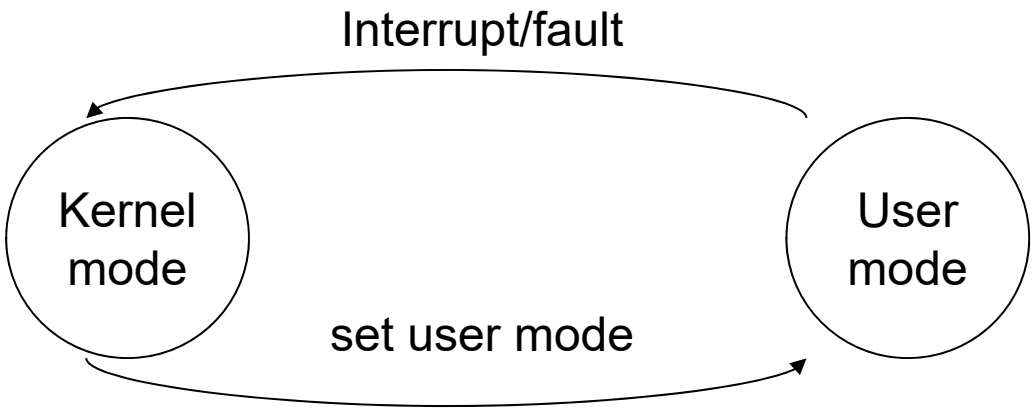
Kernel Programming (2)
Device Driver

조 진 성
경희대학교 컴퓨터공학과
Mobile & Embedded System Lab.

I/O Protection

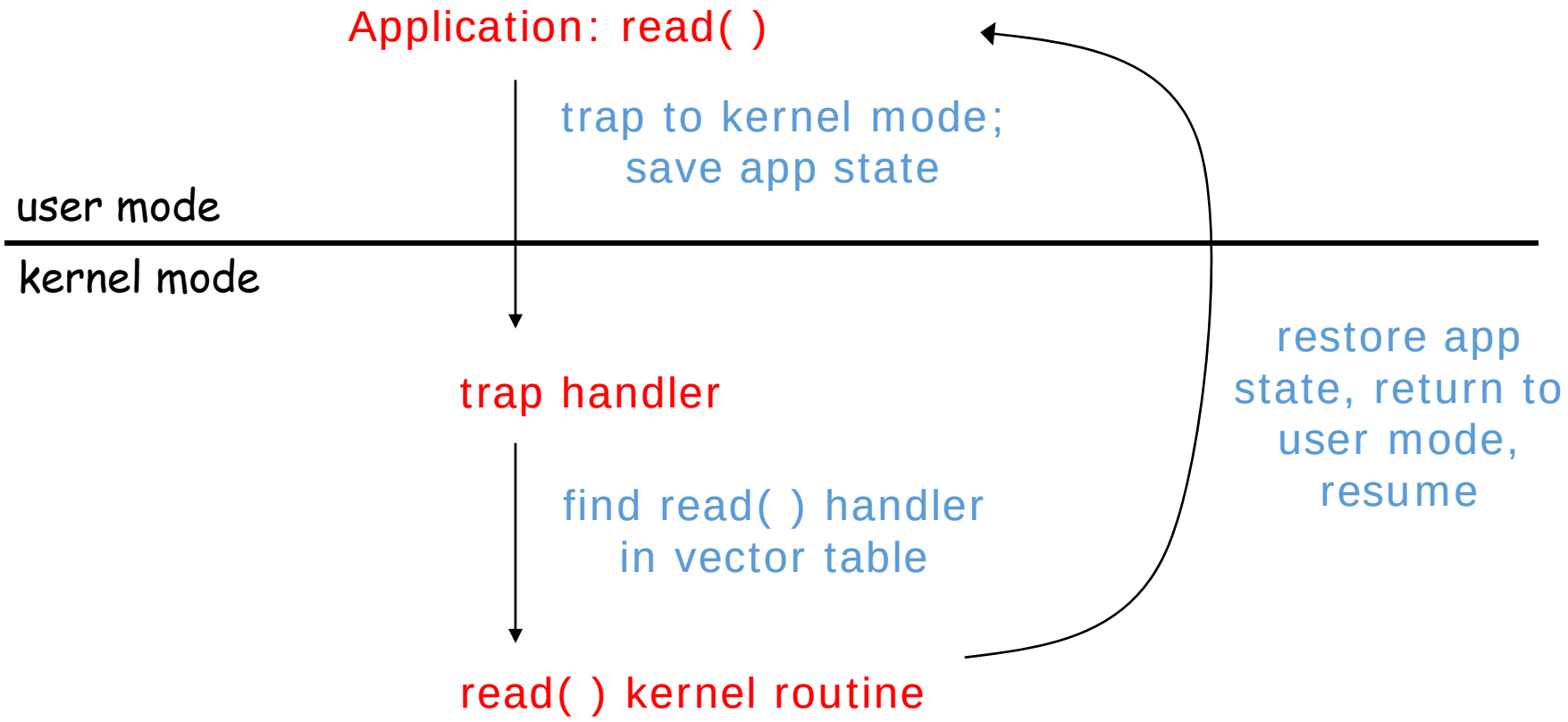
- **Dual mode operation (CPU)**



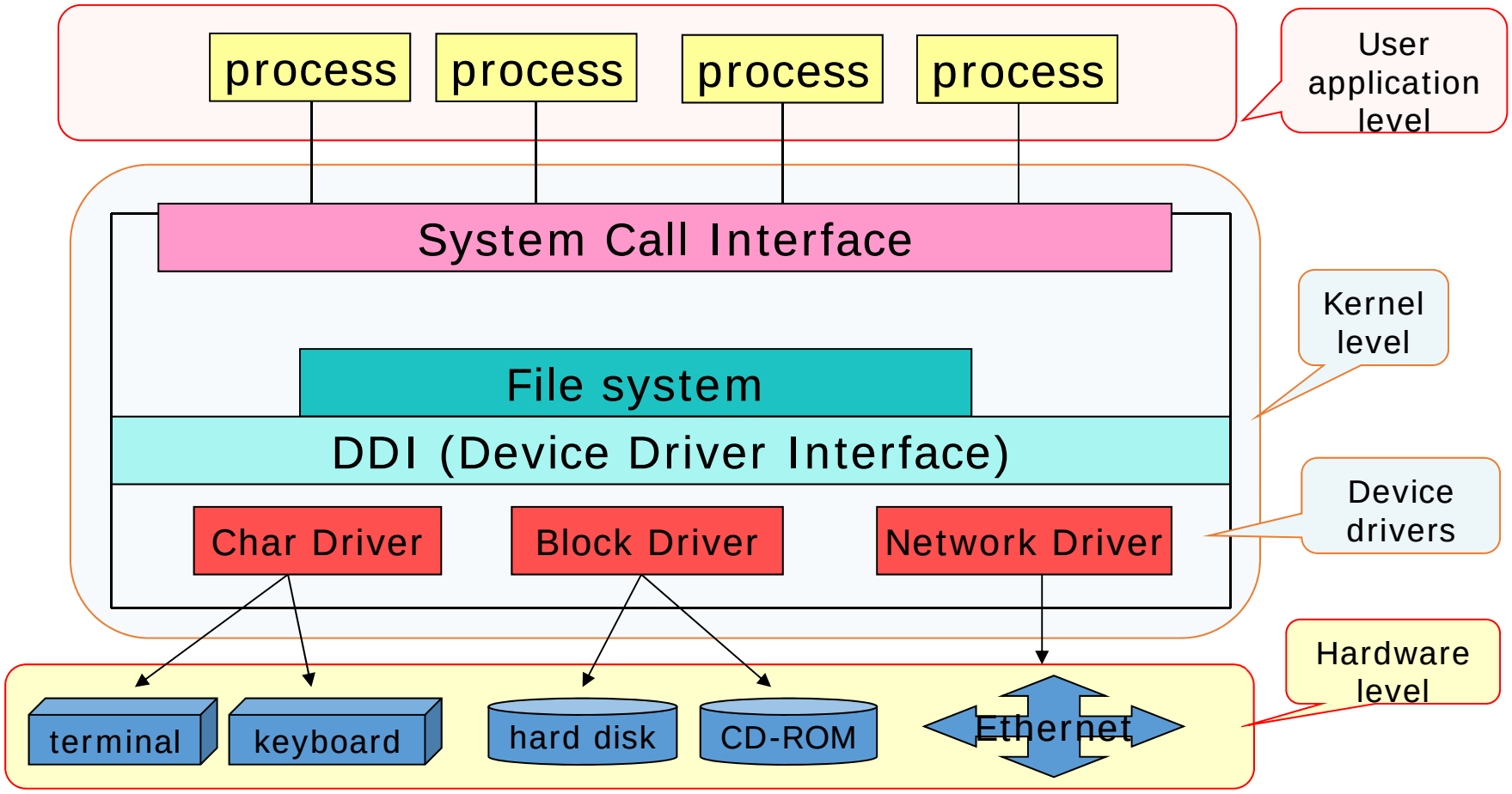
Privileged instructions can be issued only in kernel mode

How about Lab. 2) ? (Wiring Pi & /dev/mem)

I/O Protection



Device Control in Linux



Device Control in Linux

- **Device driver**
  - 특정 디바이스를 제어하는 커널 소프트웨어
  - 응용 프로그램은 특정 하드웨어에 대한 제어를 커널에 요청하고, 커널은 해당 디바이스 드라이버를 호출하여 처리
  - 응용 프로그램은 file I/O system call을 통하여 디바이스 드라이버에 요청
  - 응용 프로그램은 special device file을 통하여 특정 디바이스를 지정

- **File I/O system calls**
  - open( )
  - close( )
  - read( )
  - write( )
  - lseek( )
  - ioctl( )

Device Control in Linux

❖ Special device file

- Linux(Unix)는 시스템의 디바이스를 파일로 접근
  - 램, 키보드, 하드디스크도 파일로 표현
- 이런 파일들은 /dev/ 디렉토리에 존재하며, special device file이라고 함
  - 디바이스 파일 하나 하나는 실질적인 하드웨어를 표현
    - ex. 마우스는 /dev/mouse라는 디바이스 파일로 표현
  - Regular file의 목적이 데이터를 저장하는데 있다면, 이들은 디바이스에 대한 정보를 제공
    - 이 정보는 "**디바이스 타입 정보, 주 번호, 부 번호**"

❖ Special device file 생성

- mknod

```
# mknod example
root@ubuntu:~ # mknod /dev/ttyS4 c 4 68
```

  - 디렉토리: /dev/
  - 디바이스 파일명: ttyS4
  - 문자 디바이스 드라이버
  - 주 번호: 4
  - 부 번호: 68

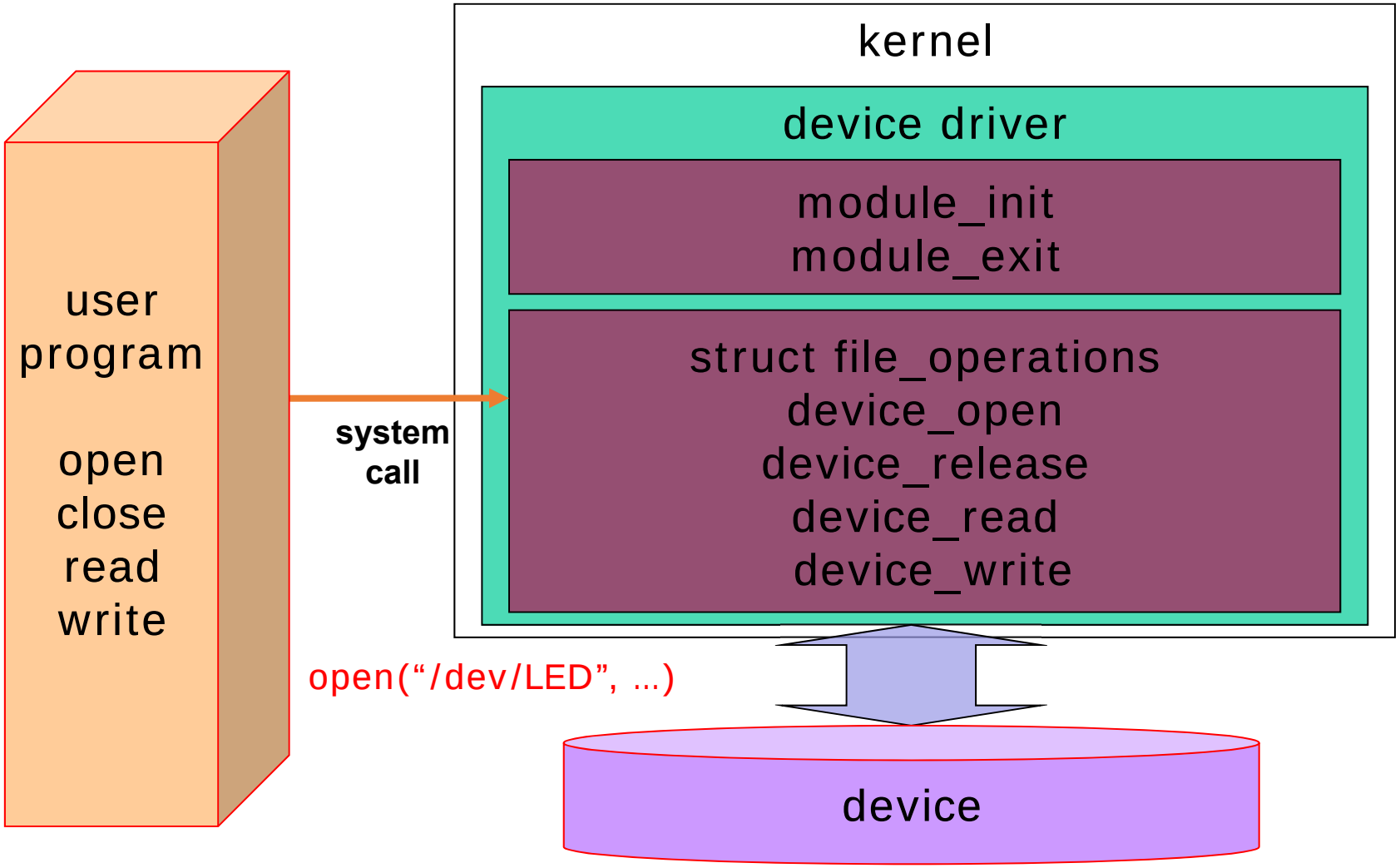
Device Control in Linux

❖ 주 번호와 부 번호

- 주 번호 (major number)
  - 커널에서 디바이스 드라이버를 구분하고 연결하는데 사용
  - 주 번호는 제어하려는 디바이스를 구분하기 위한 디바이스의 ID
- 부 번호 (minor number)
  - 디바이스 드라이버 내에서 장치를 구분하기 위해 사용
  - 같은 종류의 디바이스가 여러 개 있을 때 그 중 하나를 선택하기 위해 사용

```
root@ubuntu:~ # ls –al  /dev/ttyS*
crw------   1    root  root  4,    64   Feb   7    2018       /dev/ttyS0
crw------   1    root  root  4,    65   Feb   7    2018       /dev/ttyS1
crw------   1    root  root  4,    66   Feb   7    2018       /dev/ttyS2
crw------   1    root  root  4,    67   Feb   7    2018       /dev/ttyS3
```

Device Control in Linux



Device Control in Linux

Kernel vs. module

- 리눅스 커널이 부팅된 후 커널 모듈을 동적으로 추가하거나 제거할 수 있음
- 디바이스 드라이버의 경우 많은 경우 모듈로 동작함

모듈 프로그래밍

- 모듈은 커널 프로그램의 특징을 갖고 있으면서 커널에 동적으로 적재되고 제거되므로 일반 프로그램과는 다른 형식을 갖추어야 함
- 디바이스 드라이버의 경우, 리눅스 커널의 DDI(Device Driver Interface)에 준하여 작성되어야 함

모듈 관련 명령어

- insmod : 모듈을 커널에 적재
- rmmod : 커널에서 모듈을 제거
- lsmod : 커널에 적재된 모듈 목록을 출력
- depmod : 모듈간 의존성 정보를 생성
- modprobe : 모듈을 커널에 적재하거나 제거

Device Control in Linux

❖ 디바이스 드라이버 작성 요령

- struct file_operations 구성
  - open, release, read, write

Operation	설명
struct module *owner	어떤 모듈로 올라간 디바이스 드라이버와 연관을 가지는지를 나타내는 포인터
lseek	현재의 읽기 쓰기 포인터를 옮기고자 할 때 사용
read /write	데이터를 장치에서 읽거나 쓰고자 할 때 사용
readdir	디렉터리에 대해서만 사용되며, 디바이스에 대해서는 NULL값을 가짐
select	디바이스가 읽기나 쓰기, 혹은 예외 상황을 일으켰는지를 확인할 때 사용
ioctl	특정 디바이스에 의존적인 명령을 수행하고자 할 때 사용
mmap	디바이스의 메모리 일부를 현재 프로세스의 메모리 영역으로 매핑하고자 할 때 사용
open / release	디바이스에 대한 열기/닫기를 할 때 사용
fsyn	디바이스에 대한 모든 연산의 결과를 지연하지 않고 즉시 일어나도록 할 때 사용
fasync	FASYNC Flag의 변화를 디바이스에 알리기 위해서 사용
readv / writev	BSD 형식의 read/write를 지원하기 위한것

- module_init에서 디바이스 드라이버를 커널에 등록 (struct file_operations)
- module_exit에서 디바이스 드라이버를 커널에서 해제

Device Drivers

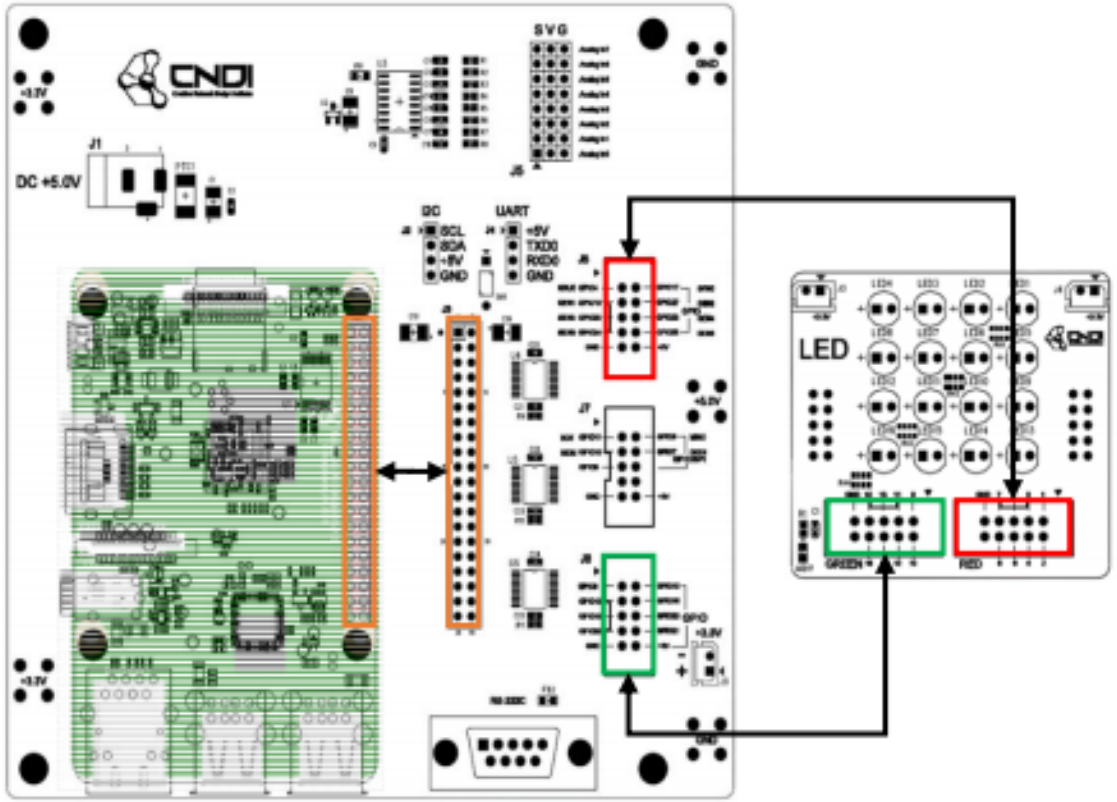
Development environment

- Host PC
  - Ubuntu 16.04 LTS
- Device
  - Board
    - Raspberry Pi 3 Model B V1.2
  - CPU
    - Hardware: BCM2837
    - Model name: ARM Cortec A53 (ARMv8)
  - Kernel
    - 4.4.50-v7+
  - OS
    - 2016-05-27-raspbian-jessie

❖ 하드웨어 구성

▪ Raspberry Pi3와 LED 모듈 결선

- Raspberry Pi3와 Raspberry Pi Adapter 연결 (20x2 케이블)
- Raspberry Pi Adapter의 포트 J6와 LED 모듈의 포트 RED 연결 (10핀 케이블)
- Raspberry Pi Adapter의 포트 J8과 LED 모듈의 포트 GREEN 연결 (10핀 케이블)



❖ Application coding (LED 1, 2, 3, 4 제어)

```
# Change Directory
ubuntu@ubuntu:~ $ mkdir working/driver; cd working/driver

# Device Driver Application [digit_app.c]
ubuntu@ubuntu:~/working/driver $ vi digit_app.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 != 2) {
    printf("Usage: %s [LED binary]\n", argv[0]);
    exit(1);
  }

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

  write(fd, argv[1], strlen(argv[1]));

  close(fd);

  return 0;
}
```

❖ Device driver coding

 # Device Driver [led_dd.c]

ubuntu@ubuntu:~/working/driver $ vi led_dd.c

```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
};
```

```c

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();

```

Device Drivers

❖ Device driver coding

 # Device Driver [led_dd.c]

ubuntu@ubuntu:~/working/driver $ vi led_dd.c

```c
  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;

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

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

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

  return len;
}

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

static int led_init(void) {
  printk("[led_dd] led_init()\n");
  register_chrdev(MAJOR_NUMBER, DEVICE_NAME, &fops);

  return 0;
}

static void led_exit(void) {
  printk("[led_dd] led_exit()\n");
  unregister_chrdev(MAJOR_NUMBER, DEVICE_NAME);
}

module_init(led_init);
module_exit(led_exit);

MODULE_LICENSE("Dual BSD/GPL");
```

Device Drivers

❖ Compile Device driver & Application

```
# Makefile
ubuntu@ubuntu:~/working/driver $ vi Makefile
```

```
1  MODULE_NAME=led_dd
2  KDIR=/home/ubuntu/working/linux/
3  PWD=$(shell pwd)
4  TARGET=arm
5  TOOLCHAIN=arm-linux-gnueabihf-
6  APP_NAME=digit_app
7
8  obj-m:=$(MODULE_NAME).o
9
10 all: dd app
11
12 dd:
13   $(MAKE) -C $(KDIR) M=$(PWD) ARCH=$(TARGET) CROSS_COMPILE=$(TOOLCHAIN) modules
14
15 app:
16   arm-linux-gnueabihf-gcc -o $(APP_NAME) $(APP_NAME).c
17
18 clean:
19   rm -f *.ko
20   rm -f *.o
21   rm -f *.mod.*
22   rm -f modules.order
23   rm -f Module.symvers
24   rm -f .*.cmd
25   rm -rf .tmp_versions
26   rm -f $(APP_NAME)
```

Compile Device driver & Application

```
# Make
ubuntu@ubuntu:~/working/driver $ make
```

```
make -C /home/sauber92/working/linux/ M=/home/sauber92/working/driver ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules
make[1]: Entering directory '/home/sauber92/working/linux'
  CC [M]  /home/sauber92/working/driver/led_dd.o
/home/sauber92/working/driver/led_dd.c:42:22: warning: function definition has qualified void return type
 static volatile void initGpioAddr(void) {
                      ^
/home/sauber92/working/driver/led_dd.c: In function ‘led_open’:
/home/sauber92/working/driver/led_dd.c:79:2: warning: function with qualified void return type called
  initGpioAddr();
  ^
/home/sauber92/working/driver/led_dd.c: In function ‘led_write’:
/home/sauber92/working/driver/led_dd.c:105:3: warning: ignoring return value of ‘copy_from_user’, declared with attribute warn_unused_result [-Wunused-result]
   copy_from_user(&state, &buf[i], 1);
   ^
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/sauber92/working/driver/led_dd.mod.o
  LD [M]  /home/sauber92/working/driver/led_dd.ko
make[1]: Leaving directory '/home/sauber92/working/linux'
arm-linux-gnueabihf-gcc -o digit_app digit_app.c
```

```
# Check device driver & application
ubuntu@ubuntu:~/working/driver $ ls -l
```

```
total 52
-rwxrwxr-x 1 sauber92 sauber92 8476  5월 28 14:27 digit_app
-rw-rw-r-- 1 sauber92 sauber92  468  5월 28 13:36 digit_app.c
-rw-rw-r-- 1 sauber92 sauber92 2796  5월 28 14:18 led_dd.c
-rw-rw-r-- 1 sauber92 sauber92 6196  5월 28 14:27 led_dd.ko
-rw-rw-r-- 1 sauber92 sauber92 1154  5월 28 14:27 led_dd.mod.c
-rw-rw-r-- 1 sauber92 sauber92 2636  5월 28 14:27 led_dd.mod.o
-rw-rw-r-- 1 sauber92 sauber92 5124  5월 28 14:27 led_dd.o
-rw-rw-r-- 1 sauber92 sauber92  458  5월 28 14:06 Makefile
-rw-rw-r-- 1 sauber92 sauber92   47  5월 28 14:27 modules.order
-rw-rw-r-- 1 sauber92 sauber92    0  5월 28 14:27 Module.symvers
```

❖ Transfer Device driver & Application (*Insert SD Card*)

```
# Check SD card
ubuntu@ubuntu:~/working/driver $ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sdb      8:16   1 14.9G  0 disk
├─sdb2   8:18   1 14.8G  0 part
└─sdb1   8:17   1   63M  0 part
sr0     11:0    1 1024M  0 rom
sda      8:0    0   30G  0 disk
├─sda2   8:2    0    1K  0 part
├─sda5   8:5    0    2G  0 part [SWAP]
└─sda1   8:1    0   28G  0 part /

# Mount SD card
ubuntu@ubuntu:~/working/driver $ sudo  mount  /dev/sdc2  /mnt/fs

# Copy Device driver & Application
ubuntu@ubuntu:~/working/driver $ mkdir  /mnt/fs/home/pi/working
ubuntu@ubuntu:~/working/driver $ cp  led_dd.ko  /mnt/fs/home/pi/working
ubuntu@ubuntu:~/working/driver $ cp  digit_app  /mnt/fs/home/pi/working

# Unmount SD card
ubuntu@ubuntu:~/working/driver $ sudo  umount /mnt/fs
```

❖ Insert Device driver

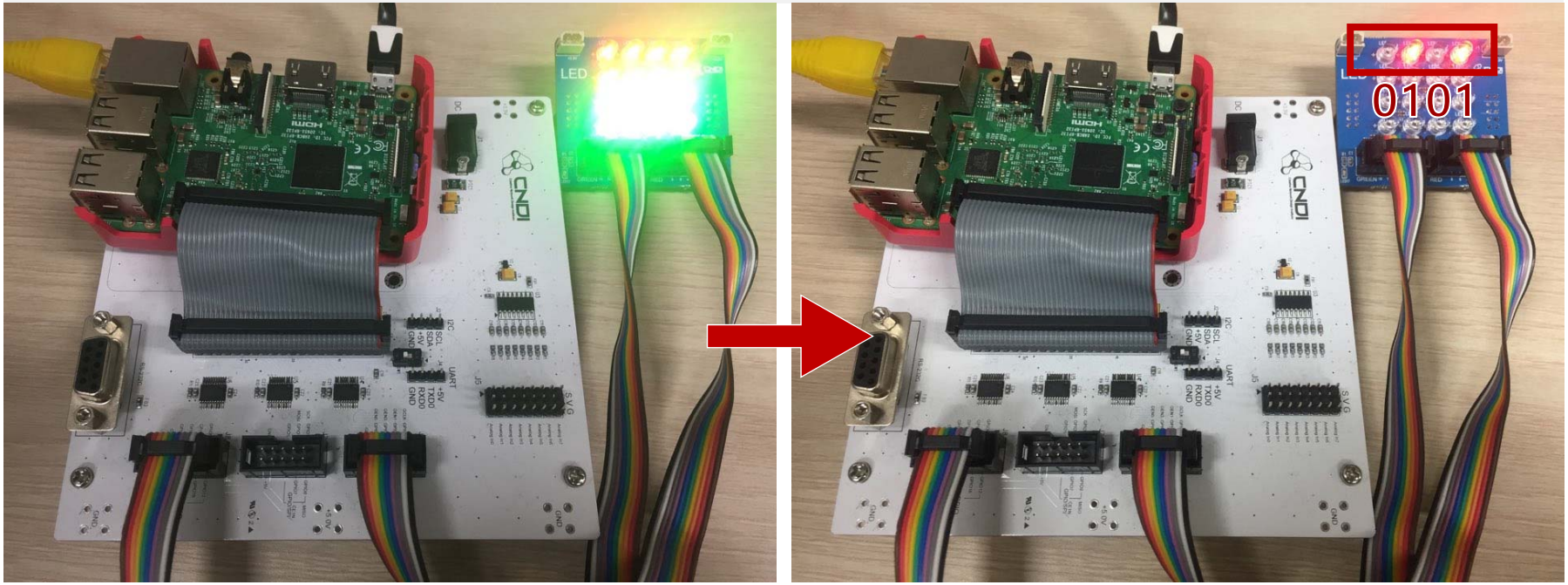
```
# Device Driver [led_dd.c]
pi@raspberry:~/working/driver $ ls -l
total 20
-rwxr-xr-x 1 pi pi 8476 May 28 05:10 digit_app
-rw-r--r-- 1 pi pi 6196 May 28 05:10 led_dd.ko

# Insert module
pi@raspberry:~/working/driver $ sudo insmod led_dd.ko
pi@raspberry:~/working/driver $ lsmod | grep led_dd
led_dd                  2160  0

# Make device file
# mknod /dev/[device_name]  c  [major number]  [minor number]
pi@raspberry:~/working/driver $ sudo mknod /dev/led_dd c 222 0
pi@raspberry:~/working/driver $ sudo chmod 666 /dev/led_dd
pi@raspberry:~/working/driver $ ls -l /dev/led_dd
crw-rw-rw- 1 root root 222, 0 May 28 05:12 /dev/led_dd
```

Execute Application

```
# Device Driver Application [led_test.c]
pi@raspberry:~/working $ ./digit_app  0101
```



Kernel message

```
# Check kernel message
pi@raspberry:~/working/application $ dmesg
[66112.042162] [led_dd] led_init()
[66125.303483] [led_dd] led_open
[66128.353709] [led_dd] led_write
[66128.353756] [led_dd] led_release
```

```
# Remove device driver
pi@raspberry:~/working/application $ sudo rmmod led_dd
pi@raspberry:~/working/application $ dmesg
[66112.042162] [led_dd] led_init()
[66125.303483] [led_dd] led_open
[66128.353709] [led_dd] led_write
[66128.353756] [led_dd] led_release
[66186.126464] [led_dd] led_exit()
```

```
# Remove device file
# rm –rf  /dev/[device_name]  c  [major number]  [minor number]
pi@raspberry:~/working/application $ sudo rm -rf /dev/led_dd c 222 0
```

실습 과제

- 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

- 해당 디바이스 드라이버를 이용하여 Red/Green LED 제어 application 작성
  - ./digit_app2 0 1010 (Red 출력)
  - ./digit_app2 1 1010 (Green 출력)

- Hint
  - static int led_ioctl(struct inode *inode, struct file *filep, unsigned int cmd, unsigned long arg) 에서 LED 색(번호) 설정
  - static int led_write( ) 에서 LED 색(번호) 설정에 따라 출력

Q & A

http://mesl.khu.ac.kr