lab4-3.

실습 과제

1) 디바이스 드라이버에 ioctl() 추가

- Device Driver [led_dd.c]

struct file_operations fops에 ioctl를 추가하니 오류가 발생하여, unlocked_ioctl을 추가하여 오류를 해결하였습니다. 오류의 원인은 커널 2.6.34 버전부터 기존의 ioctl이 사라지고, 대신 unlocked_ioctl로 대체되었기 때문입니다. BKL(Big Kernel Lock)로 보호되어 동작하는 ioctl은 SMP(Simultaneous Multi-Processing) 환경에서 불필요한 대기 시간을 발생시키는 비효율적인 면이 있어 이를 개선하고자 BKL을 사용하지 않는 unlocked_ioctl로 대체되었다고 합니다.

따라서, unlocked_ioctl에 led_ioctl()을 등록해주었고, 이 함수에서 led의 색깔을 결정해주었습니다. application program의 argv[1] 입력값에 따라 전역변수 led_color의 값을 변경해주었습니다.

이후, led_write()에서 전역변수 led_color의 값에 따라 Red 또는 Green LED를 출력하도록 하였습니다.

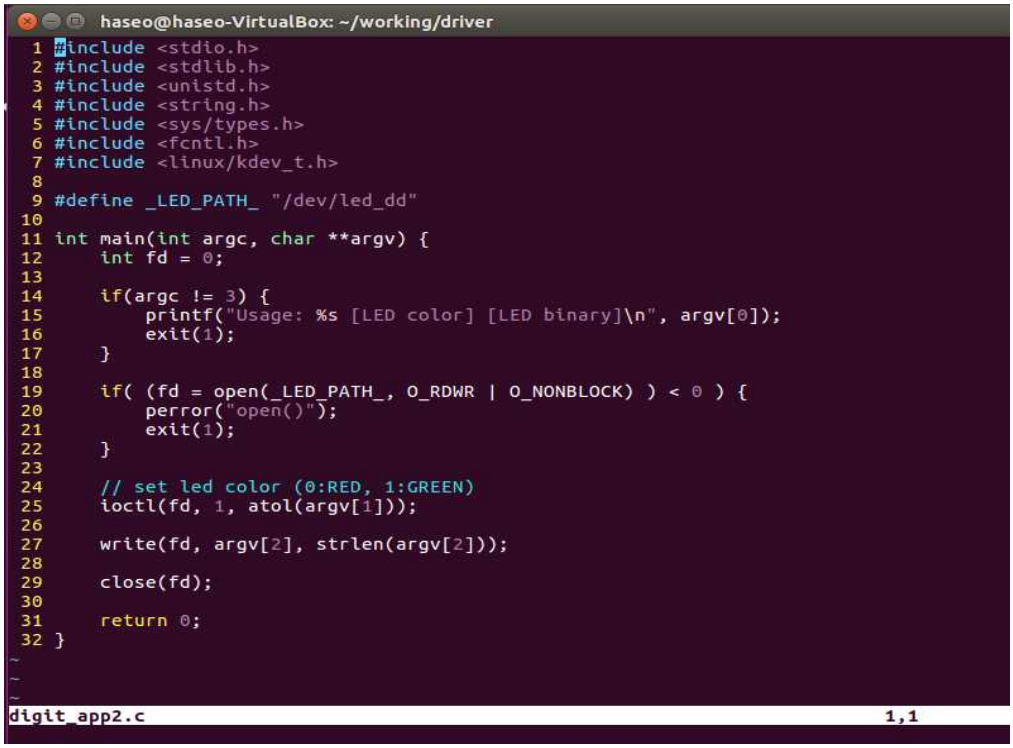
```c
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/fs.h>
#include <linux/io.h>
#include <linux/delay.h>
#include <asm/io.h>
#include <asm/uaccess.h>

#define DRIVER_AUTHOR   "KHU"
#define DRIVER_DESC     "Led Driver"
#define DEVICE_NAME     "led_dd"
#define MAJOR_NUMBER    222

#define GPIO_BASE       0x3F200000
#define GPIO_SIZE       0xC0

#define INPUT           0
#define OUTPUT          1
#define LOW             0
#define HIGH            1

void __iomem *gpioAddr;
volatile unsigned int gpioToGPFSEL[] = {
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
    8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
};
volatile unsigned int gpioToShift[] = {
    0, 3, 6, 9, 12, 15, 18, 21, 24, 27,
    0, 3, 6, 9, 12, 15, 18, 21, 24, 27,
    0, 3, 6, 9, 12, 15, 18, 21, 24, 27,
};
volatile unsigned int gpioToGPSET = 0x1C;
volatile unsigned int gpioToGPCLR = 0x28;

const int Led[16] = {
    4, 17, 18, 27, 22, 23, 24, 25,
    6, 12, 13, 16, 19, 20, 26, 21
};

int led_color;

static volatile void initGpioAddr(void) {
    if(gpioAddr == NULL) {
        gpioAddr = ioremap(GPIO_BASE, GPIO_SIZE);
    }
}

static void pinMode(int pin, int mode) {
    volatile unsigned int *gpio = (volatile unsigned int *)gpioAddr;

    unsigned int fsel = gpioToGPFSEL[pin]/sizeof(unsigned int);
    unsigned int shift = gpioToShift[pin];

    if(mode) {
        gpio[fsel] |= (1 << shift);
    }
    else {
        gpio[fsel] |= (0 << shift);
    }
}

static void digitalWrite(int pin, int value) {
    volatile unsigned int *gpio = (volatile unsigned int *)gpioAddr;

    unsigned int set = gpioToGPSET/sizeof(unsigned int);
    unsigned int clr = gpioToGPCLR/sizeof(unsigned int);

    if(value) {
        gpio[set] = (1 << pin);
    }
    else {
        gpio[clr] = (1 << pin);
    }
}

static int led_open(struct inode *inode, struct file *filp) {
    int i;

    initGpioAddr();

    for(i = 0; i < 16; i++) {
        pinMode(Led[i], OUTPUT);
        digitalWrite(Led[i], LOW);
    }

    printk("[led_dd] led_open\n");

    return 0;
}

static int led_release(struct inode *inode, struct file *filp) {
    iounmap(gpioAddr);

    printk("[led_dd] led_realse\n");

    return 0;
}

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

```
105     int led_pos;
106
107     if (led_color == 0)
108         led_pos = 3;
109     else if (led_color == 1)
110         led_pos = 11;
111
112     for(i = 0; i < len; i++) {
113         copy_from_user(&state, &buf[i], 1);
114
115         if(state == '0') {
116             digitalWrite(Led[led_pos-i], LOW);
117         }
118         else {
119             digitalWrite(Led[led_pos-i], HIGH);
120         }
121     }
122
123     printk("[led_dd] led_write\n");
124
125     return len;
126 }
127
128 static int led_ioctl(struct file *filep, unsigned int cmd, \
129                      unsigned long arg)
130 {
131     printk("[led_dd] led_ioctl() : set led_color to ");
132
133     if (arg == 0)
134     {
135         printk("Red\n");
136         led_color = 0;
137     }
138     else if (arg == 1)
139     {
140         printk("Green\n");
141         led_color = 1;
142     }
143
144     return led_color;
145 }
146
147 static struct file_operations fops = {
148     .owner          = THIS_MODULE,
149     .open           = led_open,
150     .release        = led_release,
151     .write          = led_write,
152     .unlocked_ioctl = led_ioctl,
153 };
154
155 static int led_init(void) {
156     printk("[led_dd] led_init()\n");
led_dd.c [+]
-- INSERT --
```

```
157     register_chrdev(MAJOR_NUMBER, DEVICE_NAME, &fops);
158
159     return 0;
160 }
161
162 static void led_exit(void) {
163     printk("[led_dd] led_exit()\n");
164     unregister_chrdev(MAJOR_NUMBER, DEVICE_NAME);
165 }
166
167 module_init(led_init);
168 module_exit(led_exit);
169
170 MODULE_LICENSE("Dual BSD/GPL");
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
led_dd.c [+]
-- INSERT --
```

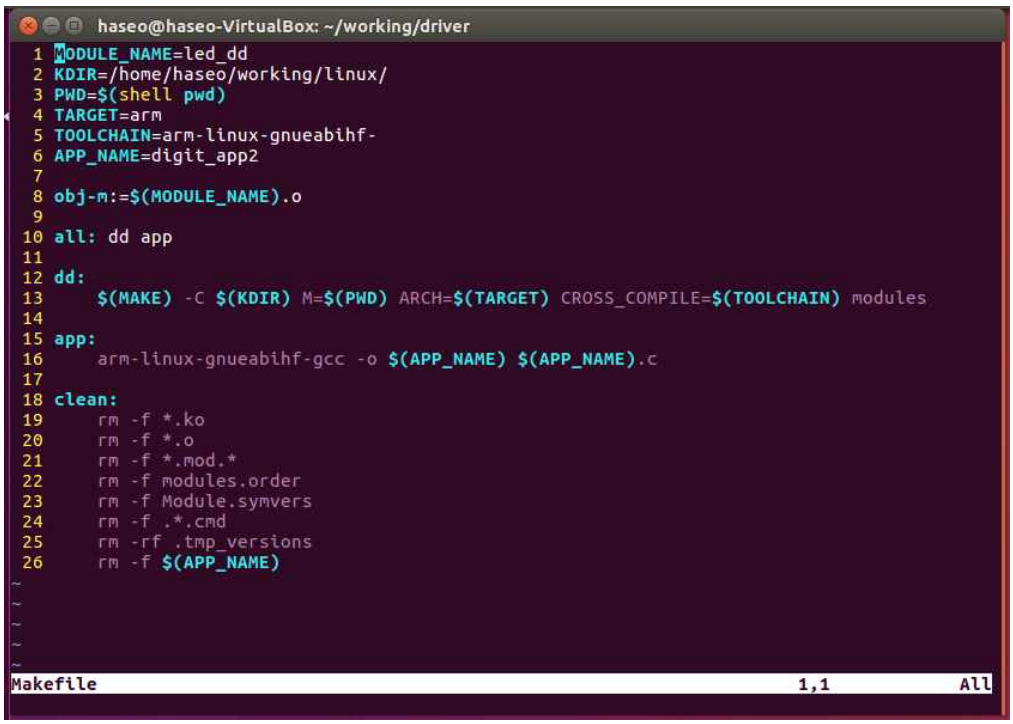
2) 해당 디바이스 드라이버를 이용하여 Red/Green LED 제어 application 작성

- Device Driver Application [digit_app2.c]



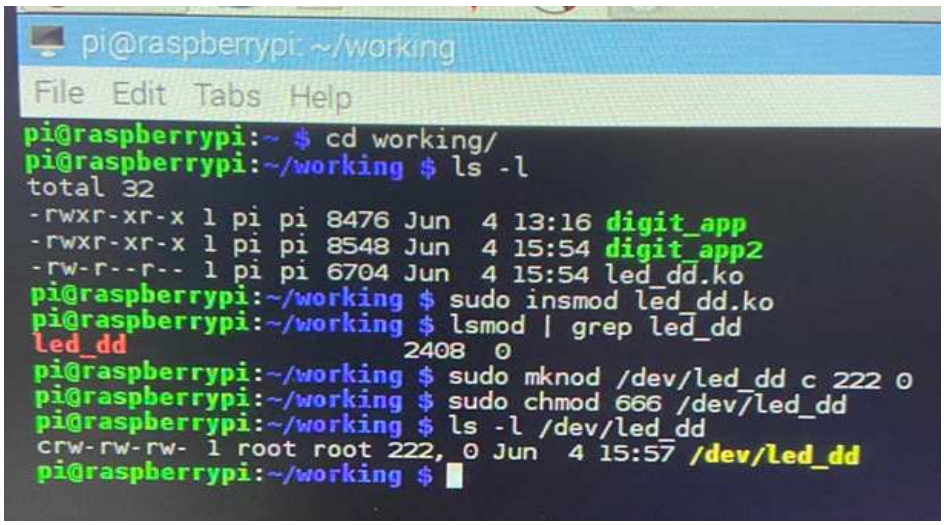

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <fcntl.h>
#include <linux/kdev_t.h>

#define _LED_PATH_ "/dev/led_dd"

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

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

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

    // set led color (0:RED, 1:GREEN)
    ioctl(fd, 1, atol(argv[1]));

    write(fd, argv[2], strlen(argv[2]));

    close(fd);

    return 0;
}
```

- Makefile



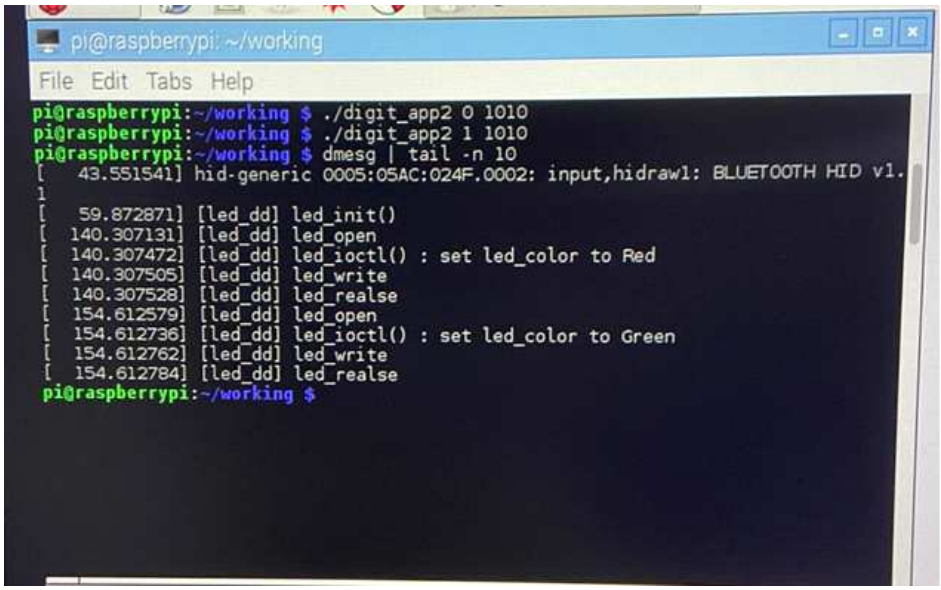

```
MODULE_NAME=led_dd
KDIR=/home/haseo/working/linux/
PWD=$(shell pwd)
TARGET=arm
TOOLCHAIN=arm-linux-gnueabihf-
APP_NAME=digit_app2

obj-m:=$(MODULE_NAME).o

all: dd app

dd:
    $(MAKE) -C $(KDIR) M=$(PWD) ARCH=$(TARGET) CROSS_COMPILE=$(TOOLCHAIN) modules

app:
    arm-linux-gnueabihf-gcc -o $(APP_NAME) $(APP_NAME).c

clean:
    rm -f *.ko
    rm -f *.o
    rm -f *.mod.*
    rm -f modules.order
    rm -f Module.symvers
    rm -f .*.cmd
    rm -rf .tmp_versions
    rm -f $(APP_NAME)
```

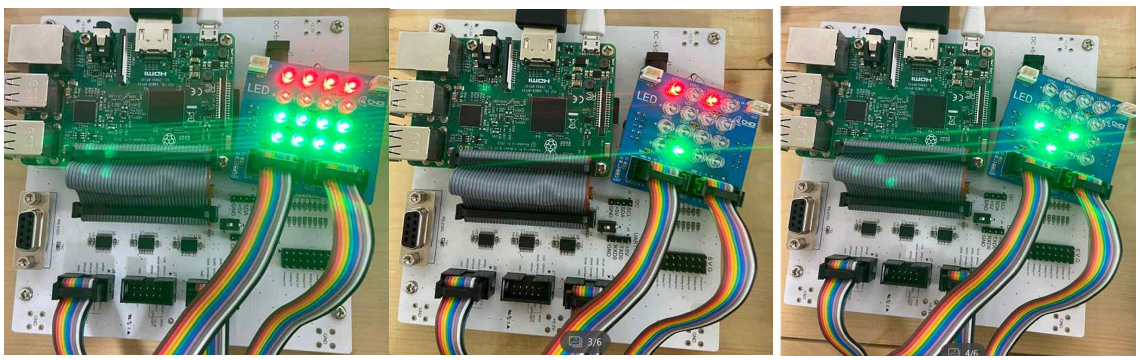
- Insert Device Driver Module && Make Device File



- Execute Device Driver Application && Check Kernel Message

모든 LED를 OFF로 초기화했음에도 불구하고, LED15는 계속 ON 상태인 문제가 있습니다. 이는 이전 과제에서 발생한 문제와 동일한 하드웨어 문제인 것 같습니다.





- Remove Device Driver Module && Remove Device File