

## Lab3

이 실습에서는 라즈베리파이를 이용하여 리눅스 드라이버 및 드라이버를 이용한 응용 프로그램의 작성을 실습합니다.

먼저, 라즈베리파이와 공유기를 유선 랜 케이블로 연결하고, 마이크로 USB 케이블(C-Type)이 연결된 어댑터를 연결합니다.

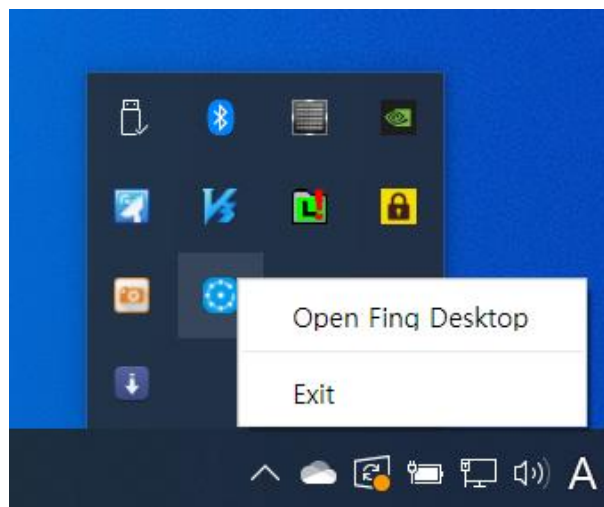
구글에서 SmarTTY 를 검색하거나 아래의 링크에서 SmarTTY 프로그램을 다운로드 받은 후 설치합니다. SmarTTY 프로그램은 터미널 프로그램으로서 telnet, SSH, serial 통신 등을 지원하는 무료 프로그램입니다. 반드시 이 프로그램이 아니라도 PuTTY 등 유사한 프로그램을 사용하는 것이 가능합니다.

<https://sysprogs.com/SmarTTY/>

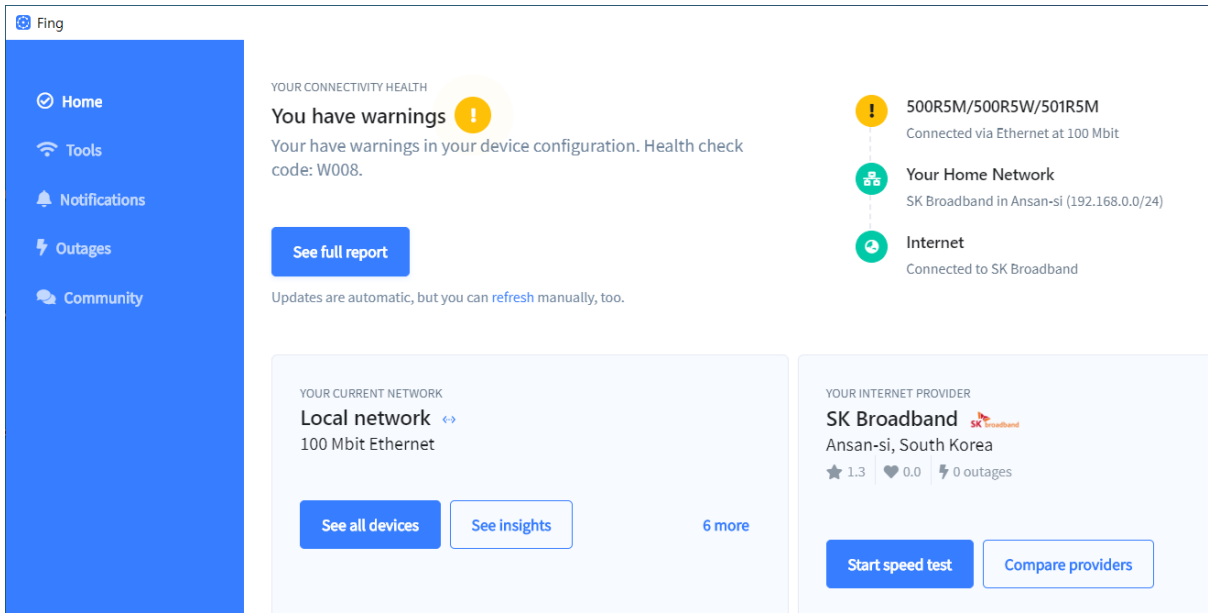
터미널 프로그램을 이용해서 라즈베리파이에 접속하기 위해서는 라즈베리파이에 할당된 IP 주소를 알아야 합니다. 이를 위해서 Fing 이라는 프로그램을 사용합니다. 이 프로그램은 같은 공유기에 접속된 유무선 장치의 IP 주소 목록을 보여줍니다.

<https://www.fing.com/products/fing-desktop>

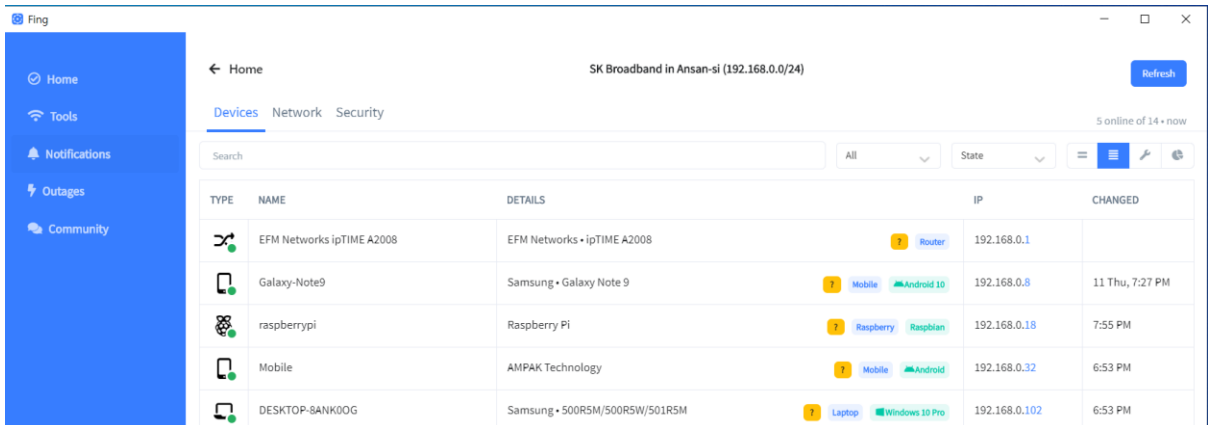
프로그램을 설치한 후 프로그램 아이콘을 아래와 같이 선택한 후 마우스 오른쪽 버튼을 누르면 아래와 같이 메뉴가 나옵니다. 여기에서 Open Fing Desktop을 선택합니다.



아래와 같은 Fing Desktop에서 See all devices 버튼을 누릅니다.



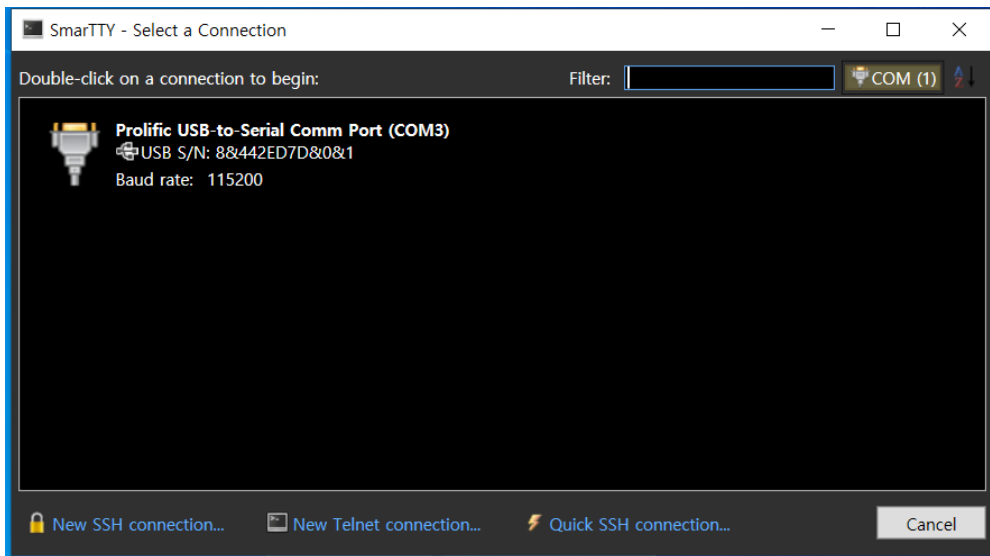
그러면 아래와 같이 접속된 장치들의 IP 주소가 보입니다. 라즈베리파이가 안 보일 경우 잠시 기다린 후 Refresh 버튼을 누르면 목록이 갱신됩니다.



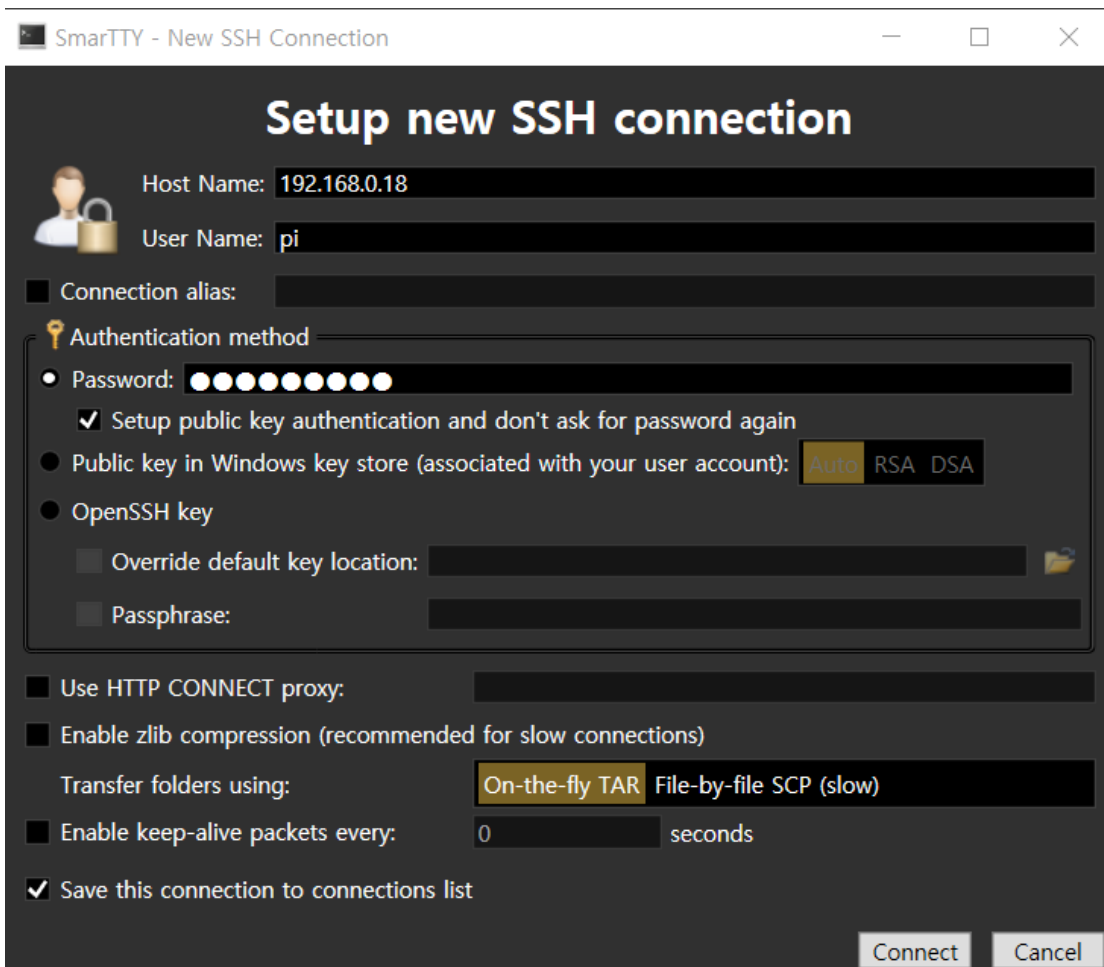
위의 화면 예에서 라즈베리파이의 IP 주소가 192.168.0.18 임을 볼 수 있습니다. 스마트폰에도 Fing 앱이 있으며 스마트폰이 같은 공유기에 접속되어 있으면 위와 같은 접속 장치 목록은 스마트폰 앱에서도 확인 가능합니다.

(참고 사항: 만약 기다려도 위의 목록에 라즈베리파이가 나타나지 않으면 부팅이 안되거나 네트워크에 접속이 안되는 경우입니다. 그 경우에는 라즈베리파이에서 HDMI 케이블을 이용하여 모니터를 연결하고 키보드를 연결한 후 부팅이 정상적으로 되는지 확인해 봅니다. 부팅이 안된다고 판단이 될 경우 별도로 문의하십시오.)

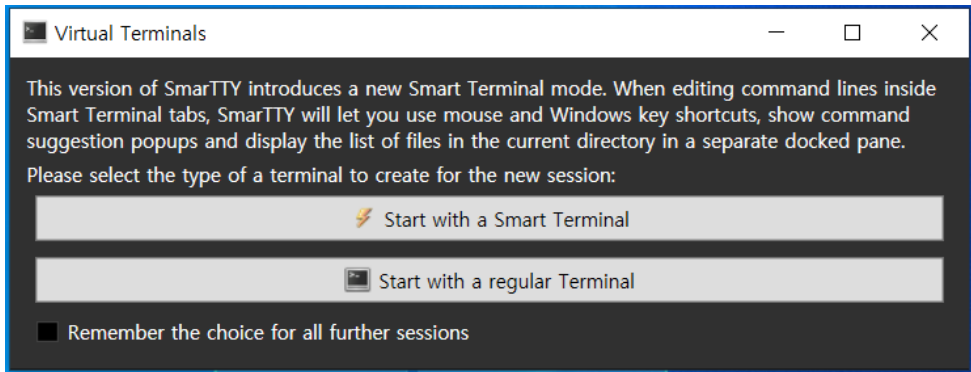
라즈베리파이의 IP 주소를 확인 후, 다음과 같이 SmarTTY를 열어서 New SSH connection을 클릭합니다.



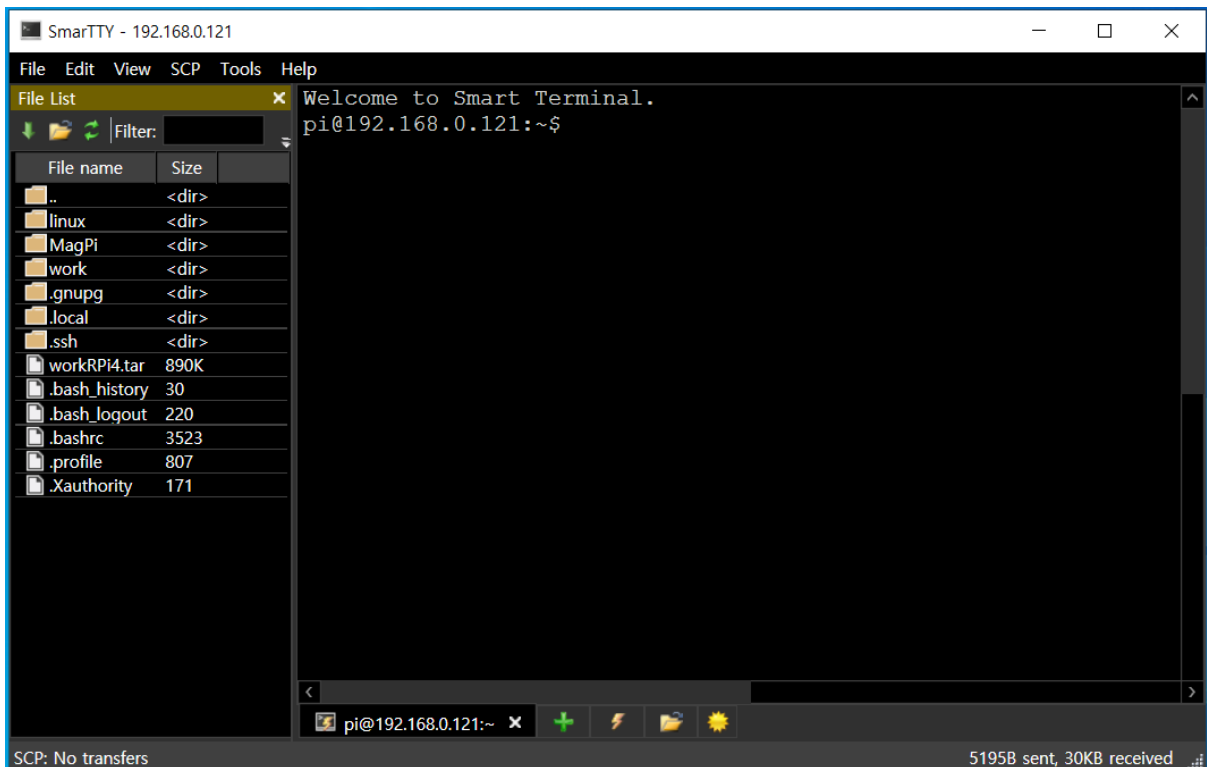
다음과 같은 창이 나오면 Host Name: 192.168.0.18, User Name: pi, Password: raspberry 를 입력하고 Connect를 선택합니다. 키를 저장할 것을 묻는 화면이 나오면 저장하는 것을 선택합니다.



다음과 같은 화면이 나오면 두가지 중의 하나를 선택합니다.



위에서 어느 것을 선택해도 가능하지만, Smart Terminal이 좀더 편리한 기능이 많으므로 이것을 선택합니다.

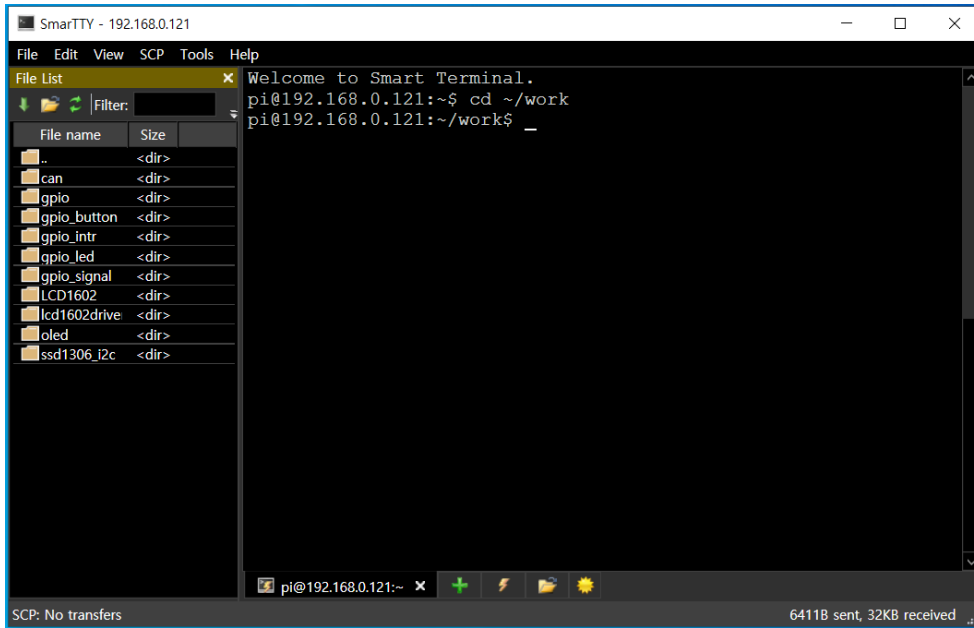


위의 화면의 좌측은 디렉토리로 이동이 가능한 브라우저 이며, 또한 파일 이름을 더블 클릭하면 편집이 가능한 편집기가 열립니다.

라즈베리파이의 user name은 pi 이므로 user 계정의 홈 디렉토리의 위치는 /home/pi 입니다. 다른 디렉토리에 있다가 홈 디렉토리로 돌아가기 위해서는 \$ cd ~ 를 입력합니다. 아래의 명령은 사용자의 홈 디렉토리내의 work 디렉토리로 이동하는 명령어로 이 실습에서는 이곳에서 작업을 합

니다.

```
$ cd ~/work
```



이 디렉토리에선 여러 개의 디렉토리가 있으나 실습에서는 gpio\_button, gpio\_intr, gpio\_led, gpio\_signal, lcd1602driver 의 파일로 실습을 합니다.

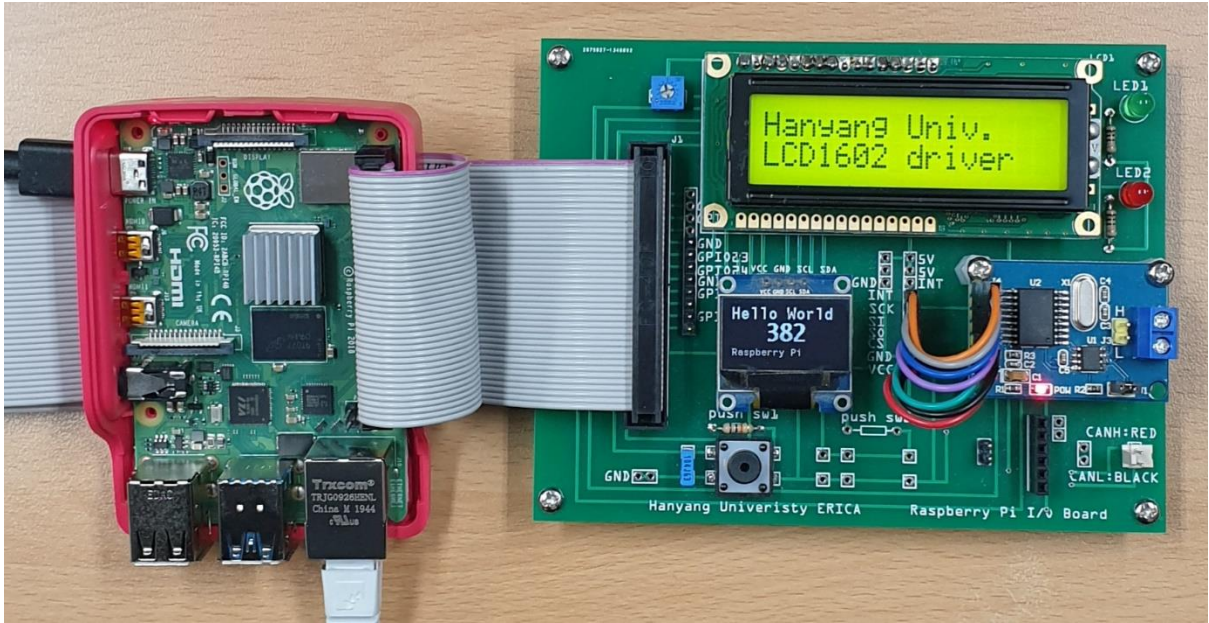
드라이버를 시험하기 전에 I/O보드와 라즈베리파이의 연결이 필요합니다. 현재 전원이 연결되어 있으면 보드의 고장 방지를 위해 전원 연결을 해제합니다. 라즈베리파이도 컴퓨터 이므로 전원을 차단할 때 적절하게 shutdown이 되어야 합니다. 아래와 같이 명령을 내린 후 전원을 차단합니다.

```
$ sudo shutdown now
```

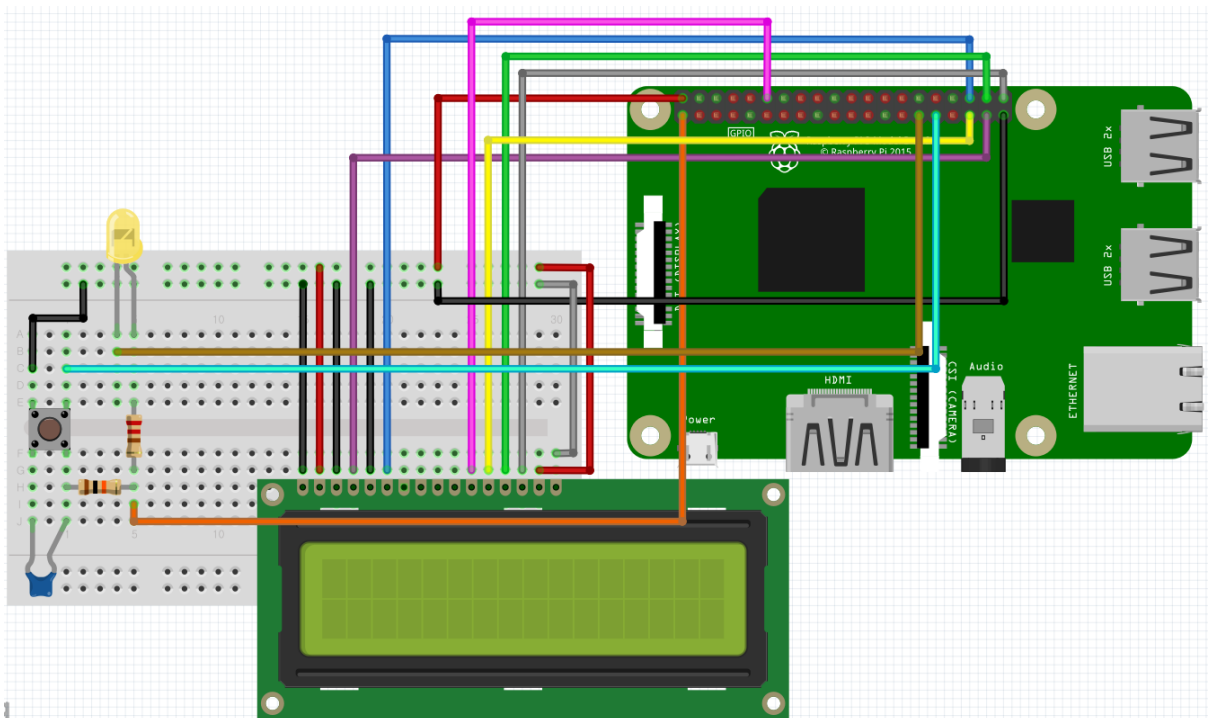
라즈베리파이와 I/O보드의 연결은 40 pin flat cable로 연결합니다. 이때 I/O보드의 커넥터에는 홈이 있어서 거꾸로 연결하는 것이 불가능하지만, 라즈베리파이에는 커넥터 핀만 있으므로 반대로 연결할 위험이 있습니다. 아래 사진을 참조하여 flat cable의 빨간 줄이 랜케이בל 커넥터의 반대 방향으로 가도록 연결합니다. 40개의 핀이 접속이 되어야 하므로 핀과 커넥터의 위치를 잘 보고 주의 해서 연결합니다. 커넥터를 잘 못 연결할 경우 고장의 우려가 있으므로 주의가 필요합니다.



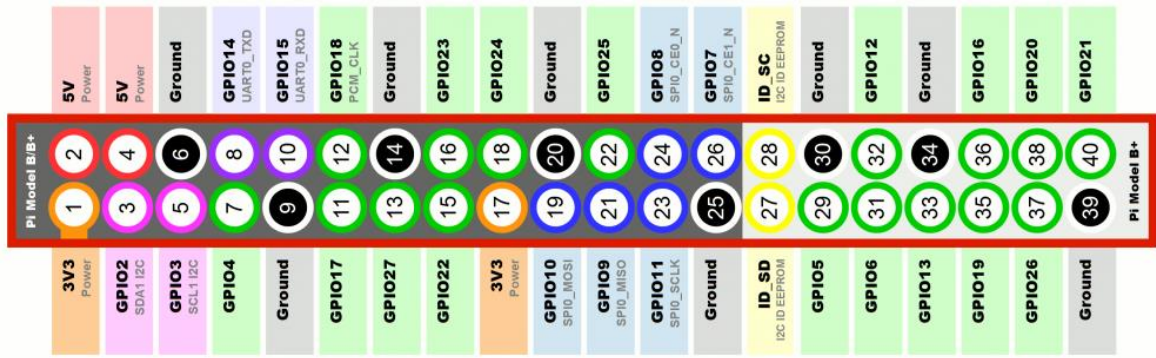
I/O보드 측의 커넥터도 케이블의 빨간색 줄이 위쪽을 향하도록 연결합니다. 커넥터의 중앙에 키홈이 있어서 반대 방향으로 커넥터가 들어가지 않습니다. 아래와 같이 라즈베리파이와 I/O보드를 연결 한 후 전원 어댑터를 연결하여 부팅 되는 것을 기다립니다.



아래의 그림들은 PCB보드를 사용하지 않고 브레드보드를 이용할 경우의 연결 방법입니다. 개인적으로 부품을 구입하여 같은 실험을 해보기를 원할 경우 아래와 같은 방법으로 연결할 수 있습니다.

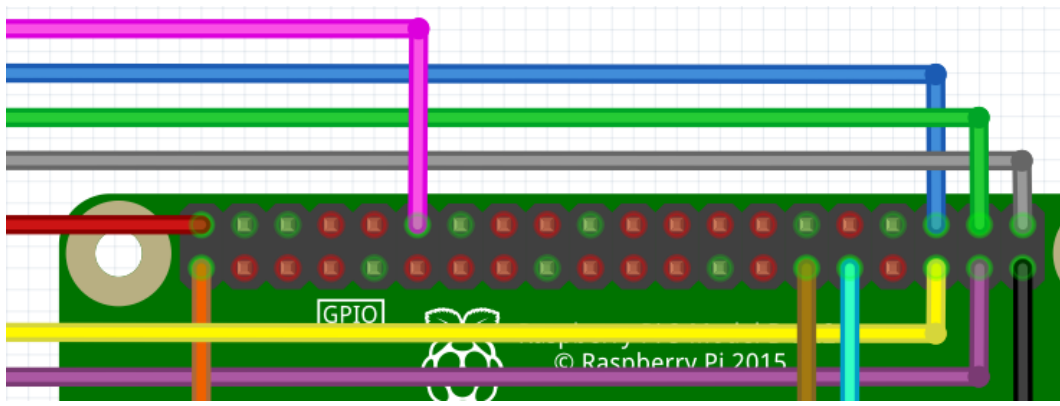


위와 같이 정확히 연결하는 것이 필요합니다. 특히 라즈베리파이는 동작 전압이 3.3볼트이며 LCD는 동작 전압이 5볼트입니다. 라즈베리파이에 5볼트의 전압이 입력되면 파손이 되므로 연결 시 실수하지 않도록 주의해야 합니다. 아래는 라즈베리파이의 핀 정의입니다.



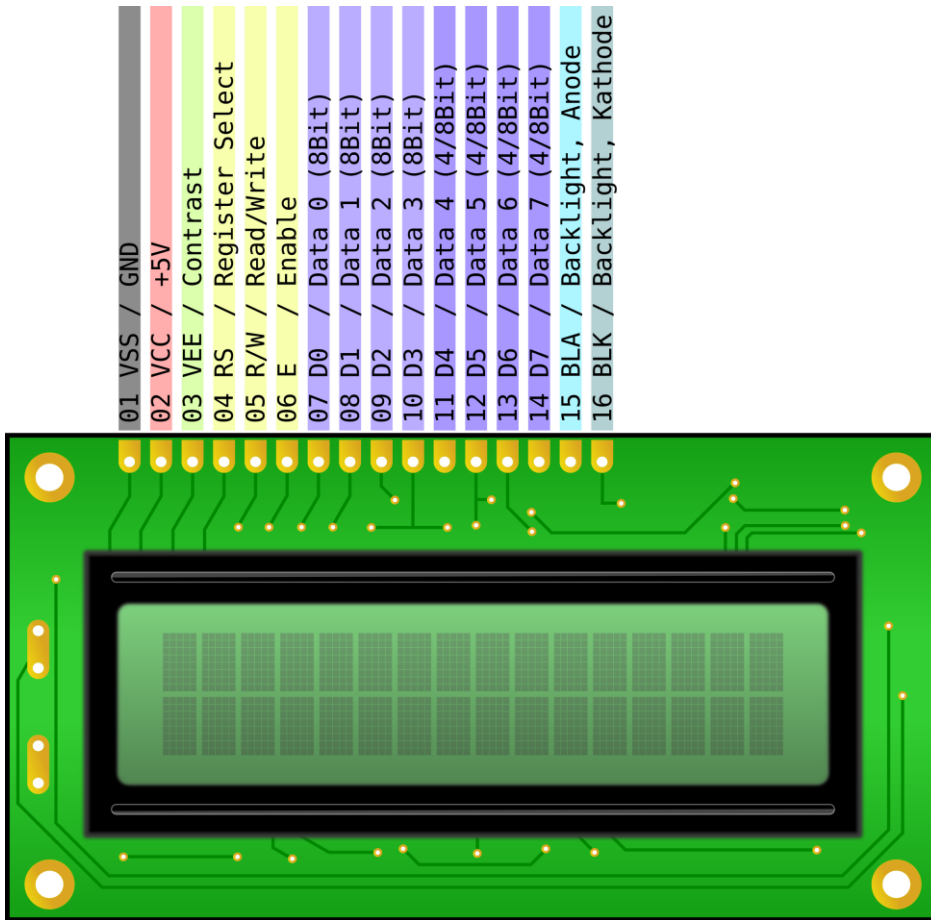
www.raspberrypi-spy.co.uk

위의 핀 정의와 아래의 연결 그림을 비교하면서 정확히 연결 합니다.



아래는 백라이트가 있는 character LCD의 핀 정의입니다.





LCD 연결을 표로 정리하면 다음과 같습니다.

LCD 핀 번호	LCD 핀 이름	라즈베리파이 핀 번호	라즈베리파이 핀 이름
14	D7	40	GPIO21
13	D6	38	GPIO20
12	D5	35	GPIO19
11	D4	12	GPIO18
6	E	36	GPIO16
4	RS	37	GPIO26

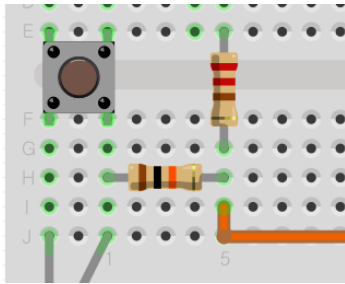
위의 연결 이외에 GND(라즈베리파이 39번핀)와 5V(라즈베리파이 2번)을 아래와 같이 브레드 보드의 GND와 5V 레일 핀에 연결합니다.



브레드보드의 LED와 Button 입력은 다음과 같습니다.

	라즈베리파이 핀 번호	라즈베리파이 핀 이름
LED Cathode(K)	29	GPIO5
Button	31	GPIO6

**중요:** LED의 저항과 Button pull-up 저항에는 반드시 라즈베리파이의 3.3V(1번 핀)을 연결해야 합니다. 여기에 5V(2번 핀)를 연결할 경우 5V 신호가 입력되어 라즈베리파이가 파손될 수 있습니다. 즉, 아래 그림의 저항에 연결된 주황색 전선은 반드시 3.3V 와 연결해야 합니다.



연결을 마친 후, 각 입출력 장치들을 시험해 봅니다. 전원 연결 후 부팅이 되면, SSH로 접속하여 아래와 같이 드라이버와 응용 프로그램을 시험합니다.

- LED 드라이버

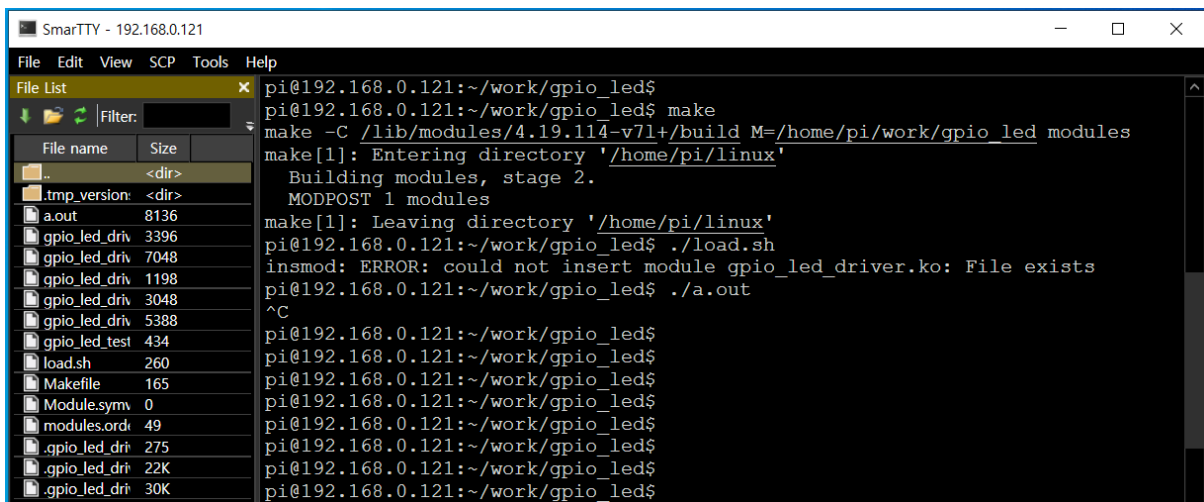
```
$ cd ~/work/gpio_led
```

```
$ make
```

```
$ ./load.sh
```

```
$ ./a.out
```

위의 명령을 실행하면 LED가 깜박이는 것을 확인할 수 있습니다.



```
SmarTTY - 192.168.0.121
File Edit View SCP Tools Help
File List
Filter:
File name Size
.. <dir>
.tmp_version: <dir>
a.out 8136
gpio_led_drv 3396
gpio_led_drv 7048
gpio_led_drv 1198
gpio_led_drv 3048
gpio_led_drv 5388
gpio_led_test 434
load.sh 260
Makefile 165
Module.sym 0
modules.order 49
_gpio_led_dri 275
_gpio_led_dri 22K
_gpio_led_dri 30K
pi@192.168.0.121:~/work/gpio_led$
pi@192.168.0.121:~/work/gpio_led$ make
make -C /lib/modules/4.19.114-v7l+/build M=/home/pi/work/gpio_led modules
make[1]: Entering directory '/home/pi/linux'
Building modules, stage 2.
MODPOST 1 modules
make[1]: Leaving directory '/home/pi/linux'
pi@192.168.0.121:~/work/gpio_led$ ./load.sh
insmod: ERROR: could not insert module gpio_led_driver.ko: File exists
pi@192.168.0.121:~/work/gpio_led$ ./a.out
^C
pi@192.168.0.121:~/work/gpio_led$
pi@192.168.0.121:~/work/gpio_led$
pi@192.168.0.121:~/work/gpio_led$
pi@192.168.0.121:~/work/gpio_led$
pi@192.168.0.121:~/work/gpio_led$
pi@192.168.0.121:~/work/gpio_led$
pi@192.168.0.121:~/work/gpio_led$
pi@192.168.0.121:~/work/gpio_led$
```

- Button 드라이버(Polling)

```
$ cd ~/work/gpio_button
```

```
$ make
```

```
$ ./load.sh
```

```
$ ./a.out
```

위와 같이 실행한 후 버튼을 누르면 프린트 되는 값이 1에서 0으로 바뀌는 것을 확인할 수 있습니다.

```

pi@192.168.0.121:~/work/gpio_button$ make
make -C /lib/modules/4.19.114-v7l+/build M=/home/pi/work/gpio_button modules
make[1]: Entering directory '/home/pi/linux'
Building modules, stage 2.
MODPOST 1 modules
make[1]: Leaving directory '/home/pi/linux'
pi@192.168.0.121:~/work/gpio_button$ ./load.sh
pi@192.168.0.121:~/work/gpio_button$ ./a.out
1
1
1
1
1
1
1
1
1
1
1
1
pi@192.168.0.121:~/work/gpio_button$
pi@192.168.0.121:~/work/gpio_button$

```

- LCD 드라이버

```
$ cd ~/work/lcd1602driver
```

```
$ make
```

```
$ ./load.sh
```

```
$ ./a.out
```

위와 같이 드라이버 모듈이 초기화 되면서 초기 메시지가 LCD에 출력되고, 응용 프로그램이 실행되면서 응용 프로그램이 출력하는 메시지를 확인할 수 있습니다.

```

pi@192.168.0.121:~/work/lcd1602driver$ make
make -C /lib/modules/4.19.114-v7l+/build M=/home/pi/work/lcd1602driver modules
make[1]: Entering directory '/home/pi/linux'
Building modules, stage 2.
MODPOST 1 modules
make[1]: Leaving directory '/home/pi/linux'
pi@192.168.0.121:~/work/lcd1602driver$ ./load.sh
pi@192.168.0.121:~/work/lcd1602driver$ ./a.out
pi@192.168.0.121:~/work/lcd1602driver$
pi@192.168.0.121:~/work/lcd1602driver$
pi@192.168.0.121:~/work/lcd1602driver$
pi@192.168.0.121:~/work/lcd1602driver$
pi@192.168.0.121:~/work/lcd1602driver$

```

- Button Interrupt 드라이버

```
$ cd ~/work/gpio_intr
```

```
$ make
```

```
$ ./load.sh
```

```
$ ./a.out
```

버튼을 누를 때 마다 인터럽트가 발생하면서 메시지가 출력되는 것을 확인할 수 있습니다.

```

pi@192.168.0.121:~/work/gpio_intr$ make
make -C /lib/modules/4.19.114-v7l+/build M=/home/pi/work/gpio_intr module s
make[1]: Entering directory '/home/pi/linux'
CC [M] /home/pi/work/gpio_intr/gpio_intr_driver.o
Building modules, stage 2.
MODPOST 1 modules
CC /home/pi/work/gpio_intr/gpio_intr_driver.mod.o
LD [M] /home/pi/work/gpio_intr/gpio_intr_driver.ko
make[1]: Leaving directory '/home/pi/linux'
pi@192.168.0.121:~/work/gpio_intr$ ./load.sh
pi@192.168.0.121:~/work/gpio_intr$ ./a.out
Please push the GPIO button!
GPIO Switch was Pushed!
GPIO Switch was Pushed!
GPIO Switch was Pushed!
GPIO Switch was Pushed!
GPIO Switch was Pushed!
GPIO Switch was Pushed!

```

- Interrupt Signal 드라이버

주의: 위의 Button Interrupt 드라이버와 Interrupt Signal 드라이버는 같은 GPIO 인터럽트를 사용하므로 새로운 드라이버를 설치하기 전에 다른 드라이버는 제거 합니다. 따라서, 아래의 예를 실행하기 전에 다음의 명령을 실행 합니다.

```
$ sudo rmmod gpio_intr_driver
```

```
$ cd ~/work/gpio_signal
```

```
$ make
```

```
$ ./load.sh
```

```
$ ./a.out
```

버튼을 누를 때 마다 인터럽트가 발생하면서 메시지가 출력되는 것을 확인할 수 있습니다. 이때, 응용 프로그램이 인터럽트 서비스 루틴에서 시그널을 받고 출력하는 메시지도 함께 출력하는 것을 확인합니다.

```

pi@192.168.0.121:~/work/gpio_signal$ sudo rmmod gpio_intr_driver
pi@192.168.0.121:~/work/gpio_signal$ make
make -C /lib/modules/4.19.114-v7l+/build M=/home/pi/work/gpio_signal modules
make[1]: Entering directory '/home/pi/linux'
Building modules, stage 2.
MODPOST 1 modules
make[1]: Leaving directory '/home/pi/linux'
pi@192.168.0.121:~/work/gpio_signal$ ./load.sh
pi@192.168.0.121:~/work/gpio_signal$ ./a.out
dev_pid=4411
Please push the GPIO button!
GPIO button signal
GPIO Switch was Pushed!
GPIO button signal
GPIO Switch was Pushed!
GPIO button signal
GPIO Switch was Pushed!
GPIO button signal

```

드라이버 소스에 대한 설명은 수업 시간에 할 예정입니다. 드라이버 소스의 이해를 위해서 아래와 같은 라즈베리파이의 GPIO address map이 필요합니다..

Address Offset	Register Name	Description	Size
0x00	GPFSSEL0	GPIO Function Select 0	32
0x04	GPFSSEL1	GPIO Function Select 1	32
0x08	GPFSSEL2	GPIO Function Select 2	32
0x0C	GPFSSEL3	GPIO Function Select 3	32
0x10	GPFSSEL4	GPIO Function Select 4	32
0x14	GPFSSEL5	GPIO Function Select 5	32
0x18	-	Reserved	-
0x1C	GPSET0	GPIO Pin Output Set 0	32
0x20	GPSET1	GPIO Pin Output Set 1	32
0x24	-	Reserved	-
0x28	GPCLR0	GPIO Pin Output Clear 0	32
0x2C	GPCLR1	GPIO Pin Output Clear 1	32
0x30	-	Reserved	-
0x34	GPLEV0	GPIO Pin Level 0	32
0x38	GPLEV1	GPIO Pin Level 1	32

```
000 = GPIO Pin is an input
001 = GPIO Pin is an output
100 = GPIO Pin takes alternate function 0
101 = GPIO Pin takes alternate function 1
110 = GPIO Pin takes alternate function 2
111 = GPIO Pin takes alternate function 3
011 = GPIO Pin takes alternate function 4
010 = GPIO Pin takes alternate function 5
```

참고사항: 윈도우의 원격 데스크 탑 기능을 이용하여 라즈베리파이의 데스크 탑을 윈도우에서 열 수 있습니다. (아래 원격 데스크 탑에 관한 주의 사항을 꼭 읽어 주세요.) 데스크 탑 서버 설치를 위해서 아래의 명령을 입력합니다. 이미 설치된 경우 아래와 같이 나오며, 설치가 안된 경우에는 설치를 실행합니다.

```
$ sudo apt install xrdp
```

```
SmarTTY - 192.168.0.18
File Edit View SCP Tools Help
File List x
Welcome to Smart Terminal.
pi@192.168.0.18:~$ sudo apt install xrdp
Reading package lists... Done
Building dependency tree
Reading state information... Done
xrdp is already the newest version (0.9.9-1+deb10u1).
0 upgraded, 0 newly installed, 0 to remove and 113 not upgraded.
pi@192.168.0.18:~$
```

원격 데스크톱 연결

원격 데스크톱 연결

컴퓨터(C): 192.168.0.18


사용자 이름: 지정 안 함

연결할 때 자격 증명을 묻는 메시지가 나타납니다.

옵션 표시(O)      연결(N)      도움말(H)

192.168.0.18: 원격 데스크톱 연결

Login to raspberrypi

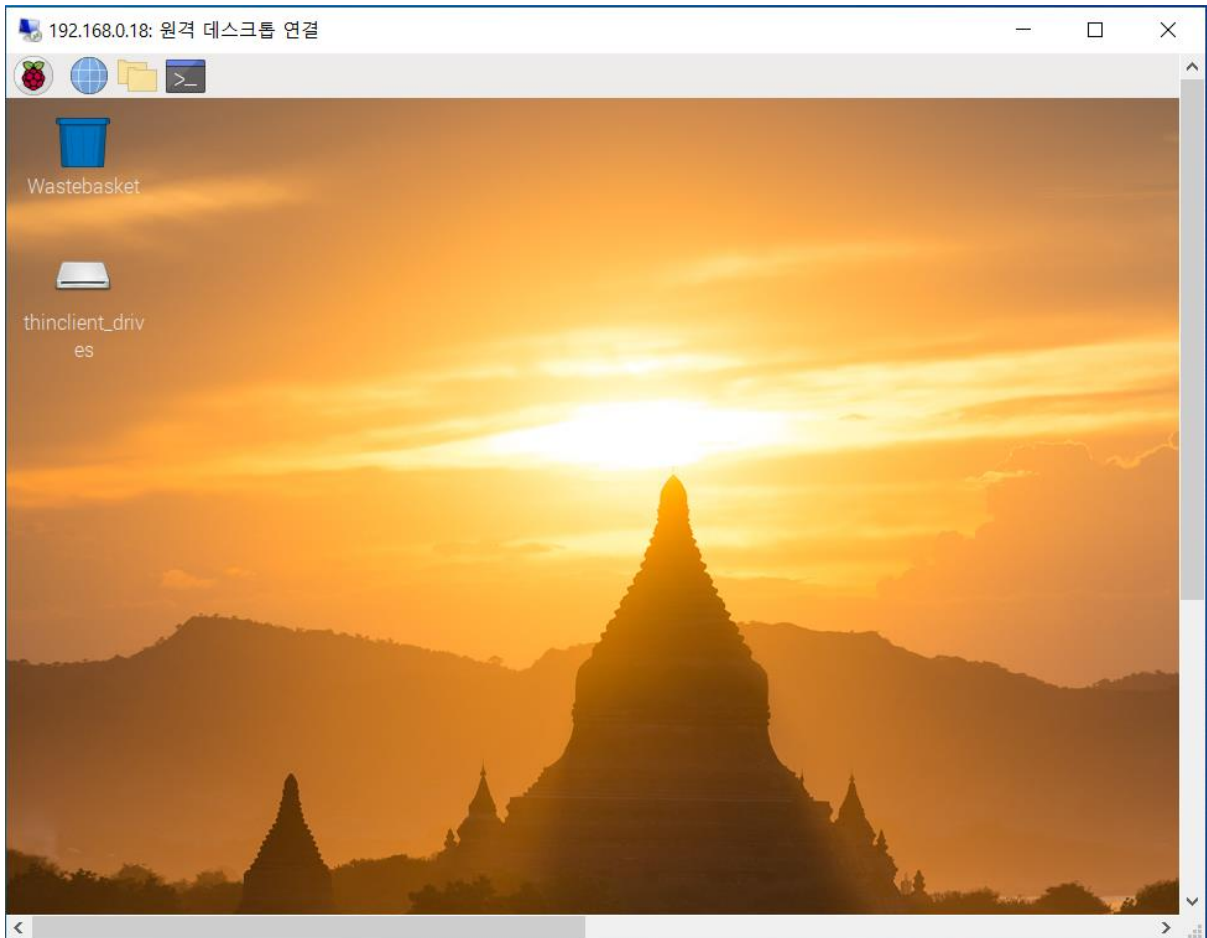


Session: Xorg

username: pi

password: \*\*\*\*\*

OK      Cancel

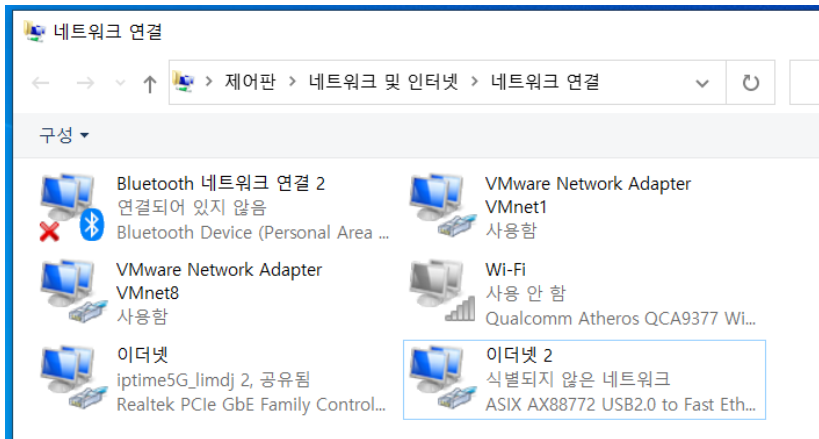


주의 사항: 원격 데스크 톱으로 라즈베리파이를 접속할 경우, 화면에 자동 업데이트나 업그레이드를 묻는 화면이 나옵니다. 이때 자동 업데이트나 자동 업그레이드를 선택할 경우 커널이 업그레이드가 되어 실습을 위해서 제공한 드라이버 소스의 빌드와 실행이 되지 않을 수 있습니다. 본인이 확실히 모를 경우에는 아무것도 선택하지 말고 창을 닫아 주세요. 이 과목에서 실행되는 실습에서는 원격 데스크 톱이 필요하지 않으므로 특별한 사유가 있지 않다면 사용하지 않을 것을 권장합니다.

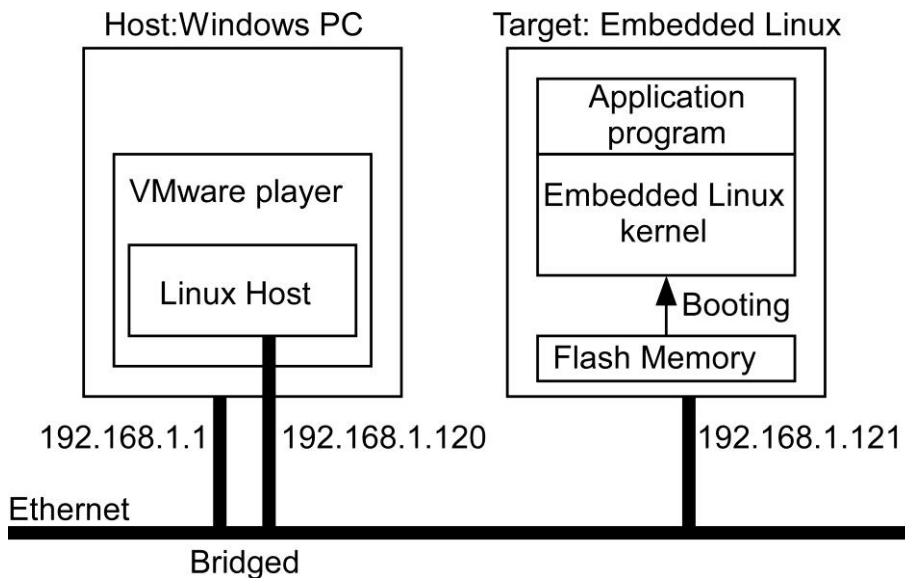
### 학교 실습실의 IP 주소 구성:

재택 실습이 아닌 학교 실습실에서 실습을 하는 경우 IP 주소가 위의 설명과 다릅니다. 재택 실습의 경우 1개의 공유기에 컴퓨터와 라즈베리파이 등 공유기에 접속된 모든 기기가 한 개의 네트워크로 구성이 되어 있지만, 학교 실습실의 경우 실습실 내의 수 십대의 컴퓨터가 한 개의 공유기에 접속되어 있으며, 또한 여러 대의 라즈베리 파이가 이 공유기에 접속될 경우 자신이 사용하는 라즈베리 파이의 주소를 파악하기가 어려울 수 있습니다. 따라서 실습실의 컴퓨터에는 USB 포트에 연결되는 이더넷 랜카드를 한 개 추가하여 별도의 네트워크를 구성합니다. 아래의 그림에서 볼 수 있듯이, 인터넷에 연결된 본체의 이더넷 랜카드와 USB 이더넷 카드가 있는 것을 볼 수 있습니다.



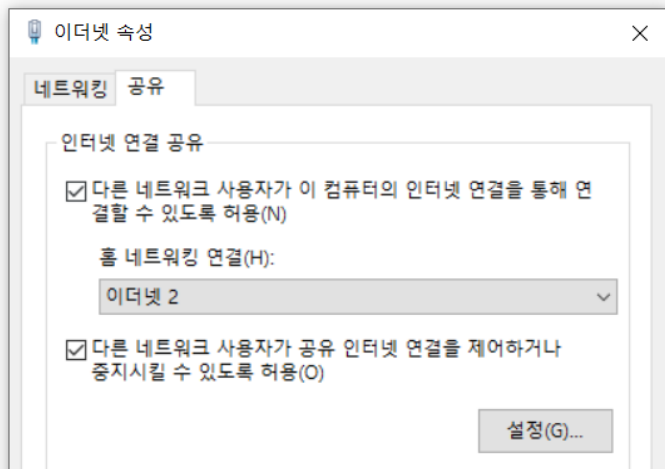


또한 실습대 위에 이더넷 허브가 있는 것을 볼 수 있는데, USB 이더넷 카드는 랜 케이블로 이 허브에 연결이 되어 있습니다. 아래의 그림은 USB 랜카드를 이용하여 별도 구성한 네트워크와 IP 주소를 보여줍니다.

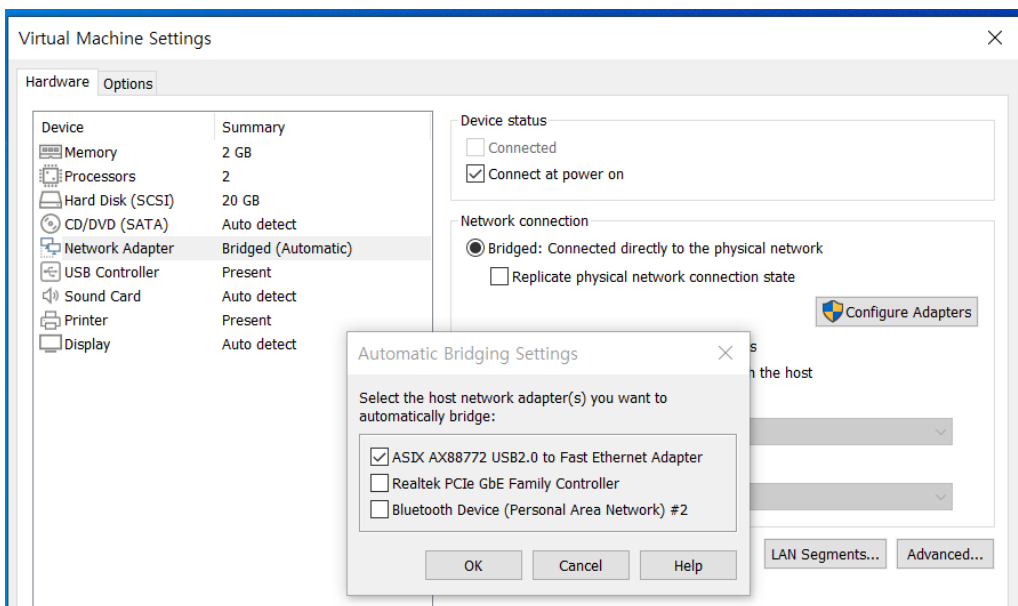


공유기는 DHCP(Dynamic Host Configuration Protocol) 서버의 기능을 가지고 있으므로 자동으로 IP 주소를 할당하지만, 이더넷 허브는 그런 기능이 없이 단순히 물리적으로 연결을 합니다. 따라서 이 경우에는 Windows와 Linux 호스트, 라즈베리 파이의 IP 주소를 수동으로 설정해야 합니다. 위의 그림은 이와 같이 설정된 주소를 보여 줍니다. 따라서 실습실에서 SmartTTY를 이용하여 라즈베리 파이를 연결할 때는 고정 IP 주소 192.168.1.122를 사용해서 접속합니다.

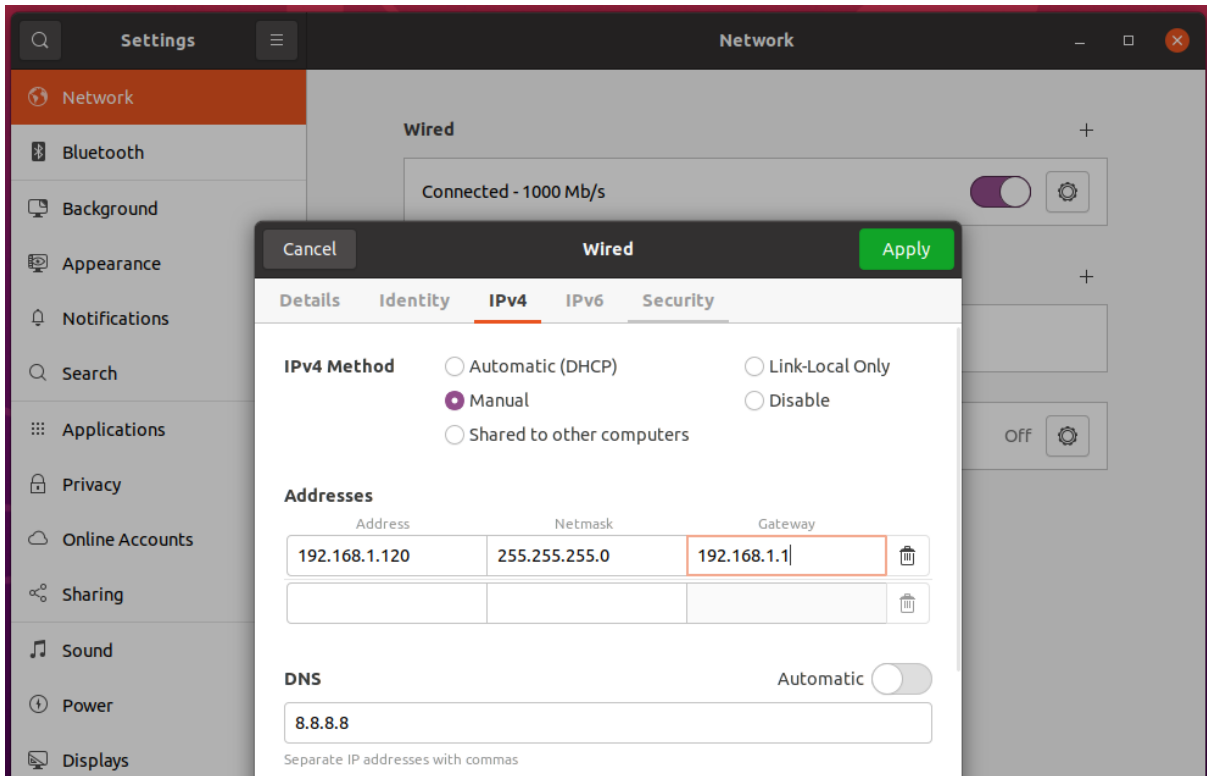
또한 위와 같이 구성할 경우 USB 이더넷 카드로 구성한 네트워크에는 인터넷이 연결되어 있지 않습니다. Ubuntu와 라즈베리 파이를 인터넷에 연결하기 위해서는 인터넷이 연결된 본체의 이더넷 설정에서 공유 기능을 활성화 합니다. 이때 주의할 점은 공유 기능을 활성화 할 때 USB 이더넷 랜 카드의 IP 주소를 임의로 바꾸게 되므로 반드시 IP 주소를 확인하고 변경해야 합니다.



또한 VMware의 네트워크 구성에서 네트워크 어댑터가 아래 그림과 같이 USB 랜 카드가 선택이 되도록 설정이 되어야 합니다.



또한 Ubuntu의 IP 주소는 아래와 같이 수동으로 설정이 되어야 합니다.



과제:

이 과제는 LED 드라이버와 Button 드라이버(Polling)를 사용합니다. 인터럽트는 사용하지 않습니다. 프로그램이 시작되면 LED를 켜고 버튼 입력을 기다립니다. 버튼을 1회 누르면 느린 속도(1초에 1번 정도)로 깜박입니다. 여기에서 다시 버튼을 1회 누르면 빠른 속도(앞의 느린 속도와 구별이 될 정도의 속도)로 깜박입니다. 이때, 다시 버튼을 1회 누르면 처음 상태로 돌아가서 깜박이지 않고 계속 켜고 있습니다. 이와 같은 모드가 무한 반복 됩니다. 즉, 모두 3개의 모드(깜박이지 않음, 느리게 깜박임, 빠르게 깜박임)가 존재 하며 버튼에 의해서 순차적으로 모드가 바뀝니다. 이 과제는 다음의 두가지 방법으로 구현합니다.

- (1) Thread를 사용하지 않고 하나의 프로그램으로 구현하십시오. 즉, single-process single-thread 방식으로 구현하십시오.
- (2) 반드시 2개의 thread를 사용하십시오. 버튼 입력 thread 와 LED를 켜는 thread 등을 구분 하여 구현하는 방법 등이 가능합니다.

리눅스에서 사용 가능한 딜레이 함수는 초 단위의 sleep, 마이크로 초 단위의 usleep 를 사용할 수 있습니다. 다음과 같이 헤더 파일을 포함합니다.

```
#include <unistd.h>
```

딜레이 함수는 딜레이 시간을 입력으로 받아들입니다. 즉, sleep(1)는 1초, usleep(1)는 1 마이크로

초의 딜레이 입니다.