# 라즈베리파이를 이용한 스마트센서 제어 실습3

한백전자 기술연구소

HANBACK ELECTRONICS CO.,LTD

학습5

# 라즈베리파이를 이용한 스마트센서 제어 실습3

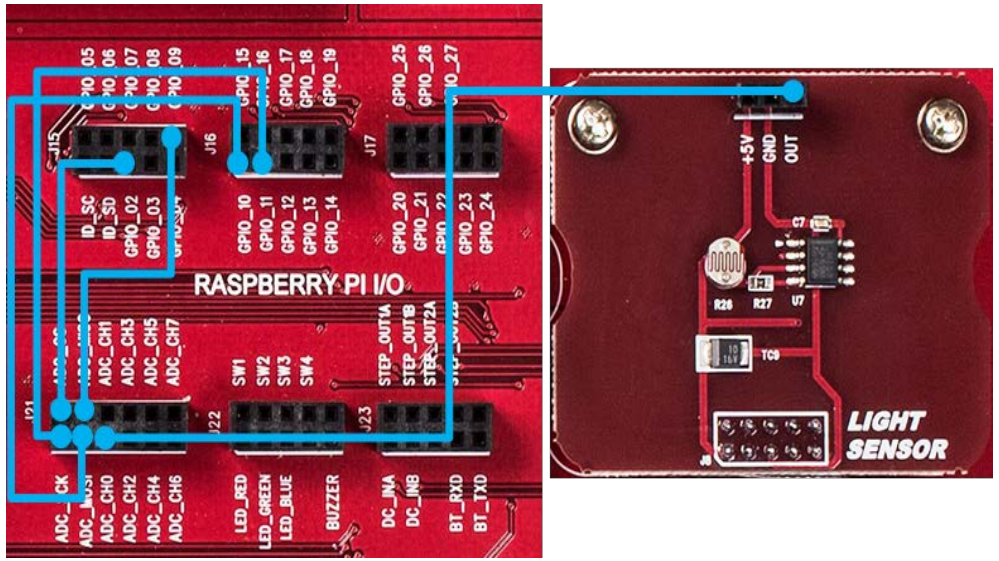
- Light 센서 실습하기
- 초음파 센서 실습하기
- 온/습도 센서 실습하기
- 가변저항 실습하기

## 1. Light 센서 핀 연결

- 라즈베리파이 에는 ADC 기능을 지원하는 핀이 없다. 따라서 외부 ADC 칩을 통해 Light 센서의 센서 값을 읽어 온다. IoT-Smart-Pioneer 에서 사용하는 ADC 칩의 명칭은 MCP3208 이다.

| 라즈베리파이 핀 번호 | Wiring Pi 핀 번호 | 핀 정보 | MCP3208 핀 명칭 |
|---|---|---|---|
| 2 | 8 | GPIO | ADC_CS |
| 9 | 13 | MISO | ADC_MISO |
| 10 | 12 | MOSI | ADC_MOSI |
| 11 | 14 | SCLK | ADC_SCK |

| MCP3208 핀 명칭 | Light 센서 모듈 핀 번호 |
|---|---|
| ADC_CH0 | OUT |



# 1. Light 센서 프로그램 작성

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <unistd.h>
#include <time.h>
#define SPI_CH 0
#define ADC_CH0 0
#define ADC_CS 8
#define SPI_SPEED 500000
int main(void){
int adcValue=0, i;
unsigned char buf[3];
char adChannel = ADC_CH0 ;
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("CDS ADC Value -> %d₩n",adcValue);
usleep(100000);
}
return 0;
}
```

## 1. 프로그램 컴파일 (in Raspberry Pi)

```
pi@raspberrypi:~/light $ make
cc -O2   -c -o main.o main.c
cc -o main main.o -lwiringPi
pi@raspberrypi:~/light $ ls
main  main.c  main.o  Makefile
pi@raspberrypi:~/light $
```

## 2. 프로그램 실행 (in Raspberry Pi)

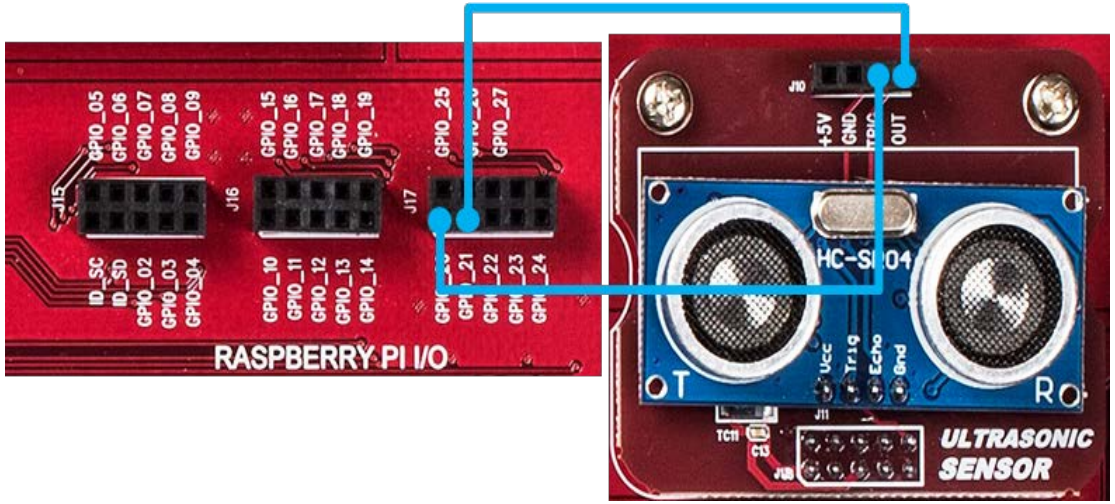
- 프로그램 컴파일이 완료 되면 "./main" 명령을 통해 프로그램을 실행한다.
  프로그램이 실행되면 현재 조도 센서에 상태를 화면에 출력한다.

```
pi@raspberrypi:~/light $ sudo ./main
Light ADC Value -> 0
Light ADC Value -> 0
Light ADC Value -> 0
Light ADC Value -> 0
Light ADC Value -> 0
Light ADC Value -> 29
Light ADC Value -> 38
Light ADC Value -> 53
Light ADC Value -> 58
Light ADC Value -> 1021
Light ADC Value -> 309
Light ADC Value -> 62
Light ADC Value -> 42
Light ADC Value -> 53
Light ADC Value -> 70
Light ADC Value -> 80
Light ADC Value -> 105
Light ADC Value -> 212
Light ADC Value -> 376
Light ADC Value -> 331
Light ADC Value -> 322
```

## 1. 초음파 센서 핀 연결

- 두 모듈을 연결하기 위해서는 케이블로 그림과 같이 초음파 센서와 라즈베리파이를 케이블로 연결한다.

| 라즈베리파이 핀 번호 | Wiring Pi 핀 번호 | 핀 정보 | ULTRA 핀 번호 |
|---|---|---|---|
| 20 | 28 | GPIO | ULTRA_TRIG |
| 21 | 29 | GPIO | ULTRA_OUT |



# 1. 초음파 센서 프로그램 작성

main.c

```
#include<stdio.h>
#include<wiringPi.h>
#define trigPin 28
#define echoPin 29
int main(void)
{
int distance=0,i;
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", distan
ce);
sleep(1);
}
}
```

# 결과 확인

## 1. 프로그램 컴파일 (in Raspberry Pi)

```
pi@raspberrypi:~/ultrasonic $ make
cc -O2   -c -o main.o main.c
cc -o main main.o -lwiringPi
pi@raspberrypi:~/ultrasonic $ ls
main  main.c  main.o  Makefile
```

## 2. 프로그램 실행 (in Raspberry Pi)

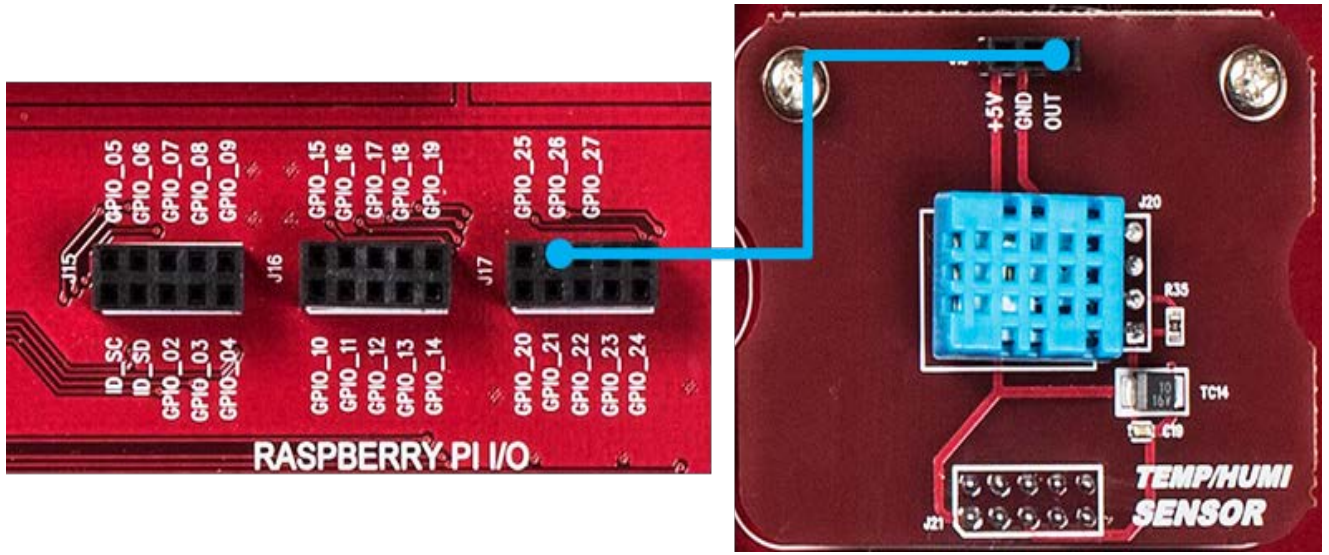
- 프로그램 컴파일이 완료 되면 "./main" 명령을 통해 프로그램을 실행한다.
  프로그램이 실행되면 현재 초음파 센서에 상태를 화면에 출력한다.

```
pi@raspberrypi:~/ultrasonic $ sudo ./main
Distance: 190 cm
Distance: 190 cm
Distance: 221 cm
Distance: 16 cm
Distance: 15 cm
Distance: 24 cm
```

## 1. 온/습도 센서 핀 연결

- 두 모듈을 연결하기 위해서는 케이블로 그림과 같이 온/습도 센서 와 라즈베리파이를 케이블로 연결한다.

| 라즈베리파이 핀 번호 | Wiring Pi 핀 번호 | 핀 정보 | 온/습도 모듈 핀 번호 |
|---|---|---|---|
| 26 | 25 | GPIO | DHT11_D |



# 1. 온/습도 센서 프로그램 작성

main.c

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#define MAX_TIME 100
#define DHT11PIN 25
int dht11_val[5]={0,0,0,0,0};
int dht11_temp[5]={0,0,0,0,0};
float farenheit_temp;
void dht11_read_val() {
uint8_t lststate=HIGH;
uint8_t counter=0;
uint8_t j=0,i;
float farenheit;
for(i=0;i<5;i++)
dht11_val[i]=0;
pinMode(DHT11PIN,OUTPUT);
digitalWrite(DHT11PIN,0);
delay(18);
digitalWrite(DHT11PIN,1);
delayMicroseconds(40);
pinMode(DHT11PIN,INPUT);
for(i=0;i<MAX_TIME;i++) {
counter=0;
while(digitalRead(DHT11PIN)==lststate){
counter++;
delayMicroseconds(1);
if(counter==255)
break;
}
lststate=digitalRead(DHT11PIN);
```

# 온/습도 센서 프로그램 작성

```
if(counter==255)
break;
if((i>=4)&&(i%2==0)){
dht11_val[j/8]<<=1;
if(counter>16)
dht11_val[j/8]|=1;
j++;
}
}
if((j>=40)&&(dht11_val[4]==((dht11_val[0]+dht11_val[1]+dht11_val[2]
+dht11_val[3])& 0xFF))){
farenheit=dht11_val[2]*9./5.+32;
printf("Humidity = %d.%d %% Temperature = %d.%d *C (%.1f *F)
₩n",dht11_val[0],dht11_val[1],dht11_val[2],dht11_val[3],farenheit);
for(i=0; i<5; i++)
```

# 온/습도 센서 프로그램 작성



```
dht11_temp[i] = dht11_val[i];
farenheit_temp = farenheit;
}else {
printf("Humidity = %d.%d %% Temperature = %d.%d *C (%.1f *F)₩n“
,dht11_temp[0],dht11_temp[1],dht11_temp[2],dht11_temp[3],farenheit_temp);
}
}
int main(void) {
int i;
if(wiringPiSetup()==-1)
 return -1;
while(1){
dht11_read_val();
sleep(1);
}
return 0;
}
```

# 결과 확인

## 1. 프로그램 컴파일 (in Raspberry Pi)

```
pi@raspberrypi:~/dht11 $ make
cc -O2   -c -o main.o main.c
cc -o main main.o -lwiringPi
pi@raspberrypi:~/dht11 $ ls
main  main.c  main.o  Makefile
```

## 2. 프로그램 실행 (in Raspberry Pi)

- 프로그램 컴파일이 완료 되면 "./main" 명령을 통해 프로그램을 실행하면 주변 온도와 습도가 화면에 출력되는 것을 확인 할 수 있다.

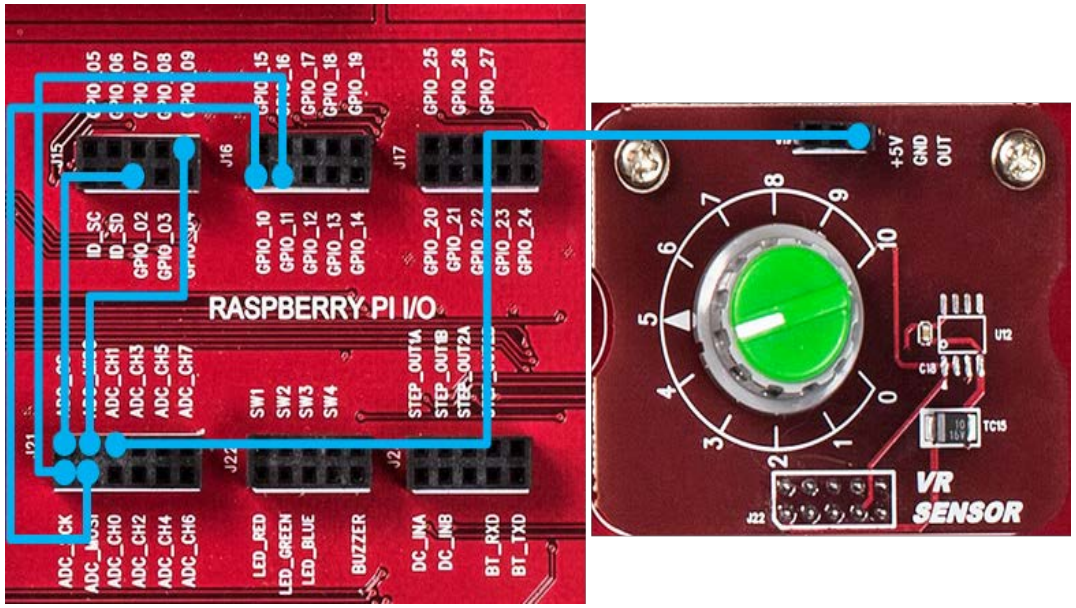
```
pi@raspberrypi:~/dht11 $ sudo ./main
Humidity = 31.0 % Temperature = 28.0 *C (82.4 *F)
Humidity = 31.0 % Temperature = 28.0 *C (82.4 *F)
Humidity = 31.0 % Temperature = 28.0 *C (82.4 *F)
Humidity = 31.0 % Temperature = 28.0 *C (82.4 *F)
```

## 1. 가변저항 핀 연결

- 라즈베리파이 에는 ADC 기능을 지원하는 핀이 없다. 따라서 외부 ADC 칩을 통해 가변저항의 값을 읽어 온다.

| 라즈베리파이 핀 번호 | Wiring Pi 핀 번호 | 핀 정보 | MCP3208 핀 명칭 |
|---|---|---|---|
| 2 | 8 | GPIO | ADC_CS |
| 9 | 13 | MISO | ADC_MISO |
| 10 | 12 | MOSI | ADC_MOSI |
| 11 | 14 | SCLK | ADC_SCK |

| MCP3208 핀 명칭 | 가변저항 모듈 핀 번호 |
|---|---|
| ADC_CH1 | OUT |



# 1. 가변저항 프로그램 작성

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <unistd.h>
#include <time.h>
#define SPI_CH 0
#define ADC_CH1 1
#define ADC_CS 8
#define SPI_SPEED 500000
int main(void){
int adcValue=0, i;
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;
}
```

# 결과 확인

## 1. 프로그램 컴파일 (in Raspberry Pi)

```
pi@raspberrypi:~/vr $ make
cc -O2   -c -o main.o main.c
cc -o main main.o -lwiringPi
pi@raspberrypi:~/vr $ ls
main  main.c  main.o  Makefile
pi@raspberrypi:~/vr $
```

## 2. 프로그램 실행 (in Raspberry Pi)

- 프로그램 컴파일이 완료 되면 "./main" 명령을 통해 프로그램을 실행한다.

```
pi@raspberrypi:~/vr $ sudo ./main
VR ADC Value -> 0
VR ADC Value -> 1
VR ADC Value -> 41
VR ADC Value -> 128
VR ADC Value -> 186
VR ADC Value -> 253
VR ADC Value -> 372
VR ADC Value -> 499
VR ADC Value -> 584
VR ADC Value -> 658
VR ADC Value -> 799
VR ADC Value -> 969
VR ADC Value -> 1199
VR ADC Value -> 1368
```

## 1. 교수 방법

- 라즈베리파이에 대한 관련 용어, 구성, 방식 이해 정도를 파악한 후 수업을 진행한다.
- 교수자의 주도로 사물인터넷 등의 내용을 PPT 자료로 제시한 후 설명한다.
- 가능하면 사전에 개인별 학습 자료를 준비하여 모든 학생이 참여할 수 있는 문제 해결식 수업, 협력 수업이 가능하도록 한다.
- 라즈베리파이의 구성과 특징 전 과정을 작업의 순서에 따라 단계적으로 실습이 이루어질 수 있도록 지도한다.
- 교재를 통해 수업을 준비하는데 있어서 필요한 모든 정보나 팁, 힌트 등 도움이 될 만한 내용들을 확인할 수 있다.
- 학생들이 만들게 될 각 모델들은 수업시간에 중점적으로 다뤄질 부분과 어휘, 질문, 답을 포함하고 있으며 또한 탐구할만한 추가적인 아이디어들도 포함하고 있다.

## 1. 학습 방법

- 라즈베리파이에 대한 관련 용어를 숙지하고, 사물인터넷의 기초 단계에 대해 이해한다.
- 라즈베리파이의 각 단계별 중점 사항을 체계적으로 학습한다.
- 라즈베리파이의 수행순서의 전 과정을 실습한다.
- 라즈베리파이의 내용을 이해한다.
- 라즈베리파이의 구성을 이해하고 숙달이 될 때까지 학습한다.