

IOT 시스템

- IoT Thing+ 플랫폼 연동 -

학과: 국제스포츠레저학부

학번: 201500483

이름: 김서연

1. Gateway 연동

- 학습내용: 스마트 센서와 클라우드 연동을 진행하기 위해 본 실습에서는 Thing+를 통한 클라우드 연동을 진행하였다. Thing+를 사용한 이유는 다음과 같다. Thing+는 AWS 기반의 IoT 서비스 일종이며, Thing+와 연동된 라즈베리파이 제품을 사용하여 IoT 서비스를 쉽게 구축할 수 있으며 다양한 성능의 IoT 디바이스와 연동이 가능한 Thing+ Embedded 서비스를 이용할 수 있다.

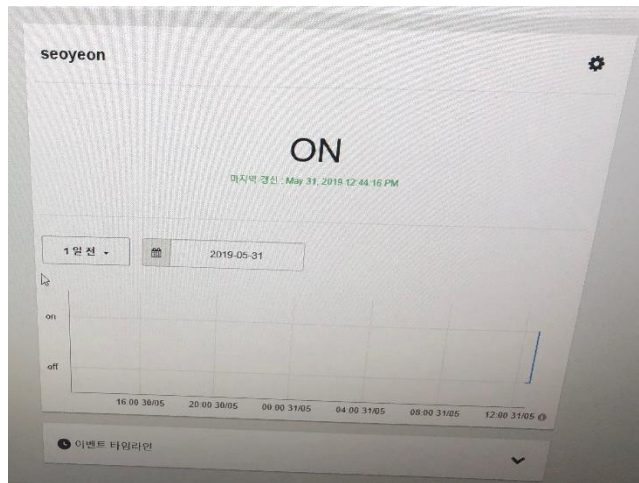
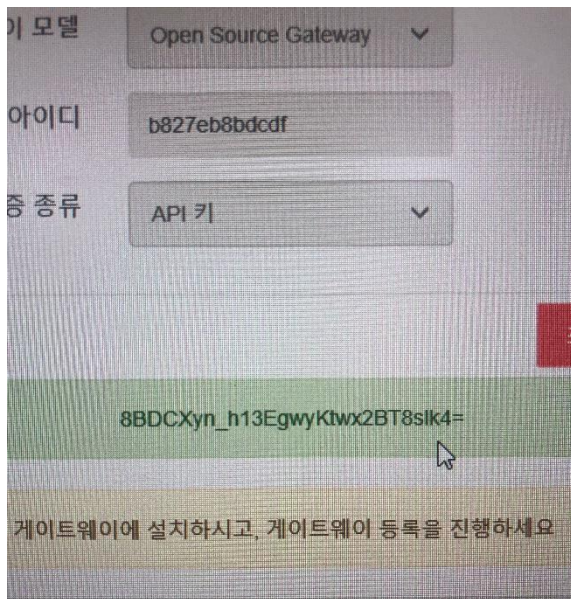
사용자는 모바일 또는 웹을 통해 플랫폼에 접근하고 플랫폼 내부에 존재하는 Application Level 프로토콜 상에서 디바이스 관리 및 센서 데이터를 수집한다. 이때 우리는 네트워크를 통해 Thing+라는 클라우드 서비스를 사용하게 된다. Thing+를 이용하기 위해서는 Gateway 연동이 필수적이다.

라즈베리파이를 부팅시키고 시스템 시간을 네트워크를 통해 업데이트한다. 시스템 시간을 변경하지 않을 경우 데이터 전송 시 문제가 발생할 수 있기 때문에 Thing+사용 환경을 구성하기 전 이 단계는 필수적이다. Thing+ 클라우드 서비스에서 각자의 라즈베리파이를 구분하기 위해서는 호스트명 변경이 필요하다. 'seoyeon'으로 호스트명을 변경하였으며 hosts 파일 또한 동일한 이름으로 변경한다.

Thing+ Gateway는 네트워크만 연결되어 있다면 파일에 권한을 부여하여 패키지를 설치하고 프로그램을 실행할 수 있다. Thing+ Gateway를 등록하기 위해서는 ID 및 호스트명이 필요한데, ID는 라즈베리파이의 Mac address를 통해 알 수 있다. 이를 통해 Gateway의 API KEY를 발급받았다면 라즈베리파이와 연결하여 Thing+ 연동을 통해 실습을 진행할 수 있다.

jprc library를 사용한 device agent 구성 소스코드를 활용하여 센서 값을 읽고 액추에이터를 동작시키도록 한다. 여기서 device란 Gateway안에 연결된 센서의 집합을 의미한다. 예를 들어, Gateway 내부에 '블루투스'로 연결되는 LED, 인체감지센서, 초음파, ADC 가변저항 센서가 있다면 블루투스 기기가 하나의 디바이스로 정의할 수 있다. 이때 device는 물리적인 하드웨어 일 수도 있고 가상의 센서 집합이 될 수도 있다.

- 프로그램 실행



```
pi@seoyeon: /opt/thingplus/gateway/scripts
File Edit Tabs Help
pi@seoyeon:~$ cd /opt/thingplus/gateway/scripts
pi@seoyeon:/opt/thingplus/gateway/scripts$ ./getGatewayID.sh
Your Gateway ID is as below
b827eb8bdcdf
pi@seoyeon:/opt/thingplus/gateway/scripts$
```

2. Cloud 연동하여 LED 제어하기

- 학습내용: 액추에이터 LED를 Thing+ 클라우드 연동을 통해 끄고, 켜는 작업을 수행하였다. Gateway가 주기적으로 센서 상태와 상태의 유효시간을 전송하여 유효시간 내에 센서 상태를 수신 받도록 한다. 이때 Thing+는 액추에이터 실행, 하드웨어 환경 설정 등의 작업을 요청하고, LED는 주어진 포맷에 따라 응답을 하여 끄고 켜는 작동을 이행할 수 있었다.

LED를 제어하는 프로그램을 작성할 때, JSON형태로 센서의 ID, Type, Name을 보내게 된다. 이것은 센서 등록 시 device model의 sensors 배열에서 사용할 수 있는 센서 모델 목록을 의미한다. 따라서 얻어 온 Gateway 정보에서 discover가 가능한지 판별해야 하므로 코드를 다음과 같이 수정하여 Thing+를 통한 LED 제어를 실행하였다.

- 프로그램 코드: Thing+ Gateway 프로그램과 연결을 수행하기 위해 jrpc_init() 함수를 사용하여 jrpc 초기화 및 연결 설정하는 작업을 수행하고 액추에이터 실행 명령어를 잘 전달받았는지 상태를 확인하는 act 데이터 수신하는 함수를 작성하도록 한다. 또한, jrpc의 server.h 프로그램에서 Thing+ 사용자가 액추에이터 동작 명령을 실행하게 되었다는 것을 확인 가능하게 해주는 act_flag() 액추에이터 제어 상태 함수를 작성하고 이 동작에 대한 데이터를 가지고 있는 act_recv() 함수를 작성해야 한다.

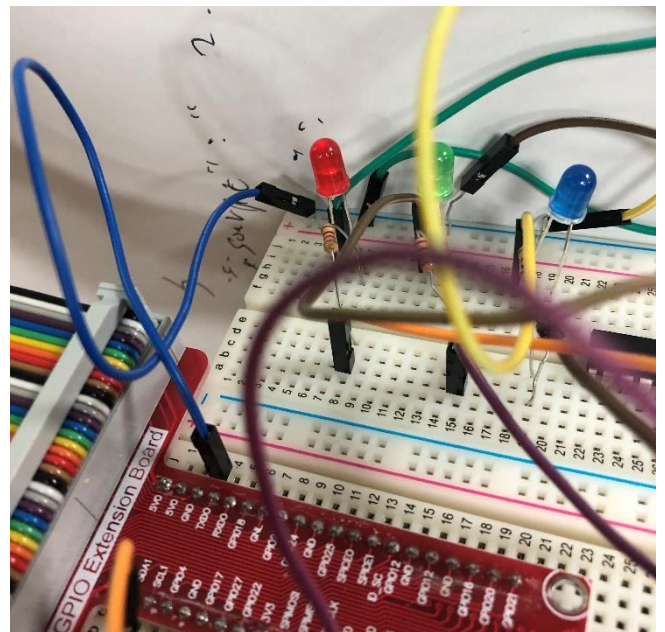
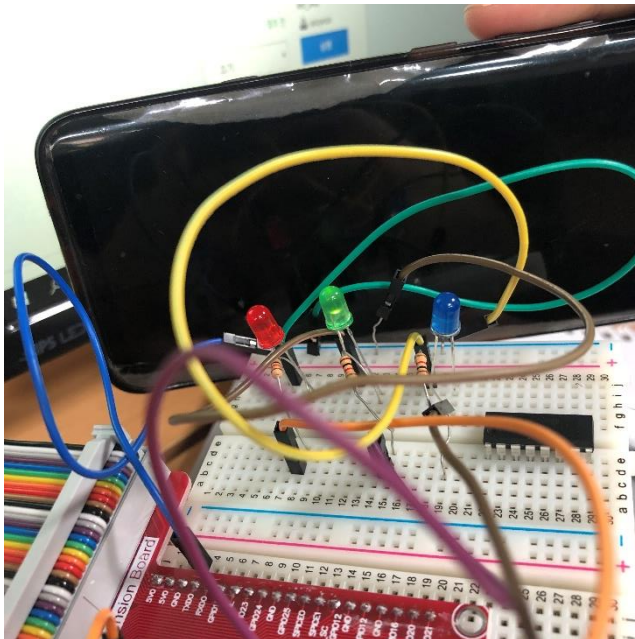
```
1 #include "b.h"
2 #include "main.h"
3 #include "wiringPi.h"
4 #define LED_RED 7
5 #define LED_GREEN 21
6 #define LED_BLUE 22
7
8 int main(void){
9     int actFlag, led_val;
10    if(wiringPiSetup()==-1)
11        return 1;
12
13    pinMode(LED_RED, OUTPUT);
14    pinMode(LED_GREEN, OUTPUT);
15    pinMode(LED_BLUE, OUTPUT);
16    jrpc_init();
17
18    while(1){
19        actFlag = getActFlag();
20        if(actFlag == 1){
21            led_val = act_recv(LED_GREEN);
22
23            switch(led_val){
24                case RED_ON:
25                    digitalWrite(LED_RED, 1);
26                    break;
27                case RED_OFF:
28                    digitalWrite(LED_RED, 0);
29                    break;
30                case GREEN_ON:
31                    digitalWrite(LED_GREEN, 1);
32                    break;
33                case GREEN_OFF:
34                    digitalWrite(LED_GREEN, 0);
35                    break;
36                case BLUE_ON:
37                    digitalWrite(LED_BLUE, 1);
38                    break;
39                case BLUE_OFF:
40                    digitalWrite(LED_BLUE, 0);
41                    break;
42            }
43        }
44    }
45}
```

```
58 // ATthing + Potat Sensor Set
59
60 cJSON *discover(jrpc_context_t *ctx, cJSON *params, cJSON *id) {
61     cJSON *devices = NULL, *device = NULL, *sensor = NULL, *all=NULL, *sensors = NULL;
62     all = cJSON_CreateObject();
63     devices = cJSON_CreateArray();
64     device = cJSON_CreateObject();
65     cJSON_AddItemToArray(devices, device);
66     cJSON_AddItemToObject(all, "version", cJSON_CreateString(JRPC_VERSION));
67     cJSON_AddItemToObject(device, "deviceid", cJSON_CreateString(DEVIDE_ID));
68     cJSON_AddItemToObject(device, "devicemodel", cJSON_CreateString("gsnrcpfv1.0"));
69     sensors = cJSON_CreateArray();
70     sensor=cJSON_CreateObject();
71     cJSON_AddStringToObject(sensor, "id", LED_RED);
72     cJSON_AddStringToObject(sensor, "type", "led");
73     cJSON_AddStringToObject(sensor, "name", "led_red");
74     cJSON_AddItemToArray(sensors, sensor);
75
76     sensor=cJSON_CreateObject();
77     cJSON_AddStringToObject(sensor, "id", LED_GREEN);
78     cJSON_AddStringToObject(sensor, "type", "led");
79     cJSON_AddStringToObject(sensor, "name", "led_green");
80     cJSON_AddItemToArray(sensors, sensor);
81
82     sensor=cJSON_CreateObject();
83     cJSON_AddStringToObject(sensor, "id", LED_BLUE);
84     cJSON_AddStringToObject(sensor, "type", "led");
85     cJSON_AddStringToObject(sensor, "name", "led_blue");
86     cJSON_AddItemToArray(sensors, sensor);
87     cJSON_AddItemToObject(device, "sensors", sensors);
88     cJSON_AddItemToObject(all, "result", devices);
89 }
```

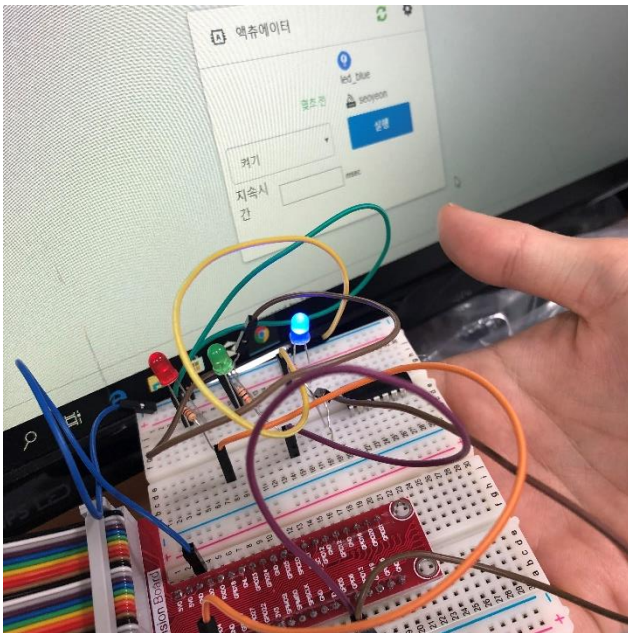
- 프로그램 실행: 프로그램을 컴파일 하기 위해 make 명령어를 통해 해주며, 파일을 수정했을 경우 make clean 작업을 이행 후 다시 컴파일 할 수 있도록 한다.

- 컴파일 실행

```
File Edit Tabs Help
{
  "jsonrpc": "2.0",
  "result": "off",
  "id": 29
}
Select receive 1 fd.
callback from fd 5
Valid JSON Received :
{
  "id": 30,
  "method": "sensor.set",
  "params": ["b827eb8bdcdf-led_green-0", "on", {
  }]
}
Method Invoked: sensor.set
JSON Response on fd 5
{
  "jsonrpc": "2.0",
  "result": "on",
  "id": 30
}
Select receive 1 fd.
callback from fd 5
Valid JSON Received :
{
  "id": 31,
  "method": "sensor.set",
  "params": ["b827eb8bdcdf-led_green-0", "off", {
  }]
}
Method Invoked: sensor.set
JSON Response on fd 5
{
  "jsonrpc": "2.0",
  "result": "off",
  "id": 31
}
Select receive 1 fd.
callback from fd 5
Valid JSON Received :
{
  "id": 32,
  "method": "sensor.set",
  "params": ["b827eb8bdcdf-led_red-0", "on", {
  }]
}
Method Invoked: sensor.set
JSON Response on fd 5
```



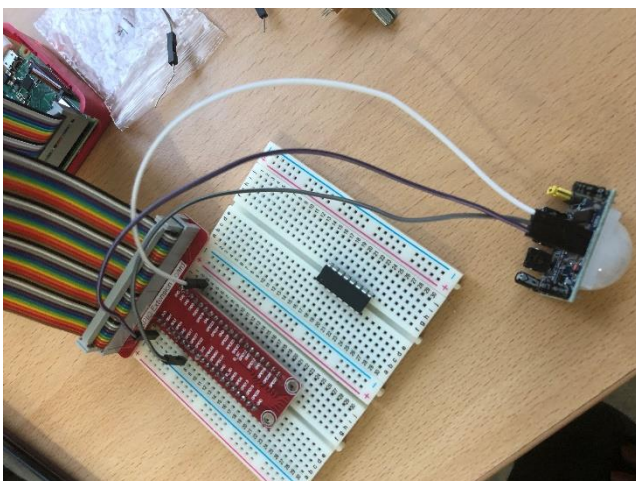
- 클라우드에서 LED 제어



3. Cloud 연동하여 인체감지센서 상태 확인하기

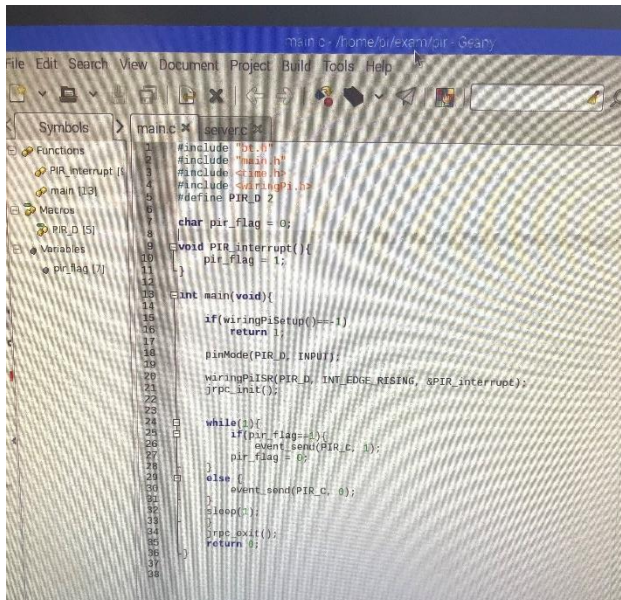
- 학습내용: 액추에이터 인체감지센서를 Thing+ 클라우드 연동을 통해 동작 제어하는 작업을 수행하였다. Gateway가 주기적으로 센서 상태와 상태의 유효시간을 전송하여 유효시간 내에 센서 상태를 수신 받도록 하는 것은 위와 동일하다. 다만 인체감지센서는 센서감도와 sleep 시간을 조절하여 인체감지시 Gateway 화면에 아이콘을 출력하도록 하였다.

인체감지센서를 제어하는 프로그램을 작성할 때, LED와 마찬가지로 JSON형태로 센서의 ID, Type, Name을 보내게 된다. discover를 하지 않는다면 Thing+ 센서를 등록할 수 없기 때문에 다음과 같이 코드를 수정해준다. 이때 등록된 센서 정보를 만들어 Gateway에 전송하게 되며 JSONRPC 프로토콜을 통해 device agent에게 센서 값 읽기, 액추에이터 동작을 요청하게 된다. 이러한 과정을 수행하여 Thing+를 통한 인체감지센서 제어를 실행하였다.

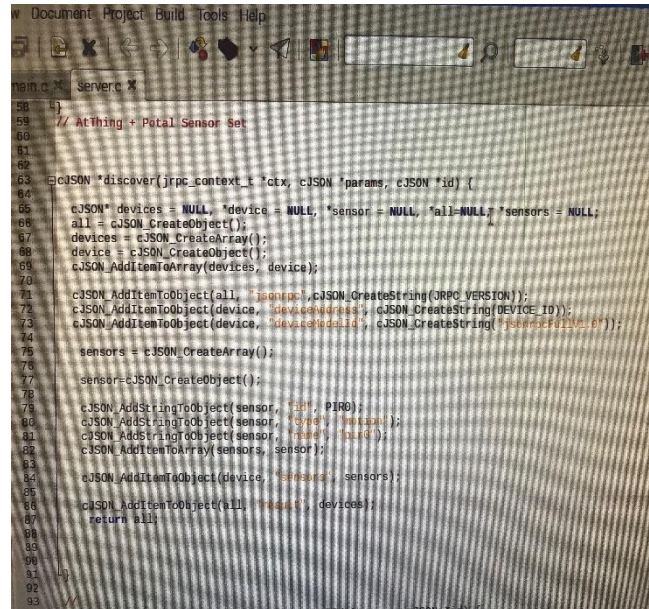


- 프로그램 코드:

Thing+ Gateway 프로그램과 연결을 수행하기 위해 `jrpc_init()` 함수를 사용하여 `jrpc` 초기화 및 연결 설정하는 작업을 수행하고 액추에이터 실행 명령어를 잘 전달받았는지 상태를 확인하는 `act` 데이터 수신하는 함수를 작성하도록 한다. 인체감지 데이터를 수신하기 위한 `pir_flag` 데이터 값을 Thing+로 전송하고, `jrpc`



```
1 #include "bt.h"
2 #include "main.h"
3 #include <stdio.h>
4 #include <unistd.h>
5 #define PIR_D 2
6
7 char pir_flag = 0;
8
9 void PIR_interrupt() {
10     pir_flag = 1;
11 }
12
13 int main(void) {
14     if(wiringPiSetup() == -1)
15         return 1;
16     pinMode(PIR_D, INPUT);
17     wiringPiISR(PIR_D, INT_EDGE_RISING, &PIR_interrupt);
18     jrpc_init();
19
20     while(1) {
21         if(pir_flag == 1) {
22             event_send(PIR_C, 1);
23             pir_flag = 0;
24         } else {
25             event_send(PIR_C, 0);
26         }
27         sleep(1);
28     }
29     jrpc_exit();
30     return 0;
31 }
```

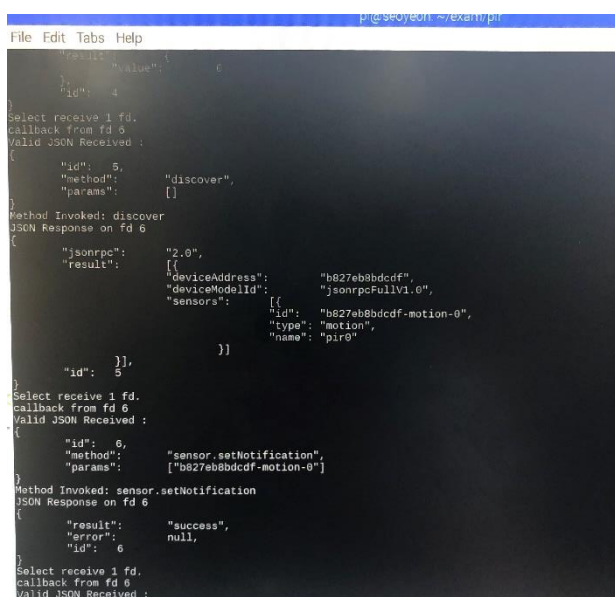


```
59 // Atthing + Potal Sensor Set
60
61 cJSON *discover(jrpc_context_t *ctx, cJSON *params, cJSON *id) {
62     cJSON *devices = NULL, *device = NULL, *sensor = NULL, *all=NULL, *sensors = NULL;
63     all = cJSON_CreateObject();
64     devices = cJSON_CreateArray();
65     device = cJSON_CreateObject();
66     cJSON_AddItemToArray(devices, device);
67
68     cJSON_AddItemToObject(all, "jsonrpc", cJSON_CreateString(JRPC_VERSION));
69     cJSON_AddItemToObject(device, "deviceAddress", cJSON_CreateString(DEVICE_ID));
70     cJSON_AddItemToObject(device, "deviceModelId", cJSON_CreateString("sensorFullV1.0"));
71
72     sensors = cJSON_CreateArray();
73     sensor = cJSON_CreateObject();
74
75     cJSON_AddStringToObject(sensor, "id", PIR0);
76     cJSON_AddStringToObject(sensor, "event", "motion");
77     cJSON_AddStringToObject(sensor, "name", "pir0");
78     cJSON_AddItemToArray(sensors, sensor);
79
80     cJSON_AddItemToObject(device, "sensors", sensors);
81     cJSON_AddItemToObject(all, "result", devices);
82     return all;
83 }
```

실행 데이터를 Thing+로 전송하기 위해 `event_send()` 함수를 사용한다.

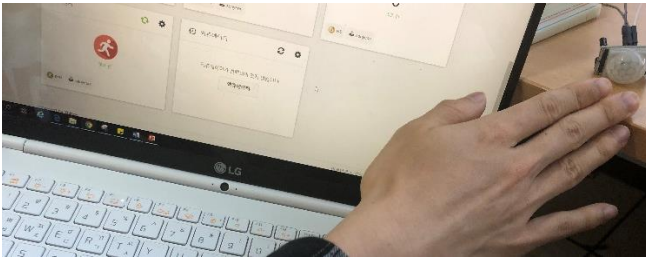
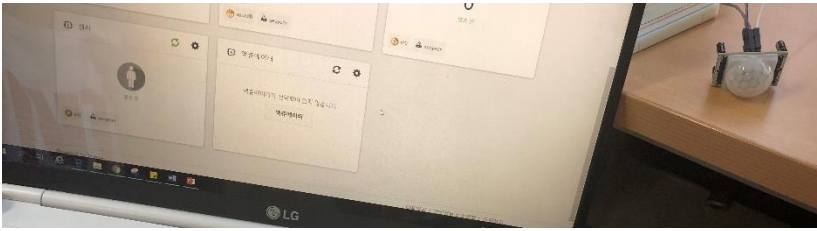
- 프로그램 실행: 프로그램을 컴파일 하기 위해 `make` 명령어를 통해 해주며, 파일을 수정했을 경우 `make clean` 작업을 이행 후 다시 컴파일 할 수 있도록 한다.

● 컴파일 실행



```
File Edit Tabs Help
p@seoyeon ~/exam/pir
$ ./client
{"id": 1, "method": "discover", "params": []}
Select receive 1 fd.
callback from fd 6
Valid JSON Received :
{
  "id": 5,
  "method": "discover",
  "params": []
}
Method Invoked: discover
JSON Response on fd 6
{
  "jsonrpc": "2.0",
  "result": {
    "deviceAddress": "b827eb8bdcdf",
    "deviceModelId": "jsonrpcFullV1.0",
    "sensors": [
      {
        "id": "b827eb8bdcdf-motion-0",
        "type": "motion",
        "name": "pir0"
      }
    ]
  },
  "id": 5
}
Select receive 1 fd.
callback from fd 6
Valid JSON Received :
{
  "id": 6,
  "method": "sensor.setNotification",
  "params": ["b827eb8bdcdf-motion-0"]
}
Method Invoked: sensor.setNotification
JSON Response on fd 6
{
  "result": "success",
  "error": null,
  "id": 6
}
Select receive 1 fd.
callback from fd 6
Valid JSON Received :
```

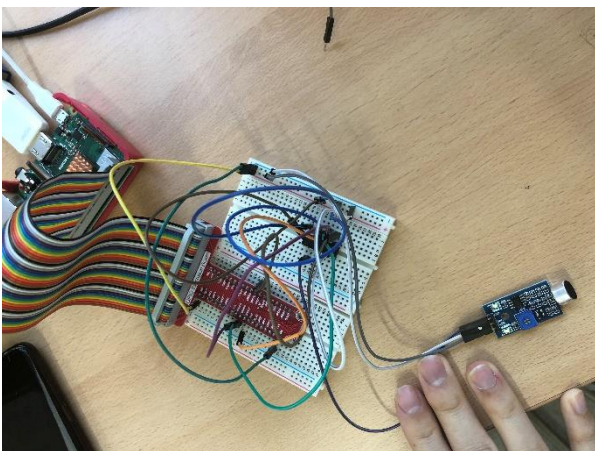
- 클라우드에서 인체감지센서 제어



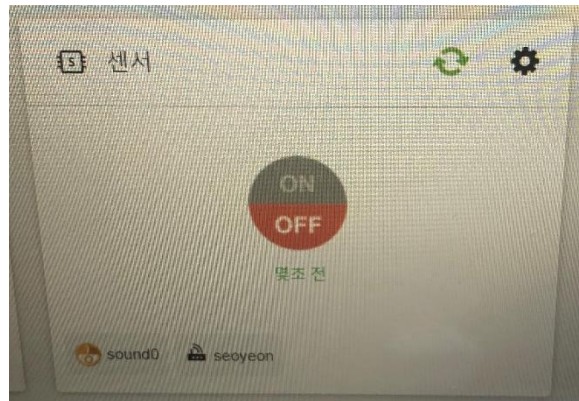
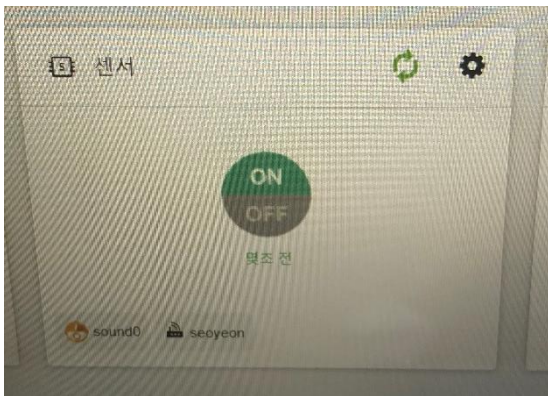
4. Cloud 연동하여 소리감지 센서 상태 확인하기

- 학습내용: 액추에이터 소리감지 센서를 Thing+ 클라우드 연동을 통해 동작 제어하는 작업을 수행하였다. Gateway가 주기적으로 센서 상태와 상태의 유효시간을 전송하여 유효시간 내에 센서 상태를 수신 받도록 하는 것은 앞서 수행했던 실습과 동일하다. 하지만 이 실습은 라즈베리파이 내에 SPI 설정을 먼저 한 후 진행이 되어야 한다.

다른 센서들과 마찬가지로 초음파 센서 또한 JSON형태로 센서의 ID, Type, Name을 보내게 된다. discover를 하지 않는다면 Thing+ 센서를 등록할 수 없기 때문에 다음과 같이 코드를 수정해준다. 따라서 discover 함수 절차를 거쳐 Thing+에 초음파 센서/액추에이터를 등록한다. 이러한 과정을 수행하여 Thing+를 통한 소리감지 센서 제어를 실행하였다.



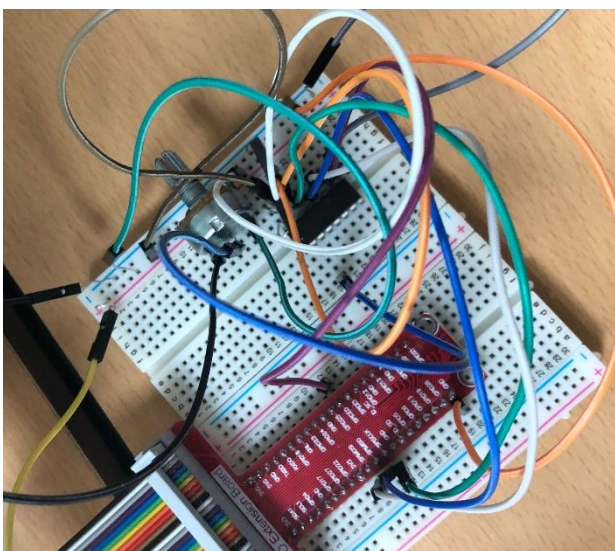
- 클라우드에서 소리감지센서 제어



5. Cloud 연동하여 ADC 가변저항 상태 확인하기

- 학습내용: 액추에이터 ADC를 이용한 가변저항 상태 확인하는 것을 Thing+ 클라우드 연동을 통해 동작 제어하는 작업을 수행하였다. ADC는 이름 그대로 아날로그 신호를 디지털 신호로 변환해주는 것을 말한다. 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.

다른 센서들과 마찬가지로 가변저항 센서 또한 JSON형태로 센서의 ID, Type, Name을 보내게 된다. discover를 하지 않는다면 Thing+ 센서를 등록할 수 없기 때문에 다음과 같이 코드를 수정해준다. 따라서 discover 함수 절차를 거쳐 Thing+에 초음파 센서/액추에이터를 등록한다. 이러한 과정을 수행하여 Thing+를 통한 ADC 가변저항 센서 제어를 실행하였다.



- 프로그램 코드: SPI 통신을 위해 변수를 선언하고 Thing+ gateway 프로그램과 연결을 수행하기 위하여 `jrpc_init()` 함수를 사용한다. 주기적으로 가변저항의 데이터를 수신하여 Thing+로 송신하기 위하여 `sensor_send()` 작업이 필요하다. 이후 SPI 통신을 통해 아날로그 데이터를 수신하여 결과 값을 얻게 된다. 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.

```

1 #include "pin.h"
2 #include "main.h"
3 #include "wiringPi.h"
4 #include "SPI.h"
5 #define SPI_CS 0
6 #define ADC_CS 8
7 #define ADC_CH1 1
8 #define SPI_SPEED 500000
9
10 int readVR() {
11     int adcValue = 0;
12     unsigned char buf[3];
13     buf[0] = 0x00 | ((ADC_CH1 & 0x07) << 2);
14     buf[1] = ((ADC_CH1 & 0x07) << 2);
15     buf[2] = 0x00;
16     digitalWrite(ADC_CS, 0);
17     wiringPiSPIDataRW(SPI_CS, buf, 3);
18     buf[1] = 0x0F & buf[1];
19     adcValue = (buf[1] << 8) | buf[2];
20     digitalWrite(ADC_CS, 1);
21     return adcValue;
22 }
23
24 int main(void) {
25     int i, vr_val;
26     int adcValue = 0;
27     if(wiringPiSetup() == -1)
28         return 1;
29     pinMode(ADC_CS, OUTPUT);
30     if(wiringPiSPISetup(SPI_CS, SPI_SPEED) == -1) {
31         printf("wiringPi SPI Setup Failed\n");
32         exit(1);
33     }
34     jrpc_init();
35     while(1) {
36         vr_val = readVR();
37         sensor_send(VR_CS, vr_val);
38         sleep(50);
39     }
40     jrpc_exit();
41     return 0;
42 }

```

```

33 void handle_kill_signal() {
34     jrpc_server_stopmy_server();
35     signal(SIGINT, SIG_DFL);
36     signal(SIGTERM, SIG_DFL);
37     signal(SIGKILL, SIG_DFL);
38 }
39 // Nothing + Potat Sensor Set
40
41 cJSON *discover(jrpc_context_t *ctx, cJSON *params, cJSON *id) {
42     cJSON *devices = NULL, *device = NULL, *sensor = NULL, *all=NULL, *sensors = NULL;
43     all = cJSON_CreateObject();
44     devices = cJSON_CreateArray();
45     device = cJSON_CreateObject();
46     cJSON_AddItemToArray(devices, device);
47     cJSON_AddItemToObject(all, "sensors", cJSON_CreateString(JRPC_VERSION));
48     cJSON_AddItemToObject(device, "deviceModel", cJSON_CreateString(DEVICE_ID));
49     cJSON_AddItemToObject(device, "deviceModelVer", cJSON_CreateString(JRPC_VERSION));
50     sensors = cJSON_CreateArray();
51     sensor = cJSON_CreateObject();
52     cJSON_AddStringToObject(sensor, "id", VR);
53     cJSON_AddStringToObject(sensor, "name", "VR");
54     cJSON_AddStringToObject(sensor, "unit", "V");
55     cJSON_AddItemToArray(sensors, sensor);
56     cJSON_AddItemToObject(device, "sensors", sensors);
57     cJSON_AddItemToObject(all, "devices", devices);
58     return all;
59 }
60
61 cJSON *sensor_set(jrpc_context_t *ctx, cJSON *params, cJSON *id) {
62     cJSON *result = cJSON_CreateObject();
63     result->id = NULL, result->id2 = NULL, result->id3 = NULL;
64     char *sensorIdStr = NULL, *sensorId2Str = NULL;
65     cJSON_AddItemToObject(result, "result", cJSON_CreateString(JRPC_VERSION));
66     return result;
67 }

```

- 프로그램 실행: 프로그램을 컴파일 하기 위해 `make` 명령어를 통해 해주며, 파일을 수정했을 경우 `make clean` 작업을 이행 후 다시 컴파일 할 수 있도록 한다.

● 컴파일 실행

```

File Edit Tabs Help
~/exam/vr

{"id": 6,
 "method": "ping",
 "params": []}

Method Invoked: ping
JSON Response on fd 6:
{"result": null,
 "id": 6}

Select receive 1 fd,
callback from fd 6
Valid JSON Received:
{"id": 7,
 "method": "sensor.get",
 "params": ["b827eb8bdcdf-VR-0"]}

Method Invoked: sensor.get
JSON Response on fd 6:
{"jsonrpc": "2.0",
 "result": {
 "value": 478
 },
 "id": 7}

VR ADC Value -> 8
Select receive 1 fd,
callback from fd 6
Valid JSON Received:
{"id": 8,
 "method": "ping",
 "params": []}

Method Invoked: ping
JSON Response on fd 6:
{"result": null,
 "id": 8}

Select receive 1 fd,
callback from fd 6
Valid JSON Received:
{"id": 9,
 "method": "sensor.get",
 "params": ["b827eb8bdcdf-VR-0"]}

Method Invoked: sensor.get

```


- 클라우드에서 초음파센서 제어

