Lab 2. Light Control

20170275 박정웅 20170616 정희진

1. Purpose

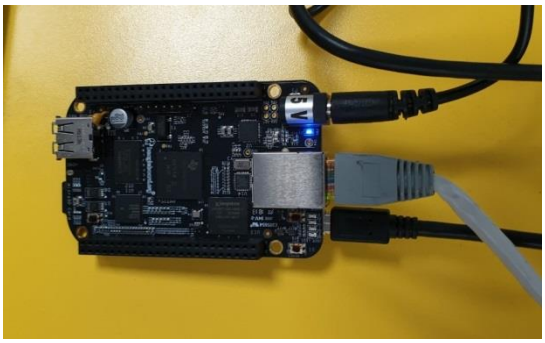
LED를 제어하기 위한 여러가지 방법에 대하여 배운다.

- Beaglebone 내의 LED를 sysfs을 사용하여 제어하여 본다.
- hard-wired LED를 제어하기 위하여 회로도를 설계하고 shell script, c program, linux module 등을 사용하여 제어한다.

2. Experiment Procedure

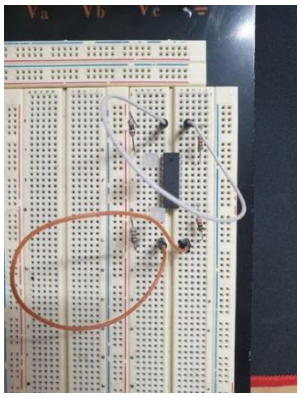
1) Prepare

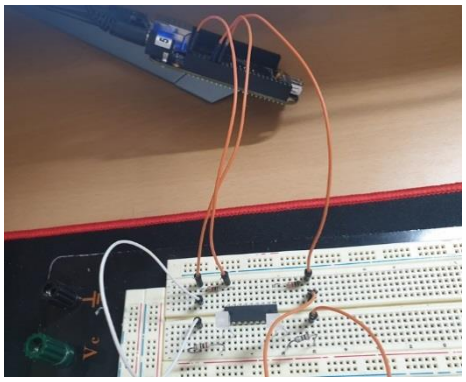
(1) Beaglebone connection



Beaglebone에는 5V adapter로 전원을 공급하고, 인터넷 케이블선을 router과 연결한다. USB선의 경우에는 컴퓨터와 연결한다. 컴퓨터 역시 인터넷 케이블선을 사용하여 router과 연결한다.

(2) Circuit design





Transistor array로는 ULN2803A를 사용하였고, LED는 CP41B-WES-CK0P0154 2개를 사용하였다. Beaglebone의 GPIO pin은 높은 voltage에 매우 취약하기 때문에 저항을 사용하여 pin에 흐르는 전류의 크기를 낮추어야 한다. 따라서 이번 실험에서는 5kΩ 저항을 GPIO output과 external logic input 사이에 연결하였다. 이 경우, GPIO output이 3.3V 임으로 흐르는 전류의 크기를 0.66mA까지 낮출 수 있다.

총 2개의 LED를 사용하는데, 각각의 (-)부분은 ULN2803A의 8C, 1C 부분에 연결하였다. 그 후 (+)부분은 200Ω 저항에 연결하고 이 저항은 다시 5kΩ저항과 연결된다. 5kΩ 저항은 각각 8B, 1B pin과 연결된다. 앞으로 위 사진에서 흰색 선과 연결된 LED를 Light1, 주황색 선과 연결된 LED를 Light2라고 명명한다. Light1은 GPIO30으로 연결되고, Light2는 GPIO31으로 연결된다. 마지막으로 ULN2803A의 GND를 beaglebone의 GND와 연결하면 된다.

2) Problem 2A: User LED0 control using sysfs and command line and Lights Control Commands

(1) Control usr0 by sysfs and command line

관리자 root 권한이 필요하여 "su" 커맨드를 친 후, "/sys/class/leds" 폴더를 확인하였다. 이 폴더 안에서 4개의 LED를 제어하기 위한 폴더 4개를 확인하였다.

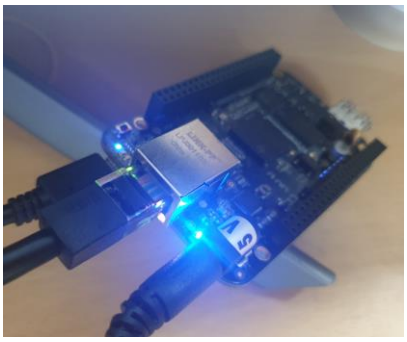
```
jungwungpark@beaglebone:~$ su
Password:
root@beaglebone:/home/jungwungpark# ls -F /sys/class
arvo/       hwmon/        mdio_bus/      regulator/     ubi/
backlight/  i2c-adapter/  mem/           rtc/           udc/
bdi/        i2c-dev/      misc/          savu/          uio/
block/      input/        mmc_host/      scsi_device/   usbmon/
bsg/        isku/         mtd/           scsi_disk/     vc/
dma/        kone/         net/           scsi_host/     video4linux/
drm/        koneplus/     power_supply/  sound/         virtio-ports/
dvb/        kovaplus/     pps/           spidev/        vtconsole/
firmware/   lcd/          ptp/           spi_master/    watchdog/
gpio/       leds/         pwm/           thermal/
graphics/   logibone/     pyra/          timed_output/
hidraw/     mbox/         rc/            tty/
root@beaglebone:/home/jungwungpark# ls -F /sys/class/leds
beaglebone:green:usr0@  beaglebone:green:usr2@
beaglebone:green:usr1@  beaglebone:green:usr3@
```

usr0 LED를 제어하기 위해 "beaglebone₩:green₩:usr0" 폴더로 들어가 보았다. 여기서 LED를 제어할 다양한 파일을 확인하였다.

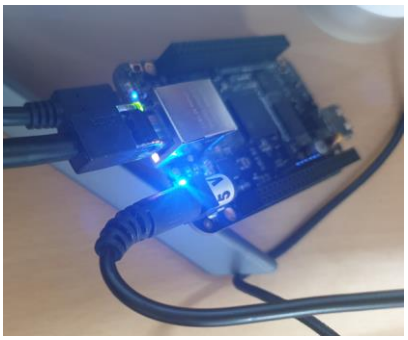
```
root@beaglebone:/home/jungwungpark# cd /sys/class/leds/
root@beaglebone:/sys/class/leds# cd beaglebone\:green\:usr0
root@beaglebone:/sys/class/leds/beaglebone:green:usr0# ls -F
brightness  device@  max_brightness  power/  subsystem@  trigger  uevent
```

"trigger" 파일을 확인하고 "hearbeat" 값으로 되어있는 것을 확인하였다. Beaglebone의 usr0 LED가 심장박동처럼 불이 켜졌다 꺼지는 것을 반복하고 있었다. "echo none > trigger" 커맨드를 통해 "none" 값으로 만들었더니 usr0 LED가 꺼졌다. 이 때, command line을 보면 알 수 있듯이 heartbeat가 켜질 때는 [heartbeat], 꺼져 있을 때는 [none]으로 terminal에 표시되는 것을 확인 할 수 있다.

```
root@beaglebone:/sys/class/leds/beaglebone:green:usr0# cat trigger
none nand-disk mmc0 mmc1 timer oneshot [heartbeat] backlight gpio cpu0 default-on transient
root@beaglebone:/sys/class/leds/beaglebone:green:usr0# echo none > trigger
root@beaglebone:/sys/class/leds/beaglebone:green:usr0# cat trigger
[none] nand-disk mmc0 mmc1 timer oneshot heartbeat backlight gpio cpu0 default-on transient
```



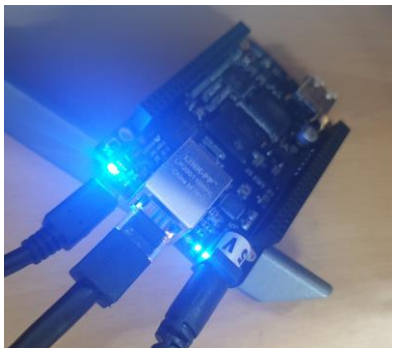
<heartbeat>



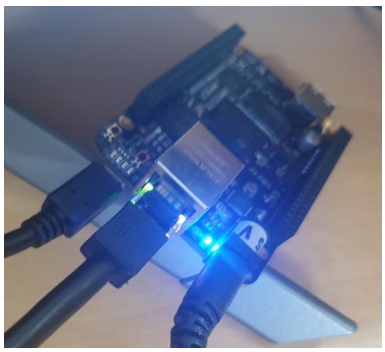
<none>

다음으로 "brightness" 파일에 값 1과 0을 차례로 넣어보았다. 1을 넣었을 때 Beaglebone의 usr0 LED가 밝게 빛났고, 0을 넣었을 때는 LED가 꺼졌다.

```
root@beaglebone:/sys/class/leds/beaglebone:green:usr0# echo 1 > brightness
root@beaglebone:/sys/class/leds/beaglebone:green:usr0# echo 0 > brightness
```



<brightness is 1>



<brightness is 0>

다음으로 "trigger" 파일에 "timer" 값을 넣어주었다. 먼저 echo timer > trigger 명령어를 사용하면 불빛이 heartbeat와 달리 일정한 시간으로 켜졌다 꺼지는 것을 확인할 수 있었다. "delay_on" 파일에 100, "delay_off" 파일에 900을 넣어주었더니 beaglebone의 usr0 LED가 짧게 켜졌다가 꺼지는 것을 반복하였다.

```
root@beaglebone:/sys/class/leds/beaglebone:green:usr0# echo timer > trigger
root@beaglebone:/sys/class/leds/beaglebone:green:usr0# echo 100 > delay_on
root@beaglebone:/sys/class/leds/beaglebone:green:usr0# echo 900 > delay_off
```

사진으로는 불빛이 켜진 것과 꺼진 것 만을 확인할 수 있음으로 timer가 작동하는 것을 사진만으로는 확인할 수 없다. 동영상을 따로 첨부할 수 없음으로 timer의 행동을 설명만 하고 넘어가도록 하겠다.

"timer" 값으로 바뀐 것을 "heartbeat"로 바꾸니 다시 심장박동처럼 켜졌다 꺼지는 것을 반복하였고 "none"으로 바꾸니 LED가 꺼졌다.

```
root@beaglebone:/sys/class/leds/beaglebone:green:usr0# echo heartbeat > trigger
```

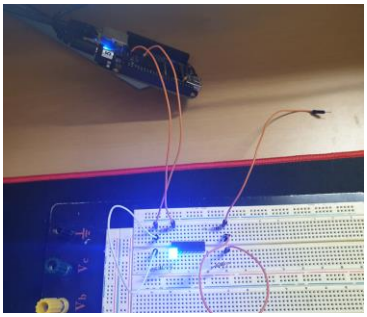
```
root@beaglebone:/sys/class/leds/beaglebone:green:usr0# echo none > trigger
```

Usr2 LED를 제어하는 폴더로 가 마찬가지로 "trigger" 파일에 "none" 값을 넣어주니 usr2 LED가 꺼졌다.

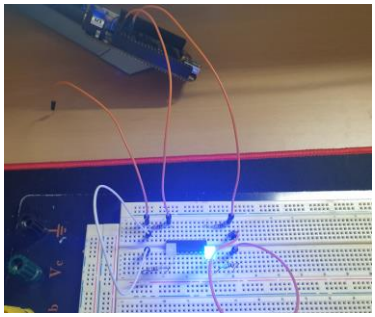
```
root@beaglebone:/sys/class/leds/beaglebone:green:usr0# cd ..
root@beaglebone:/sys/class/leds# cd beaglebone\:green\:usr2
root@beaglebone:/sys/class/leds/beaglebone:green:usr2# echo none > trigger
```

(2) Circuit test

회로에 연결된 2개의 LED를 Beaglebone의 5V pin에 각각 꽂으니 불이 들어왔다. 회로와 장비가 정상적으로 작동하는 것을 확인하였다.



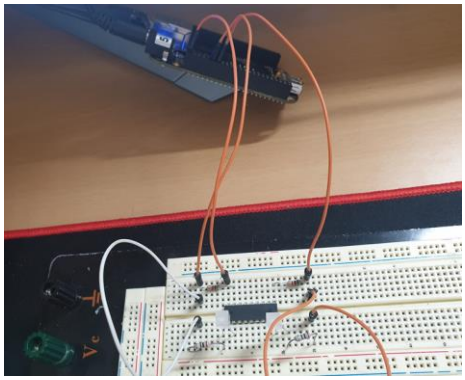
<Light1>



<Light2>

(3) Control light Control by sysfs and command line

2개의 LED를 각각 Beaglebone의 GPIO30 pin과 GPIO31 pin에 연결하였다.



다음으로 터미널창에서 "/sys/class/gpio" 폴더를 확인하였다.

```
root@beaglebone:/sys/class/leds/beaglebone:green:usr2# ls -F /sys/class/gpio/
export  gpiochip0@  gpiochip32@  gpiochip64@  gpiochip96@  unexport
```

위의 파일 경로로 이동하여 "export" 파일에 30을 넣어주었다. Beaglebone의 GPIO30 pin을 제어하기 위한 폴더가 생성되었다.

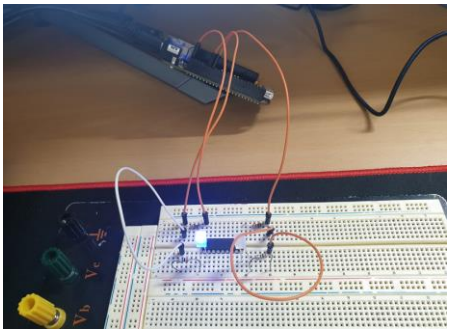
```
root@beaglebone:/sys/class/leds/beaglebone:green:usr2# cd /sys/class/gpio/
root@beaglebone:/sys/class/gpio# echo 30 > export
root@beaglebone:/sys/class/gpio# ls -F
export  gpio30@  gpiochip0@  gpiochip32@  gpiochip64@  gpiochip96@  unexport
```

다음으로 "gpio 30" 폴더로 이동하여 "direction"을 "out"으로 만들어 주었다.

```
root@beaglebone:/sys/class/gpio# cd gpio30
root@beaglebone:/sys/class/gpio/gpio30# echo out > direction
root@beaglebone:/sys/class/gpio/gpio30# cat direction
out
```

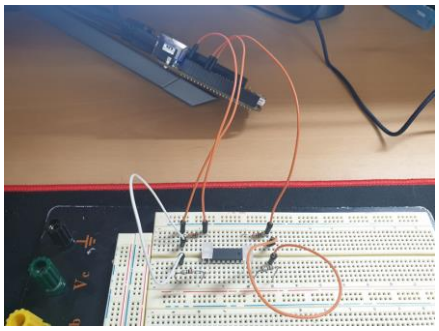
다음으로 "value"를 1로 만들어 주니 GPIO 30 pin과 연결된 Light1이 켜졌다.

```
root@beaglebone:/sys/class/gpio/gpio30# echo 1 > value
root@beaglebone:/sys/class/gpio/gpio30# cat value
1
```



"value"를 0로 만들어 주니 Light1이 다시 꺼졌다.

```
root@beaglebone:/sys/class/gpio/gpio30# echo 0 > value
root@beaglebone:/sys/class/gpio/gpio30# cat value
0
```



다시 상위폴더로가 GPIO30 pin 제어 폴더를 제거하였다.

```
root@beaglebone:/sys/class/gpio/gpio30# cd ..
root@beaglebone:/sys/class/gpio# echo 30 > unexport
root@beaglebone:/sys/class/gpio# ls -F
export  gpiochip0@  gpiochip32@  gpiochip64@  gpiochip96@  unexport
```

이번에는 GPIO 31 pin 과 연결된 LED를 보기 위해 GPIO 31 제어 폴더를 생성했다.

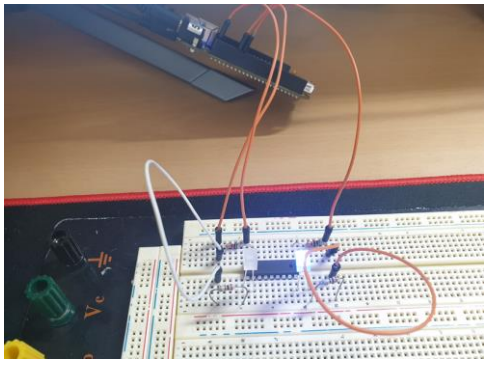
```
root@beaglebone:/sys/class/gpio# echo 31 > export
root@beaglebone:/sys/class/gpio# ls -F
export  gpio31@  gpiochip0@  gpiochip32@  gpiochip64@  gpiochip96@  unexport
```

해당 폴더로 들어가 "direction"을 "out"으로 바꿨다.

```
root@beaglebone:/sys/class/gpio# cd gpio31
root@beaglebone:/sys/class/gpio/gpio31# echo out > direction
root@beaglebone:/sys/class/gpio/gpio31# cat direction
out
```

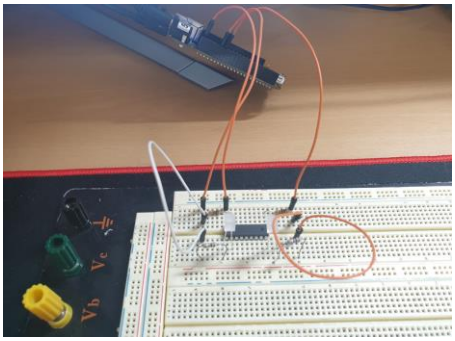
"value"를 1로 만들었더니 GPIO 31 pin과 연결된 Light2가 켜졌다.

```
root@beaglebone:/sys/class/gpio/gpio31# echo 1 > value
root@beaglebone:/sys/class/gpio/gpio31# cat value
1
```



"value"를 0로 만들어 주니 Light2이 다시 꺼졌다.

```
root@beaglebone:/sys/class/gpio/gpio31# echo 0 > value
root@beaglebone:/sys/class/gpio/gpio31# cat value
0
```



실험을 마친 뒤 다시 상위폴더로가 GPIO31 pin 제어 폴더를 제거하였다.

```
root@beaglebone:/sys/class/gpio/gpio31# cd ..
root@beaglebone:/sys/class/gpio# echo 31 > unexport
root@beaglebone:/sys/class/gpio# ls -F
export  gpiochip0@  gpiochip32@  gpiochip64@  gpiochip96@  unexport
```

3) Problem 2B: Light Control Shell Script

(1) Start NFS

PC에서 "sudo /etc/init.d/nfs-kernel-server start" 커맨드를 치고 NFS server를 시작하였다.

```
jungwung@ubuntu:~$ sudo /etc/init.d/nfs-kernel-server start
[sudo] password for jungwung:
[ ok ] Starting nfs-kernel-server (via systemctl): nfs-kernel-server.service.
jungwung@ubuntu:~$
```

Beaglebone에서 "./bone_nfs_client.sh"을 치고 NFS client를 시작하였다.

```
jungwungpark@beaglebone:~$ ./bone_nfs_client.sh
[sudo] password for jungwungpark:
Mount nfs_client in Bone Ubuntu ~/nfs_client
```

관리자 root 권한이 필요하여 "su" 커맨드를 친 후, 작업 위치를"b_GPIO_LED_Shell" 폴더가 있는 곳으로 옮겼다. b_GPIO_LED_Shell에는 ui_control_lights.sh, loop_control_lights.sh 2개의 shell script가 있다.

```
jungwungpark@beaglebone:~$ su
Password:
root@beaglebone:/home/jungwungpark# ls
bone_nfs_client.sh  nfs_client  test_scp
```

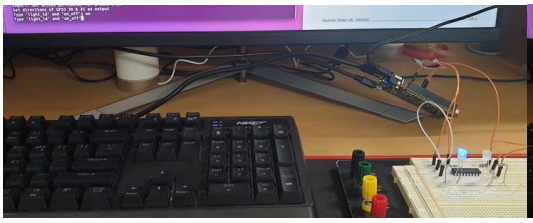
```
root@beaglebone:/home/jungwungpark# cd ./nfs_client/2_LightControl/b_GPIO_LED_Shell/
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/b_GPIO_LED_Shell# pwd
/home/jungwungpark/nfs_client/2_LightControl/b_GPIO_LED_Shell
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/b_GPIO_LED_Shell#
```

(2) Control LED with Shell Script (ui_control_lights, loop_control_lights)

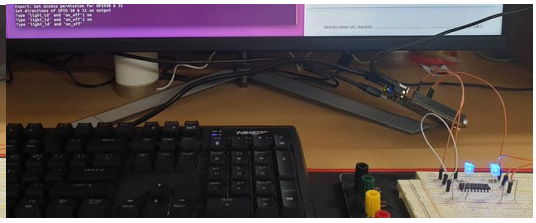
"chmod a+x" 커맨드를 이용하여 "ui_control_lights.sh" 파일이 실행 가능하도록 만들고 실행했다. 다음 그림과 같이 input을 하였을 때, LED의 빛이 꺼졌다가 켜지는 것을 확인하였다.

```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/b_GPIO_LED_Shell# chmod a+x ui_control_lights.sh
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/b_GPIO_LED_Shell# ./ui_control_lights.sh
Control_Lights_Bone.sh
Export: Get access permission for GPIO30 & 31
Set directions of GPIO 30 & 31 as output
Type 'light_id' and 'on_off'1 on
Type 'light_id' and 'on_off'2 on
Type 'light_id' and 'on_off'1 off
Type 'light_id' and 'on_off'2 off
Type 'light_id' and 'on_off'1 non
Error ON_OFF: Please type 'on' or 'off'
Type 'light_id' and 'on_off'2 non
Error ON_OFF: Please type 'on' or 'off'
Type 'light_id' and 'on_off'1 on
Type 'light_id' and 'on_off'2 on
Type 'light_id' and 'on_off'2 off
Type 'light_id' and 'on_off'1 off
Type 'light_id' and 'on_off'0 on
Input light_id is less than 1: Exit
Set directions of GPIO 30 & 31 as input
UnExport: Release access permission for GPIO30 & 31
```

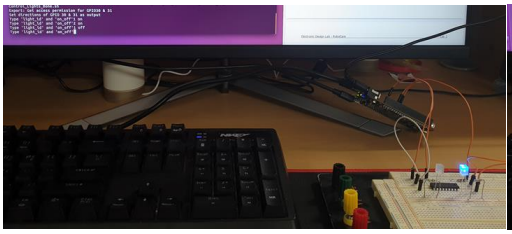
"1 on"을 치면 Light1 이 켜졌고, "2 on"을 치면 Light2가 켜졌다. 반대로 "1 off"을 치면 Light1 이 꺼졌고, "2 off"을 치면 Light2가 꺼졌다. "0 on"을 쳤을 때 shell을 종료하고 나갈 수 있었다. non과 같이 허용되지 않은 command를 입력하면 에러 메시지를 출력하였다.



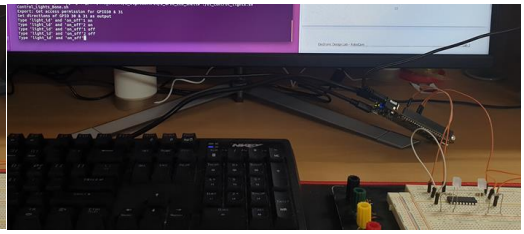
<Type 1 on>



<Type 2 on>



<Type 1 off>



<Type 2 off>

다음으로 두 LED를 10번씩 켰다 껐다 하는 시간을 측정하였다. 위와 마찬가지로 "loop_control_lights.sh" 파일을 실행 가능하게 만들고 3번 실행하였다. 결과는 다음과 같다.

```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/b_GPIO_LED_Shell# chmod a+x loop_control_lights.sh
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/b_GPIO_LED_Shell# ./loop_control_lights.sh
Loop_Control.sh
Export: Get access permission for GPIO30 & 31
Set directions of GPIO 30 & 31 as output
Start:1633589634.892864923*I End:1633589634.937654756
Set directions of GPIO 30 & 31 as input
UnExport: Release access permission for GPIO30 & 31
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/b_GPIO_LED_Shell# ./loop_control_lights.sh
Loop_Control.sh
Export: Get access permission for GPIO30 & 31
Set directions of GPIO 30 & 31 as output
Start:1633589644.647541090*I End:1633589644.692723049
Set directions of GPIO 30 & 31 as input
UnExport: Release access permission for GPIO30 & 31
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/b_GPIO_LED_Shell# ./loop_control_lights.sh
Loop_Control.sh
Export: Get access permission for GPIO30 & 31
Set directions of GPIO 30 & 31 as output
Start:1633589647.841221507*I End:1633589647.886103716
Set directions of GPIO 30 & 31 as input
UnExport: Release access permission for GPIO30 & 31
```

평균적으로 두 LED를 10번 켰다 끄는데 걸린 시간은 0.045초가 소요되었다.

4) Problem 2C: C Program for Two Lights

(1) Compile and run

Problem 2B에서 NFS를 시작하고 있는 상태에서 Beaglebone의 작업 위치를 "c_LightControl_C" 폴더가 있는 곳으로 옮겼다. 그런 다음 해당 폴더에 있는 파일을 확인하였다. (이 때, 이미 컴파일 된 파일들이 있었기 때문에 컴파일 과정은 생략되었다. PC에서 Makefile에 있는 대로 "make"를 쳐서 컴파일 할 수 있다.)

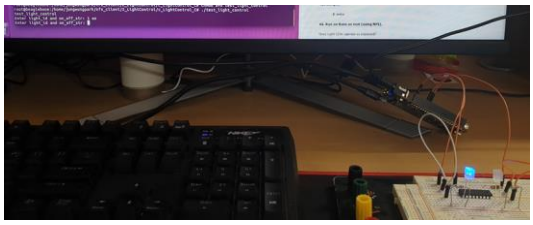
```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl# cd c_LightControl_C/
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/c_LightControl_C# pwd
/home/jungwungpark/nfs_client/2_LightControl/c_LightControl_C
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/c_LightControl_C# ls
gpio_control.c  loop_light_control    Makefile            test_light_control.c
gpio_control.h  loop_light_control.c  test_light_control
```

다음으로 "test_light_control" 파일을 실행 가능하게 만들고 실행하였다. 다음 그림과 같이 input을 하였을 때, LED의 빛이 꺼졌다가 켜지는 것을 확인하였다.

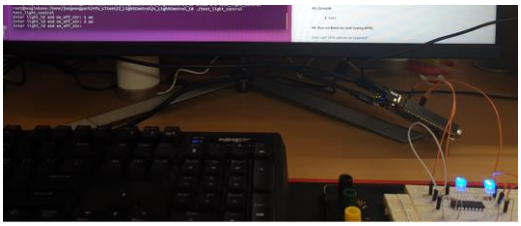
```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/c_LightControl_C# chmod a+x test_light_control
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/c_LightControl_C# ./test_light_control
test_light_control
Enter light_id and on_off_str: 1 on
Enter light_id and on_off_str: 2 on
Enter light_id and on_off_str: 1 off
Enter light_id and on_off_str: 2 off
Enter light_id and on_off_str: 1 on
Enter light_id and on_off_str: 2 on
Enter light_id and on_off_str: 1 none
Enter light_id and on_off_str: 2 none
Enter light_id and on_off_str: 0 on
```

"1 on"을 치면 Light1 이 켜졌고, "2 on"을 치면 Light2가 켜졌다. 반대로 "1 off"을 치면

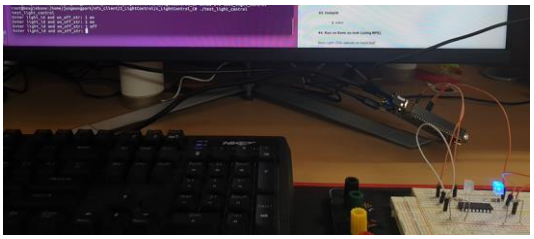
Light1 이 꺼졌고, "2 off"을 치면 Light2가 꺼졌다. "1 none"과 "2 none"을 쳤을 때는 변화가 없었다. "0 on"을 치면 나갈 수 있었다.



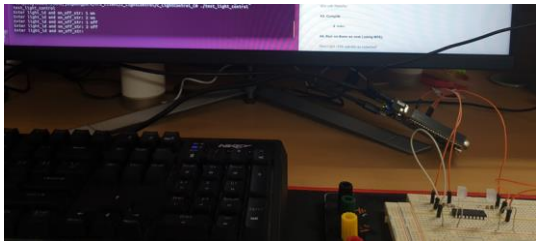
<Type 1 on>



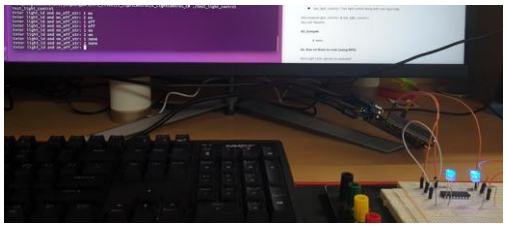
<Type 2 on>



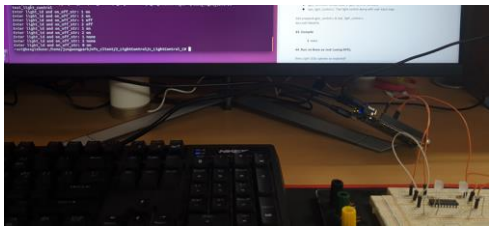
<Type 1 off>



<Type 2 off>



<Type 1 none and 2 none after Type 1 on and 2 on>



<Type 0 on>

다음으로 두 LED를 10번씩 켰다 껐다 하는 시간을 측정하였다. 위와 마찬가지로 "loop_light_control" 파일을 실행 가능하게 만들고 3번 실행하였다. 결과는 다음과 같다.

```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/c_LightControl_C# chmod a+x loop_light_control
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/c_LightControl_C# ./loop_light_control
loop_light_control
start: 1633591104.462698 end: 1633591104.465350
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/c_LightControl_C# ./loop_light_control
loop_light_control
start: 1633591105.867231 end: 1633591105.867705
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/c_LightControl_C# ./loop_light_control
loop_light_control
start: 1633591107.034968 end: 1633591107.037618
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/c_LightControl_C# ./loop_light_control
loop_light_control
start: 1633591107.908024 end: 1633591107.910680
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/c_LightControl_C# ./loop_light_control
loop_light_control
start: 1633591108.710737 end: 1633591108.713394
```

평균적으로 0.0019초가 소요됨을 확인하였다.

5) Problem 2D: Test HelloDev

(1) Compile and run and fail.

먼저 작업 위치를 "d_HelloDev" 폴더가 있는 곳으로 옮기고 cross-development 환경을 만들어주었다.

```
## Set MACHINE
MACHINE=beaglebone

## Set SYSROOTSDIR & STAGEDIR
SYSROOTSDIR=/usr/bin
STAGEDIR=${SYSROOTSDIR}

## Set CROSSBINDIR (where cross compiler exists)
CROSSBINDIR=/usr/bin

## Set KERNELDIR (where the Linux kernel source is located)
## NOTE: This path to KernelDir should be exact.
export KERNELDIR=/home/jungwung/DesignLab/linux

## Set PATH
PATH=${CROSSBINDIR}:${PATH}

unset CFLAGS CPPFLAGS CXXFLAGS LDFLAGS MACHINE

export ARCH="arm"
export CROSS_COMPILE="arm-linux-gnueabihf-"
export CC="arm-linux-gnueabihf-gcc"
export LD="arm-linux-gnueabihf-ld"
export STRIP="arm-linux-gnueabihf-strip"
echo "Set cross-development environment for Beaglebone Debian (3.8.13-bone79)."
```

```
jungwung@ubuntu:~/DesignLab/2_LightControl/d_HelloDev$ source cde_bd_k3813_bone79
Set cross-development environment for Beaglebone Debian (3.8.13-bone79).
```

이때 KERNELDIR를 실험 시 사용하는 컴퓨터의 환경에 맞게 변경하였다.

다음으로 "make"를 쳐 모듈을 만들어주었다.

```
jungwung@ubuntu:~/DesignLab/2_LightControl/d_HelloDev$ make
make -C /home/jungwung/DesignLab/linux M=/home/jungwung/DesignLab/2_LightControl/d_HelloDev modules
make[1]: Entering directory '/home/jungwung/DesignLab/linux'
  CC [M]  /home/jungwung/DesignLab/2_LightControl/d_HelloDev/HelloDev.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/jungwung/DesignLab/2_LightControl/d_HelloDev/HelloDev.mod.o
  LD [M]  /home/jungwung/DesignLab/2_LightControl/d_HelloDev/HelloDev.ko
make[1]: Leaving directory '/home/jungwung/DesignLab/linux'
jungwung@ubuntu:~/DesignLab/2_LightControl/d_HelloDev$ ls
cde_bd_k3813_bone79  HelloDev.ko     HelloDev.mod.o  Makefile       Module.symvers
HelloDev.c           HelloDev.mod.c  HelloDev.o      modules.order  Test_HelloDev.c
```

Beaglebone으로 돌아와 생성된 "HelloDev.ko" 모듈을 넣어주었다.

```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/d_HelloDev# insmod HelloDev.ko
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/d_HelloDev# dmesg | tail
[    54.232186] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[    56.961662] net eth0: initializing cpsw version 1.12 (0)
[    56.966287] net eth0: phy found : id is : 0x7c0f1
[    56.966318] libphy: PHY 4a101000.mdio:01 not found
[    56.971379] net eth0: phy 4a101000.mdio:01 not found on slave 1
[    56.986993] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[    58.974215] libphy: 4a101000.mdio:00 - Link is Up - 100/Full
[    58.974281] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 8208.473693] HelloDev Init - debug mode is disabled
[ 8208.473767] HelloDev: registered successfully!
```

PC로 돌아와 "make app"을 쳐 "Test_HelloDev.c" 파일을 컴파일을 한 후 다시 Beaglebone으로 돌아와 실행해보았지만 에러가 났다.

```
jungwung@ubuntu:~/DesignLab/2_LightControl/d_HelloDev$ make app
arm-linux-gnueabihf-gcc -o Test_HelloDev Test_HelloDev.c
```

```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/d_HelloDev# ./Test_HelloDev
./Test_HelloDev: entered
open failed: No such file or directory
```

(2) Make node and run again

"/dev/HelloDev" 장치 파일을 생성한 후 다시 실행해보았더니 잘 실행이 되었다.

```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/d_HelloDev# mknod /dev/HelloDev c 234 0
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/d_HelloDev# ./Test_HelloDev
./Test_HelloDev: entered
./Test_HelloDev: open successful
./Test_HelloDev: read: returning 0 bytes!
```

```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/d_HelloDev# dmesg | tail
[    56.966318] libphy: PHY 4a101000.mdio:01 not found
[    56.971379] net eth0: phy 4a101000.mdio:01 not found on slave 1
[    56.986993] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[    58.974215] libphy: 4a101000.mdio:00 - Link is Up - 100/Full
[    58.974281] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 8208.473693] HelloDev Init - debug mode is disabled
[ 8208.473767] HelloDev: registered successfully!
[ 8361.564495] HelloDev_open: successful
[ 8361.567622] HelloDev_read: returning zero bytes
[ 8361.568978] HelloDev_release: successful
```

마지막으로 모듈을 제거하였다.

```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/d_HelloDev# rmmod HelloDev
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/d_HelloDev# dmesg | tail
[    56.971379] net eth0: phy 4a101000.mdio:01 not found on slave 1
[    56.986993] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[    58.974215] libphy: 4a101000.mdio:00 - Link is Up - 100/Full
[    58.974281] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 8208.473693] HelloDev Init - debug mode is disabled
[ 8208.473767] HelloDev: registered successfully!
[ 8361.564495] HelloDev_open: successful
[ 8361.567622] HelloDev_read: returning zero bytes
[ 8361.568978] HelloDev_release: successful
[ 8423.476788] Goodbye, HelloDev
```

6) Problem 2E: Compile and run Light control with LED driver module

(1) Compile and run

먼저 작업 위치를 "e_LightLEDs_DeviceDriver" 폴더가 있는 곳으로 옮겼다. 해당 폴더 안 파일을 확인했을 때 다음과 같았다.

```
jungwung@ubuntu:~/DesignLab/2_LightControl/e_LigthLEDs_DeviceDriver$ ls
am33xx.h  gpio.h  LightLEDs_Module.c  Makefile  Test_LightLEDs_Module.c
```

Problem 2D와 비슷하게 Makefile안에 명시된 "make"을 쳐 모듈을 만들고 "make app"을 쳐 "Test_LightLEDs_Module.c" 파일을 컴파일 했다.

```
'oe0' [-Wunused-variable]
  unsigned int oe0, oe0new;
               ^
/home/jungwung/DesignLab/2_LightControl/e_LigthLEDs_DeviceDriver/LightLEDs_Module.c: In function 'LightLEDs_release':
/home/jungwung/DesignLab/2_LightControl/e_LigthLEDs_DeviceDriver/LightLEDs_Module.c:150:20: warning: unused variable
'oe0new' [-Wunused-variable]
  unsigned int oe0, oe0new;
                    ^
/home/jungwung/DesignLab/2_LightControl/e_LigthLEDs_DeviceDriver/LightLEDs_Module.c:150:15: warning: unused variable
'oe0' [-Wunused-variable]
  unsigned int oe0, oe0new;
               ^
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/jungwung/DesignLab/2_LightControl/e_LigthLEDs_DeviceDriver/LightLEDs_Module.mod.o
  LD [M]  /home/jungwung/DesignLab/2_LightControl/e_LigthLEDs_DeviceDriver/LightLEDs_Module.ko
make[1]: Leaving directory '/home/jungwung/DesignLab/linux'
jungwung@ubuntu:~/DesignLab/2_LightControl/e_LigthLEDs_DeviceDriver$ ls
am33xx.h  LightLEDs_Module.c   LightLEDs_Module.mod.c  LightLEDs_Module.o  modules.order   Test_LightLEDs_Module.c
gpio.h    LightLEDs_Module.ko  LightLEDs_Module.mod.o  Makefile            Module.symvers
jungwung@ubuntu:~/DesignLab/2_LightControl/e_LigthLEDs_DeviceDriver$ make app
arm-linux-gnueabihf-gcc -o Test_LightLEDs_Module Test_LightLEDs_Module.c
jungwung@ubuntu:~/DesignLab/2_LightControl/e_LigthLEDs_DeviceDriver$ ls
am33xx.h  LightLEDs_Module.c   LightLEDs_Module.mod.c  LightLEDs_Module.o  modules.order   Test_LightLEDs_Module
gpio.h    LightLEDs_Module.ko  LightLEDs_Module.mod.o  Makefile            Module.symvers  Test_LightLEDs_Module.c
```

Beaglebone으로 돌아와 해당 폴더의 파일을 확인하였다.

```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/e_LigthLEDs_DeviceDriver# ls
am33xx.h  LightLEDs_Module.c   LightLEDs_Module.mod.c  LightLEDs_Module.o  modules.order   Test_LightLEDs_Module
gpio.h    LightLEDs_Module.ko  LightLEDs_Module.mod.o  Makefile            Module.symvers  Test_LightLEDs_Module.c
```

다음으로 생성된 "LightLEDs_Module.ko" 모듈을 넣었다. 이때 major number은 239이다.

```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/e_LigthLEDs_DeviceDriver# insmod LightLEDs_Module.ko
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/e_LigthLEDs_DeviceDriver# dmesg | tail
[   56.986993] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[   58.974215] libphy: 4a101000.mdio:00 - Link is Up - 100/Full
[   58.974281] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 8208.473693] HelloDev Init - debug mode is disabled
[ 8208.473767] HelloDev: registered successfully!
[ 8361.564495] HelloDev_open: successful
[ 8361.567622] HelloDev_read: returning zero bytes
[ 8361.568978] HelloDev_release: successful
[ 8423.476788] Goodbye, HelloDev
[ 8905.617302]     A. Init LightLEDs: The major number is 239
```

"/dev/LightLEDs" 장치 파일을 생성하였다 이때 major number은 239이다.

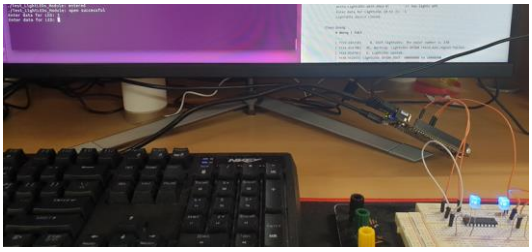
```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/e_LigthLEDs_DeviceDriver# mknod /dev/LightLEDs c 239 0
```

컴파일 된 파일을 실행하고 다음과 같은 순서로 input을 넣었다. Input 값에 따라서 2개

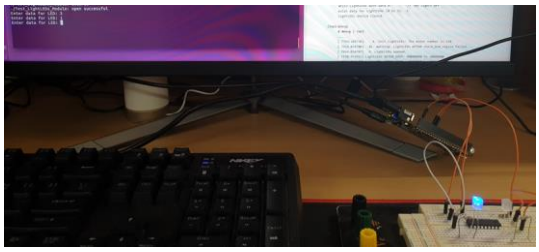
의 LED가 켜지기도 하고 꺼지기도 했다.

```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/e_LigthLEDs_DeviceDriver# ./Test_LightLEDs_Module
./Test_LightLEDs_Module: entered
./Test_LightLEDs_Module: open successful
Enter data for LED: 3
Enter data for LED: 1
Enter data for LED: 2
Enter data for LED: 0
Enter data for LED: -1
```

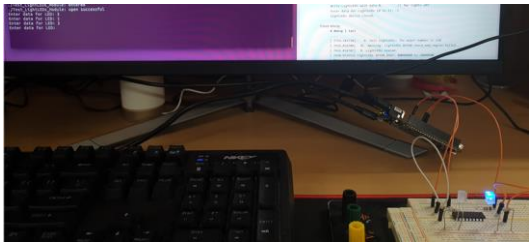
Input에 3을 입력했을 때, 두 LED 모두 켜졌다. 1을 입력했을 때는 Light1이 켜지고 Light2는 꺼졌다. 2을 입력했을 때는 Light2가 켜지고 Light1은 꺼졌다. 0을 입력하면 두 LED 모두 꺼졌다. -1을 입력하고 실행을 종료할 수 있었다.



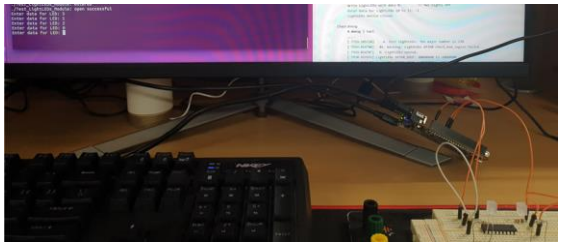
<Type 3>



<Type 1>



<Type 2>



<Type 0>

실험을 마친 후 모듈을 제거하였다.

```
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/e_LigthLEDs_DeviceDriver# dmesg | tail
[ 9110.076583]   D. LightLEDs opened.
[ 9133.773836] 3
[ 9133.773883] LightLEDs GPIO0_DOUT: 00000000 to c0000000
[ 9151.666738] 1
[ 9151.666785] LightLEDs GPIO0_DOUT: 00000000 to 40000000
[ 9159.913821] 2
[ 9159.913868] LightLEDs GPIO0_DOUT: 00000000 to 80000000
[ 9166.437054] 0
[ 9166.437100] LightLEDs GPIO0_DOUT: 00000000 to 00000000
[ 9195.625157]   -D. LightLEDs released.
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/e_LigthLEDs_DeviceDriver# rmmod LightLEDs_Module
root@beaglebone:/home/jungwungpark/nfs_client/2_LightControl/e_LigthLEDs_DeviceDriver# dmesg | tail
[ 9133.773836] 3
[ 9133.773883] LightLEDs GPIO0_DOUT: 00000000 to c0000000
[ 9151.666738] 1
[ 9151.666785] LightLEDs GPIO0_DOUT: 00000000 to 40000000
[ 9159.913821] 2
[ 9159.913868] LightLEDs GPIO0_DOUT: 00000000 to 80000000
[ 9166.437054] 0
[ 9166.437100] LightLEDs GPIO0_DOUT: 00000000 to 00000000
[ 9195.625157]   -D. LightLEDs released.
[ 9269.451497]   -A. Cleanup LightLEDs: Unregistered.
```