

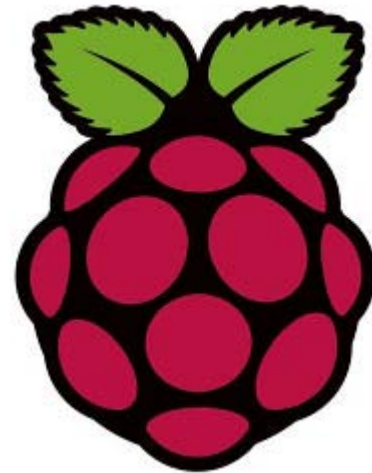
라즈베리파이 시작하기.

리눅스 배우기 입문편

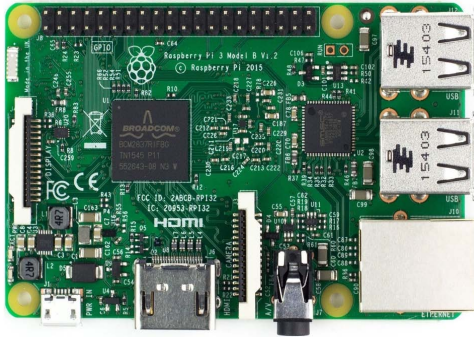

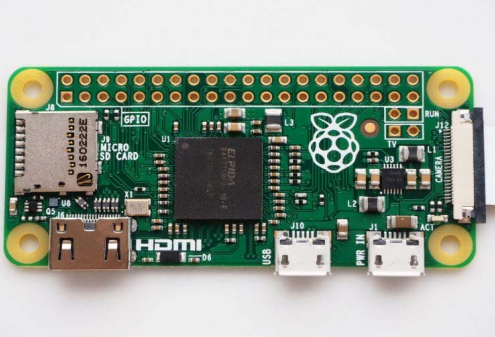
RASPBERRY PI!

라즈베리파이란

- 라즈베리파이란 영국 라즈베리파이 재단에서 교육용 저가 PC보급 프로젝트에 의해 탄생한 초소형 저가 PC
- PC이기 때문에 간단한 서버로도 운용 가능
- GPIO를 이용한 센서 및 하드웨어 제어 가능

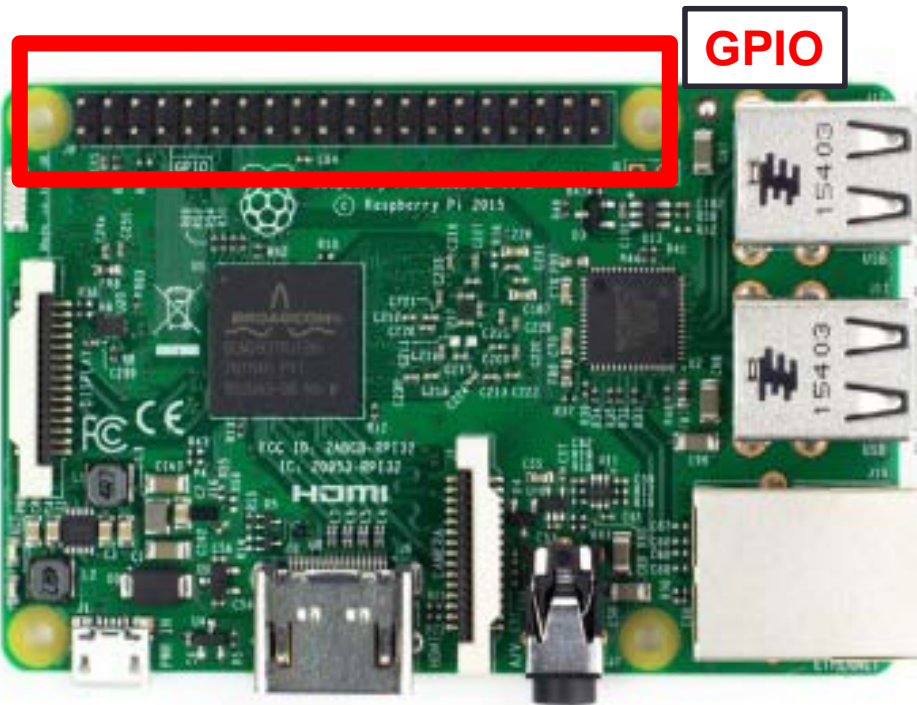


라즈베리파이 버전 별 사양

	Pi3	Pi2	PI ZERO
Image			
SOC	BCM2837	BCM2836	BCM2835
CPU	Quad Cortex A53 1.2GHz	Quad Cortex A7 900MHz	ARM11 700MHz
RAM	1GB SDRAM	1GB SDRAM	512MB SDRAM
GPU	400MHz Video Core IV	250MHz Video Core IV	250MHz Video Core IV
Ethernet	10/100Mbps	10/100Mbps	NONE

라즈베리파이 GPIO

- GPIO(General-purpose input/output)
- 라즈베리파이에 달린 입출력제어용 핀
- 일반 GPIO핀을 이용해서 입출력 제어 가능
- SPI, I2C, UART통신을 지원



Pi 2											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
		3.3v			1	2		5v			
2	8	SDA.1	ALT0	0	3	4		5V			
3	9	SCL.1	ALT0	1	5	6		0v			
4	7	GPIO. 7	IN	1	7	8	1	ALT0	TxD	15	14
		0v			9	10	1	ALT0	RxD	16	15
17	0	GPIO. 0	IN	0	11	12	1	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	1	13	14		0v			
22	3	GPIO. 3	IN	1	15	16	1	IN	GPIO. 4	4	23
		3.3v			17	18	0	IN	GPIO. 5	5	24
10	12	MOSI	ALT0	0	19	20		0v			
9	13	MISO	ALT0	0	21	22	0	IN	GPIO. 6	6	25
11	14	SCLK	ALT0	0	23	24	1	OUT	CE0	10	8
		0v			25	26	1	OUT	CE1	11	7
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31	1
5	21	GPIO.21	IN	1	29	30		0v			
6	22	GPIO.22	IN	0	31	32	0	IN	GPIO.26	26	12
13	23	GPIO.23	IN	1	33	34		0v			
19	24	GPIO.24	IN	0	35	36	0	IN	GPIO.27	27	16
26	25	GPIO.25	IN	0	37	38	0	IN	GPIO.28	28	20
		0v			39	40	0	IN	GPIO.29	29	21

쉴드(하드웨어 확장 보드).

하드웨어를 설계할
필요가 없다!



www.icbanq.com/

www.leocom.kr/

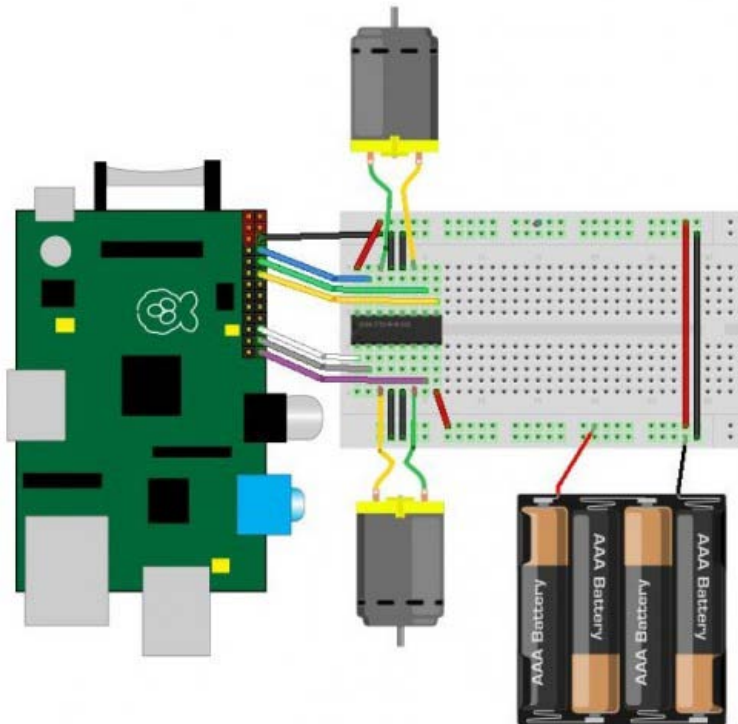
kr.element14.com

www.devicemart.co.kr/

www.eleparts.co.kr

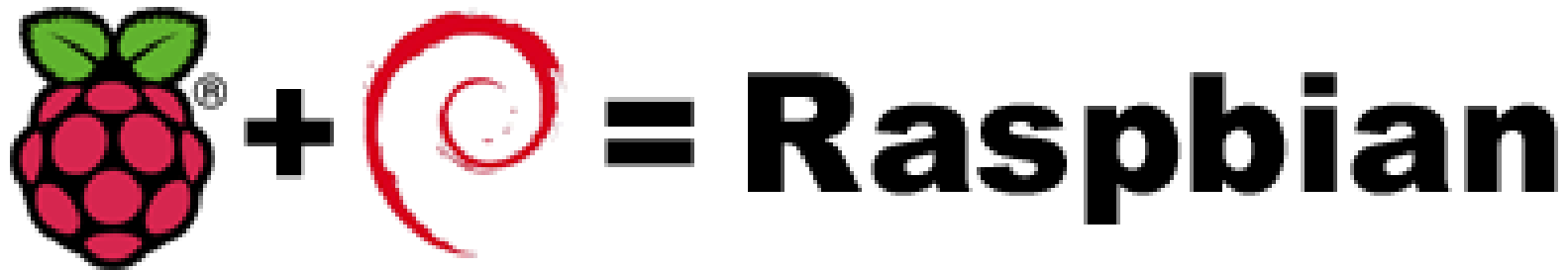
GPIO 활용.

- http://www.icbanq.com/pblogger/board_View.aspx?number=362



라즈베리파이 OS – 라즈비안 소개

- 라즈베리파이에서 가장 많이 사용하는 운영체제
- 라즈베리파이 재단에서 권장하는 리눅스 배포판
- 데비안 리눅스를 라즈베리파이 시스템에 포팅한 운영체제
- SW관련해서 타기업의 지원을 받고 있으며
MATHEMATICA, Minecraft등 유료 소프트웨어도
라즈베리파이 버전으로 무료 사용이 가능하다



RPI 유틸리티 살펴보기 – 사용 가능언어

- C, C++, C#, JAVA, PYTHON등 여러 언어를 지원한다.
- 터미널에서 텍스트 편집기를 통해 코드 작성이 가능하다
- GCC, JDK등을 사용하여 개발이 가능하다.
- 라즈베리파이 전용 라이브러리도 많이 제공한다.



RPI 유틸리티 살펴보기 – wiringPi

- 라즈베리파이 GPIO 제어를 쉽게 도와주는 유틸리티
- gpio 명령어를 이용한 간단한 제어가능
- C, C++용 wiringPi 라이브러리를 제공

The logo for Wiring Pi, featuring the text "Wiring Pi" in a blue, pixelated font. The letters are blocky and have a slight shadow effect, giving it a retro, digital appearance. The text is centered within a light gray rectangular background.

RPI 유틸리티 살펴보기 – 파이썬

- Guido van Rossum 이개발한 인터프리터 언어
- 공동 작업과 유지보수가 매우 쉽고 편함
- 간결하고 문법이 쉽다
- 개발 속도가 빠르다
- 외국에서는 교육 뿐만 아니라 실무에서도 많이 쓰이고 있다



RPI 유틸리티 살펴보기 – OPENCV

- OpenCV(Open Computer Vision)란 오픈 소스 컴퓨터 비전 C라이브러리다.
- 얼굴인식, 사물인식 등 사진이나 영상 매칭에 쓰인다
- 사용자가 직접 인식 라이브러리를 만들 수 있다.
- Pi CAM, USB 웹 캠을 이용해서 카메라 이용이 가능하다
- openCV라이브러리 패키지 설치 후 이용 가능하다



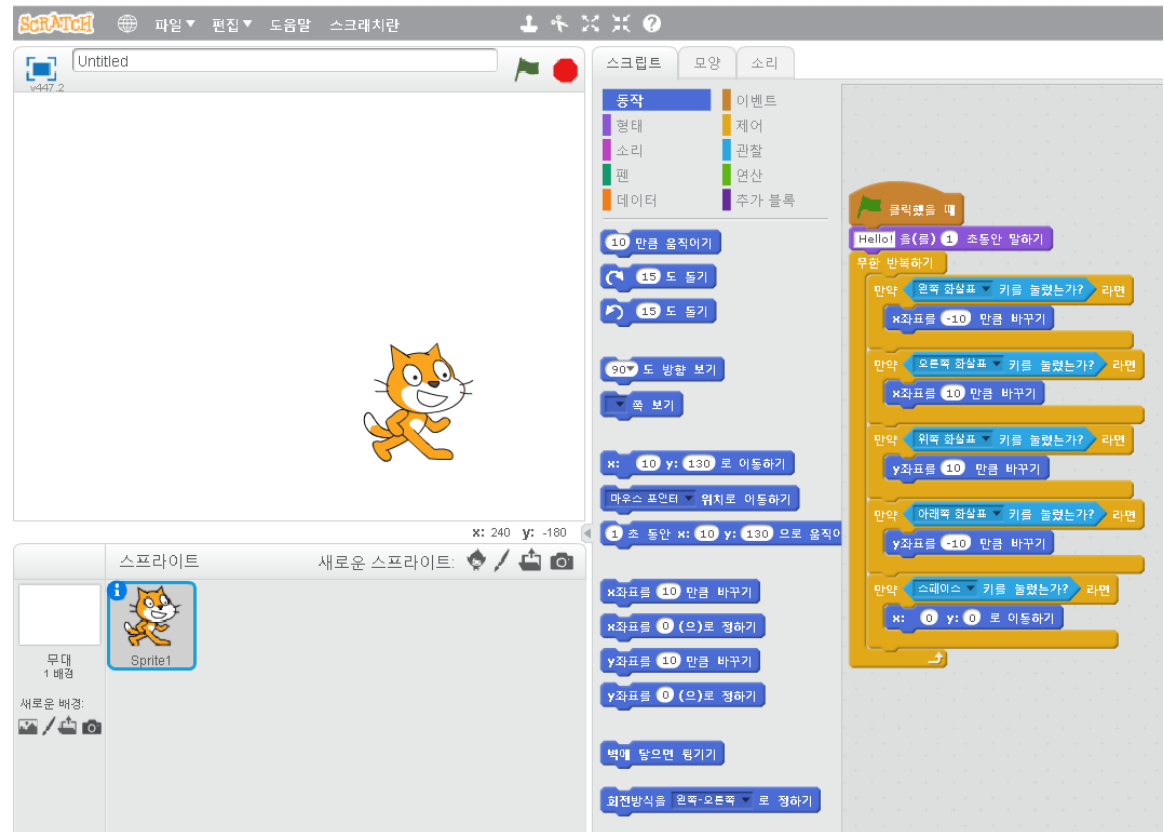
RPI 유틸리티 살펴보기 – gstreamer

- 스트리밍을 쉽게 처리할 수 있게 만든 오픈소스 프레임 워크
- 파이프라인 기반의 멀티미디어 프레임 워크
- 다양한 코덱과 플러그인을 제공
- 플러그인을 사용하여 어플리케이션을 만들 수 있는 API 제공



RPI 유틸리티 살펴보기 - 스크래치

- MIT에서 발표한 교육용 프로그래밍 언어
- 블록을 조립 하듯 코딩을 할 수 있어 초보자에게 적합하다



라즈베리 파이 살리기.

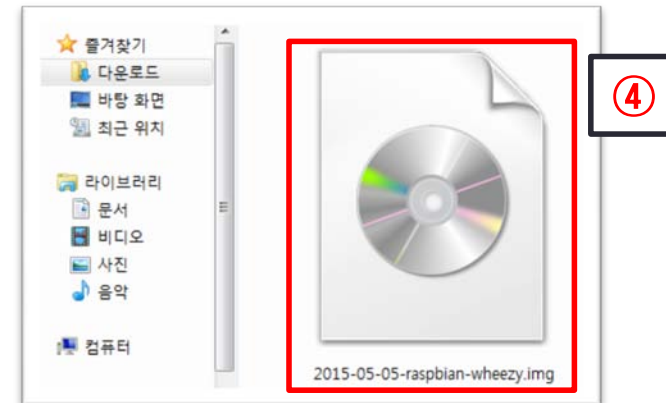
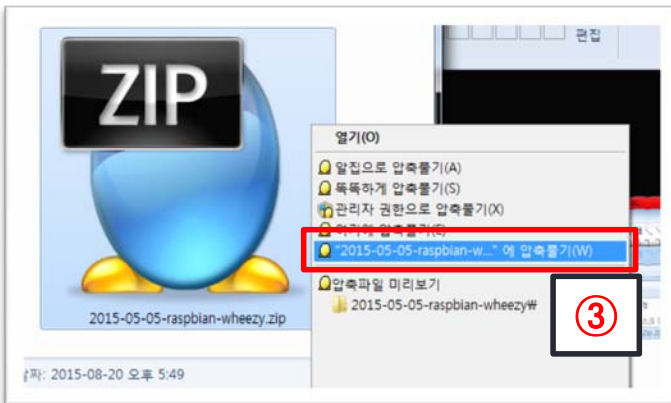
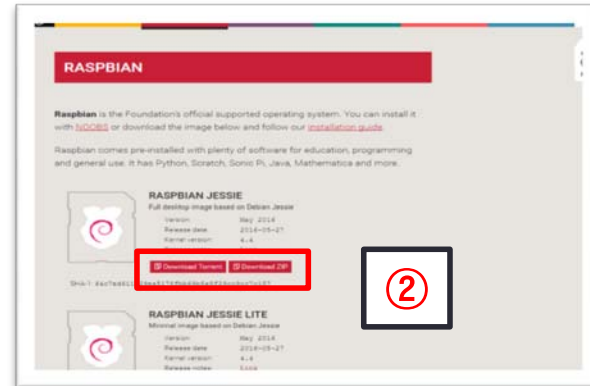
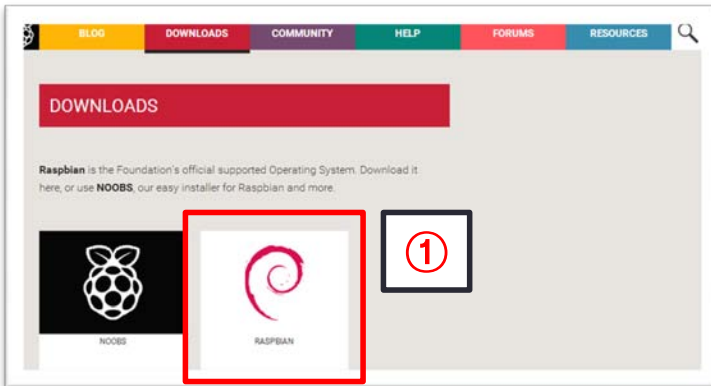
라즈비안 설치

라즈비안 설치 방법

- 라즈베리 파이 공식홈페이지 다운로드 탭에서 (<https://www.raspberrypi.org/downloads/>) RASPBIAN 이미지 파일을 다운받은 후 압축을 해제한다.
- SD Formatter 프로그램을 다운받은 뒤 (https://www.sdcard.org/downloads/formatter_4/) 디스크 이름을 잘 확인해서 SD카드를 포맷한다.
- 포맷이 완료된 SD카드에 라즈비안 이미지 파일을 씩운다. Win32 Disk Imager 프로그램을 설치한 뒤 (<http://sourceforge.net/projects/win32diskimager/>) 이미지 파일을 선택하고 디스크 이름을 확인한 뒤 White로 씩워준다.

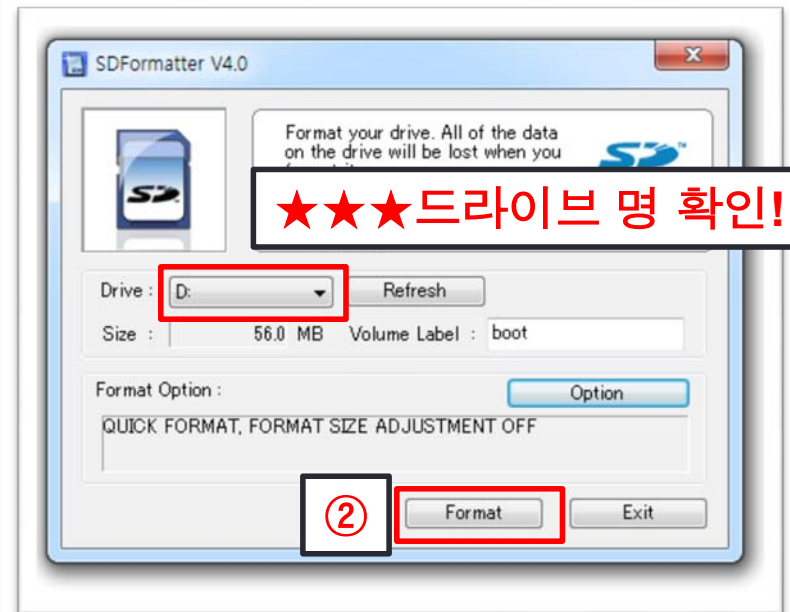
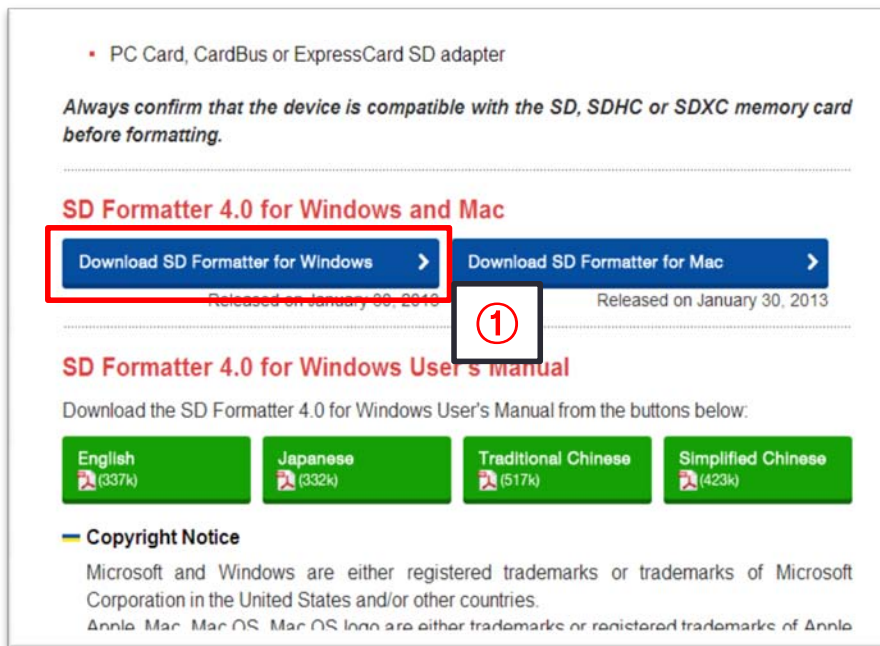
라즈비안 설치(1)-이미지 다운로드

- 라즈베리 파이 공식홈페이지 다운로드 탭에서 (<https://www.raspberrypi.org/downloads/>) RASPBIAN 이미지 파일을 다운로드 한 뒤 압축을 해제한다.



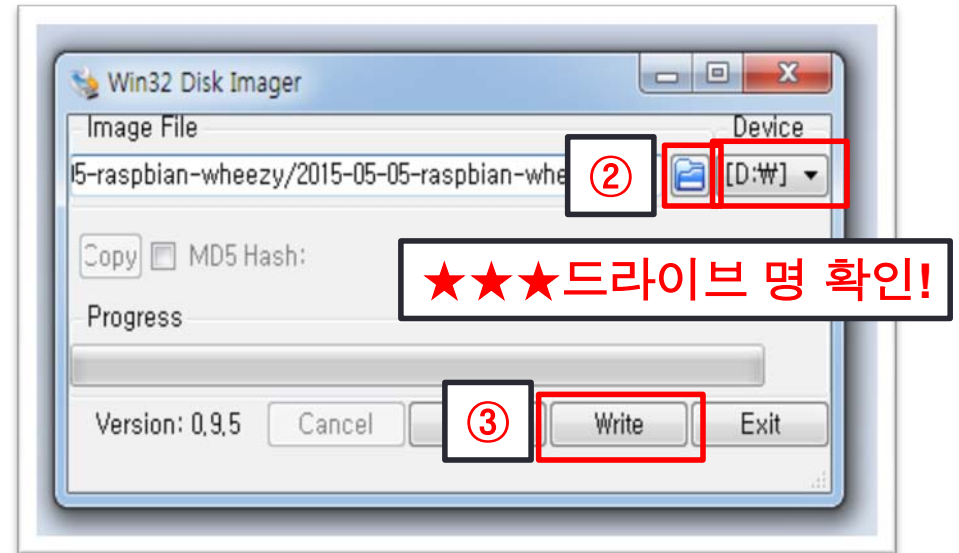
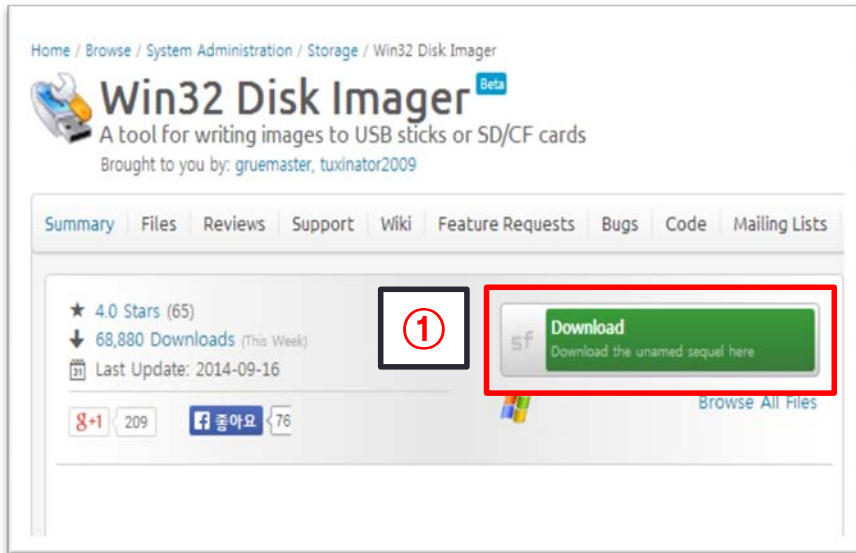
라즈비안 설치(2)-SD카드 포맷하기(1)

- SD Formatter 프로그램을 다운받은 뒤 (https://www.sdcard.org/downloads/formatter_4/) 디스크 명을 잘 확인해서 SD카드를 포맷한다. (8GB이상 권장)



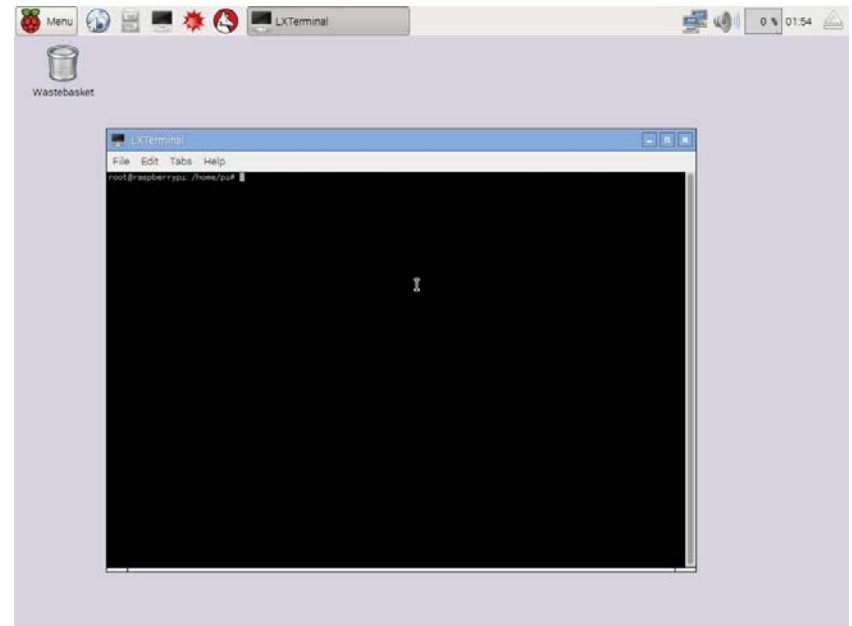
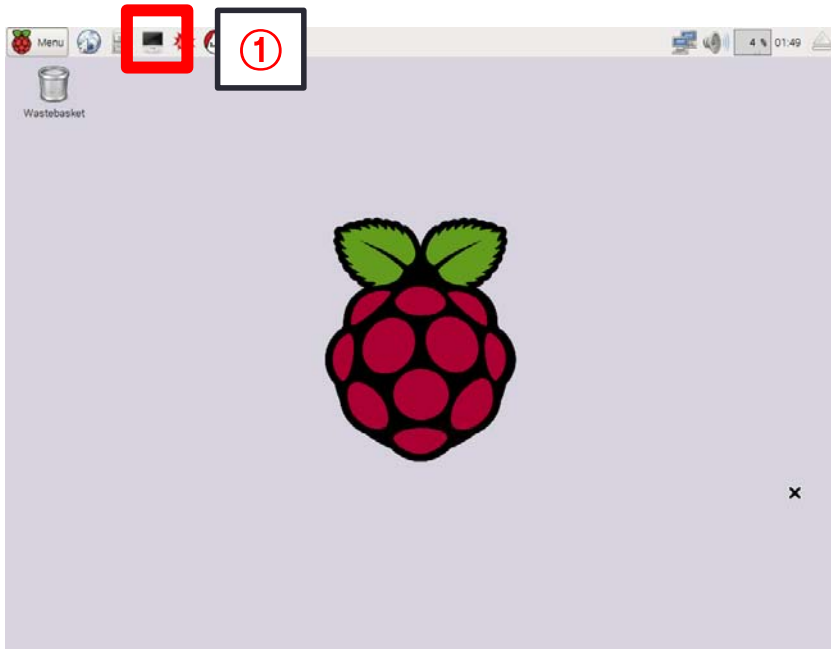
라즈비안 설치(3)-이미지 쓰기

- 포맷이 완료된 SD카드에 라즈비안 이미지 파일을 씩운다.
Win32 Disk Imager 프로그램을 설치한 뒤
(<http://sourceforge.net/projects/win32diskimager/>) 이미지 파일을 선택하고 디스크 명을 확인한 뒤 Write로 씩워준다.



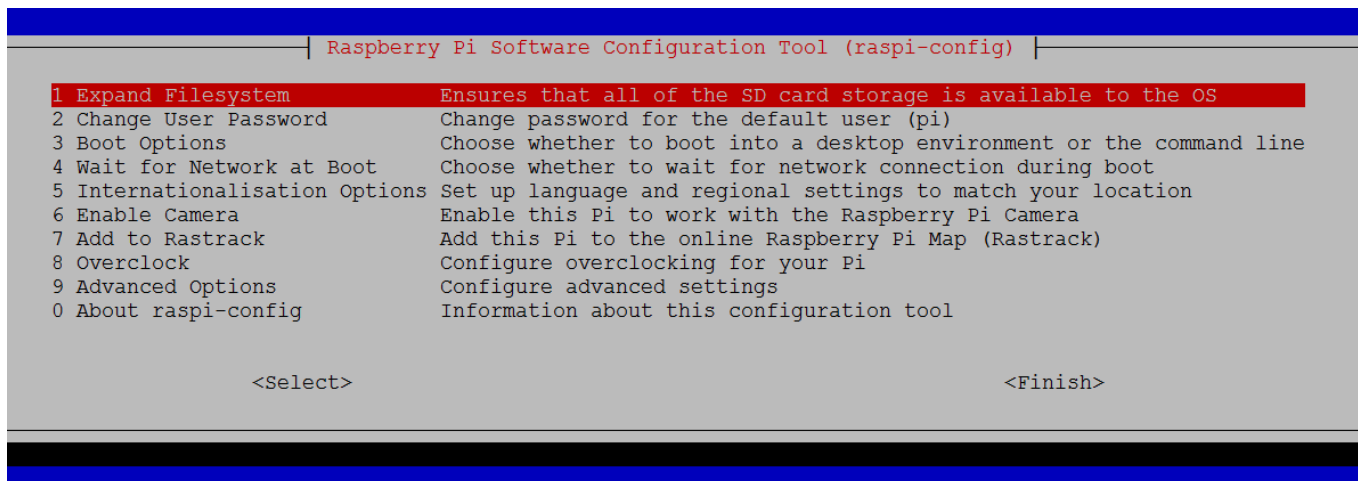
라즈베리파이 사용하기 – 초기 부팅화면

- 기본 초기 부팅화면은 X-window 화면으로 부팅된다.
- 좌측 상단의 X-window를 켜준 뒤 기본 설정을 해준다.
- `sudo raspi-config`



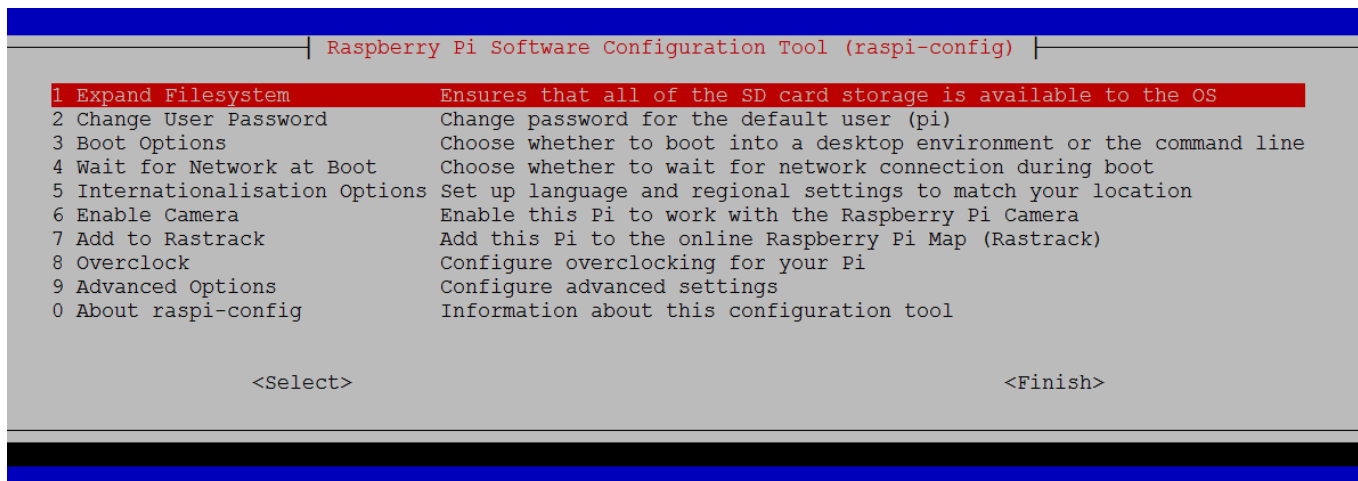
라즈베리파이 사용하기 – 설정메뉴(1)

- 1. Expand filesystem : 파일시스템의 크기를 SD카드의 용량에 맞춰서 최대로 늘려준다.
- 2. Change User Password : pi계정의 비밀번호 변경
- 3. Boot option : 부팅 설정 변경 가능
- 4. Wait for Network at Boot : 네트워크 부팅 설정
- 5. Internationalisation Options : 나라별 환경 설정



라즈베리파이 사용하기 – 설정메뉴(2)

- 6. Enable Camera : 라즈베리파이 전용 카메라의 사용 설정
- 7. Add to Rastrack : Rastrack을 등록한다
- 8. Overclock : 오버클럭 설정
- 9. Advanced Options : SHH, SPI, I2C등을 설정
- 10. About raspi-config : raspi-config에 대한 설명



라즈베리파이 사용하기 – 기본 설정(1)

- 1. Expand Filesystem으로 파일시스템의 크기를 키운다.
- 5. Internationalisation Options에서 키보드 레이아웃 설정
 - I3 Change Keyboard Layout(키보드 연결 필수)->
 - Generic 105-key (Intl) PC->
 - Korean - Korean (101/104 key compatible)후 나머지 엔터키

라즈베리파이 사용 - 키보드 레이아웃 변경

- 키보드 연결 없이 SSH연결 후 레이아웃 변경 방법
- `sudo nano /etc/default/keyboard`
- 아래와 같이 내용 변경후 재부팅한다.

```
GNU nano 2.2.6   File: /etc/default/keyboard

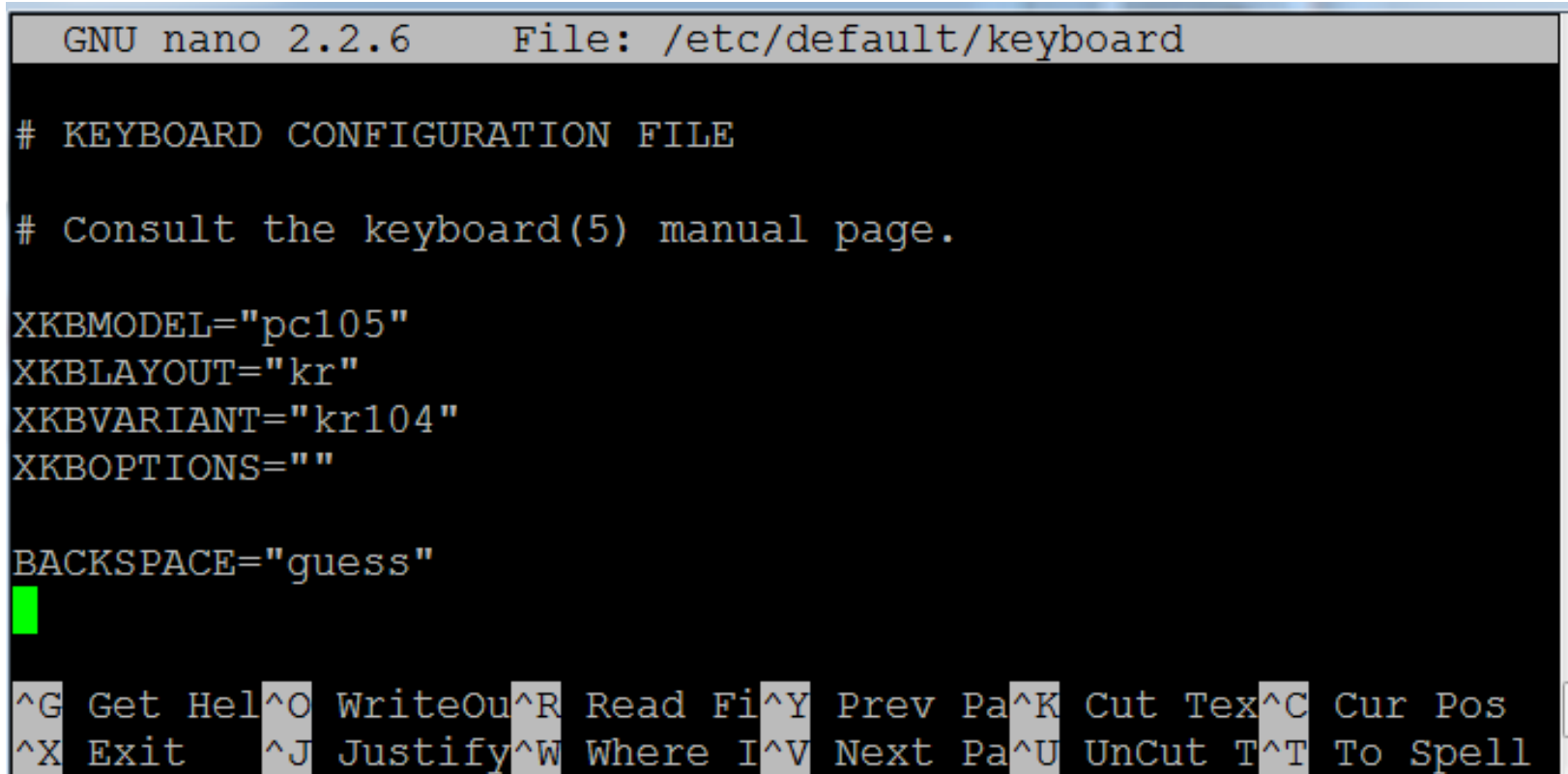
# KEYBOARD CONFIGURATION FILE

# Consult the keyboard(5) manual page.

XKBMODEL="pc105"
XKBLAYOUT="kr"
XKBVARIANT="kr104"
XKBOPTIONS=""

BACKSPACE="guess"

```



라즈베리파이 사용 – LAN케이블 연결

- 패키지 설치등 네트워크를 사용 하려면 인터넷 연결 필요
- 랜 케이블 연결, 와이파이를 이용해서 인터넷 사용 가능
- USB커넥터 옆의 RJ45커넥터에 LAN케이블 연결

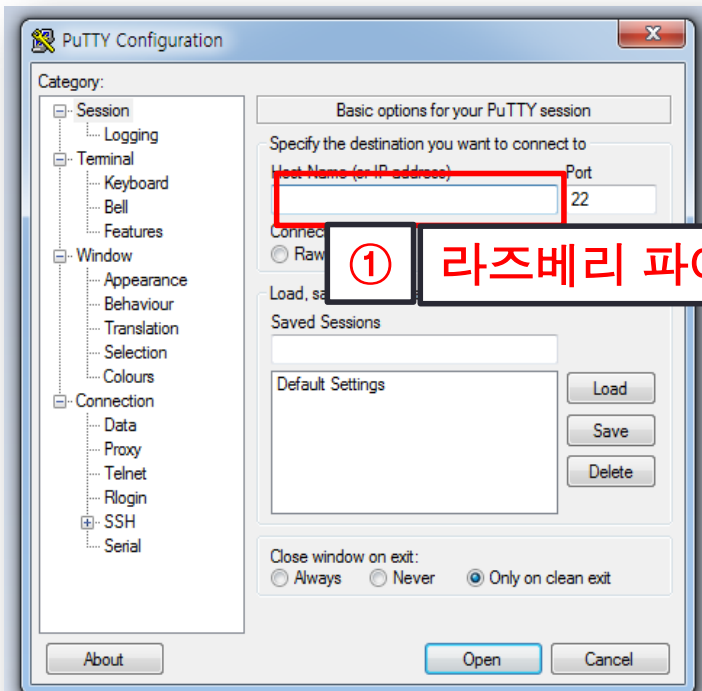


라즈베리파이 사용 – 고정IP 설정

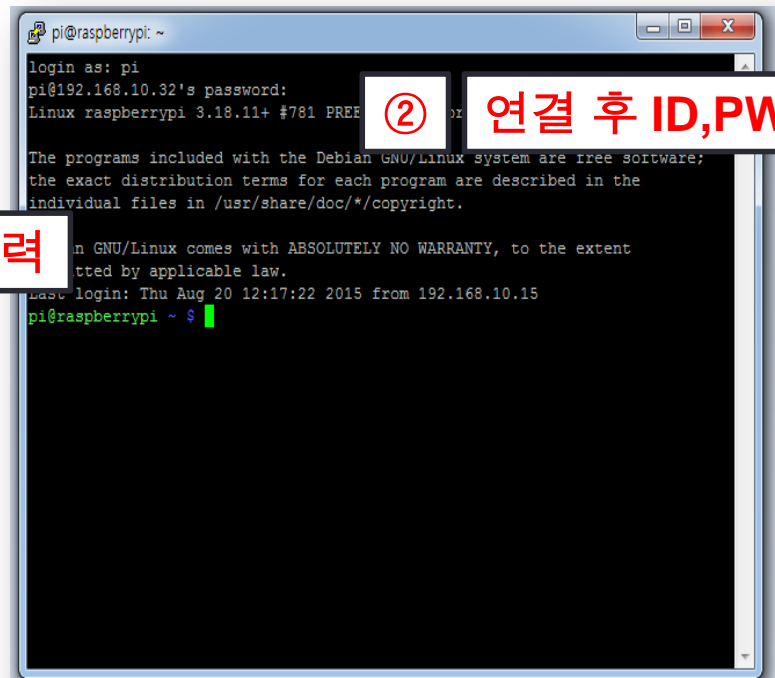
- `sudo nano /etc/network/interfaces`
- eth0 부분
 - `allow-hotplug eth0`
`iface eth0 inet static`
`address` 사용할 IP주소
`netmask 255.255.255.0`(서브넷 마스크)
`gateway` 사용할 IP주소
- 변경 후 저장
- 네트워크 재 시작
 - `ifdown eth0`
 - `ifup eth0`
- `Ifconfig` 명령어로 IP 변경여부 확인

라즈베리파이 사용 - PUTTY

- PUTTY는 SSH, 텔넷, TCP등을 위한 클라이언트
- 라즈베리파이는 기본적으로 SSH 사용이 가능
- 설치 없이 바로 실행
- <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>



① 라즈베리 파이 IP입력



② 연결 후 ID,PW입력

라즈베리파이 사용 – SAMBA(1)

- SAMBA란 리눅스에서 디렉터리를 네트워크에 공유시켜서 윈도우 공유 폴더같이 디렉터리 공유를 할 수 있게 해주는 패키지



라즈베리파이 사용 – SAMBA(2)

- 라즈베리 파이 업데이트 확인을 해 준다.

```
pi@raspberrypi ~ $ sudo apt-get update
Hit http://raspberrypi.collabora.com wheezy Release.gpg
Hit http://mirrordirector.raspbian.org wheezy Release.gpg
Hit http://archive.raspberrypi.org wheezy Release.gpg
```

삼바 관련 패키지를 설치해준다.

```
pi@raspberrypi ~ $ sudo apt-get install samba samba-common-bin
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

- 삼바에 접근 시 사용할 로그인 계정을 등록한다.

```
pi@raspberrypi ~ $ sudo smbpasswd -a pi
New SMB password:
Retype new SMB password:
pi@raspberrypi ~ $
```

라즈베리파이 사용 – SAMBA(3)

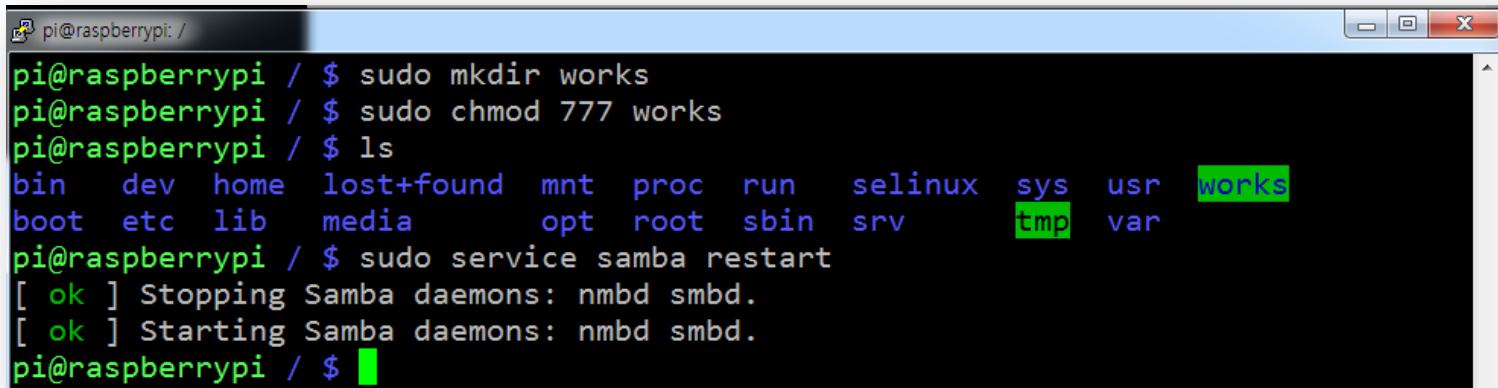
- `sudo vi /etc/samba/smbconf` 설정 파일을 수정한다

```
; postexec = /bin/umount /cdrom
[pi]
comment = Rpi samba
path = /works
valid user = pi
writable = yes
browsable = yes
```

- [pi] 부터 내용을 추가한다.
- comment : 사용자에 대한 간단한 메모
- path : 공유할 폴더의 위치
- valid user : samba 디렉터리 접근 가능한 유저
- writable : 쓰기 가능 여부 확인
- browsable : 탐색 가능 여부 확인

라즈베리파이 사용 – SAMBA(4)

- path 폴더 추가와 폴더의 권한을 변경해준 뒤 삼바를 재 시작 한다.

A terminal window titled 'pi@raspberrypi: /' showing the following commands and output:

```
pi@raspberrypi / $ sudo mkdir works
pi@raspberrypi / $ sudo chmod 777 works
pi@raspberrypi / $ ls
bin  dev  home  lost+found  mnt  proc  run  selinux  sys  usr  works
boot  etc  lib  media      opt  root  sbin  srv      tmp  var
pi@raspberrypi / $ sudo service samba restart
[ ok ] Stopping Samba daemons: nmbd smbd.
[ ok ] Starting Samba daemons: nmbd smbd.
pi@raspberrypi / $
```

- 윈도우 키+R 로 실행창을 띄운 뒤 \\라즈베리IP 를 입력하고 삼바가 잘 동작하고 있는지 확인한다.

리눅스 기본 명령어(1)

- sudo(substitute user do)
 - `$ sudo <옵션> <명령어>`
 - 슈퍼 유저(root) 권한으로 명령어 실행
 - `-s` : 현재 사용 계정을 루트 계정으로 전환
 - ex) `sudo nano main.c`, `sudo -s`
- pwd(print working directory)
 - 현재 디렉터리의 위치를 표시

리눅스 기본 명령어(2)

- ls (list segment)

- \$ ls <옵션> <표시할 위치>
- 디렉터리의 내용을 표시, 표시할 위치 인자 없을 경우 현재 디렉터리
- -a : 디렉터리 내 모든 내용 표시
- -l : 파일정보 자세히 표시
- -h: -l에서의 바이트 정보를 읽기 쉽게 표시
- ex) ls -ahl, ls -ahl /etc/network

- ln (link)

- \$ ln <옵션> <원본파일명> <링크파일명>
- 파일의 링크 생성
- -s 심볼릭 링크 파일(바로가기) 생성
- ex) ln test.c test.hd , ln -s test.c test.sl

리눅스 기본 명령어(3)

- `cd(change directory)`
 - `$cd <경로>`
 - 디렉터리 이동 명령어, 절대경로 또는 상대경로를 이용하여 이동
 - 절대경로 : 최상위 디렉터리 루트(/)디렉터리 부터 표시
 - 상대경로 : 현재 디렉터를 기준으로 ..(하위 디렉터리), .(현재 디렉터리) 를 이용해서 경로 표시
 - ex)(절대경로) `cd /home/pi`, (상대경로) `cd ../pi`
- `mkdir(make directory)`
 - `$ mkdir <디렉터리명>`
 - 디렉터리 생성 명령어
 - 두 개 이상의 디렉터리 생성시 디렉터리명을 여러 개 써주면 된다
 - ex) `mkdir code`

리눅스 기본 명령어(4)

- **rm(remove)**
 - `$ rm <옵션> <파일, 디렉터리명>`
 - 파일, 디렉터리 삭제 명령어
 - 두 개 이상의 파일 삭제 시 파일명을 여러 개 써주면 된다.
 - `-r` : 디렉터리의 경우 하위 경로의 파일들까지 삭제한다
 - `-f` : 파일, 디렉터리 삭제 시 사용자에게 묻지 않고 삭제한다
 - `-v` : 파일 삭제 정로를 표시한다
 - ex) `rm -rf tst tst.c`
- **rmdir(remove directory)**
 - `$ rmdir <옵션> <디렉터리명>`
 - 디렉터리 삭제 명령어
 - `-p` : 상위 디렉터리까지 삭제
 - ex) `rmdir test, rmdir -p test/test1/test2`

리눅스 기본 명령어(5)

- cp (copy)
 - \$ cp <옵션> <복사할 대상> <복사할 위치>
 - 파일, 디렉터리 복사 명령어
 - 두 개 이상 파일 복사 시 복사할 대상을 여러 개 써주면 된다
 - -r : 디렉터리 복사 시 하위 디렉터리 까지 전부 복사한다
 - -f : 사용자에게 확인 없이 복사한다.
 - ex) cp -r tst test/.
- cat (catenate)
 - \$cat <옵션> <파일명>
 - -n : 행번호와 함께 출력한다
 - ex) cat /etc/network/interfaces, cat -n /etc/network/interfaces

리눅스 기본 명령어(6)

- mv(move)
 - `$mv <옵션> <이동할 대상> <이동할 위치>`
 - 파일 이동 명령어
 - 두 개 이상의 파일 이동시 이동할 대상을 여러 개 써주면 된다.
 - 파일명 변경 시에도 사용
 - `-f` : 사용자에게 확인 없이 이동시킨다.
 - `-i` : 이동할 위치에 동일한 파일이나 디렉터리가 존재 할 경우 확인
 - ex) `mv test.c test.c.bak, mv -f test.c test.c.bak`

리눅스 기본 명령어(7)

- `chmod(change mode)`
 - `$ chmod <권한수준> <파일명>`
 - 파일의 권한을 변경하는 명령어
 - 권한 수준은 8진수 세자리로 구성
 - `<111><111><111>` 소유자, 그룹, 그외의 사용자들 순
 - 8진수 앞자리 부터 읽기, 쓰기, 실행 권한순
 - ex) `chmod 750 main.c`

라즈베리파이 사용하기 – nano(1)

- nano 텍스트 에디터는 리눅스에 익숙지 않은 초보자에게 쉽게 텍스트 편집을 할 수 있게 해주는 유틸리티다.

```

:::
iLE88Dj. :jD88888Dj:
.LGitE888D.f8GjjjL8888E;
iE :8888Et. .G8888.
:i E888, ,8888,
D888, :8888: 88888b. 8888b. 88888b. .d88b.
D888, :8888: 888 "88b "88b 888 "88b d88""88b
D888, :8888: 888 888 .d888888 888 888 888 888
D888, :8888: 888 888 888 888 888 888 Y88..88P
888W, :8888: 888 888 "Y888888 888 888 "Y88P"
W88W, :8888:
W88W: :8888:
DGGD: :8888: Text Editor Homepage
      :8888:
      :W888:
      :8888:
      E888i
      tW88D
```

라즈베리파이 사용하기 – nano(2)

- 파일 생성
 - nano <파일명>
- 파일 저장
 - Ctrl+o <ENTER>
- 빠져 나오기
 - Ctrl+x , 파일 변경 내역이 있을시 저장여부 확인함

라즈베리파이 사용하기 – nano(3)

- 파일 생성해보기
- mkdir example
- cd example
- nano hello.c
 - #include<stdio.h>
int main()
{
 printf("Hello World!\n");
 return 0;
}
- Ctrl + o <ENTER> 저장
- Ctrl + x로 빠져나옴
- cat hello.c 로 내용 확인

```
root@raspberrypi:/home/pi# mkdir example
root@raspberrypi:/home/pi# cd example/
root@raspberrypi:/home/pi/example# nano hello.c
root@raspberrypi:/home/pi/example# cat hello.c
#include<stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}

root@raspberrypi:/home/pi/example#
```

리눅스에서 C컴파일 하기 – gcc(1)

- gcc(GNU Compiler)
 - `$gcc <옵션> <컴파일 대상 코드> <옵션>`
 - 리눅스에서 기본으로 제공하는 컴파일러
 - `-o` 파일명 : 출력파일명을 지정해준다
 - `-c` : 출력 파일을 생성 안하고 `.o` 파일인 오브젝트 파일 생성
 - `-l<추가 라이브러리>` : 추가 라이브러리를 사용하여 컴파일
 - `-L<추가 라이브러리 위치>` : 추가 라이브러리의 위치 설정 가능
- 코드가 여러 개일 경우 컴파일 방법
 - `gcc <코드1> <코드2> ... <코드x>` 식으로 나열 후 컴파일

리눅스에서 C컴파일 하기 – gcc(2)

- 컴파일 해보기
 - `gcc -o hello hello.c`
 - `./hello`
 - 결과 확인

```
root@raspberrypi:/home/pi/example# gcc -o hello hello.c
root@raspberrypi:/home/pi/example# ./hello
Hello World!
root@raspberrypi:/home/pi/example#
```

동적 · 정적 라이브러리 생성(1)

- 동적 라이브러리
 - 라이브러리를 여러 프로그램에서 공유하며 사용
 - 메모리, 용량 절약, 라이브러리 업데이트 등의 유연성
 - 라이브러리 의존성에 따른 관리 필요
- 정적 라이브러리
 - 파일 생성 시 라이브러리 내용 포함하여 사용
 - 라이브러리 연동 과정이 필요 없음
 - 메모리 활용성이 떨어짐
- 동적 · 정적 라이브러리 생성시 ar유틸리티 이용

동적 · 정적 라이브러리 생성(2)

- 라이브러리 생성 실습
- 아래 내용의 파일 3개를 만든다

myfunc.h

```
1. extern void say_hello(void);
```

myfunc.c

```
1. #include<stdio.h>
2. #include"myfunc.h"
3. void say_hello(void)
4. {
5.     printf("Hello World!!\n");
6. }
```

hello.c

```
1. #include"myfunc.h"
2. int main(void)
3. {
4.     say_hello();
5.     return 0;
6. }
```

동적 · 정적 라이브러리 생성(3)

- 정적 라이브러리 생성 실습
- myfunc.o 오브젝트 생성
 - gcc -c myfunc.c
- 라이브러리 등록
 - ar r libmylib.a myfunc.o
 - ar t libmylib.a myfunc.o
- 컴파일 후 결과 확인
 - gcc -o my_lib hello.c -lmylib -L.
 - ./my_lib
 - Hello World!!

동적 · 정적 라이브러리 생성(4)

- 동적 라이브러리 생성 실습
- myfunc.c를 컴파일 하여 libmyfunc.so.1 생성
 - so(share object), 1 : 동적 라이브러리 버전 의미
- 오브젝트 파일 생성
 - gcc -fPIC -c myfunc.c
- 공유 라이브러리 파일 생성
 - gcc -shared -Wl,-soname,libmyfunc.so.1 -o libmyfunc.so.1 myfunc.o

동적 · 정적 라이브러리 생성(5)

- 심볼릭 링크 파일 생성후 기본 라이브러리 폴더에 복사
 - `ln -s libmyfunc.so.1 libmyfunc.so`
 - `cp -apf libmyfunc.so* /lib`
- hello.c 컴파일
 - `gcc -o hello hello.c -lmyfunc`
- 실행 후 확인
 - `./hello`

```
root@raspberrypi:/home/pi/example# gcc -shared -Wl,-soname,libmyfunc.so.1 -o libmyfunc.so.1 myfunc.o
root@raspberrypi:/home/pi/example# ln -s libmyfunc.so.1 libmyfunc.so
root@raspberrypi:/home/pi/example# cp -apf libmyfunc.so* /lib/
root@raspberrypi:/home/pi/example# gcc -o hello hello.c -lmyfunc
root@raspberrypi:/home/pi/example# ./hello
Hello World!!
root@raspberrypi:/home/pi/example#
```

WIRINGPI

WiringPi란?

- Raspberry Pi 보드를 위해 C 언어로 작성되어진 라이브러리 이다.
 - `#gpio readall`
- 하드웨어 접근을 쉽게 해준다.
- C, C++, Node.JS 등을 지원한다.
- PiFace, Gertboard, PiGlow 등의 보드 지원.
- <http://wiringpi.com> 에서 운영되어지고 있다.
- GNU LGPLv3 라이선스를 사용한다.
- Git Hub 를 통해서 설치가 가능하다.
 - `# git clone git://git.drogon.net/wiringPi`

GPIO를 이용한 LED 제어

- GPIO제어 유틸리티인 wiringPi를 설치한다.
- GPIO핀 번호를 확인한 뒤 LED를 연결한다.
- GPIO명령어를 이용하여 LED를 제어해본다.
- C언어를 이용하여 LED를 제어해본다.

GPIO 사용해 보기 - wiringPi 설치

- wiringPi 설치
 - `git clone git://git.drogon.net/wiringPi`
 - `cd wiringPi`
 - `git pull origin`
 - `./build`
- `gpio -v` 명령어로 라즈베리파이 정보 확인 가능

```
root@raspberrypi:/home/pi# gpio -v
gpio version: 2.32
Copyright (c) 2012-2015 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
  Type: Pi 2, Revision: 01, Memory: 1024MB, Maker: Sony
  * Device tree is enabled.
  * This Raspberry Pi supports user-level GPIO access.
    -> See the man-page for more details
    -> ie. export WIRINGPI_GPIOMEM=1
root@raspberrypi:/home/pi#
```

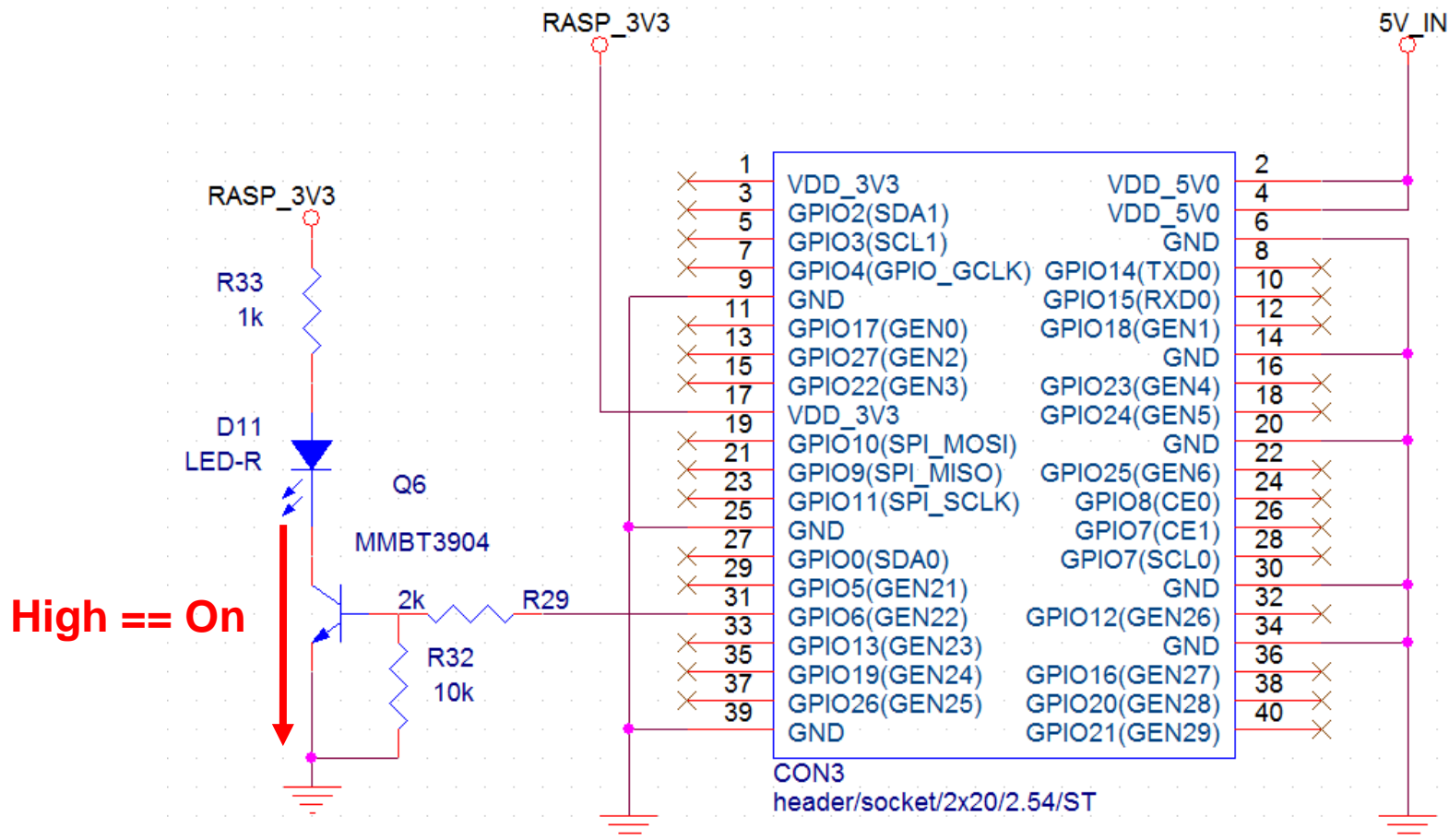
GPIO 사용해 보기 - wiringPi번호 확인

- `gpio readall` 명령어로 핀번호 확인가능
- wPi번호는 wiringPi에서 지정한 gpio 핀번호
- BCM은 메인 칩 데이터시트에 나와있는 칩 핀번호이다

[illegible]

GPIO 사용해보기 – LED 연결도

- 키트의 wPi 22번에 연결된 LED를 제어해 본다



GPIO 사용해보기 – gpio명령어(1)

- \$ gpio <옵션>
 - 파이 버전 확인 등 간단한 gpio 제어용 명령어
 - -v : 버전 확인
 - mode <핀번호> <모드>
해당 핀의 기능을 바꾼다
 - read <핀번호>
해당 핀의 논리상태 확인
 - write <핀번호> <상태>
output 상태의 핀의 논리 상태를 변경한다

GPIO 사용해 보기 – gpio명령어(2)

- gpio 명령어를 이용하여 LED를 켜보자
- gpio 22 번 핀 출력상태로 변경
 - gpio mode 22 output
- 0번핀 논리상태를 HIGH로 변경
 - gpio write 22 1 (ON), gpio write 22 0 (OFF)
- LED제어가 잘 되는지 확인

WIRINGPI LIBRARY

GPIO 사용해 보기 – gpio 제어 함수

- **wiringPiSetup (void) ;**

- This initialises wiringPi and assumes that the calling program is going to be using the wiringPi pin numbering scheme. This is a simplified numbering scheme which provides a mapping from virtual pin numbers 0 through 16 to the real underlying Broadcom GPIO pin numbers. See the pins page for a table which maps the wiringPi pin number to the Broadcom GPIO pin number to the physical location on the edge connector.

This function needs to be called with root privileges.

GPIO 사용해 보기 – gpio 제어 함수

- **void pinMode (int pin, int mode) ;**
 - This sets the mode of a pin to either INPUT, OUTPUT, or PWM_OUTPUT. Note that onlywiringPi pin 1 (BCM_GPIO 18) supports PWM output. The pin number is the number obtained from the pins table.
This function has no effect when in Sys mode.
- **void digitalWrite (int pin, int value) ;**
 - Writes the value HIGH or LOW (1 or 0) to the given pin which must have been previously set as an output.

GPIO 사용해 보기 – gpio 제어 함수

- **void delay (unsigned int howLong)**
 - This causes program execution to pause for at least howLong milliseconds. Due to the multi-tasking nature of Linux it could be longer. Note that the maximum delay is an unsigned 32-bit integer or approximately 49 days

GPIO 사용해 보기 – LED제어 코드

- nano led.c

```
1.  #include<stdio.h>
2.  #include<wiringPi.h>
3.  #define LED 22                                //LED 연결핀 설정
4.  int main(void)
5.  {
6.      wiringPiSetup();
7.      pinMode(LED, OUTPUT);                      //핀 모드 설정
8.      digitalWrite(LED,0);                       //LED 초기화
9.      while(1)
10.     {
11.         digitalWrite(LED,1);
12.         delay(1000);
13.         digitalWrite(LED,0);
14.         delay(1000);
15.     }
16.     return 0;
17. }
```

GPIO 사용해 보기 – LED제어해 보기

- 컴파일
 - `gcc -o led led.c -lwiringPi`
- 파일 실행
 - `./led`
- LED가 잘 깜빡이는지 확인한다.

SEE U!

THE END