

2020.2학기 임베디드시스템 프로그래밍

6조 팀프로젝트 결과보고서

학번 : 2016112989

이름 : 김동규

담당교수 : 조정훈 교수님

제출일 : 2020.12.23.

1. 프로젝트 수행 목표 소개

(1) 수행 목표

라즈베리파이와 senseHAT을 이용하여 senseHAT의 온습도 정보와 인터넷의 미세먼지 데이터를 이용자에게 알려줄 수 있는 기능을 수행하는 시스템을 구성한다. 해당 시스템은 멀티쓰레드, 소켓프로그래밍, 디바이스 드라이버, 웹서비스를 통해 구현된다.

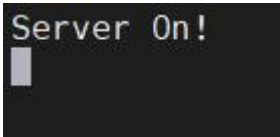
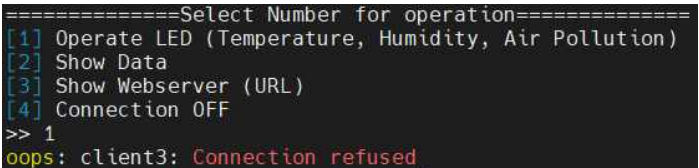
(2) 주요 기능

- senseHAT LED에 senseHAT 온습도 센서를 통해 측정한 결과와 웹에서 가져온 미세먼지 데이터 표시
- 소켓통신을 통해 서버에서 클라이언트에 온습도 및 미세먼지 데이터 즉시 제공
- 웹서버에 온습도 및 미세먼지 데이터 로그 기록 (날짜-시간 별)
- 소켓통신을 통한 서버 제어

2. 프로젝트 결과 개요

클라이언트는 서버에 기능에 대한 명령을 전달하고(1, 2, 3, 4 선택), 해당 명령에 따른 동작 혹은 결과 전달을 수행한다.

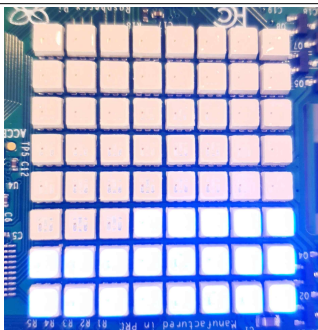
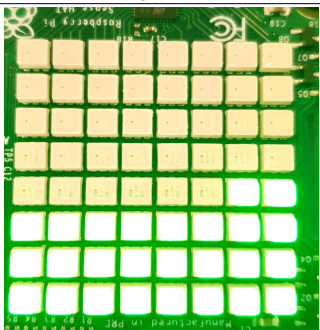
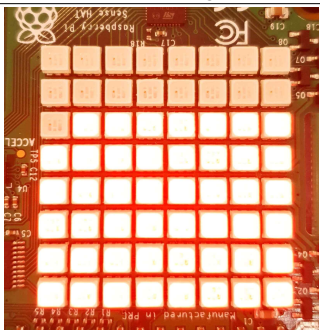
(1) 서버 동작 및 클라이언트와 통신

서버가 열린 상태	클라이언트가 서버에 연결되지 않은 경우
(서버 상태) 	(클라이언트 상태) 

(2) LED로 온도, 습도, 미세먼지(좋음, 보통, 나쁨 단계로 표시) 데이터 출력

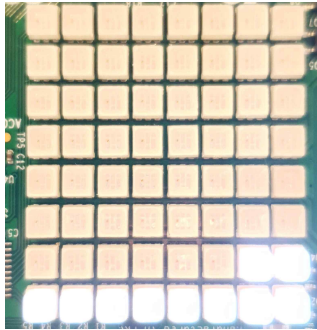
• 온도

온도에 따라 사용자가 느끼는 정도를 색으로 표현하여 한눈에 체감하기 쉽게 설정하였다. 상대적으로 값도 대략적으로 확인할 수 있다.

추움(BLUE) 10°C 이하	적당(GREEN) 11°C이상 28°C이하	더움(RED) 29°C 이상
		

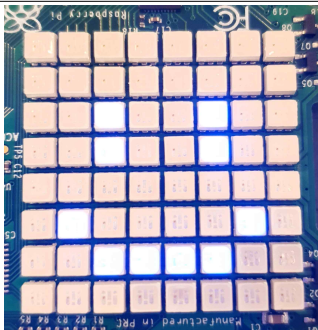
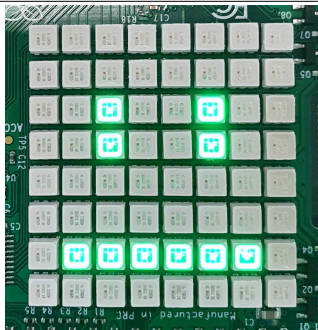
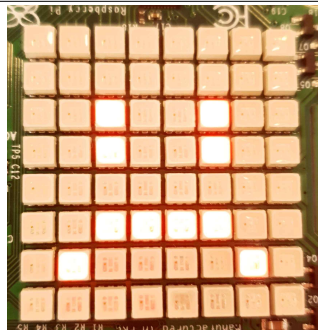
- 습도

전체 100%를 64개의 LED로 표현할 수 있도록 환산하였다.



- 미세먼지

미세먼지 상태 분류 기준에 따라 표정별로 분류하였다. 데이터는 에어코리아 (www.airkorea.or.kr)의 데이터를 이용하였으며 위치는 학교 근처 측정소를 지정하였다.

좋음(BLUE) 30 $\mu\text{g}/\text{m}^3$ 이하	보통(GREEN) 31~80 $\mu\text{g}/\text{m}^3$	나쁨(RED) 81 $\mu\text{g}/\text{m}^3$ 이상
		

<학교와 가까운 측정소 데이터> (출처 : 에어코리아)



(3) 클라이언트에 온도, 습도, 미세먼지 데이터 즉시 제공

클라이언트에서 정보를 요청할 경우 즉시 정확한 데이터를 제공한다.

```
=====Select Number for operation=====
[1] Operate LED (Temperature, Humidity, Air Pollution)
[2] Show Data
[3] Show Webserver (URL)
[4] Connection OFF
>> 2
You Chose Number [2] -> Temperature:30℃ , Humidity:17%, Air Pollution:43μg/m³
```

(4) 웹서버에 데이터 로그 기록 (날짜 및 시간 추가)

클라이언트가 요청한 날짜 및 시간에 측정된 해당 정보를 웹서버에 기록한다. 데이터 로그 형식으로 데이터를 이후에 조회할 수 있게 하는 목적이다. 클라이언트는 해당 정보가 게시되어 있는 URL을 전달받는다.

- 웹서버 화면

Raspberry PI Server Data Log

날 짜	시 간	온 도(℃)	습 도(%)	미세먼지PM2.5(μg/m³)
2020-12-18	21:25:57	33	33	26
2020-12-18	21:25:58	33	33	26
2020-12-18	21:26:01	33	33	26
2020-12-18	23:37:27	34	36	17
2020-12-18	23:37:49	34	36	17
2020-12-21	22:23:50	30	18	46
2020-12-23	02:24:57	36	42	47

- 클라이언트 화면(URL전달)

```
=====Select Number for operation=====
[1] Operate LED (Temperature, Humidity, Air Pollution)
[2] Show Data
[3] Show Webserver (URL)
[4] Connection OFF
>> 3
You Chose Number [3] -> http://192.168.219.104/log.html
```

(5) 클라이언트에서 서버와의 통신 종료

클라이언트와 서버 모두 종료하여 소켓 통신을 종료한다.

서버 화면	클라이언트 화면
<pre>Server On! Server OFF pi@raspberrypi:~/Project1 \$</pre>	<pre>=====Select Number for operation===== [1] Operate LED (Temperature, Humidity, Air Pollution) [2] Show Data [3] Show Webserver (URL) [4] Connection OFF >> 4 You Chose Number [4] -> End Connection</pre>

3. 코드 구성 및 내용

사용되는 함수들을 기능별로 나누어 모듈화 하였다. 크게 측정, 파싱, LED제어, 웹서버 업로드, 소켓통신의 기능으로 나뉜다.

```
pi@raspberrypi:~/Project1 $ ls
colors.h      log.html      Makefile      thd_led.c     utils.h
dust_data.c   main.c        th_data.c     thread.c      web.c
```

(1) 온도 습도 측정 (디바이스 드라이버 활용) - th_data.c

수업시간에 다룬 senseHAT 디바이스 드라이버를 활용하여 senseHAT을 통해 측정한 온도 및 습도를 리턴하는 함수를 만들고, 해당 데이터를 다른 함수에 전달하도록 한다.

<th_data.c>

```
1 #include <stdio.h>
2 #include <stdint.h>
3 #include <unistd.h>
4 #include <linux/i2c-dev.h>
5 #include <stdlib.h>
6 #include <fcntl.h>
7 #include <i2c/smbus.h>
8 #include <sys/ioctl.h>
9 #include "utils.h"
10
11 #define DEV_PATH      "/dev/i2c-1"
12 #define DEV_ID        0x5F
13 #define WHO_AM_I      0X0F
14
15 #define CTRL_REG1      0x20
16 #define CTRL_REG2      0x21
17
18 #define T0_OUT_L       0x3C
19 #define T0_OUT_H       0x3D
20 #define T1_OUT_L       0x3E
21 #define T1_OUT_H       0x3F
22 #define T0_degC_x8     0X32
23 #define T1_degC_x8     0x33
24 #define T1_T0_MSB      0x35
25
26 #define TEMP_OUT_L     0x2A
27 #define TEMP_OUT_H     0x2B
28
29 #define H0_T0_OUT_L     0x36
30 #define H0_T0_OUT_H     0x37
31 #define H1_T0_OUT_L     0x3A
32 #define H1_T0_OUT_H     0x3B
33 #define H0_rH_x2        0x30
34 #define H1_rH_x2        0x31
35
36 #define H_T_OUT_L       0x28
37 #define H_T_OUT_H       0x29
38
39 void delay(int t)
40 {
41     usleep(t * 1000);
42 }
```



```

44 int temp_out(void)
45 {

```

```

46     int fd = 0;
47     int temp = 0;
48     uint8_t status = 0;
49
50     /* open i2c comms */
51     if ((fd = open(DEV_PATH, O_RDWR)) < 0) {
52         perror("Unable to open i2c device");
53         exit(1);
54     }
55
56     /* configure i2c slave */
57     if (ioctl(fd, I2C_SLAVE, DEV_ID) < 0) {
58         perror("Unable to configure i2c slave device");
59         close(fd);
60         exit(1);
61     }
62
63     /* check we are who we should be */
64     if (i2c_smbus_read_byte_data(fd, WHO_AM_I) != 0x84) {
65         printf("%s\n", "who_am_i_error");
66         close(fd);
67         exit(1);
68     }
69
70     /* Power down the device (clean start) */
71     i2c_smbus_write_byte_data(fd, CTRL_REG1, 0x00);
72
73     /* Turn on the pressure sensor analog front end in single shot mode */
74     i2c_smbus_write_byte_data(fd, CTRL_REG1, 0x84);
75
76     /* Run one-shot measurement (temperature and pressure), the set bit will be reset by the
77      * sensor itself after execution (self-clearing bit)
78      */
79     i2c_smbus_write_byte_data(fd, CTRL_REG2, 0x01);
80
81     /* Wait until the measurement is complete */
82     do{
83         delay(25); /* 25 milliseconds*/
84         status = i2c_smbus_read_byte_data(fd, CTRL_REG2);
85     } while(status != 0);
86
87     /* Read calibration temperature LSB (ADC) data
88      * (temperature calibration x-data for two points)
89      */
90     uint8_t t0_out_l = i2c_smbus_read_byte_data(fd, T0_OUT_L);

```

```

91     uint8_t t0_out_h = i2c_smbus_read_byte_data(fd, T0_OUT_H);
92     uint8_t t1_out_l = i2c_smbus_read_byte_data(fd, T1_OUT_L);
93     uint8_t t1_out_h = i2c_smbus_read_byte_data(fd, T1_OUT_H);
94
95     /* Read calibration temperature data
96      * (temperature calibration y -data for two points)
97      */
98     uint8_t t0_degC_x8 = i2c_smbus_read_byte_data(fd, T0_degC_x8);
99     uint8_t t1_degC_x8 = i2c_smbus_read_byte_data(fd, T1_degC_x8);
100     uint8_t t1_t0_msb = i2c_smbus_read_byte_data(fd, T1_T0_MSB);
101
102     /* make 16 bit values (bit shift)
103      * (temperature calibration x-values)
104      */
105     int16_t T0_OUT = t0_out_h << 8 | t0_out_l;
106     int16_t T1_OUT = t1_out_h << 8 | t1_out_l;
107
108     /* make 16 and 10 bit values (bit mask and bit shift) */
109     uint16_t T0_DegC_x8 = (t1_t0_msb & 3) << 8 | t0_degC_x8;
110     uint16_t T1_DegC_x8 = ((t1_t0_msb & 12) >> 2) << 8 | t1_degC_x8;
111
112     /* Calculate calibration values
113      * (temperature calibration y-values)
114      */
115     double T0_DegC = T0_DegC_x8 / 8.0;
116     double T1_DegC = T1_DegC_x8 / 8.0;
117
118     /* Solve the linear equations 'y = mx + c' to give the
119      * calibration straight line graphs for temperature and humidity
120      */
121     double t_gradient_m = (T1_DegC - T0_DegC) / (T1_OUT - T0_OUT);
122     double t_intercept_c = T1_DegC - (t_gradient_m * T1_OUT);
123
124     /* Read the ambient temperature measurement (2 bytes to read) */
125     uint8_t t_out_l = i2c_smbus_read_byte_data(fd, TEMP_OUT_L);
126     uint8_t t_out_h = i2c_smbus_read_byte_data(fd, TEMP_OUT_H);
127
128     /* make 16 bit value */
129     int16_t T_OUT = t_out_h << 8 | t_out_l;
130
131     /* Calculate ambient temperature */
132     double T_DegC = (t_gradient_m * T_OUT) + t_intercept_c;
133
134     /* Output */
135     temp = (int)T_DegC;

```

```

136
137 /* Power down the device */
138 i2c_smbus_write_byte_data(fd, CTRL_REG1, 0x00);
139 close(fd);
140
141 return temp;
142 }
143
144 int humid_out(void)
145 {
146     int humid = 0;
147     int fd = 0;
148     uint8_t status = 0;
149
150     /* open i2c comms */
151     if ((fd = open(DEV_PATH, O_RDWR)) < 0) {
152         perror("Unable to open i2c device");
153         exit(1);
154     }
155
156     /* configure i2c slave */
157     if (ioctl(fd, I2C_SLAVE, DEV_ID) < 0) {
158         perror("Unable to configure i2c slave device");
159         close(fd);
160         exit(1);
161     }
162
163     /* check we are who we should be */
164     if (i2c_smbus_read_byte_data(fd, WHO_AM_I) != 0xBC) {
165         printf("%s\n", "who_am_i error");
166         close(fd);
167         exit(1);
168     }
169
170     /* Power down the device (clean start) */
171     i2c_smbus_write_byte_data(fd, CTRL_REG1, 0x00);
172
173     /* Turn on the pressure sensor analog front end in single shot mode */
174     i2c_smbus_write_byte_data(fd, CTRL_REG1, 0x84);
175
176     /* Run one-shot measurement (temperature and pressure), the set bit will be reset by the
177      * sensor itself after execution (self-clearing bit)
178      */
179     i2c_smbus_write_byte_data(fd, CTRL_REG2, 0x01);
180

```

```

181 /* Wait until the measurement is complete */
182 do{
183     delay(25); /* 25 milliseconds*/
184     status = i2c_smbus_read_byte_data(fd, CTRL_REG2);
185 } while(status != 0);
186
187
188 /* Read calibration relative humidity LSB(ADC) data
189  * (humidity calibration x-data for two points)
190  */
191 uint8_t h0_out_l = i2c_smbus_read_byte_data(fd, H0_T0_OUT_L);
192 uint8_t h0_out_h = i2c_smbus_read_byte_data(fd, H0_T0_OUT_H);
193 uint8_t h1_out_l = i2c_smbus_read_byte_data(fd, H1_T0_OUT_L);
194 uint8_t h1_out_h = i2c_smbus_read_byte_data(fd, H1_T0_OUT_H);
195
196 /* Read relative humidity (% rH) data
197  * (humidity calibration y-data for two points)
198  */
199 uint8_t h0_rh_x2 = i2c_smbus_read_byte_data(fd, H0_RH_X2);
200 uint8_t h1_rh_x2 = i2c_smbus_read_byte_data(fd, H1_RH_X2);
201
202
203 /* make 16 bit values (bit shift)
204  * (humidity calibration x-values)
205  */
206 int16_t H0_T0_OUT = h0_out_h << 8 | h0_out_l;
207 int16_t H1_T0_OUT = h1_out_h << 8 | h1_out_l;
208
209 /* Humidity calibration values
210  * (humidity calibration y-values)
211  */
212 double H0_rH = h0_rh_x2 / 2.0;
213 double H1_rH = h1_rh_x2 / 2.0;
214
215 /* Solve the linear equations 'y = mx + c' to give the
216  * calibration straight line graphs for temperature and humidity
217  */
218 double h_gradient_m = (H1_rH - H0_rH) / (H1_T0_OUT - H0_T0_OUT);
219 double h_intercept_c = H1_rH - (h_gradient_m * H1_T0_OUT);
220
221 /*Read the ambient humidity measurement (2 bytes to read) */
222 uint8_t h_t_out_l = i2c_smbus_read_byte_data(fd, H_T_OUT_L);
223 uint8_t h_t_out_h = i2c_smbus_read_byte_data(fd, H_T_OUT_H);
224
225 /* make 16 bit value */

```

```

226     int16_t H_T_OUT = h_t_out_h << 8 | h_t_out_l;
227
228     /*Calculate ambient humidity */
229     double H_rH = (h_gradient_m * H_T_OUT) + h_intercept_c;
230
231     /* Output */
232     humid = (int)H_rH;
233
234     /* Power down the device */
235     i2c_smbus_write_byte_data(fd, CTRL_REG1, 0x00);
236     close(fd);
237
238     return humid;
239 }

```

해당 코드의 경우, 수업시간에 다룬 라즈베리파이와 SenseHAT의 I2C 통신을 통한 온도 센서와 습도 센서의 디바이스 드라이버 코드를 활용하여 측정한 온도와 습도 데이터를 리턴하는 함수를 정의하였다. 해당 과정을 통한 리턴 값은 LED에 표시 될 때 구분의 편의를 위해 정수 값으로 전달되도록 하였다.

(2) 미세먼지 데이터 측정 (스레드 활용) - dust_data.c, thread.c

에어코리아 페이지에서 학교와 가까운 미세먼지 측정소의 데이터를 가져오는 코드이다. 해당 코드 중 웹서버의 상태를 확인하고 데이터를 가져오는 동작을 스레드를 사용하여 처리하였다.

<dust_data.c>

```

#include <stdlib.h>
#include <stdio_ext.h>
#include <string.h>
#include "utils.h"

int webcrawling() {
    char Site[100] = {"www.airkorea.or.kr/web/vicinityStation?item_code=10007\"&\"station_code=422154"};
    char Command[100] = {"wget -O index.html "};

    // _fpurge(stdin);
    strcat(Command, Site);
    system(Command);
    system("clear");
    return 1;
}

int webSelecting() {
    int d = 0;
    FILE *fp = fopen("index.html", "r");
    if (fp == NULL) {
        printf("index.html is not EXIST!\n");
        return 0;
    }
    else {
        char *buff = malloc(sizeof(char) * 1024);
        while (fgets(buff, 100, fp)) {
            char *ptr = strstr(buff, "font-size: 30px;\">");
            if (ptr != NULL) {
                d = atoi(ptr+18);
                break;
            }
        }
        free(buff);
        return d;
    }
}

```

우선 해당 사이트(에어코리아)의 내용 wget 명령어를 통해 index.html 파일에 저장한다. 이후, 해당 파일에서 미세먼지 데이터가 저장된 태그의 style을 나타내는 ...font-size: 30px;\">의 포인터를 전달 받아 이후 나오는 미세먼지 데이터를 atoi로 정수 형태로 리턴하였다.

<thread.c>

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <string.h>
5 #include <pthread.h>
6 #include "utils.h"
7 int dust_data; // global parameter
8
9 void *dust_thread();
10
11 // use thread for web crawling to get pollution
12 void use_thread(void)
13 {
14     int res;
15     pthread_t a_thread;
16     void *thread_result;
17
18     res = pthread_create(&a_thread, NULL, dust_thread, NULL);
19
20     if (res != 0) {
21         perror("Thread creation failed\n");
22         exit(EXIT_FAILURE);
23     }
24     res = pthread_join(a_thread, &thread_result);
25
26     if (res != 0) {
27         perror("Thread join failed\n");
28         exit(EXIT_FAILURE);
29     }
30 }
31
32
33 void *dust_thread()
34 {
35     int d;
36     if(webcrawling())
37         d = webSelecting();
38     dust_data = d;
39     system("rm index.html");
40 }
41
```

위에서 정의된 webcrawling() 함수와 webSelecting 함수를 스레드를 이용하여 처리해 주었다.

(3) 데이터를 기반으로 LED 작동 - colors.h, thd_led.c

LED제어 디바이스 드라이버를 활용하여 상황에 따른 LED동작을 구현하였다.

LED의 색상은 colors.h파일에 별도로 정의해 두었다. (실제로 사용한 색은 RGBW정도이다.)

다른 부분은 수업에서 다룬 내용과 유사하고, 색상 작동 부분은 다르게 처리하였다.

<thd_led.c>

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <sys/stat.h>
5 #include <unistd.h>
6 #include <fcntl.h>
7 #include <sys/mman.h>
8 #include <stdint.h>
9 #include <string.h>
10 #include <linux/fb.h>
11 #include <sys/ioctl.h>
12 #include "colors.h"
13 #include "utils.h"
14
15
16 #define FILEPATH "/dev/fb0"
17 #define NUM_WORDS 64
18 #define FILESIZE (NUM_WORDS * sizeof(uint16_t))
19
20 void t_led(int temperature)
21 {
22     int i;
23     int fbfd;
24     uint16_t color;
25     uint16_t *map;
26     uint16_t *p;
27     struct fb_fix_screeninfo fix_info;
28
29     /*temp data*/
30     int t_out = temperature+10;
31
32     /* open the led frame buffer device */
33     fbfd = open(FILEPATH, O_RDWR);
34     if (fbfd == -1) {
35         perror("Error (call to 'open')");
36         exit(EXIT_FAILURE);
37     }
38
39     /* read fixed screen info for the open device */
40     if (ioctl(fbfd, FBIOGET_FSCREENINFO, &fix_info) == -1) {
41         perror("Error (call to 'ioctl')");
42         close(fbfd);
43         exit(EXIT_FAILURE);
44     }
45
46     /*now check the correct device has been found*/
47     if (strcmp(fix_info.id, "RPi-Sense FB") != 0) {
48         printf("%s\n", "Error: RPi-Sense FB not found");
49         close(fbfd);
50         exit(EXIT_FAILURE);
51     }
52
53     /*map the led frame buffer device into memory*/
54     map = mmap(NULL, FILESIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fbfd, 0);
55     if (map == MAP_FAILED) {
56         close(fbfd);
57         perror("Error mmaping the file");
58         exit(EXIT_FAILURE);
59     }
60
61     /*set a pointer to the start of the memory area */
62     p = map;
63
64     /*clear the led matrix */
65     memset(map, 0, FILESIZE);
66
67     if (t_out <= 26) { // below 10°C
68         color = RGB565_BLUE;
69     } else if (t_out > 26 && t_out <= 44) { // 10°C < t < 28°C
70         color = RGB565_GREEN;
71     } else if (t_out > 44) { // higher than 28°C
72         color = RGB565_RED;
73     }
74     /*light up*/
75     for (i = 0; i < t_out; i++) {
76         *(p + i) = color;
77         delay(25);
78     }
79
80     delay(2000);
81
82     /* clear the led matrix */
83     memset(map, 0, FILESIZE);
84
85     /* un-map and close */
86     if (munmap(map, FILESIZE) == -1) {
87         perror("Error un-mmaping the file");
88     }
89     close(fbfd);
90 }
```

```

92 void h_led(int humidity)
93 {
94     int i;
95     int fbfd;
96     uint16_t *map;
97     uint16_t *p;
98     struct fb_fix_screeninfo fix_info;
99
100     /*humidity data*/
101     int h_out = (int)(humidity*0.64);
102
103     /* open the led frame buffer device */
104     fbfd = open(FILEPATH, O_RDWR);
105     if (fbfd == -1) {
106         perror("Error (call to 'open')");
107         exit(EXIT_FAILURE);
108     }
109
110     /* read fixed screen info for the open device */
111     if (ioctl(fbfd, FBIOGET_FSCREENINFO, &fix_info) == -1) {
112         perror("Error (call to 'ioctl')");
113         close(fbfd);
114         exit(EXIT_FAILURE);
115     }
116
117     /*now check the correct device has been found*/
118     if (strcmp(fix_info.id, "RPi-Sense FB") != 0) {
119         printf("%s\n", "Error: RPi-Sense FB not found");
120         close(fbfd);
121         exit(EXIT_FAILURE);
122     }
123
124     /*map the led frame buffer device into memory*/
125     map = mmap(NULL, FILESIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fbfd, 0);
126     if (map == MAP_FAILED) {
127         close(fbfd);
128         perror("Error mmaping the file");
129         exit(EXIT_FAILURE);
130     }
131
132     /*set a pointer to the start of the memory area */
133     p = map;
134

```

```

135     /*clear the led matrix */
136     memset(map, 0, FILESIZE);
137
138     /*light up*/
139     for (i = 0; i < h_out; i++) {
140         *(p + i) = RGB565_WHITE;
141         delay(25);
142     }
143
144     delay(2000);
145
146     /* clear the led matix */
147     memset(map, 0, FILESIZE);
148
149     /* un-map and close */
150     if (munmap(map, FILESIZE) == -1) {
151         perror("Error un-mmapping the file");
152     }
153     close(fbfd);
154 }
155

```

```

156 void d_led(int dust)
157 {
158     int i;
159     int fbfd;
160     uint16_t color;
161     uint16_t *map;
162     uint16_t *p;
163     struct fb_fix_screeninfo fix_info;
164
165     /*temp data*/
166     int d_out = dust;
167
168     /* open the led frame buffer device */
169     fbfd = open(FILEPATH, O_RDWR);
170     if (fbfd == -1) {
171         perror("Error (call to 'open')");
172         exit(EXIT_FAILURE);
173     }
174
175     /* read fixed screen info for the open device */
176     if (ioctl(fbfd, FBIOGET_FSCREENINFO, &fix_info) == -1) {
177         perror("Error (call to 'ioctl')");
178         close(fbfd);
179         exit(EXIT_FAILURE);
180     }
181
182     /*now check the correct device has been found*/
183     if (strcmp(fix_info.id, "RPi-Sense FB") != 0) {
184         printf("%s\n", "Error: RPi-Sense FB not found");
185         close(fbfd);
186         exit(EXIT_FAILURE);
187     }
188
189     /*map the led frame buffer device into memory*/
190     map = mmap(NULL, FILESIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fbfd, 0);
191     if (map == MAP_FAILED) {
192         close(fbfd);
193         perror("Error mmapping the file");
194         exit(EXIT_FAILURE);
195     }
196

```

```

197     /*set a pointer to the start of the memory area */
198     p = map;
199
200     /*clear the led matrix */
201     memset(map, 0, FILESIZE);
202
203     /*light up*/
204     int ledArr[10];
205     if (d_out <= 30) {
206         int Arr[10] = {11, 12, 13, 14, 18, 23, 35, 38, 43, 46};
207         memcpy(ledArr, Arr, sizeof(Arr));
208         color = RGB565_BLUE;
209     } else if (d_out > 30 && d_out <= 80) {
210         int Arr[10] = {10, 11, 12, 13, 14, 15, 35, 38, 43, 46};
211         memcpy(ledArr, Arr, sizeof(Arr));
212         color = RGB565_GREEN;
213     } else if (d_out > 80) {
214         int Arr[10] = {10, 15, 19, 20, 21, 22, 35, 38, 43, 46};
215         memcpy(ledArr, Arr, sizeof(Arr));
216         color = RGB565_RED;
217     }
218
219     for (i = 0; i < sizeof(ledArr)/sizeof(int); i++) {
220         *(p + ledArr[i] - 1) = color;
221     }
222     delay(2000);
223
224     /* clear the led matix */
225     memset(map, 0, FILESIZE);
226
227     /* un-map and close */
228     if (munmap(map, FILESIZE) == -1) {
229         perror("Error un-mmapping the file");
230     }
231     close(fbfd);
232 }

```

(4) 웹서버에 데이터 업로드 - web.c

웹서버에 html형식의 파일을 업로드 한다. 명령어 수행마다 새로운 내용이 추가되도록 구성하기 위해 html파일 작성 시 시작점의 포인터가 </body></html> 태그 앞에 오도록 하고, 형식에 맞게 파일에 내용을 작성하였다. (* log.txt는 테스트용 파일에서 사용)

<web.c>

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <string.h>
5 #include "utils.h"
6
7 void upload_txt(int temp, int humid, int dust)
8 {
9     FILE *fp = fopen("log.txt", "a");
10    char Msg[1024];
11    time_t t = time(NULL);
12    struct tm tm = *localtime(&t);
13
14    sprintf(Msg, "%4d-%02d-%02d %02d:%02d:%02d %3d %3d %3d\n", tm.tm_year+1900, tm.tm_mon+1, tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec, temp, humid, dust);
15    fputs(Msg, fp);
16    fclose(fp);
17    system("sudo cp log.txt /var/www/html/log.txt");
18 }
19
20 void upload_html(int temp, int humid, int dust)
21 {
22     FILE *fp = fopen("log.html", "r+");
23     char Msg[1024];
24     time_t t = time(NULL);
25     struct tm tm = *localtime(&t);
26     fseek(fp, -17, SEEK_END);
27     sprintf(Msg, "\n\t\t\t<tr style='font-size:100%'>\n<th>%4d-%02d-%02d</th>\n<th>%02d:%02d:%02d</th>\n<th>%3d</th>\n<th>%3d</th>\n<th>%3d</th>\n\t\t\t</tr>\n\t\t<tr>\n\t\t<td>\n<html>-, tm.tm_year+1900, tm.tm_mon+1, tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec, temp, humid, dust);
28     fputs(Msg, fp);
29     fclose(fp);
30     system("sudo cp log.html /var/www/html/log.html");
31 }
32
33
```

(5) 메인함수 (소켓 통신을 통해 기능 제어) - utils.h, main.c

메인함수에서는 소켓 통신을 통해 앞에서 구현한 함수들을 제어할 수 있는 서버를 구축하고, 클라이언트에서 요청한 작업을 수행하도록 구성하였다.

<utils.h>

```
1 #ifndef UTILS_DOT_H
2 #define UTILS_DOT_H
3
4 extern int dust_data;
5 void delay(int t);
6 int temp_out(void);
7 int humid_out(void);
8 void t_led(int temperature);
9 void h_led(int humidity);
10 void d_led(int dust);
11 void use_thread();
12 int webcrawling();
13 int webSelecting();
14 void upload_txt(int temp, int humid, int dust);
15 void upload_html(int temp, int humid, int dust);
16
17 #endif
```

따로 정의된 함수들을 메인 함수에서 사용하기 위해 헤더 파일에 재정의하였다.

<main.c>

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <string.h>
5 #include <sys/types.h>
6 #include <sys/socket.h>
7 #include <netinet/in.h>
8 #include "utils.h"
9
10 #define MSG_S      60
11 #define MSG_L      1024
12 #define PORT_NUM   3655
13
14 void main(void)
15 {
16     int server_sockfd, client_sockfd;
17     int server_len, client_len;
18     struct sockaddr_in server_address;
19     struct sockaddr_in client_address;
20     char Msg1[MSG_S] = "You Chose Number [1] -> Check LED!\n";
21     char Msg2[MSG_L];
22     char Msg3[MSG_S] = "You Chose Number [3] -> http://192.168.219.104/log.html\n"; // local ip address
23     char Msg4[MSG_S] = "You Chose Number [4] -> End Connection\n";
24     char Msgd[MSG_S] = "Err - Wrong Number -> Try Again\n";
25     int strlen;
26     int t, h, d;
27
28     unlink("server_socket");
29     server_sockfd = socket(AF_INET, SOCK_STREAM, 0);
30
31     server_address.sin_family = AF_INET;
32     server_address.sin_addr.s_addr = htonl(INADDR_ANY);
33     server_address.sin_port = htons(PORT_NUM);
34     server_len = sizeof(server_address);
35     bind(server_sockfd, (struct sockaddr *)&server_address, server_len);
36
37     listen(server_sockfd, 5);
38
39     while(1) {
40         char num;
41         printf("Server On!\n");
42         client_len = sizeof(client_address);
43         client_sockfd = accept(server_sockfd, (struct sockaddr *)&client_address, &client_len);
44         recv(client_sockfd, &num, 1, 0);
45         switch(num) {
46
47             case '1':
48                 use_thread();
49                 send(client_sockfd, Msg1, MSG_S, 0);
50                 t = temp_out();
51                 h = humid_out();
52                 d = dust_data;
53                 printf("LED LIGHTING\n");
54                 t_led(t);
55                 h_led(h);
56                 d_led(d);
57                 break;
58             case '2':
59                 use_thread();
60                 d = dust_data;
61                 t = temp_out();
62                 h = humid_out();
63                 sprintf(Msg2, "You Chose Number [2] -> Temperature:%d", Humidity:%d, Air Pollution:%dug/m\n", t, h, '%', d);
64                 send(client_sockfd, Msg2, MSG_L, 0);
65                 break;
66             case '3':
67                 use_thread();
68                 send(client_sockfd, Msg3, MSG_S, 0);
69                 t = temp_out();
70                 h = humid_out();
71                 d = dust_data;
72                 upload_html(t, h, d);
73                 break;
74             case '4':
75                 send(client_sockfd, Msg4, MSG_S, 0);
76                 printf("Server OFF\n");
77                 exit(0);
78             default:
79                 send(client_sockfd, Msgd, MSG_S, 0);
80         }
81     }
82 }
```

메인 함수의 경우, 소켓 통신을 위한 서버로 작동하며, 클라이언트가 전달한 숫자에 따라 케이스를 나누어 다른 명령어 및 함수가 실행 되도록 하였다. 혹시나 모를 함수 명 충돌의 방지를 위해 소켓 통신의 구현에 있어 recv(), send() 함수를 사용하였다.

(5) 클라이언트 (소켓 통신을 통해 서버에 명령 전달) - sock_client.c

<sock_client.c>

```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3 #include <stdio.h>
4 #include <netinet/in.h>
5 #include <arpa/inet.h>
6 #include <unistd.h>
7 #include <stdlib.h>
8
9 int main()
10 {
11     int sockfd;
12     int len;
13     struct sockaddr_in address;
14     int result;
15     char ch;
16     char Msg[1024] = {0, };
17     char flag = 0;
18     sockfd = socket(AF_INET, SOCK_STREAM, 0);
19
20     address.sin_family = AF_INET;
21     address.sin_addr.s_addr = inet_addr("127.0.0.1");
22     //192.168.219.104
23
24     address.sin_port = htons(3655);
25
26     len = sizeof(address);
27     printf("=====Select Number for operation=====\n");
28     printf("[1] Operate LED (Temperature, Humidity, Air Pollution)\n");
29     printf("[2] Show Data\n");
30     printf("[3] Show Webserver (URL)\n");
31     printf("[4] Connection OFF\n");
32     printf(">>> ");
33     scanf("%c", &ch);
34     flag = ch;
35     result = connect(sockfd, (struct sockaddr *)&address, len);
36
37     if (result == -1) {
38         perror("oops: client3");
39         exit(1);
40     }
41     send(sockfd, &ch, 1, 0);
42     recv(sockfd, Msg, 1024, 0);
43     printf("%s\n", Msg);
44     close(sockfd);
45     if (flag != '4') {
46         system("./client");
47     } else {
48         exit(0);
49     }
50 }
```

클라이언트 부분 역시 recv(), send()함수를 활용하여 서버와 통신하였다. 명령에 따른 플래그를 두어 수행 종료(4)가 아닐 경우 서버와 지속적으로 통신이 가능하게끔 해당 파일을 다시 실행하도록 구성하였다. (리눅스 명령어를 수행하는 system() 함수 활용)

4. 수행 과정 중 문제점 및 해결방안

(1) 웹에서 미세먼지 데이터 파싱

C언어에서 웹사이트의 데이터를 가져오는 과정에서 API를 활용하지 않고 가져 오는 방법에 어려움이 있었다. 해당 과정은 웹서버의 특정 데이터를 바로 가져오는 방법을 활용하는 대신, 리눅스의 wget명령어를 통해 해당 웹서버 내용(HTML)을 파일에 저장하고 해당 파일을 읽어 오는 방법으로 원하는 데이터를 추출하였다. 이 과정에서도 해당 데이터의 포인터를 결정하는데 오래 걸렸다.

(2) 다른 디렉토리의 파일 쓰기(root권한)

웹서버에 올라갈 파일을 작성하는 경우, sudo 명령어를 통해 파일을 쓰는 작업을 수행한다. 작성 과정 중에 sudo 명령어를 작성하지 않아 불편함이 생기는 경우가 많았다. 그래서 작성 및 수정을 쉽게 하기 위해 메인파일이 있는 디렉토리에 해당 html파일을 작성하고, 그 파일을 서버 디렉토리에 복사하는 방식을 사용하였다. sudo 명령어를 코드에서 사용하기 위해 system 함수(stdlib.h)를 사용하였다. 이를 활용하면 굳이 www/html/ 디렉토리에 접근하지 않고도 웹서버를 쉽게 수정할 수 있었다.

5. 느낀점

라즈베리파이를 다루어 보는 것은 이번 학기가 처음이지만, 흥미로웠던 부분들이 많았다. 특히 단순히 파이썬과 같은 라이브러리 제공 언어를 활용하여 디바이스를 제어하는 것 보다, 디바이스 드라이버를 구성하여 직접 제어해보는 경험도 새롭고 많은 의미가 있었다. 이번에는 수업에서 다룬 내용들을 바탕으로 구성했지만, 다음에는 다른 새로운 디바이스와 기능을 이용하여 프로젝트를 진행해보고 싶다.