

# IOT 시스템

## - 라즈베리파이 센서 실습 -

학과: 국제스포츠레저학부

학번: 201500483

이름: 김서연

## 1. LED & Switch

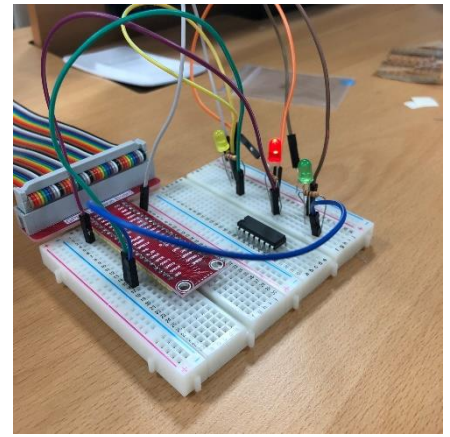
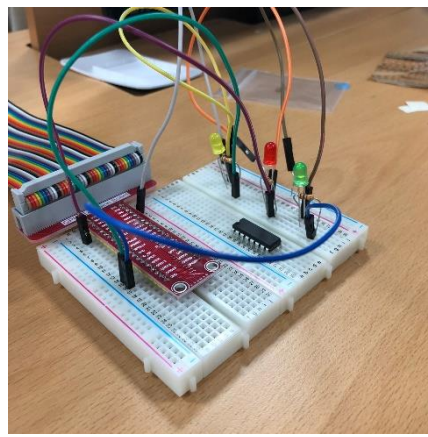
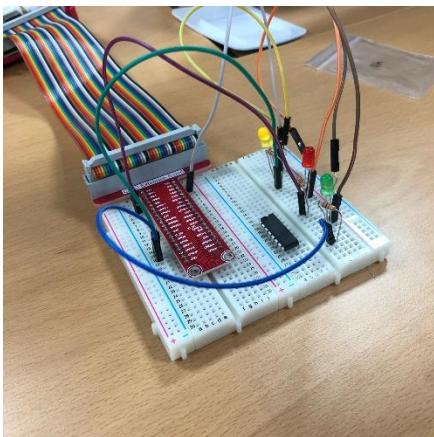
### [LED]

- 학습내용: 라즈베리파이 GPIO를 이해하기 위하여 간단히 LED를 연결하고 키고 끄는 프로그램을 실습해본다. RED, GREEN, BLUE 총 3개의 LED 핀을 사용하였으며, 프로그램 코드는 C언어이다.

- LED 회로: GPIO Output 테스트에 사용될 GPIO는 GPIO 4번, 5번, 6번이다. 이는 코드에 입력한 LED 모듈 핀 번호에 대응하는 WiringPi 핀 번호이다. LED\_RED는 7번, LED\_GREEN은 21번, LED\_BLUE는 22번이며 WiringPi는 실제 라즈베리파이의 핀 맵과는 번호가 다르기 때문에 gpio readall을 입력하여 대응하는 번호를 확인할 수 있다.

LED 2개, 저항 1K를 아래와 같이 라즈베리파이의 브레드 보드에 연결하여 GPIO의 Output 값을 1로 설정해주면 LED가 On, 0으로 설정해주면 Off가 된다.

LED는 다리가 짧은 쪽이 마이너스(-), 긴 쪽이 플러스(+)이기 때문에 점퍼케이블을 이용해 전류가 흐르는 GPIO(Vcc)와 저항을 먼저 연결한 후, 이를 LED의 긴 쪽에 연결해준다. (저항이 없으면 전류의 크기가 커서 LED가 고장나게 된다.) 그 후, LED의 짧은 쪽은 마이너스로 전류를 빼내어 GND(Ground, 접지)에 연결한다.



- ```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <unistd.h>
#include <time.h>

#define LED_RED 7
#define LED_GREEN 21
#define LED_BLUE 22

int main(void) {
    if(wiringPiSetup () == -1)
        return 1;

    pinMode(LED_RED,OUTPUT);
    pinMode(LED_GREEN, OUTPUT);
    pinMode(LED_BLUE, OUTPUT);

    digitalWrite(LED_RED,0);
    digitalWrite(LED_BLUE, 0);
    printf("3 color LED Control Start !! \n");
    for(int i= 0; i < 20; i++){
        printf("Red LED ON !! \n");
        digitalWrite(LED_RED, 1);
        usleep(1000000);
        printf("Red LED OFF!! \n GREEN LED ON !!\n");
        digitalWrite(LED_RED, 0);
        digitalWrite(LED_GREEN, 1);
        usleep(1000000);
        printf("GREEN LED OFF!! \n BLUE ON !!\n");
        digitalWrite(LED_GREEN, 0);
        digitalWrite(LED_BLUE, 1);
        usleep(1000000);
        printf("BLUE LED OFF!! \n");
        digitalWrite(LED_BLUE, 0);
    }

    return 0;
}
```

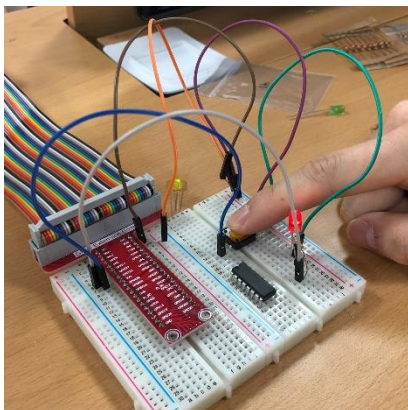
- [illegible]

## [Switch]

- 학습내용: GPIO Input 기능을 이해하기 위해서 앞서 실행했던 LED에, 추가로 스위치 회로를 추가하였다. 이 스위치의 ON/OFF 신호를 입력 신호로 받고 LED를 그에 따라 키고 끄는 실습을 진행하였다. 프로그램 코드는 C언어이다.

- 스위치 회로: GPIO Output 테스트에 사용될 GPIO는 27번(BUTTON), 17번(LED)이다. BUTTON은 WiringPi 핀 번호 2번, LED 하나는 0번에 대응된다. GPIO 27번에 연결된 점퍼 케이블을 통해 스위치의 ON/OFF 값을 받을 수 있도록 연결하고 스위치가 ON이면 Low값을 받고 이를 프로그래밍적으로 LED가 ON이 되게 한다. 그리고 스위치가 OFF이면 HIGH값을 받게 되어 LED를 OFF시키게 된다. 다시 스위치에 흐르는 전류를 +로 보내어 GND가 받을 수 있도록 점퍼 케이블을 연결한다.

LED는 전류가 흐르는 GPIO 17번(Vcc)과 저항을 먼저 연결한 후, 이를 LED의 긴 쪽에 연결해준다. 그 후, LED의 짧은 쪽은 마이너스로 전류를 빼내어 TXD(데이터를 송신)에 연결한다.



- 프로그래밍: GPIO 27번 스위치 값이 HIGH이면 LED를 ON 시키고, LOW이면 OFF 시키도록 한다. 1초 대기 후 다시 값이 반복되도록 반복문 while을 이용해 처리하도록 한다.

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <unistd.h>

#define BUTTON 2//
#define LED 0 //

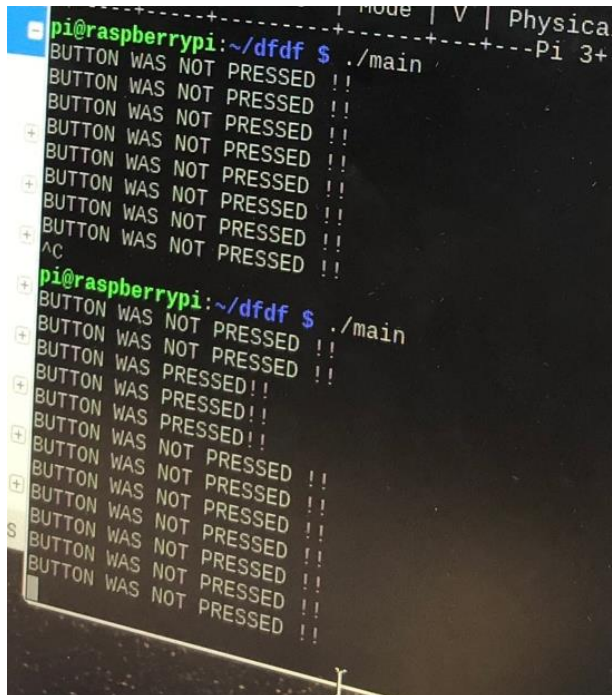
int main(void) {
    if(wiringPiSetup () == -1)
        return -1;

    pinMode(BUTTON, INPUT);
    pinMode(LED, OUTPUT);

    while (1) {
        if(digitalRead(BUTTON)==HIGH)
        {
            printf("BUTTON WAS PRESSED!! \n");
            digitalWrite(LED, HIGH);
        }
        else
        {
            printf("BUTTON WAS NOT PRESSED !! \n");
            digitalWrite(LED, LOW);
        }
        sleep(1);
    }
    return 0;
}
```



## - 프로그램 실행



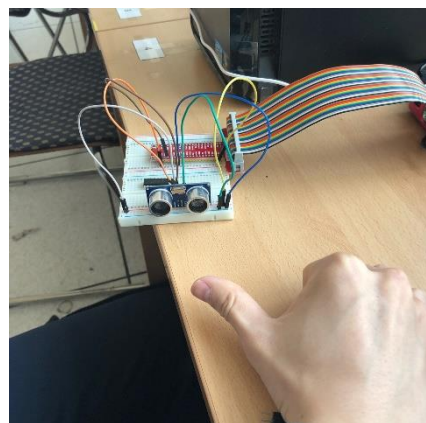
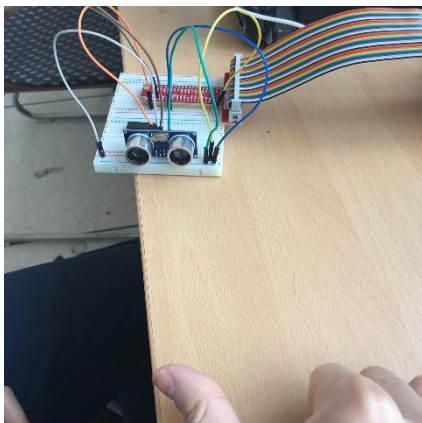
```
pi@raspberrypi:~/dfdf $ ./main
BUTTON WAS NOT PRESSED !!
BUTTON WAS NOT PRESSED !!
BUTTON WAS NOT PRESSED !!
BUTTON WAS NOT PRESSED !!
BUTTON WAS NOT PRESSED !!
BUTTON WAS NOT PRESSED !!
BUTTON WAS NOT PRESSED !!
BUTTON WAS NOT PRESSED !!
pi@raspberrypi:~/dfdf $ ./main
BUTTON WAS NOT PRESSED !!
BUTTON WAS NOT PRESSED !!
BUTTON WAS PRESSED!!
BUTTON WAS PRESSED!!
BUTTON WAS PRESSED!!
BUTTON WAS NOT PRESSED !!
BUTTON WAS NOT PRESSED !!
BUTTON WAS NOT PRESSED !!
BUTTON WAS NOT PRESSED !!
BUTTON WAS NOT PRESSED !!
BUTTON WAS NOT PRESSED !!
```

## 2. 초음파센서

### [초음파]

- 학습내용: 초음파 센서는 초음파를 내보내는 송신부, 초음파가 돌아오는 (수신부) 것을 감지하는 방식이다. 대략 2cm~4m 정도의 거리를 감지를 할 수 있다. 센서 가운데 아래 부분에 보면 Vcc, Trig, Echo, Gnd 네 부분이 있고, 이 부분을 GPIO와 연결한다.

- 연결 회로: Trig, Echo의 WiringPi 번호에 대응되도록 GPIO 20번과 Trig를, GPIO 21번과 Echo를 연결하고, GND는 마이너스로 보내 GPIO GND와 연결된 점퍼 케이블로 전류를 제어한다. Vcc는 초음파 센서의 Echo핀에서 5V출력을 받아준다.



While(digitalRead(echoPin)==LOW); // 부분을 살펴보면 while 뒤에 세미콜론이 붙어 조건부 대기 상태의 역할을 수행한다. 즉, LOW(0)가 될 때까지 계속 대기한다.

start\_time=micros(); // micros()는 프로그램이 시작된 후 흐른 시간을 microseconds(1/1000000초)  
단위로 리턴을 해주는 것이며 while문으로 대기를 하고 있다가 원하는 상태가 되면 micro()로 시  
간을 입력받는 것이다. 그리고 시간의 차이를 이용하여 거리를 도출하게 되는데

travelTime = micros()-startTime; // 을 이용하여 초음파의 travelTime을 구해 거리를 측정한다.  
 startTime을 빼는 이유는 수신한 값에서 수신부가 처음 초음파를 수신하지 못했던 값을 빼면 실  
 제 초음파가 송신부에서 나갔다가 수신부로 돌아오는 시간을 구할 수 있기 때문이다.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <wiringPi.h>
#define trigPin 28 // 20
#define echoPin 29 // 21

int main()
{
    int distance = 0;
    long startTime, travelTime;

    if(wiringPiSetup() == -1)
        return 1;

    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    printf("ultra: \n");
    while(1)
    {
        digitalWrite(trigPin, LOW);
        usleep(2);
        digitalWrite(trigPin, HIGH);
        usleep(20);

        digitalWrite(trigPin, LOW);
        while(digitalRead(echoPin) == LOW);
        startTime = micros();
        while(digitalRead(echoPin) == HIGH);
        travelTime = micros() - startTime;
        distance = travelTime*17/1000;
        printf("Distance: %d cm\n", distance);
        sleep(1);
    }
}
```

- 프로그램 실행

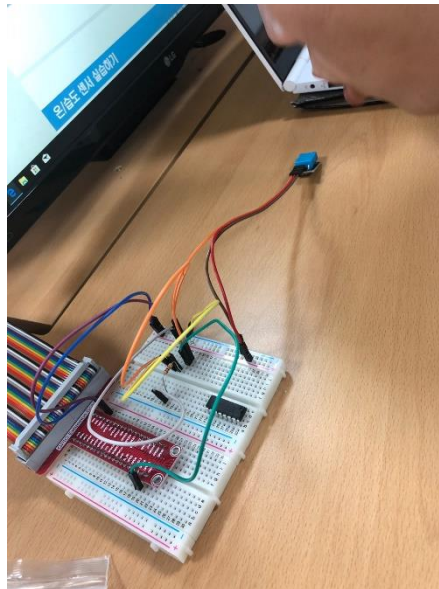
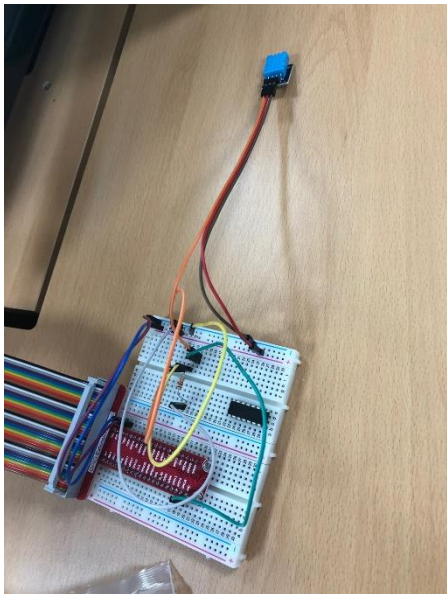
```
File Edit View Help
Distance: 7 cm
Distance: 8 cm
Distance: 4 cm
Distance: 4 cm
Distance: 4 cm
Distance: 4 cm
Distance: 4 cm
Distance: 4 cm
Distance: 4 cm
Distance: 8 cm
Distance: 8 cm
Distance: 8 cm
Distance: 8 cm
Distance: 8 cm
Distance: 8 cm
Distance: 9 cm
Distance: 28 cm
Distance: 27 cm
Distance: 27 cm
Distance: 26 cm
Distance: 26 cm
```

```
Distance: 4 cm
Distance: 4 cm
Distance: 4 cm
Distance: 4 cm
Distance: 7 cm
Distance: 7 cm
Distance: 7 cm
Distance: 8 cm
Distance: 4 cm
Distance: 4 cm
Distance: 4 cm
Distance: 4 cm
Distance: 4 cm
Distance: 4 cm
Distance: 4 cm
Distance: 8 cm
Distance: 8 cm
Distance: 8 cm
Distance: 8 cm
Distance: 8 cm
Distance: 8 cm
```

### 3. 온/습도 센서

- 학습내용: 온/습도 센서 DHT11는 내부에서 온습도 데이터를 측정 후 지정된 GPIO핀에 디지털 신호를 보낸다. 라즈베리파이는 통신을 시작함에 있어 DHT11 센서의 데이터 핀을 18msec 동안 LOW 상태로 만들어 유지하다가 아주 짧은 20~40 msec 동안 HIGH 상태를 유지한다. 이어서 DHT11 센서는 데이터 핀을 80msec 동안 LOW 그리고 80 msec 동안 HIGH 상태 신호를 만들어 라즈베리에 반응하게 된다. 이 시점에 라즈베리는 데이터를 수신할 준비가 된다. DHT11 반응 신호 전송을 할 때 26~28us 사이라면 0을 송신하게 되고 70us일 때 1을 송신하게 된다. 이때, 1의 주기 길이가 길수록 DHT11의 반응 시간이 길기 때문에 너무 작은 수를 데이터 전송하면 원하는 결과를 가질 수 없다.

- 연결 회로: DHT11은 선이 4개가 나와있으나 3개만 연결해주면 되기 때문에 1번은 신호 데이터 (Input, Output), 2번은 전압(Vcc) 5V, 3번은 접지(GND)에 연결한다. WiringPi 핀 번호에 대응되는 GPIO 26번을 데이터 신호를 받을 수 있게 연결하고, Vcc는 5V를 받을 수 있도록 점퍼 케이블을 사용해 연결해준다. 센서에서 데이터 값을 라즈베리파이로 정확히 전달하기 위해서 데이터 핀에 풀업 저항(10K옴)을 달아주어야 한다.



- 프로그래밍: DHT11은 다른 센서와 다르게 신호의 길이로 데이터를 내보낸다. 처음에 신호선으로 LOW를 18ms동안, 그리고 20~40us동안 HIGH 신호를 주면 START신호로 인식하게 된다.

`(i >= 4) && (i % 2 == 0) // 위 코드는 처음 3비트를 버린 상태로, 그리고 짝수 번째 비트인지를 확인한다. 홀수 번째는 LOW상태이기 때문에 필요가 없다. 배열 dht11_val에 조건이 부합하면 i를 8로 나눈 곳에 데이터를 쓰고 앞에서 비트가 반전될 때 까지 count를 측정하면서 26이상이면 1이라고 값을 쓴다. 26을 쓰는 이유는 26~28us정도를 데이터 0으로 지정하고 있기 때문에 26이라는 값을 기준으로 비교해주는 것이다.`



값이 다 쓰이면 0, 1, 2, 3번 값을 더한 값과 마지막 4번값을 비교하며 마지막 4번째 8비트는 패리티 비트로써 값이 잘 전송되었는지 확인하기 위해 작성해준다. 이와 비교하여 잘 나왔다면 값을 출력한다.

```
main.c % temp.c %
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stdint.h>
4 #include <wiringPi.h>
5 #include <unistd.h>
6 #define MAX_TIME 83
7 #define DHT11PIN 25
8
9
10 int dht11_val[5]={0,0,0,0,0};
11 int dht11_temp[5]={0,0,0,0,0};
12 float fahrenheit_temp;
13
14 void dht11_read_val()
15 {
16     uint8_t lststate=HIGH;
17     uint8_t counter=0;
18     uint8_t j=0, i;
19     float fahrenheit;
20
21     for(i = 0; i<5; i++)
22     {
23         dht11_val[i] = 0;
24         pinMode(DHT11PIN, OUTPUT);
25         digitalWrite(DHT11PIN, 0);
26         delay(10);
27         digitalWrite(DHT11PIN, 1);
28         delayMicroseconds(30);
29         pinMode(DHT11PIN, INPUT);
30         for(i=0; i<MAX_TIME; i++)
31         {
32             counter = 0;
33             while(digitalRead(DHT11PIN)==lststate)
34             {
35                 counter++;
36                 delayMicroseconds(1);
37                 if(counter==255)
38                     break;
39             }
40             lststate = digitalRead(DHT11PIN);
41             if(counter==255)
42                 break;
43             if(((i>=3)&&(i%2==0)))
44             {
45                 dht11_val[j/8]<=1;
46                 if(counter==255)
47                     dht11_val[j/8]=1;
48                 j++;
49             }
50         }
51     }
52 }
```

```
53
54 if (((j>=4)&&(dht11_val[4]==(dht11_val[0] + dht11_val[1] + dht11_val[2] + dht11_val[3])&&0xFF)))
55 {
56     fahrenheit = dht11_val[2]*9/5+32;
57     printf("Humidity = %d.%d%% Temperature = %d.%d *C (%.1f *F)\n", dht11_val[0],dht11_val[1],dht11_val[2],dht11_val[3], fahrenheit);
58     for(i=0; i<5; i++)
59         dht11_val[i] = dht11_val[i];
60     fahrenheit_temp = fahrenheit;
61 }
62 else
63 {
64     printf("Humidity = %d.%d%% Temperature = %d.%d *C (%.1f *F)\n", dht11_val[0],dht11_val[1],dht11_val[2],dht11_val[3], fahrenheit_temp);
65 }
66
67
68
69
70
71 int main(void)
72 {
73     //int i;
74     if(wiringPiSetup() == -1)
75         return -1;
76     while(1)
77     {
78         dht11_read_val();
79         sleep(1);
80     }
81     return 0;
82 }
83
84
85
```

## - 프로그램 실행

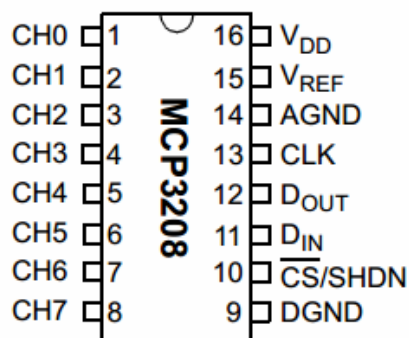
```
pi@raspberrypi:~/dfdf $ gcc -o temp temp.c -lwiringPi
pi@raspberrypi:~/dfdf $ ./temp
Humidity = 0.0 % Temperature = 0.0 *C (0.0 *F)
Humidity = 0.0 % Temperature = 0.0 *C (0.0 *F)
Humidity = 39.0 % Temperature = 23.3 *C (73.4 *F)
Humidity = 39.0 % Temperature = 23.3 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.5 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.5 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.4 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.4 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.4 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.4 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.5 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.5 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.6 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.6 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.5 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.5 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.5 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.5 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.6 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.6 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.6 *C (73.4 *F)
Humidity = 37.0 % Temperature = 23.6 *C (73.4 *F)
Humidity = 72.0 % Temperature = 23.7 *C (73.4 *F)
Humidity = 72.0 % Temperature = 23.7 *C (73.4 *F)
```



#### 4. ADC를 이용한 가변 저항 센서

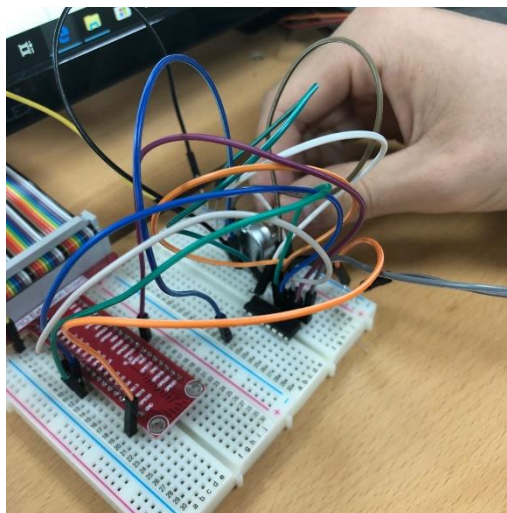
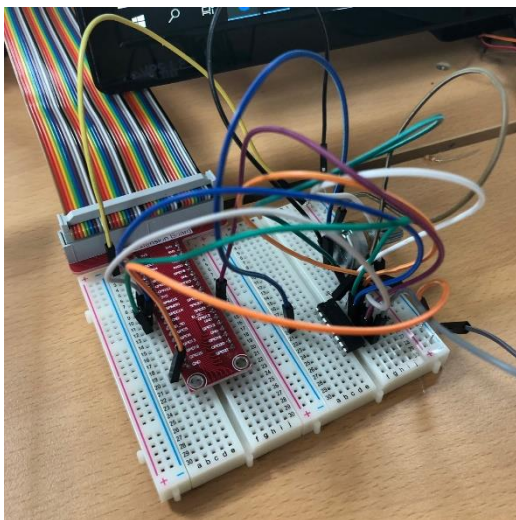
- 학습내용: ADC는 이름 그대로 아날로그 신호를 디지털 신호로 변환해주는 것을 말한다. 라즈베리파이 GPIO의 SPI를 테스트하고, SPI는 컴퓨터간의 데이터 통신을 위해 사용되는 것으로 ADC와의 상호간 통신을 위해 필요하다. MCP3208을 이용하여 센서 값을 받아오도록 하였다. MCP3208은 센서의 출력값인 아날로그 값을 12비트의 디지털 값으로 변환하는 ADC 칩이다. 라즈베리파이는 입출력을 담당하는 GPIO를 가지고 있지만 ADC 기능이 없어 센서를 활용하지 못한다는 단점이 있다. 그래서 외부 ADC 기능을 가져 SPI 통신을 이루도록 하였다.

- 연결 회로: MCP3208은 총 8개의 ADC포트(채널0~채널7)를 가지고 있는데 CH1에 가변저항을 달아 돌려가면서 ADC 값을 변경하였다.



| Name             | Function                   |
|------------------|----------------------------|
| V <sub>DD</sub>  | +2.7V to 5.5V Power Supply |
| DGND             | Digital Ground             |
| AGND             | Analog Ground              |
| CH0-CH7          | Analog Inputs              |
| CLK              | Serial Clock               |
| D <sub>IN</sub>  | Serial Data In             |
| D <sub>OUT</sub> | Serial Data Out            |
| CS/SHDN          | Chip Select/Shutdown Input |
| V <sub>REF</sub> | Reference Voltage Input    |

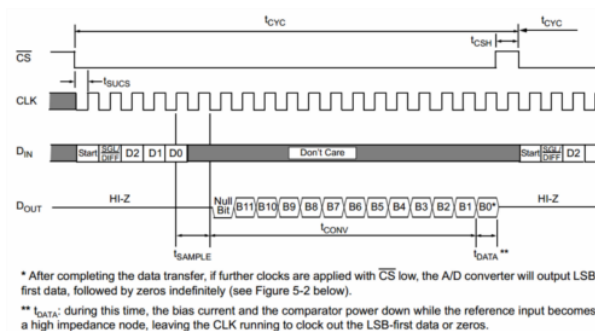
9번부터 16번까지 모든 곳에 점퍼 케이블을 연결하여 각각이 어떤 역할을 하는지 이해해 회로를 연결한다. 핀 정보 CE0, MISO, MOSI, SCLK를 MCP3208이 가진 핀 명칭에 대응되는 것을 확인하여 각각 CS/SHDN, Dout, Din, CLK에 연결해준다. 가변저항은 3개의 다리가 있는데 왼쪽부터 1번은 GND(-), 2번은 아날로그 저항(채널 1), 3번은 Vcc(+ 전류)를 가진다. 이것 또한 MCP3208에 각각이 가진 기능을 확인하여 올바른 곳에 점퍼 케이블로 연결한다.



- 프로그래밍: 라즈베리파이는 기본적으로 SPI 통신이 활성화 되어있지 않아 SPI 기능을 수행할 수 있도록 raspi-config를 이용해 통신 연결을 해준다. SPI\_CH, ADC\_CH1, ADC\_CS, SPI\_SPEED를 아래와 같이 설정하고 프로그램을 실행한다.

SPI통신은 마스터와 슬레이브로 나뉘는데, 라즈베리파이가 마스터이고 MCP3208이 슬레이브이다. 즉, 마스터인 라즈베리파이가 정해진 명령어를 슬레이브인 MCP3208에게 정해진 명령을 알려주면 그에 해당되는 값을 전송해주는 시스템이다. 센서값을 ADC하여 라즈베리파이가 받을 필요가 있기 때문에 아래와 같은 규칙으로 데이터를 전송해야 한다.

라즈베리파이는 총 3개의 바이트를 보내며 그와 동시에 MCP3208로부터 데이터를 받게 된다. 그 데이터 중에서 밑으로부터 1,2비트가 ADC결과값이다. 이를 프로그램 작성 코드와 같이 비트 연산인 하나의 데이터 값으로 변환해준다.



| 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                    |
| 0                      | 0  | 0  | 0  | differential        | CH0 = IN+<br>CH1 = IN- |
| 0                      | 0  | 0  | 1  | differential        | CH0 = IN-<br>CH1 = IN+ |
| 0                      | 0  | 1  | 0  | differential        | CH2 = IN+<br>CH3 = IN- |
| 0                      | 0  | 1  | 1  | differential        | CH2 = IN-<br>CH3 = IN+ |
| 0                      | 1  | 0  | 0  | differential        | CH4 = IN+<br>CH5 = IN- |
| 0                      | 1  | 0  | 1  | differential        | CH4 = IN-<br>CH5 = IN+ |
| 0                      | 1  | 1  | 0  | differential        | CH6 = IN+<br>CH7 = IN- |
| 0                      | 1  | 1  | 1  | differential        | CH6 = IN-<br>CH7 = IN+ |

```

#include <wiringPi.h>
#include <wiringPiSPI.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <unistd.h>
#include <time.h>
#define SPI_CH 0
#define ADC_CH1 1
#define ADC_CS 10
#define SPI_SPEED 50000

int main(void){
    int adcValue=0;
    unsigned char buf[3];
    char adChannel = ADC_CH1;

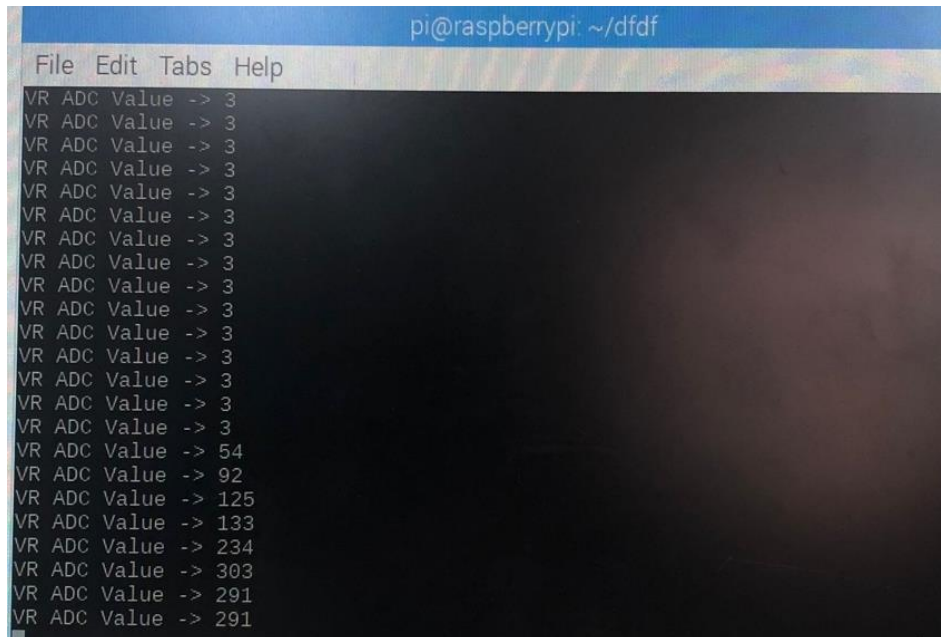
    if(wiringPiSetup() == -1)
        return 1;

    pinMode(ADC_CS, OUTPUT);
    if(wiringPiSPISetup(SPI_CH, SPI_SPEED) == -1){
        printf("wiringPi SPI Setup failed\n");
        exit(0);
    }

    while(1) {
        buf[0] = 0x06 | ((adChannel & 0x07)>>2);
        buf[1] = ((adChannel & 0x07)<<6);
        buf[2] = 0x00;
        digitalWrite(ADC_CS, 0);
        wiringPiSPIDataRW(SPI_CH, buf, 3);
        buf[1] = 0x0F & buf[1];
        adcValue = (buf[1] << 8) | buf[2];
        digitalWrite(ADC_CS, 1);
        printf("VR ADC Value -> %d\n", adcValue);
        usleep(100000);
    }
    return 0;
}

```

- 프로그램 실행



```
pi@raspberrypi: ~/dfdf
File Edit Tabs Help
VR ADC Value -> 3
VR ADC Value -> 3
VR ADC Value -> 3
VR ADC Value -> 3
VR ADC Value -> 3
VR ADC Value -> 3
VR ADC Value -> 3
VR ADC Value -> 3
VR ADC Value -> 3
VR ADC Value -> 3
VR ADC Value -> 3
VR ADC Value -> 3
VR ADC Value -> 3
VR ADC Value -> 3
VR ADC Value -> 3
VR ADC Value -> 54
VR ADC Value -> 92
VR ADC Value -> 125
VR ADC Value -> 133
VR ADC Value -> 234
VR ADC Value -> 303
VR ADC Value -> 291
VR ADC Value -> 291
```