

실습과제)

v 디바이스 드라이버에 ioctl() 추가

\$ ioctl(fd, 1, 0): Red LED 1, 2, 3, 4 (default)

\$ ioctl(fd, 1, 1): Green LED 9, 10, 11, 12

v 해당 디바이스 드라이버를 이용하여 Red/Green LED 제어 application 작성

\$./digit_app2 0 1010 (Red 출력)

\$./digit_app2 1 1010 (Green 출력)

우선 새로운 ioctl() 함수를 추가하기 위해서 다음과 같이 함수를 작성 하였습니다.

```
static struct file_operations fops={
    .owner    =THIS_MODULE,
    .open     =led_open,
    .release  =led_release,
    .write    =led_write,
    .unlocked_ioctl  =led_ioctl,
};
```

Led_dd.c 파일에서 구조체에 추가로 led_ioctl 함수를 추가해 주었습니다. 후에 digital_app에서 사용되는 함수는 ioctl 로 사용됩니다.

그다음 해당 파일 안에서 추가로 ioctl 의 함수 내용을 추가 하였습니다.

```
static int led_ioctl( struct file *filp,
                     unsigned int cmd, unsigned long arg) {

    if(arg == 0)
        RofG = 0;
    else
        RofG = 1;

    printk("[led_dd] led_ioctl\n");
    return 0;
}
```

Led_ioctl 함수를 쓰기 이전에, static int 로 RofG 라는 전역 변수를 선언하였습니다. ioctl 은 arg 로 받아온 값에 따라 RofG 에 red 혹은 green 의 led 를 켜게 값을 할당 해줍니다.

이번 과제에서의 ioctl 함수는 cmd가 1로 고정되어 있으므로, 따로 cmd에 관한 코드를 작성 할 필요가 없습니다. 그리고 Arg 파라미터를 받으면 arg 값에 따라 RofG 를 0 혹은 1 이 되게 합니다.

```
static int RofG = 0;

static int led_write(struct file *filp, const char *buf,
                    size_t len, loff_t *f_pos) {
    int i;
    char state;

    for(i = 0; i < len; i++){
        copy_from_user(&state, &buf[i], 1);

        if(RofG == 0){
            if(state == '0') {
                digitalWrite(Led[3-i], LOW);
            }
            else {
                digitalWrite(Led[3-i], HIGH);
            }
        }
        else if(RofG == 1) {
            if(state == '0') {
                digitalWrite(Led[11-i], LOW);
            }
            else {
                digitalWrite(Led[11-i], HIGH);
            }
        }
    }
}
```

Arg 값이 0이었을 때 RofG 는 0 값이 할당되고, 첫 번째 줄의 red led 가 켜지게 되고, Arg 값이 1이었을 때는 RofG 는 1 값이 할당되고 세 번째 줄의 green led 가 켜지게 됩니다. Green 일 때는 led번호가 8,9,10,11번 이므로, led[11-i] 로 설정해준다.

나머지 코드는 실습 예제의 코드와 동일합니다.

```

#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <fcntl.h>
#include <linux/kdev_t.h>

#define _LED_PATH_ "/dev/led_dd"

int main(int argc, char **argv) {
    int fd = 0;
    char* pos;

    if(argc != 3){
        printf("Usage: %s [LED binary]\n", argv[0]);
        exit(1);
    }

    if((fd = open(_LED_PATH_, O_RDWR | O_NONBLOCK)) < 0) {
        perror("open()");
        exit(1);
    }

    ioctl(fd, 1, strtoul(argv[1], &pos, 10));
    write(fd, argv[2], strlen(argv[2]));

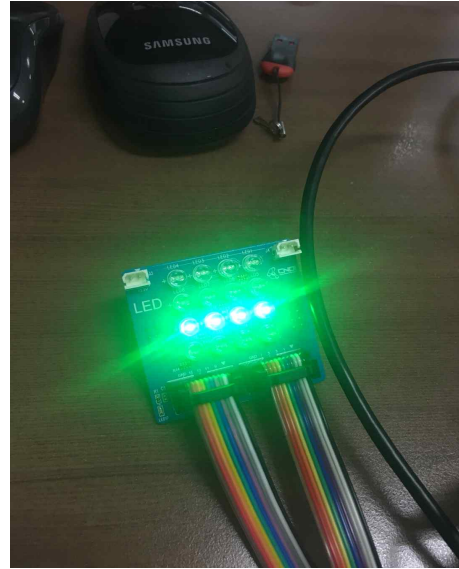
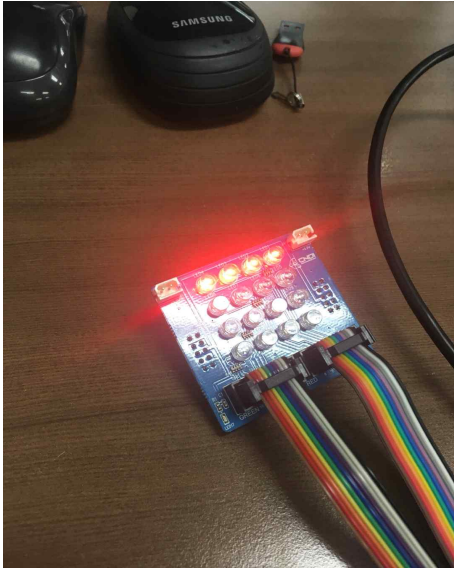
    close(fd);

    return 0;
}

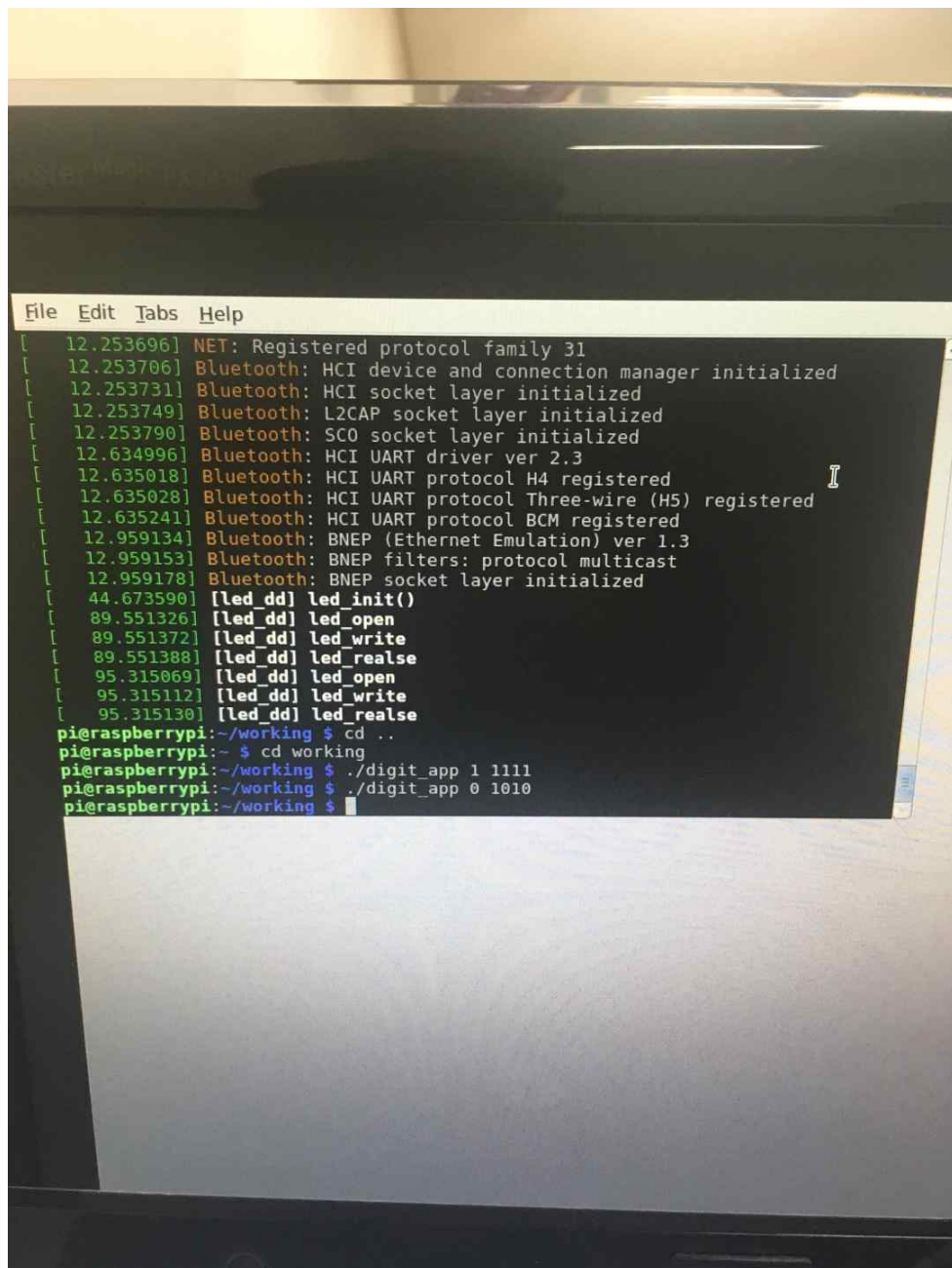
```

Digit_app.c 에서 수정된 코드는, 우선 digit_app 이 ./digit_app 1 1010 과 같이 실행될 때 입력 파라미터가 3개 이므로, argc != 3으로 수정해 주었습니다.

그런 다음 ioctl 함수를 호출 하는데, ioctl 이 받는 arg 파라미터는 unsigned long 타입입니다. main 함수에서 받는 argv 값은 char 형이므로 strtoul 함수를 이용해서 unsigned long 타입으로 바꿔주었습니다.



<컴파일 후 실행 결과모습>



```
File Edit Tabs Help
[ 12.253696] NET: Registered protocol family 31
[ 12.253706] Bluetooth: HCI device and connection manager initialized
[ 12.253731] Bluetooth: HCI socket layer initialized
[ 12.253749] Bluetooth: L2CAP socket layer initialized
[ 12.253790] Bluetooth: SCO socket layer initialized
[ 12.634996] Bluetooth: HCI UART driver ver 2.3
[ 12.635018] Bluetooth: HCI UART protocol H4 registered
[ 12.635028] Bluetooth: HCI UART protocol Three-wire (H5) registered
[ 12.635241] Bluetooth: HCI UART protocol BCM registered
[ 12.959134] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
[ 12.959153] Bluetooth: BNEP filters: protocol multicast
[ 12.959178] Bluetooth: BNEP socket layer initialized
[ 44.673590] [led_dd] led_init()
[ 89.551326] [led_dd] led_open
[ 89.551372] [led_dd] led_write
[ 89.551388] [led_dd] led_realse
[ 95.315069] [led_dd] led_open
[ 95.315112] [led_dd] led_write
[ 95.315130] [led_dd] led_realse
pi@raspberrypi:~/working $ cd ..
pi@raspberrypi:~ $ cd working
pi@raspberrypi:~/working $ ./digit_app 1 1111
pi@raspberrypi:~/working $ ./digit_app 0 1010
pi@raspberrypi:~/working $
```

<라즈베리 파이 동작 화면>