11/dht11_example.py

```
import RPi.GPIO as gpio
import dht11
import time
import datetime

# initialize GPIO
gpio.setwarnings(False)
gpio.setmode(gpio.BCM)
gpio.cleanup()

# read data using pin 5
instance = dht11.DHT11(pin = 5)
```

```
try:
    while True:
        result = instance.read()
        if result.is_valid():
            print("Last valid input: " +
str(datetime.datetime.now()))
            print("Temperature: %d C" % result.temperature)
            print("Humidity: %d %%" % result.humidity)

        time.sleep(3)

except KeyboardInterrupt:
    gpio.cleanup()
```

데이터 읽기

아까 패리티비트로 오류 검사했는데 오류가아니면

- 예제 코드 동작
  - GPIO 5번 핀을 통하여 DHT11 센서에서 3초 간격으로 신호를 읽고 그 신호에 에러가 없으면, 현재 시간과 온도, 습도 데이터를 출력

  - DHT11 파이썬 모듈 이용
    - DHT11 센서의 데이터 신호를 읽고 해석하기 위해서 만들어진 모듈
    - 소스 코드 및 예제 코드 링크: https://github.com/szazo/DHT11_Python
      - dht11.py

4. 아날로그 조도/온도 센서 이용하기

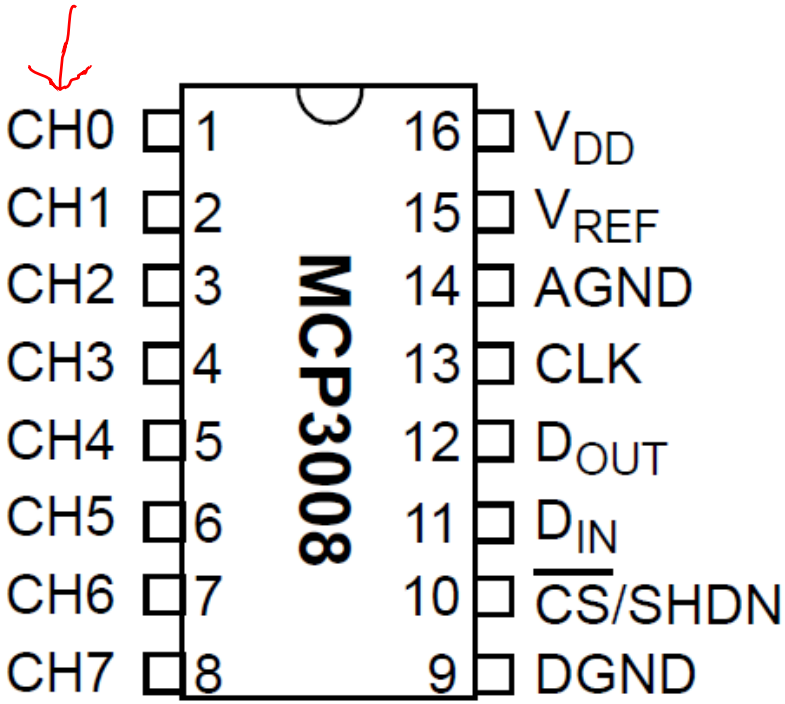
- 라즈베리 파이에서 아날로그 센서 이용
  - 라즈베리 파이 GPIO 핀은 디지털 신호의 입출력 용도로만 사용 가능 숭요
  - 아날로그 센서를 이용하기 위해서는 외부 ADC 칩을 사용 ADC칩의 중요성
- 준비
  - 라즈베리 파이와 GPIO 케이블로 연결된 코블러 브레이크아웃 보드와 브레드보드
  - MCP3008 칩 1개
  - TMP36 온도 센서 1개
  - 조도 센서 1개
  - 저항 1개
  - 점퍼 와이어

- MCP3008 칩
  - 아날로그 데이터를 디지털 데이터로 바꿔주는 ADC
    - 8개 아날로그 채널 입력
    - 10비트 디지털 데이터로 변환
    - 초당 200 샘플링

http://www.microchip.com/wwwproducts/Devices.aspx?product=MCP3008

- MCP3008 핀 (총 16개)
  - 좌 8개: 아날로그 입력 채널
  - 9번 DGND(Digital Ground): 내부 디지털 회로와 연결되는 디지털 그라운드 핀
  - 10번 CS/SHDN: Chip Select/Shutdown Input 핀
    - 통신을 시작하고 종료하기 위해 사용
    - Low 상태가 되면 통신을 시작하고, High 상태가 되면 AD 변환을 종료하고 기기를 저전력 대기모드로 변경
  - 11번 DIN: 직렬 데이터 입력 핀
    - 칩의 채널 설정 데이터를 입력 받는데 사용
    - 여기에서는 라즈베리 파이에서 전송하는 데이터가 이 핀을 통해 입력
  - 12번 DOUT: 직렬 데이터 출력 핀
    - AD 변환이 이루어진 결과 데이터를 전송하는데 사용
  - 13번 CLK: 직렬 클럭 핀
    - AD 변환을 시작시키기 위해 사용
  - 14번 AGND(Analog Ground): 내부 아날로그 회로와 연결되는 아날로그 그라운드 핀
  - 15번 VREF: 레퍼런스 전압 입력 핀
    - 이 레퍼런스 전압이 아날로그 입력 전압 범위를 결정
    - 여기서는 라즈베리 파이에서 공급되는 3.3V 전원을 인가
  - 16번 VDD: 전원 공급 핀
    - 2.7V에서 5.5V 전원 공급이 가능
    - 여기서는 라즈베리 파이에서 공급되는 3.3V 전원을 인가

아날로그 신호를 받을 수 있는 채널



라즈베리파이에서 spi를 지원하는 핀이 있다

- SPI(Serial Peripheral Interface, 직렬 주변기기 인터페이스)
  - MCP3008 칩은 SPI를 통하여 라즈베리 파이와 같은 마이크로컨트롤러 혹은 마이크로컴퓨터와 통신
  - 외부 주변장치와 클럭을 통한 동기식 통신 방식
  - 마스터 슬레이브 모드를 기반으로 하여 하나의 마스터와 하나 혹은 다수의 슬레이브 장치 간 통신을 지원
  - 마스터 장치와 슬레이브 장치 사이에 4가지 연결이 필요
    - SCLK, MISO, MOSI, CS M는 라즈베리파이 S는 mcp칩
    - MCP3008 칩의 13번 CLK 핀, 12번 DOUT 핀, 11번 DIN 핀, 10번 CS/SHDN 핀에 대응

라즈베리파이 GPIO에선 2개까지 이원이 됨

CE0
CE1이 칩 셀렉트

- 라즈베리 파이의 SPI
  - 1개의 SPI 버스
  - 2개의 Chip Select
  - GPIO 핀 중에서 SPI 통신용으로 지정된 핀
    - SCLK(SPI_CLK): 클럭 신호를 출력
      - MCP3008의 13번 CLK 핀에 연결
    - MISO(SPI_MISO, SPI Master In Slave Out): SPI 마스터 기기가 ADC 데이터 입력을 받음
      - MCP3008의 12번 DOUT 핀에 연결
    - MOSI(SPI_MOSI, SPI Master Out Slave In): SPI 마스터 기기가 데이터 출력을 보냄
      - MCP3008의 11번 DIN 핀에 연결
    - CE0(Chip Enable 0) / CE1(Chip Enable 1): 2개의 Chip Select
      - MCP3008의 10번 CS/SHDN 핀을 이 둘 중 하나에 연결

- 라즈베리 파이에서 SPI 통신 활성화
  - 터미널의 프롬프트에서 raspi-config 명령어를 실행 (sudo raspi-config)
  - Advanced Options을 선택
  - SPI를 선택한 후 SPI를 사용하도록 설정

라즈베리파이 prefences -> Resabary pi Configuration

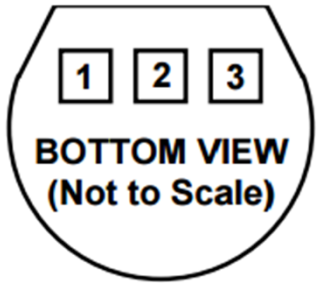
- 참고 링크:

https://learn.sparkfun.com/tutorials/raspberry-pi-spi-and-i2c-tutorial

- SPI 디바이스와 인터페이스하는 파이썬 모듈 설치
  - 1) 라즈베리 파이 최신 업데이트
    - sudo apt-get update
    - sudo apt-get upgrade
  - 2) Python Dev 패키지가 설치되어 있지 않다면 설치
    - sudo apt-get install python-dev
  - 3) 파이썬 SPI 모듈이 공개된 github 저장소를 클론하여 다운로드
    - git clone https://github.com/Gadgetoid/py-spidev.git
  - 4) 다운로도 받은 저장소(py-spidev) 디렉토리로 이동
    - cd py-spidev/
  - 5) 파이썬 모듈을 설치한다.
    - sudo python setup.py install
  - 6) 재부팅 한다.
    - sudo shutdown –r now

- TMP36 센서
  - 2.7V에서 5.5V 사이 전압에서 동작하는 온도 센서
  - 섭씨 온도에 비례하는 전압을 출력
    - 섭씨 25도: 750mV 출력
    - 섭씨 1도당 10mV 씩 증감
  - -40도에서 125도 범위에서 최대 ±2도의 오차
  - 3개의 핀
    - 전원 공급 핀, 출력 전압 핀, 그라운드 핀

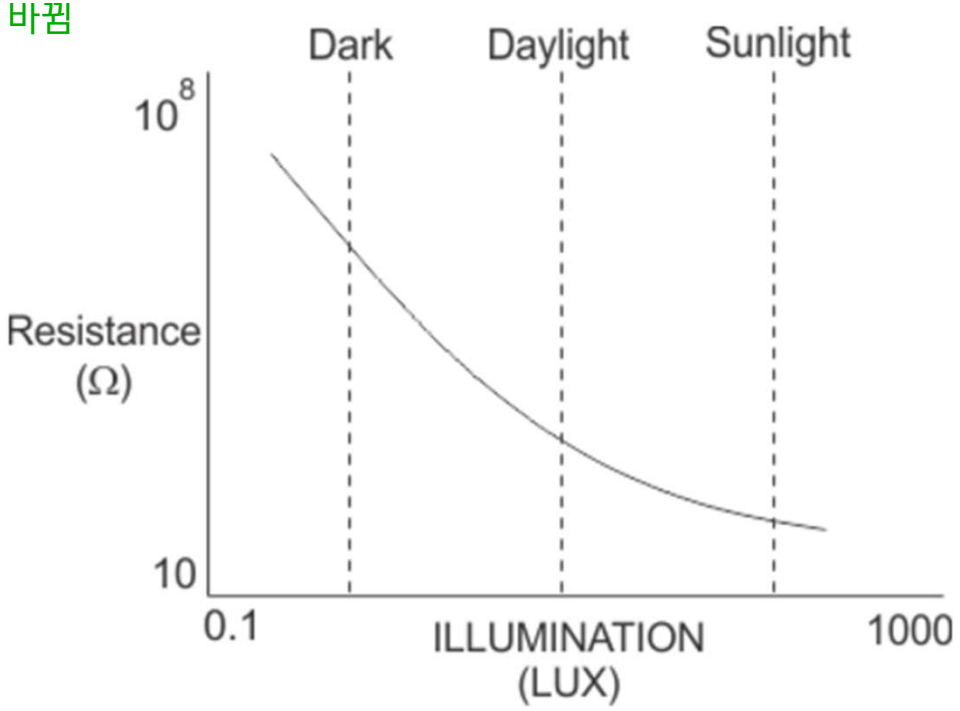
https://www.sparkfun.com/products/10988



PIN 1, $+V_S$; PIN 2, V_{OUT}; PIN 3, GND

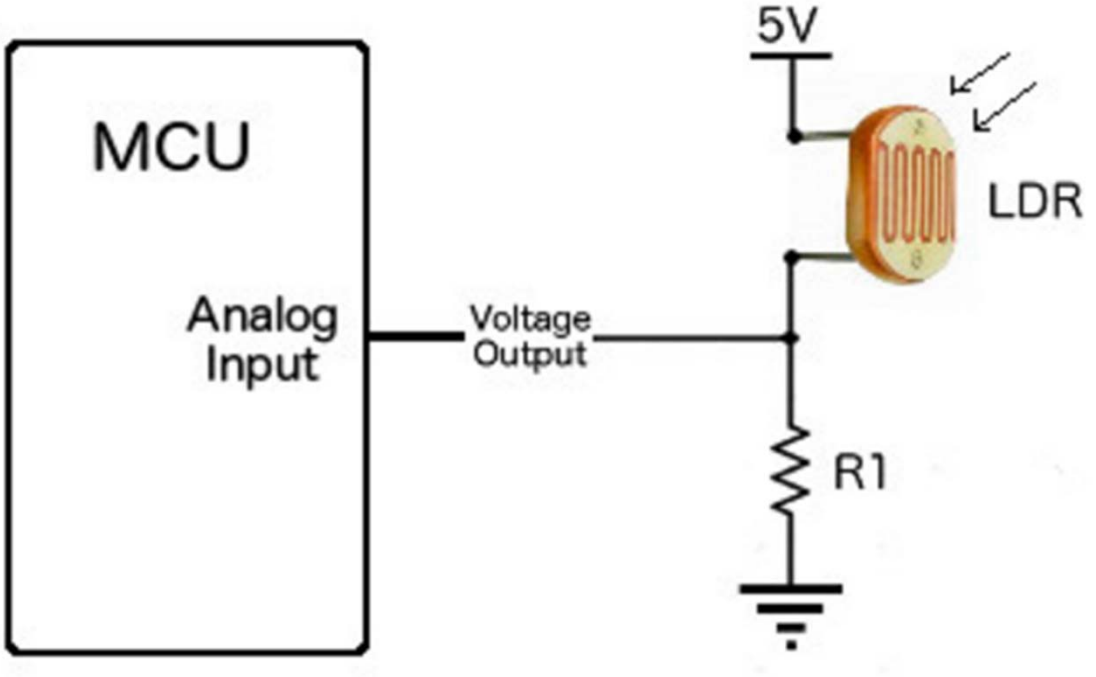
- 조도 센서
  - LDR(Light Dependent Resistor)
  - 조도에 따라 저항값이 바뀜
    - 조도가 낮은 어두운 곳에서는 저항이 매우 커져서 수M옴 이상으로 증가
    - 조도가 높은 밝은 곳에서는 저항이 작아져서 수K옴 이하로 감소

말그대로 저항인데 조도에따라 바뀜



http://www.electrical4u.com/light-dependent-resistor-ldr-working-principle-of-ldr/

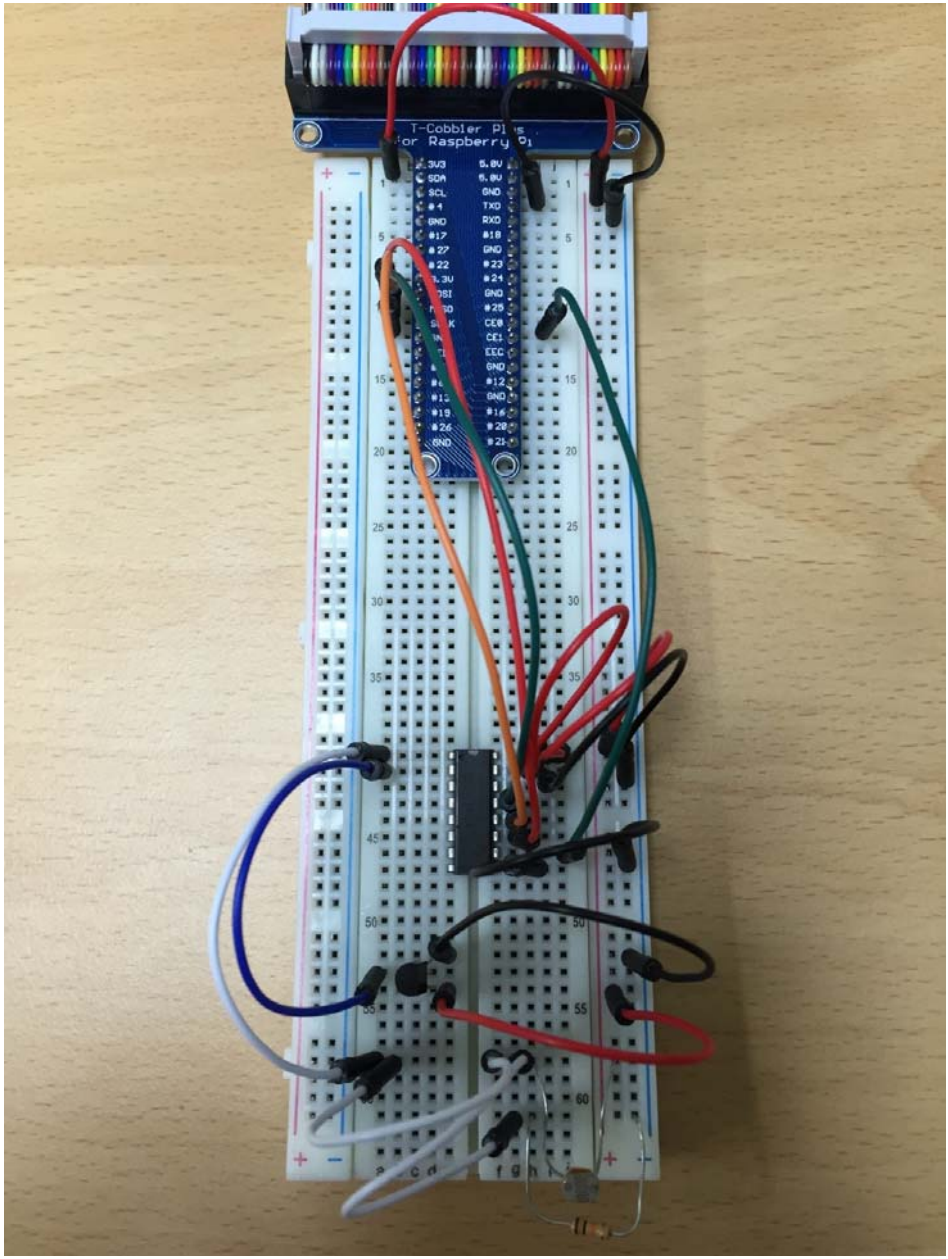
- 조도 센서 사용 방법
  - Voltage divider 회로로 연결하여 사용
  - 조도에 따라 저항값이 달라지므로 R1에 걸리는 전압이 그에 따라 달라짐
  - 매우 어두운 곳
    - 조도 센서의 저항이 매우 커지므로 R1에 걸리는 전압 감소
    - R1이 상대적으로 많이 작은 경우 0V에 가까운 값을 가질 수 있음
  - 매우 밝은 곳
    - 조도 센서의 저항이 매우 작아지므로 상대적으로 R1에 걸리는 전압 증가



http://cactus.io/hookups/sensors/light/ldr/hookup-arduino-to-ldr-sensor

  - R1에 걸리는 전압 출력 → 아날로그 입력
    - MCP3008의 아날로그 입력 채널로 연결되어 디지털 값으로 변환된 후 라즈베리 파이로 전송

- 회로 구성
  - 전원 연결
    - 5V 전원
  - 그라운드 연결
  - MCP3008 칩 연결
    - 9번 DGND 핀, 14번 AGND 핀 ←→ 그라운드 (-) 홀
    - 15번 VREF 핀, 16번 VDD 핀 ←→ 전원 (+) 홀
    - 10번 CS/SHDN 핀 ←→ CE0 핀
    - 11번 DIN 핀 ←→ MOSI 핀
    - 12번 DOUT 핀 ←→ MISO 핀
    - 13번 CLK 핀←→ SCLK 핀
  - 조도 센서 연결
    - 한쪽 핀 ←→ 전원 (+) 홀
    - 다른 쪽 핀 ←→ MCP3008 칩 1번 CH0 핀
    - 저항(1K옴) 한쪽 핀 ←→ 그라운드 (-) 홀
    - 저항 다른 쪽 핀 ←→ 조도 센서 핀
  - TMP36 온도 센서 연결
    - GND 핀 ←→ 그라운드 (-) 홀
    - 전원 핀 ←→ 전원 (+) 홀
    - 데이터 핀 ←→ MCP3008 칩 2번 CH1 핀



예제 코드: /sensor_SPI_ADC/light-temperature_mcp3008.py

```
import spidev
import time

spi = spidev.SpiDev()
spi.open(0,0)

light_channel = 0
temp_channel = 1

def readChannel(channel):
    adc = spi.xfer2([1, (8 + channel) << 4, 0])
    adc_out = ((adc[1] & 3) << 8) + adc[2]
    return adc_out
```

채널설정

```
def convert2volts(data, places):
    volts = (data * 3.3) / float(1023)
    volts = round(volts, places)
    return volts

def convert2temp(data, places):
    temp = ((data * 330) / float(1023)) - 50
    temp = round(temp, places)
    return temp

try:
    while True:
        light_level = readChannel(light_channel)
        light_volts = convert2volts(light_level, 2)
```

```
        temp_level = readChannel(temp_channel)
        temp_volts = convert2volts(temp_level, 2)
        temp = convert2temp(temp_level, 2)

        print("--------------------------------------")
        print("Light: %d (%f V)" %(light_level, light_volts))
        print("Temp: %d (%f V) %f deg C" %(temp_level, temp_volts, temp))

        time.sleep(1)

except KeyboardInterrupt:
    print("Finished")
    spi.close()
```

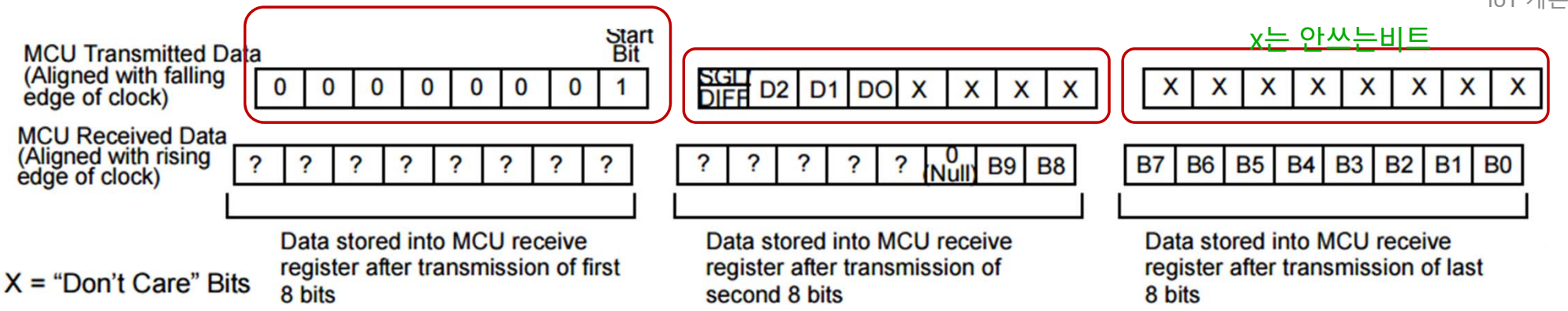
- readChannel 함수
  - 선택된 아날로그 채널에서 아날로그 신호를 읽어서 디지털 데이터로 변환된 결과를 반환
  - xfer2 함수
    - 라즈베리 파이에서 SPI 디바이스인 MCP3008로 통신을 시작하고 아날로그 입력 채널을 설정하는 제어 데이터를 보내고 그 결과 디지털 변환된 데이터를 받는 역할
    - 입력과 출력은 3개의 8비트 데이터

몇번핀에서 나오는 데이터를 보내라 그런 명령을 설정함

빨간색 테투리: 3개의 8비트 데이터의 구조

3개의 8비트 데이터가 전송될 것이다.

x는 안쓰는비트

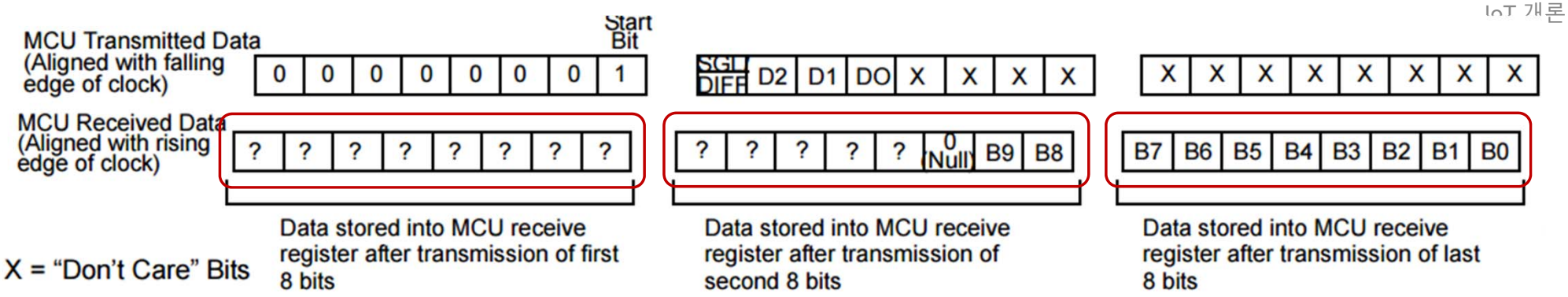


adc = spi.xfer2([1, (8 + channel) << 4, 0])

- 1
  - 첫 8비트 세그먼트의 Start Bit
- (8 + channel) << 4 이렇게하면 1000 0000이 된다. 나는 0번 채널을 쓸려고하니까
  - 둘째 8비트 세그먼트 (channel이 0이면 → 1000 0000)
    - SGL/DIFF 비트만 1이고, D2, D1, D0 비트는 모두 0

    → 하나의 채널만 사용하는 싱글 입력이고 그 채널은 0번
- 0
  - 세 번째 8비트 세그먼트
  - don't care 비트로 프로그램에서는 0으로 지정

Control Bit Selections				Input Configuration	Channel Selection
Single /Diff	D2	D1	D0		
1	0	0	0	single-ended	CH0
1	0	0	1	single-ended	CH1
1	0	1	0	single-ended	CH2
1	0	1	1	single-ended	CH3
1	1	0	0	single-ended	CH4
1	1	0	1	single-ended	CH5
1	1	1	0	single-ended	CH6
1	1	1	1	single-ended	CH7



- 변수 adc (앞의 spi.xfer2([1, (8 + channel) << 4, 0])의 반환 값)
  - 배열
  - adc[0]: 첫째 8비트 세그먼트
  - adc[1]: 둘째 8비트 세그먼트
  - adc[2]: 셋째 8비트 세그먼트
  - 유효 데이터
    - 두 번째 세그먼트에서는 하위 2비트
    - 세 번째 세그먼트 8비트
    - 이를 10비트 데이터로 변환하기 위하여 **((adc[1] & 3) << 8) + adc[2]** 연산 수행

앞에 물음표는 쓸모없는 데이터

(주의)조도에서 나오는 값이 몇lux 이런 값이 나오는 게아님

학습 정리

- 센서를 이용한 액추에이터 제어
  - 초음파 센서를 이용하여 거리에 따라 다른 색의 LED가 켜지도록 제어하는 프로그램을 작성해보자
    - 30cm 이상: 녹색 LED
    - 10-30cm: 노란색 LED
    - 10cm 이하: 빨간색 LED
  - 조도 센서를 이용하여 일정 밝기 이하에서 LED가 켜지고, 그 이상이면 꺼지도록 제어하는 프로그램을 작성해보자
  - 온도 센서를 이용하여 일정 온도 이상이면 서보 모터가 동작하고 이하이면 멈추는 프로그램을 작성해보자