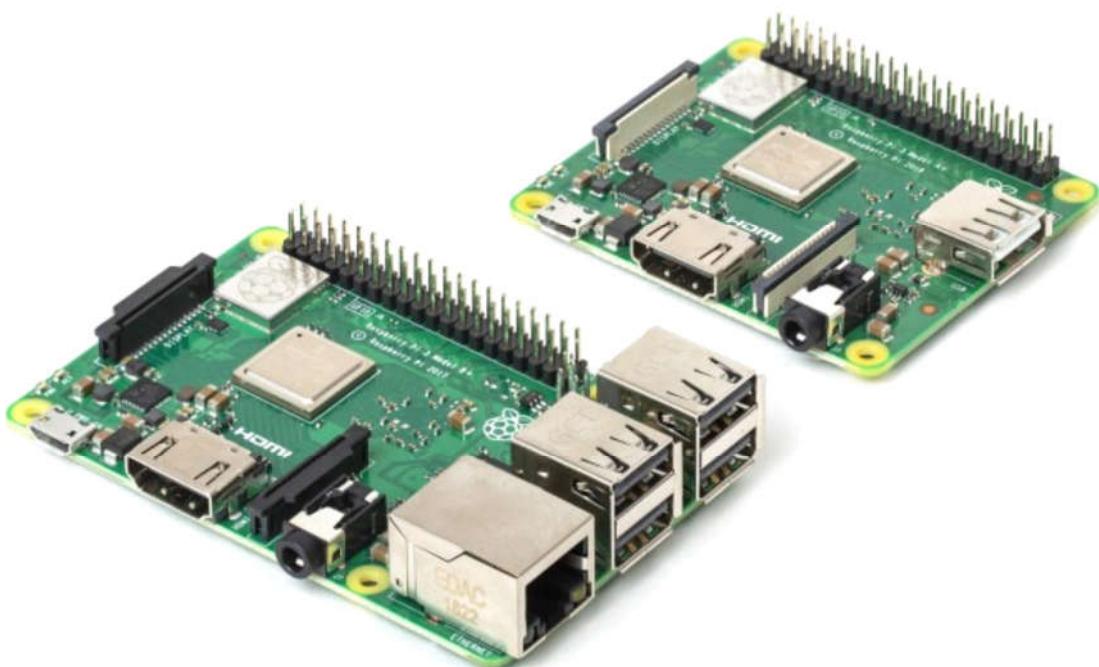


# 라즈베리파이를 이용한 IoT 시스템 구현



<https://blog.naver.com/thumbdown>  
<https://github.com/greattoe>

# 라즈베리파이를 활용한 IoT 시스템 구현

1. 라즈베리파이(RaspberryPi)는.....	4
1.1. 라즈베리파이 버전 .....	4
1.1.1. 모델 B 시리즈 .....	4
1.1.2. 모델 A 시리즈 .....	5
1.1.3. 모델 Zero 시리즈.....	6
1.2. 지원되는 운영체제 .....	6
2. 라즈베리파이 시작하기 .....	7
2.1. 준비할 것들 .....	7
2.1.1. 하드웨어 .....	7
2.1.2. 소프트웨어.....	7
2.2. 시스템 디스크 작성 .....	7
2.2.1. Win32DiskImager 다운로드 및 설치 .....	7
2.2.2. RASPBIAN 다운로드 .....	8
2.2.3. 시스템 디스크 작성 .....	8
2.3. raspi-config 를 이용한 시스템 기본 설정 .....	10
2.3.1. Change User Password .....	11
2.3.2. Network Options.....	11
2.3.3. Boot option .....	12
2.3.4. Localisation option.....	14
2.3.5. Interfaceing option .....	16
2.3.6. Overclock.....	18
2.3.7. Advanced Option.....	18
2.3.8. Update .....	19
2.3.9. About raspi-config .....	19
2.4. GUI 에서의 설정 .....	19

2.4.1. Raspberry Pi Configuration.....	20
2.4.2. WiFi 설정 .....	20
2.4.3. Bluetooth 설정.....	22
2.4.4. 한글 사용을 위한 설정 .....	24
3. LINUX 기본 .....	28
3.1. 기초 리눅스 명령어 및 용어 .....	28
3.1.1. 가상 터미널 .....	28
3.1.2. 사용자 계정 관련 명령어.....	30
3.1.3. 파일 및 디렉토리 관련 명령어 .....	33
3.1.4. 시스템 종료 명령 .....	38
3.2. 리눅스 네트워크 기초 .....	38
3.2.1. 기본 개념 및 용어 .....	38
3.2.2. 네트워크 관련 설정파일 .....	40
4. 원격접속(Remote Access) 환경 설정 .....	41
4.1. SSH(Secure Shell) 원격접속 .....	41
4.2. GUI 원격접속 by VNC(Virtual Network Computing) .....	42
4.3. "Notepad++"를 이용한 원격 개발환경 .....	43
5. GPIO(General Purpose I/O) 포트 제어 .....	46
5.1. 라즈베리파이의 GPIO 포트.....	46
5.2. GPIO 제어 프로그래밍.....	47
5.2.1. GPIO 출력 .....	48
5.2.2. GPIO 입력 .....	50
5.2.3. 동작 검지 센서 .....	55
5.2.4. 버저( Buzzer ).....	56
5.2.5. 초음파 센서( HC-SR04 ).....	58
5.2.6. 온습도 센서( DHT11 ) .....	61

5.2.7. 서보( Servo ) 모터 제어하기 .....	64
6. Web 을 통한 GPIO 제어.....	69
6.1. HTML, Javascript, Node.js .....	69
6.1.1. 간단히 살펴보는 HTML 과 Javascript .....	69
6.1.2. Node.js.....	76
6.2. GPIO 제어 웹서버 구현 .....	77
6.2.1. LED 제어 웹 서버 .....	77
6.2.2. 스위치 입력에 의한 웹페이지 변경.....	83
6.3. GPIO 음성제어.....	86
6.3.1. 음성 입력장치 설정 .....	86
6.3.2. 음성 명령으로 LED 제어하기.....	88

# 1. 라즈베리파이(RaspberryPi)는...

현 라즈베리파이재단 이사 에반 업튼( Eben Upton )이 캠브리지 대학 컴퓨터 공학부 재직 당시, 학생들의 시스템에 대한 이해를 높이고자 시작한 프로젝트에서 비롯된 SBC( Single Board Computer )이며, 다음과 같은 특징을 가지고 있다.

- Intel 의 x86 계열이 아닌 ARM 프로세서
- MS-Windows 아닌 LINUX 운영체제
- 사용자 프로그램 제어 가능한 H/W 제공
- 저렴한 가격( \$35 )

사용자가 제어하고자 하는 H/W 를 연결하여, 프로그램으로 제어할 수 있는 저가의( \$35 ) 손바닥만한 PC 가 바로 '라즈베리파이'라고 할 수 있다. 이런 특징을 이용한 수 많은 사용자 컨텐츠가 인터넷을 통해 전해지면서, '라즈베리파이'는 2012년 2월 29일 첫 출시 이후, 2019년 3월 집계 누적 판매량 2천 5백만대를 넘어선 것으로 알려졌다. 2018년 3월 14일에는 '라즈베리파이 3 모델 B+'가 출시되어 현재까지 판매되고 있다.

## 1.1. 라즈베리파이 버전

2012년 2월 29일 라즈베리파이 1 세대 모델 B 와 모델 A 출시를 시작으로 2018년 3월 14일(Pi-Day) 라즈베리파이 3 모델 B+가 출시되어 현재까지 판매되고 있다. 스펙도 초기 모델의 32bit ARM11 Single-Core 600MHz / 512MB RAM 에서 64bit ARM Cortex-A53 Quad-Core 1.4GHz / 1GB RAM( 라즈베리파이 3 모델 B+ )으로 성능면에서도 큰 변화가 나타났다.

출시일	모델
2012년 02월 29일	RaspberryPi Model B
	RaspberryPi Model A
2014년 07월 14일	RaspberryPi Model B+
2014년 11월 10일	RaspberryPi Model A+
2015년 02월 02일	RaspberryPi 2 Model B
2015년 11월 26일	RaspberryPi Zero
2016년 02월 29일	RaspberryPi 3 Model B
2017년 09월 22일	RaspberryPi Zero W
2018년 03월 14일	RaspberryPi 3 Model B+
2018년 11월 15일	RaspberryPi 3 Model A

### 1.1.1. 모델 B 시리즈

Model	SoC	RAM	VIDEO	ISA	Ethernet	USB	WiFi	BT
 Raspberry Pi Model B	BCM2835 Arm-11 600MHz Single	512 MB	Broadcom Dual Video CoreIV®	ARMv6	10/100M 1 Port	USB2.0 2 Port	x	x

	( 32bit )							
	BCM2836 Coretex-A7 900MHz Quad ( 32bit )		Broadcom Dual Video CoreIV®	ARMv7				
	BCM2837 Coretex-A53 1.2GHz Quad ( 64bit )	1GB		ARMv8		USB2.0 4 Port	802.11n 4.1 BLE	
	BCM2837 Coretex-A53 1.4GHz Quad ( 64bit )			ARMv8			5, 2.5G Dual Band 802.11 b/g/n/ac	4.2 BLE

### 1.1.2. 모델 A 시리즈

Model	SoC	RAM	VIDEO	ISA	Ethernet	USB	WiFi	BT
	BCM2835 Arm-11 600MHz Single ( 32bit )	256 MB		ARMv6			x	x
			Broadcom Dual Video CoreIV®		x	USB2.0 1 Port		
	BCM2837 Coretex-A53 1.4GHz Quad ( 64bit )	512 MB		ARMv8			2.4/5GHz Dual Band IEEE 802.11 b/g/n/ac	4.2/BLE

### 1.1.3. 모델 Zero 시리즈

Model	SoC	RAM	VIDEO	ISA	Ethernet	USB	WiFi	BT
Raspberry Pi Model Zero	BCM2835 Arm-11 1GHz Single ( 32bit )	512 MB	Broadcom Dual Video CoreIV®	ARMv6	x	USB OTG 1Port	x	x
Raspberry Pi Model Zero W							802.11n 4.1 BLE	

## 1.2. 지원되는 운영체제

공식 운영체제( Official OS ) : 라즈비안( raspbian )

데비안 리눅스를 라즈베리파이에 적합하게 Modify 한 리눅스(Linux) 시스템으로 역시 데비안(Debian) 리눅스의 한 가지인 우분투(Ubuntu)와 유사하다. 데비안 리눅스의 wheezy, Jessie, stretch로 버전업 됨에 따라 raspbian 역시 현재 raspbian-stretch-2019-04-08 버전까지 배포되었다.

서드파티( 3rd party OS ) 운영체제

	우분투 메이트( Ubuntu Mate )		우분투 코어( Ubuntu Core )
	윈도우 10 코어( Windows 10 Core)		OSMC( Open Source Media Center )
	LIBREELEC		PINET

이 외에도 3rd party 운영체제 다수 존재.

라즈베리파이 2 출시 이 후, 데비안에서는 라즈베리파이 재단에 ARM ISA( Instruction Set Architecture ) 버전 ARMv6 와 ARMv7 을 모두 지원하는 Kernel 을 제공하고 있다. Official OS 인 라즈비안은 이 커널로 만들어 졌기 때문에 그 동안 출시된 어떤 라즈베리파이 기종에서도 동작하지만, 3rd Party OS 의 경우는 라즈베리파이 버전 1 과 버전 2/3 용으로 따로 제공되거나 사용자가 이를 알아서 설치해야 하는 경우가 많으므로 주의를 요한다.

## 2. 라즈베리파이 시작하기

### 2.1. 준비할 것들

#### 2.1.1. 하드웨어

라즈베리파이 보드, USB 키보드, 마우스, HDMI 모니터와 케이블, DC 5V / 2.5A 어댑터, 8GB 이상의 마이크로 SD 카드, USB 마이크로 SD 카드리더, 이미지 다운로드 및 기록을 위한 PC

#### 2.1.2. 소프트웨어

시스템 디스크 작성을 위해서는 최신 버전의 라즈비안 이미지, 7-ZIP, Win32DiskImager 만 준비하면 되지만, 그 이 후 운영을 위해서 putty, VNC Viewer, WinSCP, NotePAD++ 등의 PC에서 자주 사용하는 유ти리티 프로그램들을 설치해 두자.

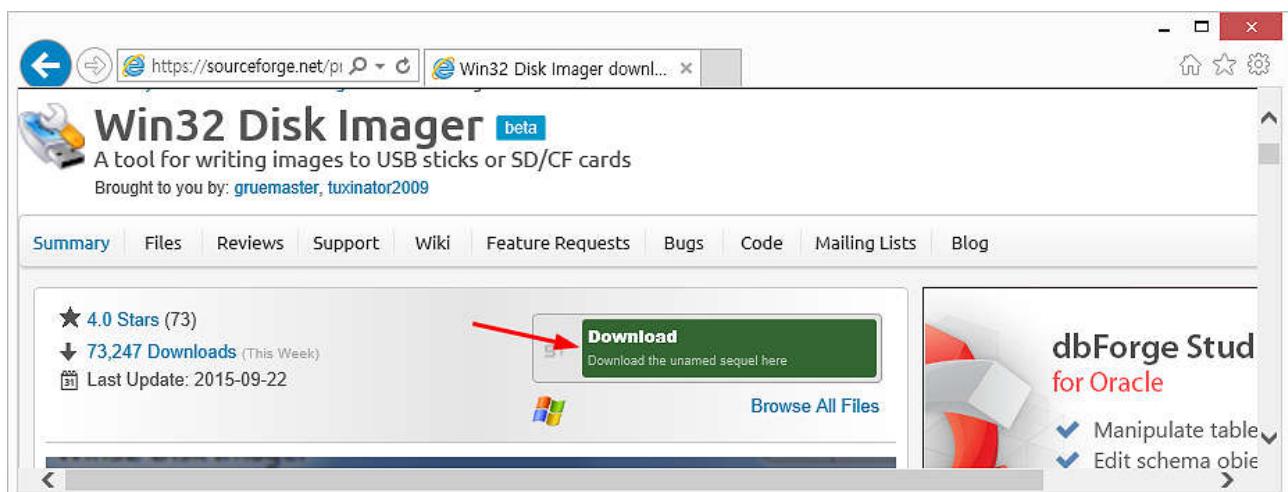
### 2.2. 시스템 디스크 작성

RASPBIAN은 데비안 계열의 리눅스이다. 윈도우와 리눅스의 파일 시스템이 서로 다르므로 윈도우 환경에서 라즈베리파이를 위한 시스템디스크는 단순히 마이크로 SD 카드에 필요한 파일을 복사해 넣는 것으로는 만들어지지 않는다.

#### 2.2.1. Win32DiskImager 다운로드 및 설치

Win32DiskImager는 시스템 이미지를 읽어, 타겟 디스크(라즈베리파이에서 사용할 마이크로 SD 카드)에 리눅스 형식으로 기록해 주는 프로그램이다.

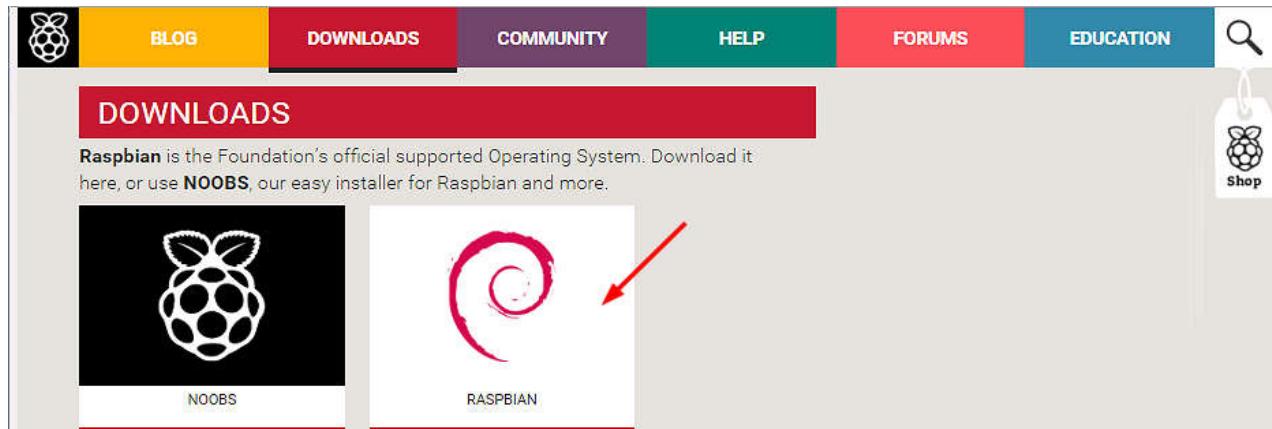
웹 브라우저에서 주소 <http://sourceforge.net/projects/win32diskimager/>를 열어 아래 표시된 Download 버튼을 클릭하여 다운로드 후, 설치한다.



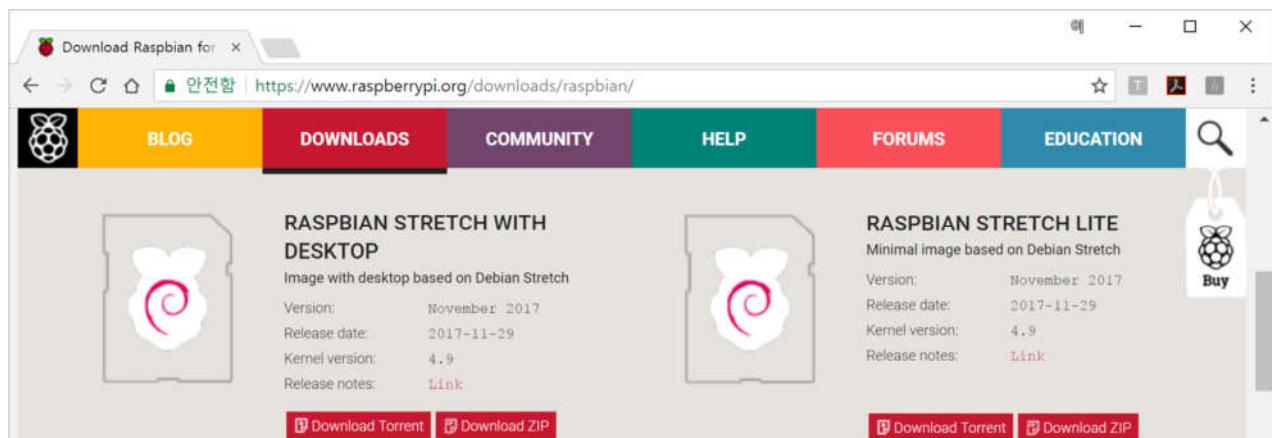
## 2.2.2. RASPBIAN 다운로드

웹 브라우저에서 주소 <http://raspberrypi.org/downloads> 를 열어 아래 표시한 다운로드 링크를 클릭한다.

아래 표시한 "Download ZIP" 버튼을 클릭하여 라즈베리파이 재단의 공식 시스템 이미지인 RASPBIAN 을



다운로드한다.



다운로드한 zip 파일의 압축을 해제하여 얻어진 img 파일( 2019. 01. 04. 현재 2019-04-08-raspbian-stretch.img )의 위치를 기억(또는 메모)해 둔다.

Rspbian Stretch with desktop & recommended SW : Rspbian Stretch with desktop + 추천 SW

Rspbian Stretch with desktop : GUI 환경( X-Windows )이 포함된 버전

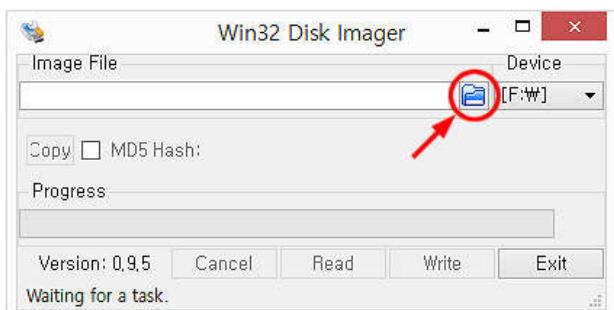
Rspbian Stretch lite Rspbian Stretch with desktop : GUI 환경( X-Windows )이 제외된 버전

## 2.2.3. 시스템 디스크 작성

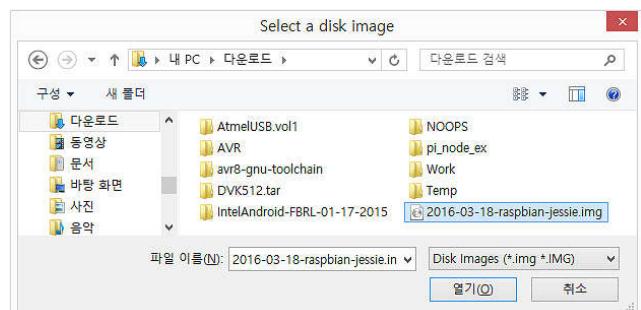
이제 필요한 이미지 파일과 이미지 기록 프로그램이 준비되었으니 시스템 디스크를 작성해보자.

먼저 라즈베리파이에서 사용할 8GB 이상의 마이크로 SD 카드를 카드리더에 삽입하여 PC 에 연결하고 나서 Win32DiskImager 프로그램을 실행( 시작 - 프로그램 - Image Writer - Win32DiskImager )후, 다음 과정을 진행한다.

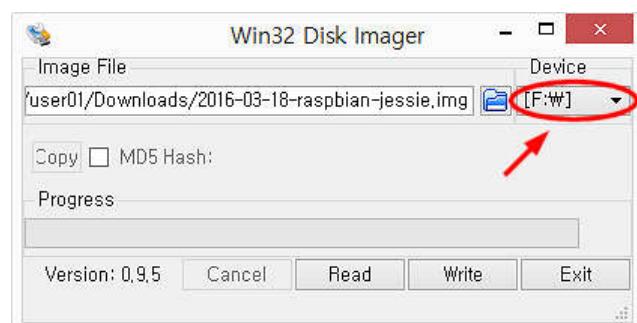
### 1. 아래 표시된 Browse 아이콘 클릭



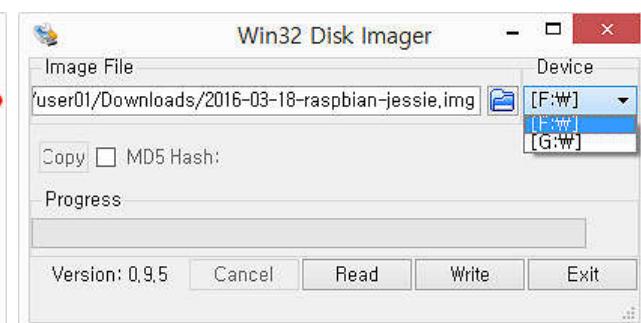
### 2. 앞서 다운받은 RASPBIAN 이미지 선택 후 "열기(O)"



### 3. 디스크 선택

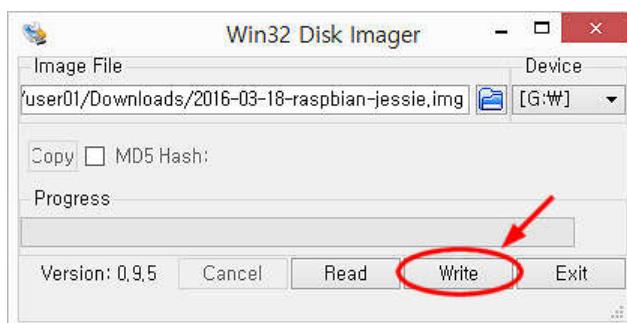


### 4. 타겟 드라이브 선택( 디스크가 초기화되므로 주의! )

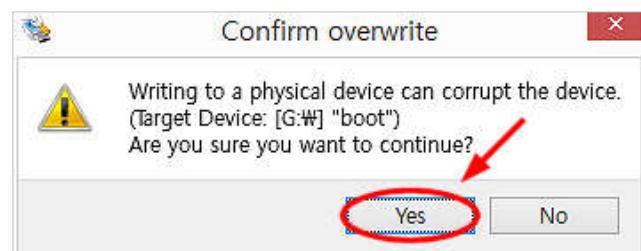


※ Device 에 디스크가 2 개이상 표시될 경우 선택된 디스크의 모든 내용이 지워지므로 주의한다.  
( 실수로 공인인증서 같은 중요한 데이터를 잃게 될 수 있다. )

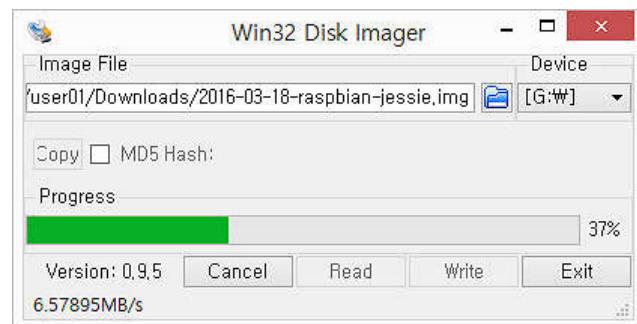
### 5. "Write" 버튼을 클릭하여 디스크 기록이 시작.



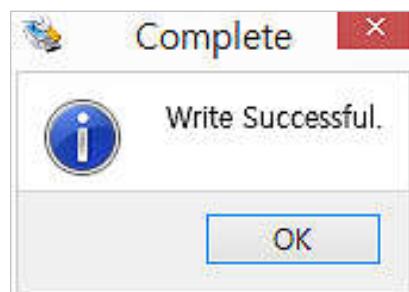
### 6. 타겟 드라이브 선택 재확인. "Yes"버튼 클릭



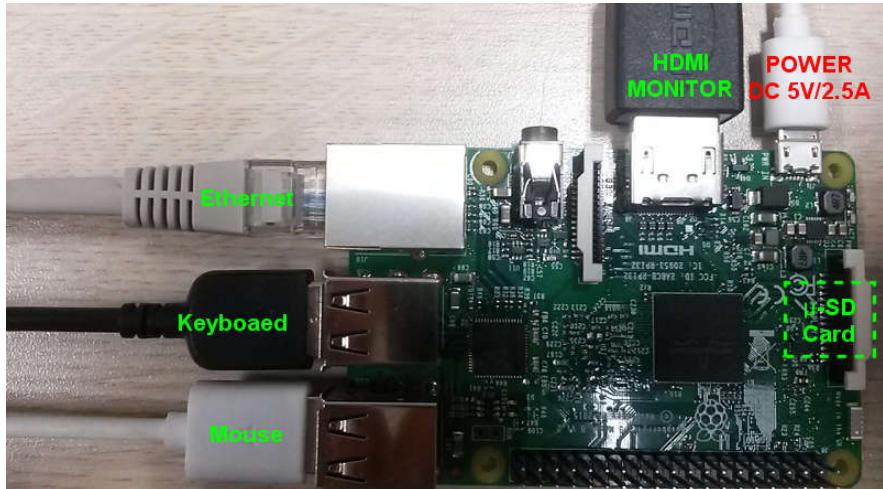
### 7. 진행 상황 표시



### 8. 완료



시스템 디스크가 만들어졌다.  
이제 라즈베리파이를 사용할  
준비가 된 것이다. 오른 편  
사진처럼 작성된 시스템  
디스크를 라즈베리파이에  
삽입하고, 모니터, 키보드,  
마우스, Ethernet 을 연결하고  
전원을 연결한다.



## 2.3. raspi-config 를 이용한 시스템 기본 설정

"pi" 사용자 계정의 패스워드를 변경하거나, 라즈베리파이의 호스트네임을 변경하려면 'passwd' 명령을 사용하거나, "/etc/hosts" 파일과 "/etc/hostname"을 편집하여야 하는데, 익숙하지 않은 사용자에게는 매우 불편한 방법이다. 라즈베리파이는 이러한 시스템 설정 변경을 편리하게 할 수 있도록 "raspi-config"라는 프로그램을 제공한다. 이를 이용하여, 라즈베리파이를 사용하기 위한 기본적인 시스템 설정을 해보자.

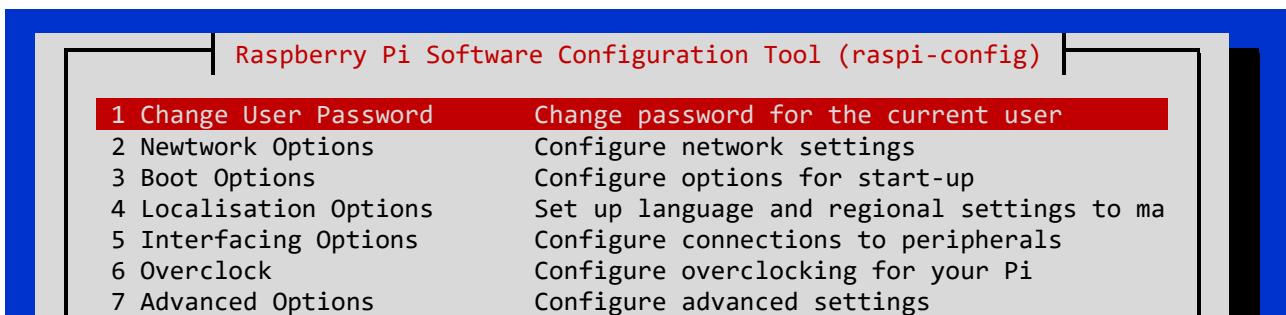
아래 표시한 터미널 아이콘을 클릭하여 터미널을 실행한다.

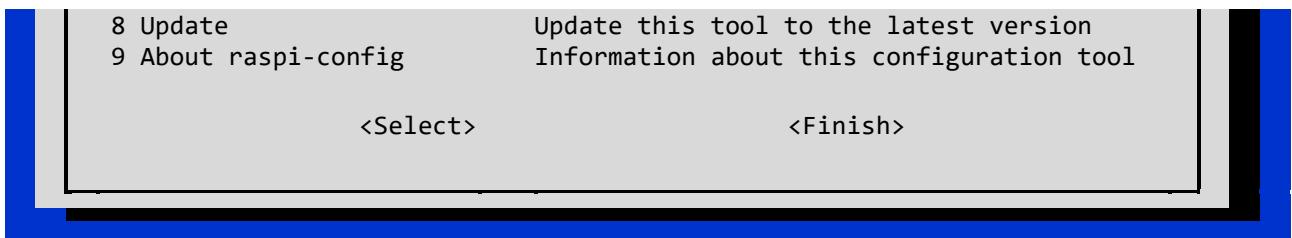


터미널 창에서 아래와 같이 입력한다.

```
$ sudo raspi-config
```

다음 그림에서 보여지듯이 9 개의 메인 메뉴가 나타난다. 메뉴간의 이동은 방향키, 탭 키를 선택은  
스페이스 바, 엔터 키를 이용한다.





각 메뉴에서 설정할 수 있는 기능들을 살펴보자.

### 2.3.1. Change User Password

라즈베리파이 현재 로그인된 사용자(사용자를 추가하지 않았다면 기본계정인 "pi")의 패스워드를 변경해주는 메뉴로, 콘솔 창에서 "passwd" 명령으로 변경해야 하는 작업을 명령어 입력 없이 가능하게 해준다.

아래와 같이 새로운 패스워드를 입력을 요구하는데, 이 때 새 패스워드 입력 시 화면상에 어떤 변화도 나타나지 않지만 제대로 입력되고 있는 것이므로 당황하지 말자. ( 입력한 글자 수 만큼 \*\*\*\* 처럼 화면에 표시되지 않는다. 엔터키를 입력할 때까지 화면상에 아무 변화도 나타나지 않는다. )

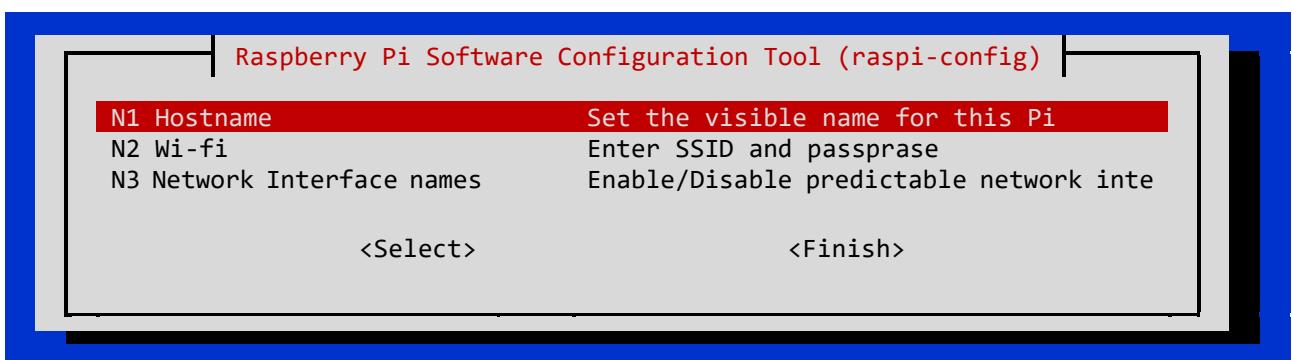
```
Enter new UNIX password:  
Retype new UNIX password:
```

처음 입력한 패스워드와 다시 입력한 패스워드가 일치하면 " password updated successfully"라는 메시지에 화면에 출력되고 다시 "raspi-config"의 메인 메뉴화면으로 복귀한다.

```
passwd: password updated successfully
```

### 2.3.2. Network Options

네트워크관련 설정 중 hostname 변경, WiFi 연결, 네트워크 인터페이스 이름 부여방식 변경 방법을 제공한다.



## 1. N1. Hostname

다음 터미널 화면의 prompt( )에서 "raspberrypi"가 바로 호스트 네임에 해당한다.

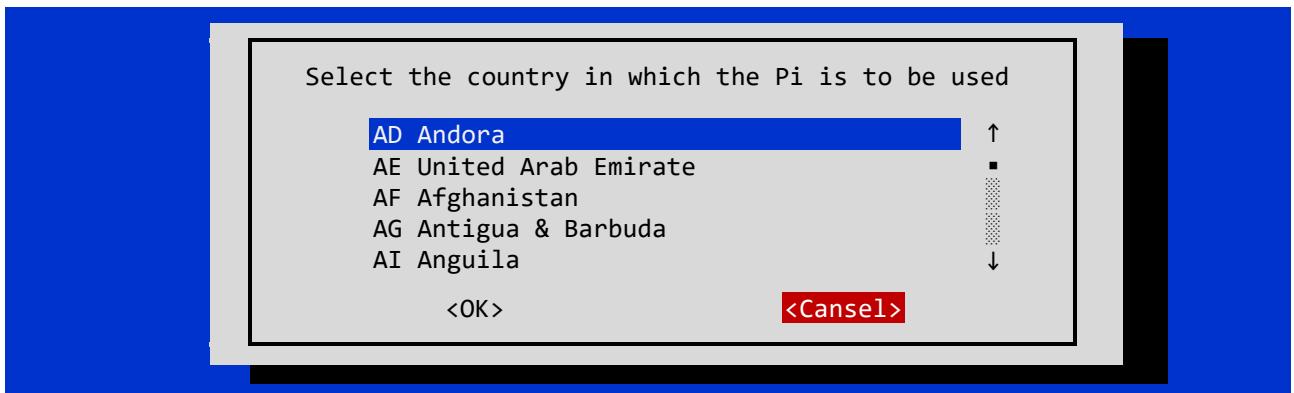
```
pi@raspberrypi:~ $
```

바로 이 호스트네임을 변경해 준다. (" /etc/hostname" 파일과 "/etc/hosts" 파일의 내용을 변경해 준다. )

## 2. N2. Wi-Fi

라즈베리파이( Wi-Fi )를 사용하게 될 국가를 선택하는 설정에서는 Tab 키를 이용하여 "<Cancel>" 선택.

( 또는 GB English 나 US America 를 선택해도 되지만 KR Korea 는 선택하지 않는다. KR 로 선택할 경우 Wi-Fi 연결이 되지 않는 버그가 있다. )



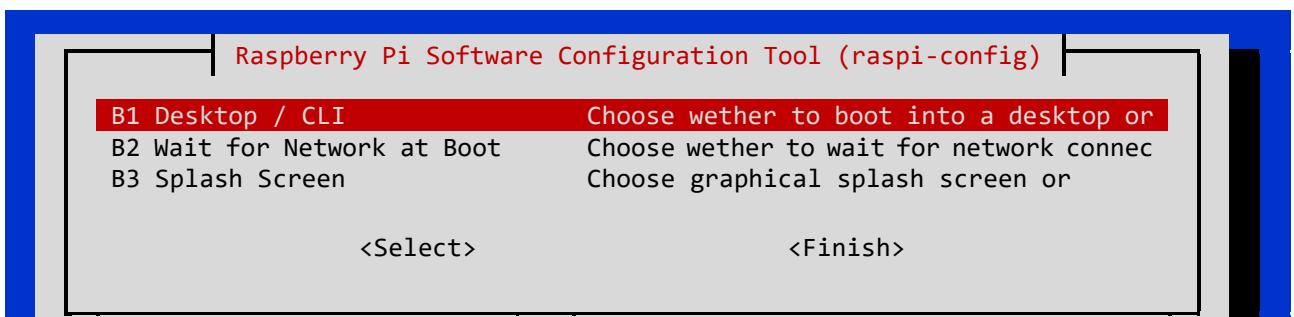
이 후는 사용하려는 무선 AP( Access Pointer )의 SSID 와 Password( passphrase )를 입력 하면, 해당 무선 AP( 공유기 ) 에 Wi-Fi 연결을 설정해 주는 기능.( '/etc/wpa\_supplicant/wpa\_supplicant.conf' 파일에 설정 값 기록 )

## 3. N3. Network interfaces names

리눅스에서 그 동안 사용해 오던 전통적인 네트워크 인터페이스 이름( eth0, eth1, ..., wlan0, wlan1, ... )과 같이 예측 가능한 네트워크 인터페이스 명 대신, 예측하기 어려운 고유한 인터페이스 이름을 사용하도록 설정

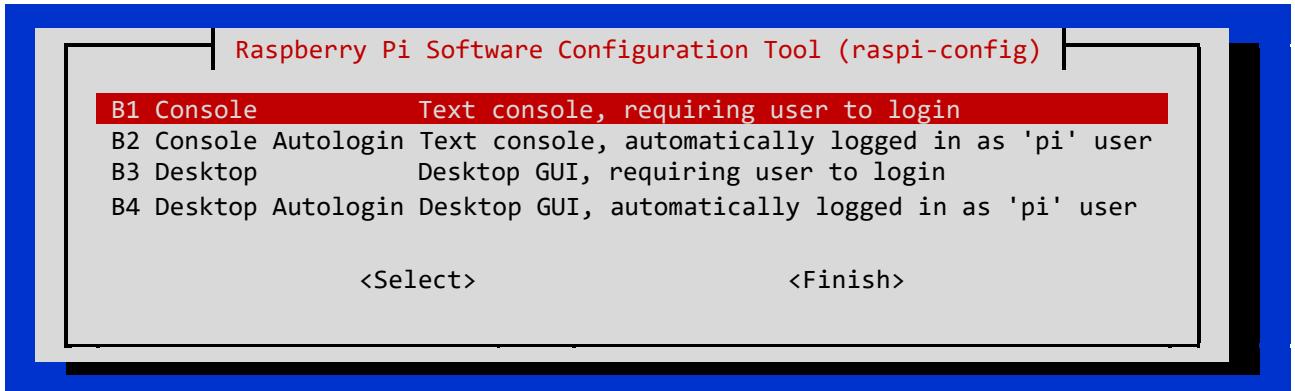
### 2.3.3. Boot option

라즈베리파이의 사용자 환경( User Interface ) 선택, 부팅 시 네트워크 연결 대기 여부, 부팅시 Splash Screen 표시여부 등을 설정할 수 있다.



## 1. B1. Desktop / CLI

시작 시 GUI( Graphic User Interface : X-Windows )로 시작할 것인 지, CLI( Command Line Interface : 명령 프롬프트 환경 )으로 시작할 것인지를 선택할 수 있다. 아래 4 가지 중 하나를 선택한다.



**B1. Console -----** Text Console 시작 ( ID, Password 입력 필요 )

**B2. Console, Autologin -----** Text Console 시작, 자동 로그인( ID, Password 입력 불필요 )

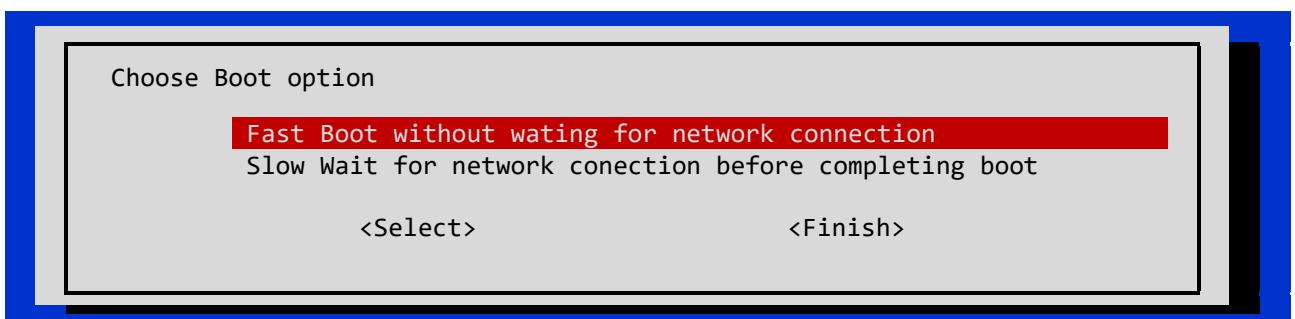
**B3. Desktop GUI -----** Desktop GUI 시작 ( ID, Password 입력 필요 )

**B4. Desktop GUI, Autologin ---** Desktop GUI 시작, 자동 로그인 ( ID, Password 입력 불필요 )

( 기본 설정은 "B4. Desktop GUI, Autologin", Desktop GUI 는 리눅스의 X-Windows 환경을 말한다. )

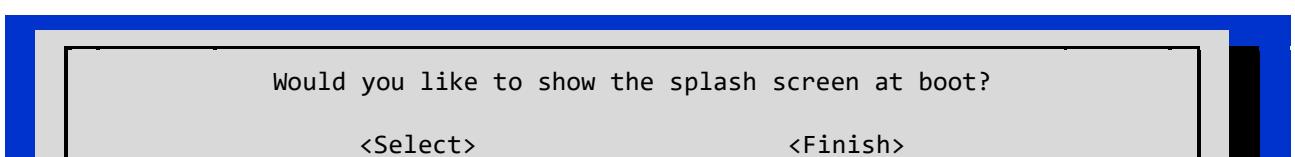
## 2. B2. Wait for Network at Boot

부팅 시 네트워크 연결상태 확인 과정 없이( Fast ) 진행할 것인지, 확인 후( Slow ) 진행할 것인지를 선택



## 3. B3. Splash Screen

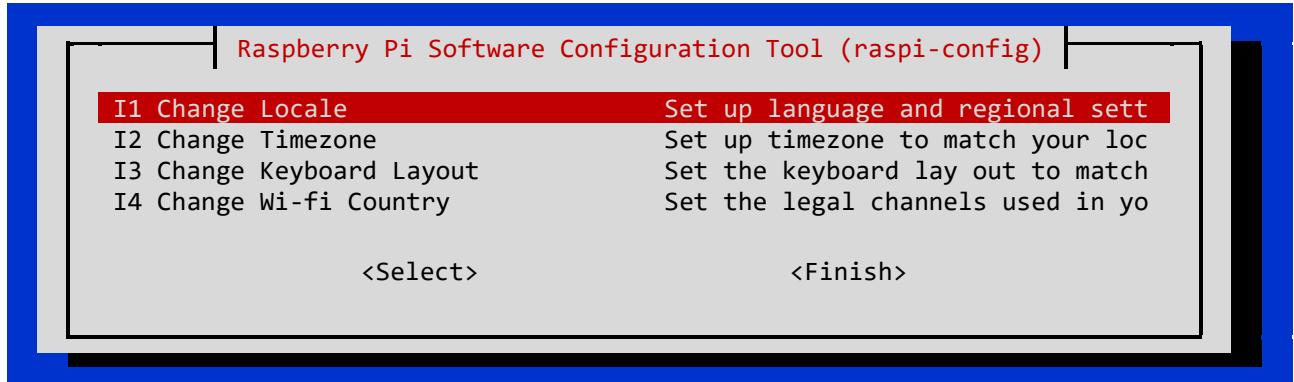
부팅 시 화면에 디스플레이되는 Splash Screen 을 표시 할 것인지를 선택한다.





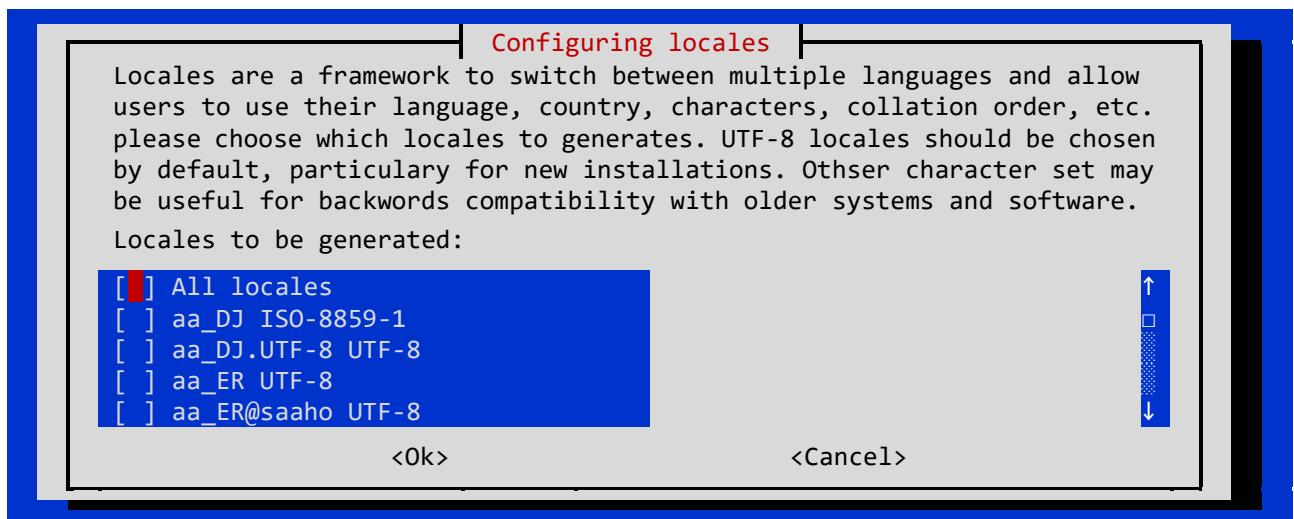
## 2.3.4. Localisation option

지역, 국가 등에 관련된 설정으로 로케일( Locale ), 표준시, 키보드 레이아웃을 설정



### 1. I1. Change Locale

윈도우의 제어판 항목 중, “국가 또는 지역”에 해당한다. 날짜, 시간, 통화단위 등의 표시 등을 설정된 Locale의 형식을 따르게 된다. 기본 값으로 설정된 “영국”을 “대한민국”으로 변경해준다.



위, 아래 방향키를 이용하여 현재 설정되어 있는 로케일( en\_GB.UTF-8 )로 이동, 스페이스 바를 이용해 해제하고, 대한민국( ko\_KR.UTF-8 UTF8 )로케일을 추가한다. ( 영문모드로 사용하려면 en\_US.UTF-8 UTF8 을 사용 )

#### 기본으로 설정되어 있는 'en\_GB(영국)' 해제

[ ] en\_GB ISO-8859-1  
[ ] en\_GB.ISO-8859-15 ISO-8859-15  
[\*] en\_GB.UTF-8 UTF-8  
[ ] en\_HK ISO-8859-1  
[ ] en\_HK.UTF-8 UTF-8

[ ] en\_GB ISO-8859-1  
[ ] en\_GB.ISO-8859-15 ISO-8859-15  
[\*] en\_GB.UTF-8 UTF-8  
[ ] en\_HK ISO-8859-1  
[ ] en\_HK.UTF-8 UTF-8

'en\_US(미국)' 추가 (모든 설정이 완료될 때 까지 메뉴 항목 등이 제대로 표시 되려면 필요하다.)

```
[ ] en_US ISO-8859-1  
[ ] en_US.ISO-8859-15 ISO-8859-15  
[*] en_US.UTF-8 UTF-8  
[ ] en_ZA ISO-8859-1  
[ ] en_ZA.UTF-8 UTF-8
```

```
[ ] en_US ISO-8859-1  
[ ] en_US.ISO-8859-15 ISO-8859-15  
[*] en_US.UTF-8 UTF-8  
[ ] en_ZA ISO-8859-1  
[ ] en_ZA.UTF-8 UTF-8
```

'ko\_KR(대한민국)' 설정

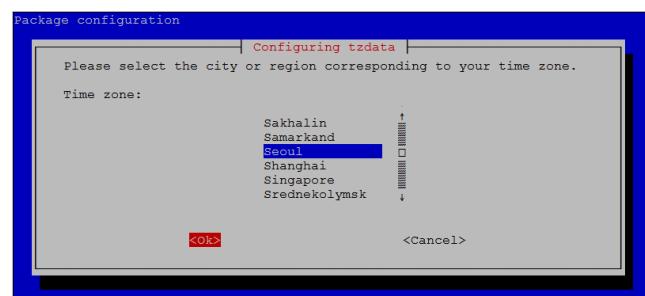
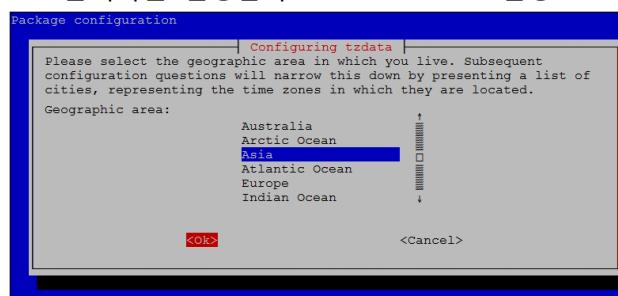
```
[ ] kn_IN UTF-8  
[ ] ko_KR.EUC-KR EUC-KR  
[*] ko_KR.UTF-8 UTF-8  
[ ] kok_IN UTF-8  
[ ] ks_IN UTF-8
```

```
[ ] kn_IN UTF-8  
[ ] ko_KR.EUC-KR EUC-KR  
[*] ko_KR.UTF-8 UTF-8  
[ ] kok_IN UTF-8  
[ ] ks_IN UTF-8
```

위 그림에서 보여지듯이 같은 로케일이 en\_GB 와 en\_US 의 경우는 3 가지, ko\_KR 의 경우는 2 가지가 있지만, 반드시 그 중 UTF-8 을 선택해야 한다. 추가한 Locale 을 설치 후, "Default locale for system" 설정은 일단 en\_US 로 설정한다. ( 모든 설정 완료 후, ko\_KR 로 설정할 수 있다. )

## 2. I2. Change Timezone

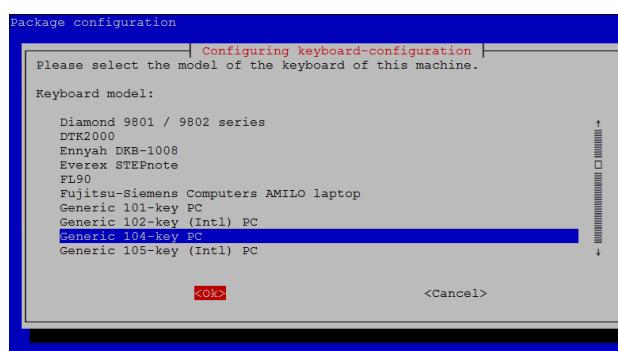
표준시각을 설정한다. Asia – Seoul 로 설정.



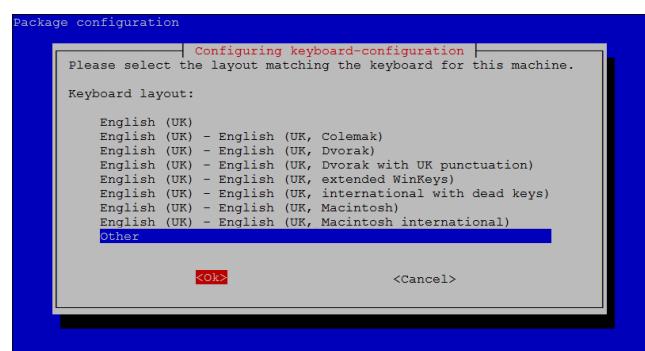
## 3. I3. Change Keyboard Layout

기본설정이 UK(영국)로 설정되어 있어서, 변경하지 않을 경우 "#"을 입력할 때 "£"( 영국 통화단위 파운드 기호 )가 입력된다

### 1. Generic 104-key PC 선택

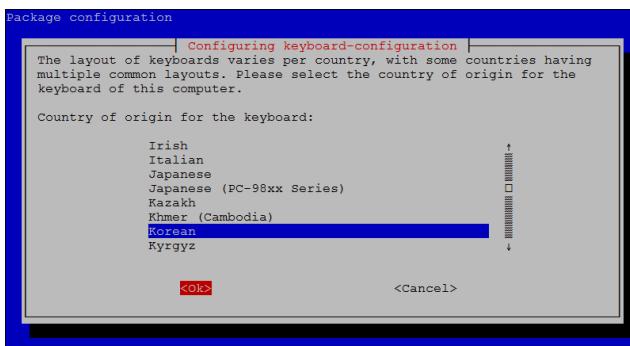


### 2. Other 선택

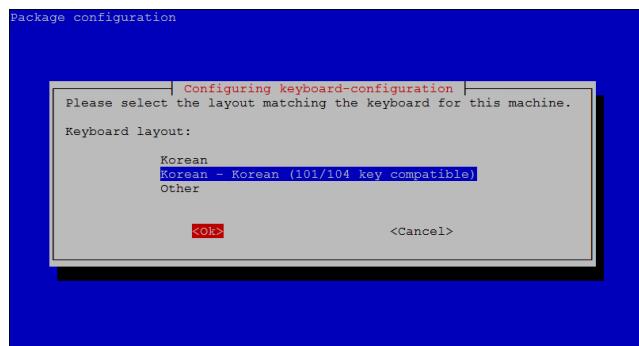


### 3. Korean 선택

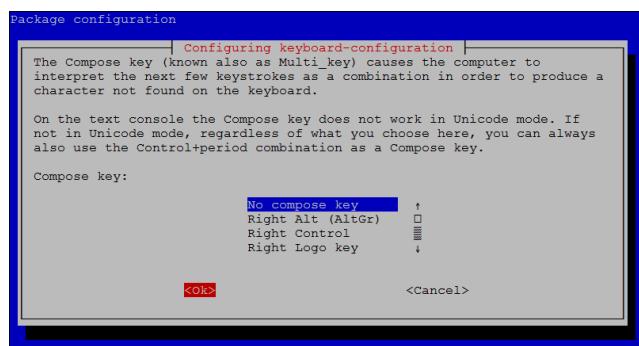
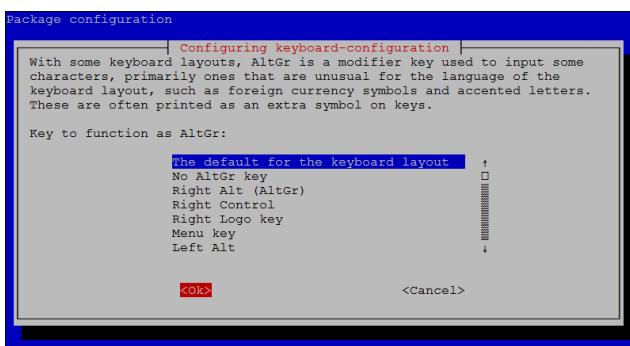
### 4. Korean - Korean (103/104 Competible) 선택



### 5. The default for the keyboard layout 선택



### 6. No compose key 선택



### 7. "Ctrl+Alt+Backspace" 로 X-Windows 종료 <Yes>

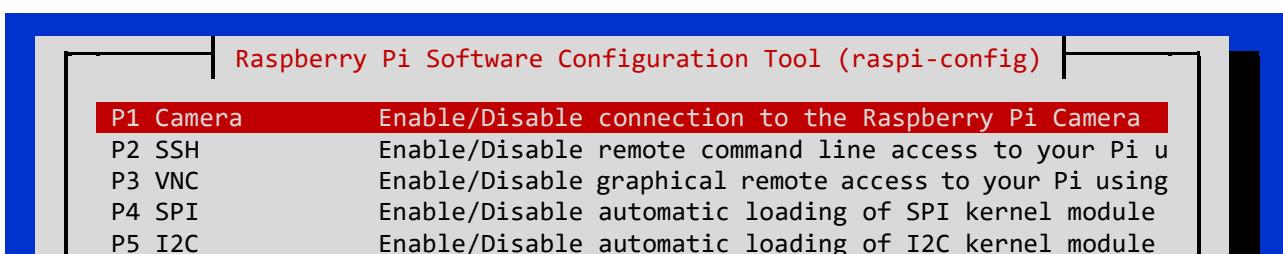


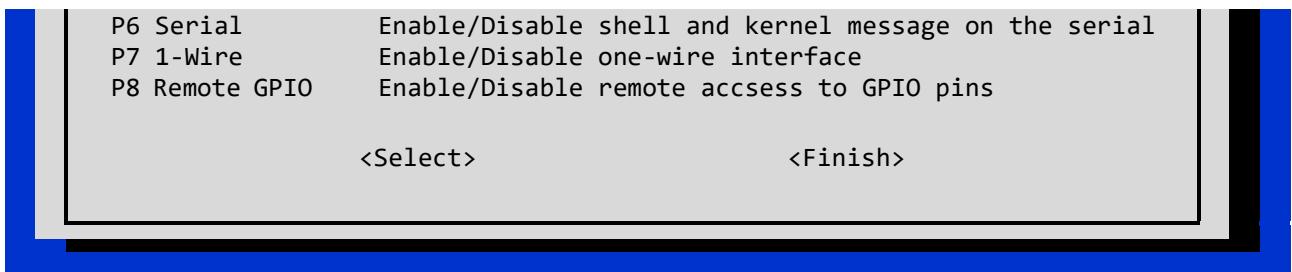
## 4. I4. Change Wi-fi country

기본설정이 GB(영국)로 설정되어 있지만, 굳이 KR로 변경하지 않는다. 변경할 경우, GUI 환경에서 Wifi 설정을 해도 반영되지 않는 버그가 있다.

### 2.3.5. Interfaceing option

라즈베리파이와 주변장치의 연결에 대한 옵션들을 설정





## 1. P1. Camera

라즈베리파이의 카메라 연결 포트를 활성화 여부를 선택한다. ( 파일 카메라 사용 시 Enable 선택 )

## 2. P2. SSH

Secure Shell Server Demon( 보안이 강화된 원격 접속 서버 )의 활성 여부를 설정한다. 특별한 이유가 없다면 그대로 사용함으로 설정한다. ( 매우 유용한 기능이다. )

## 3. P3. VNC

GUI 원격 접속을 위한 VNC( Virtual Network Computing ) 서버 데몬 활성화 여부 선택. Real VNC 클라이언트 프로그램을 이용한 GUI 원격접속을 위해서 선택해 두자.

## 4. P4. SPI

라즈베리파이 BCM GPIO 07, 08, 09, 10, 11 을 각각 SPI 통신포트의 CE1, CE0, MISO, MOSI, SCLK 의 기능으로 사용하도록 설정.

## 5. P5. I2C

라즈베리파이 BCM GPIO 00, 01, 02, 03 을 각각 I2C 통신포트의 SDA0, SCL0, SDA1, SCL1 의 기능으로 사용하도록 설정.

## 6. P6. Serial

라즈베리파이 1, 2 버전에서 /dev/ttyAMA0 시리얼 포트를 Serial Console로 사용할 것인지를 설정한다. <yes>를 선택할 경우 모니터, 키보드 연결 없이 시리얼통신 연결을 통해 라즈베리파이의 상태 모니터링과 설정 변경 등이 가능하다. /dev/ttyAMA0 포트를 다른 장치와의 통신에 사용하려면 이 설정에서 <No>를 선택해야 한다.

라즈베리파이 3 의 경우에는 시리얼콘솔을 사용할 경우, /boot/config.txt 내용의 마지막에 아래 내용을 추가하여 리부팅 한다. ( 이 경우, 블루투스가 비활성화 된다. )

```
# add below for using serial console with /dev/ttyAMA0
enable_uart=1
dtoverlay=pi3-miniuart-bt
```

## 7. P7. 1-Wire

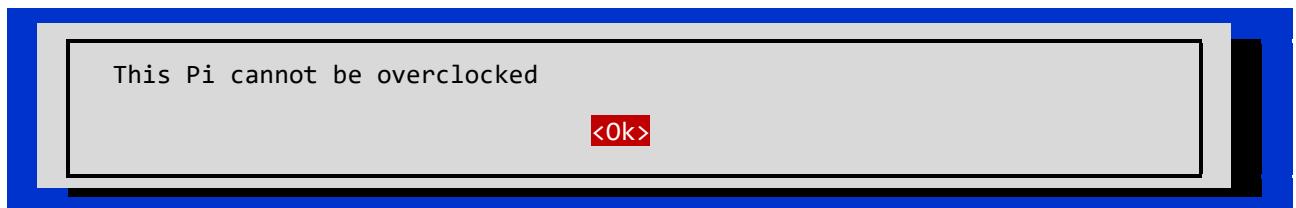
1-Wire 통신 인터페이스 사용 여부를 선택한다.

## 8. P8. Remote GPIO

네트워크를 통해 GPIO 를 제어할 수 있도록 하는 서버 사용 여부를 선택한다.

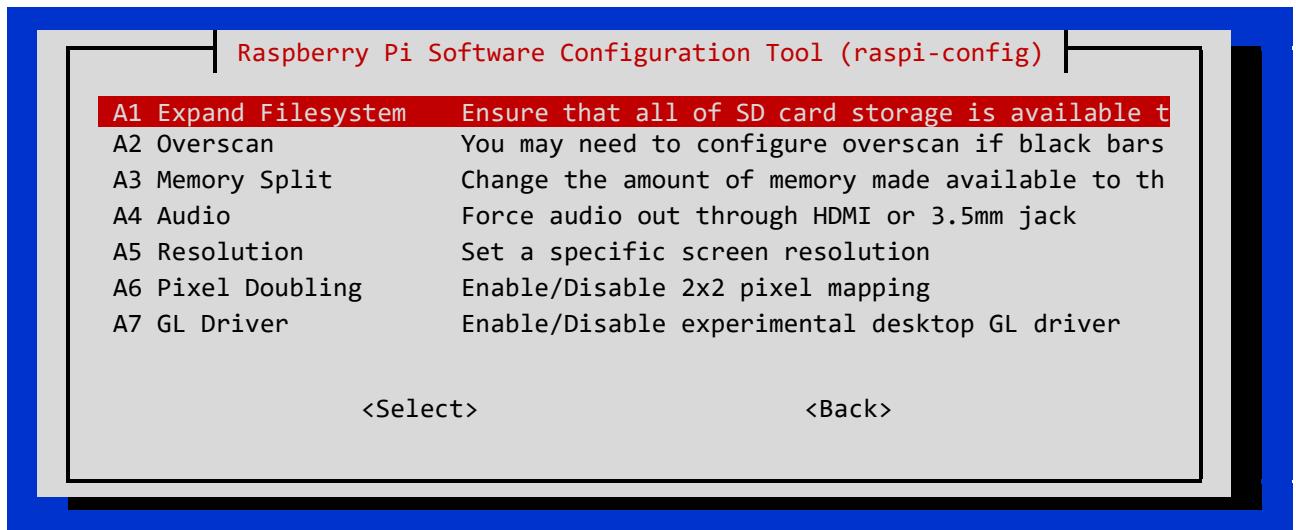
### 2.3.6. Overclock

라즈베리파이 3 이전 버전의 라즈베리파이에서는 사용하는 클럭주파수를 기준 주파수 보다 높게 설정할 수 있는 메뉴를 제공하지만, 1.2GHz 로 동작하는 라즈베리파이 3 의 경우는 더 이상 높은 주파수를 설정할 수 없다.



### 2.3.7. Advanced Option

SSH 서버 데몬 구동, 시리얼 콘솔 사용여부, GPIO 의 I2C, SPI 포트 활성화 여부 등을 설정한다.



#### 1. A1. Expand Filesystem

RASPBIAN 시스템 이미지는 최소 4GB 의 Micro SD 카드에서 실행되도록 작성( 최근 배포된 RASPBIAN 이미지의 크기 3.75GB )되어 배포된다. 이 이미지를 Micro SD 카드에 기록할 경우, 4GB 의 경우 0.25GB, 8GB 의 경우 4.25GB, 16GB 의 경우 12.25GB 의 할당되지 않은 디스크 공간이 발생한다.

Expand Filesystem 메뉴는 파일시스템을 확장하여 할당되지 않은 디스크 공간을 사용할 수 있도록 해준다.

( raspbian-stretch 의 경우 최초 시스템 기동 시 자동 실행된다. )

## 2. A2. Overscan

구형 CRT 모니터 연결 시 화면 표시 이상 개선( LCD 모니터 사용 시에는 설정할 필요 없다. ).

## 3. A3. Memory Spilt

기본메모리 중 비디오 메모리로 할당할 용량을 설정한다. ( CLI: 128 MB / GUI: 256MB 정도가 적당하다. )

## 4. A4. Audio

Audio 출력방향을 HDMI 와 3.5Φ Audio Jack 중 하나로 설정.( 0. 자동선택 / 1. 3.5Φ Audio Jack / 2. HDMI )

## 5. A5. Device Tree

리눅스 커널의 Device Tree 사용 여부를 설정. ( 사용하도록 설정한다. )

## 6. A6. GL Driver

라즈베리파이 2, 3 에서 X-Windows 사용 시, OpenGL 라이브러리를 통한 하드웨어 그래픽 가속기능 사용 여부 설정

## 2.3.8. Update

리눅스 시스템에서 어플리케이션을 추가 할 경우, 네트워크의 해당 어플리케이션이 저장되어있는 Repository 라는 곳에서 가져다 설치한다. 따라서 시스템 안에 설치 가능한 모든 어플리케이션의 Repository 주소가 기록되어 있는 목록( /etc/apt/sources.list )을 가지고 있다. 어플리케이션의 Repository 가 변경되었을 경우 이 목록을 새 정보로 갱신하는 작업이 바로 업데이트이다.( sudo apt-get install update )

## 2.3.9. About raspi-config

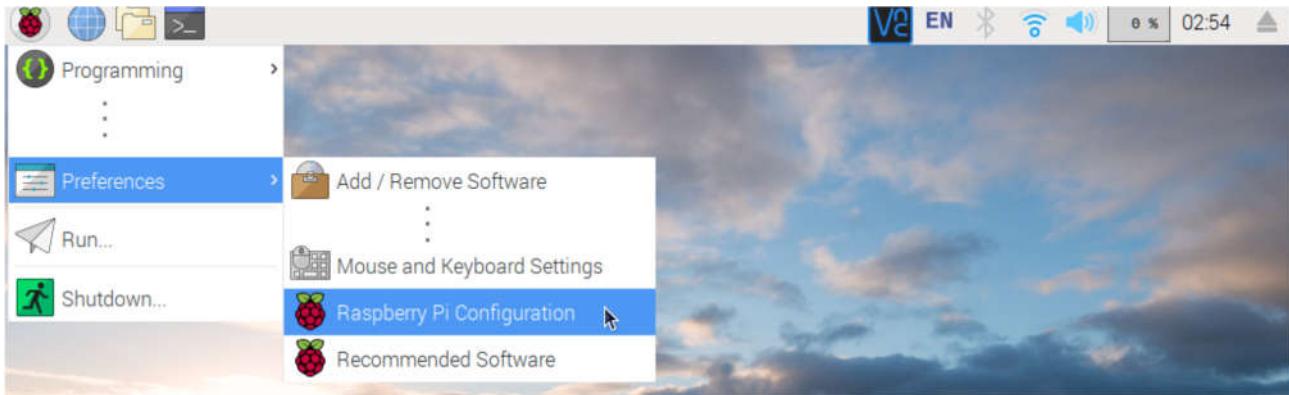
설정 메뉴는 아니다. raspi-config 프로그램을 만든 의도, 설명이 간단히 언급되어 있다. ( 리눅스의 시스템 설정은 해당 설정 파일들을 편집, 수정하도록 되어있다. 처음 접하는 사용자는 각 설정파일의 위치나, 무엇을 어떻게 편집해야 하는지 조차 모르더라도, raspi-config 프로그램을 통해 쉽게 설정할 수 있다. )

## 2.4. GUI 에서의 설정

Desktop 사용자환경(Graphic User Interface)을 사용할 경우 많은 부분에서 CLI(Command Line interface) 환경에 비해 편리하다(반면 한정된 시스템 자원중 상당량이 사용자 환경에 할당된다). Desktop 사용자 환경에서 "IoT 시스템 개발"을 위해 유용한 시스템 설정 몇 가지를 알아보자.

## 2.4.1. Raspberry Pi Configuration

앞서 "2.3 raspi-config 를 이용한 시스템 기본 설정"에서 터미널 창에서 "raspi-config"를 구동해 설정했던 기능들을 X-Windows Desktop GUI에서도 설정 가능하도록 만들어 추가한 어플리케이션이다.



앞의 그림과 같이 "시작 – Performance – Raspberry Pi Configuration"을 실행하면, 4 개의 탭( System, Interfaces, Performance, Localisation )으로 구성된 Dialog Box 가 화면에 나타난다. "raspi-config"에 나오는 내용 중 일부 자주 사용되는 기능들이 4 개의 탭에 배치되어 있다.

System 탭	Interfaces 탭	Performance 탭	Localisation 탭
The System tab contains settings for the host name (raspberrypi), boot options (To Desktop or To CLI), auto login (As current user checked), network at boot (Wait for network unchecked), splash screen (Enabled), resolution (Set Resolution...), underscan (Enabled), and pixel doubling (Enabled). Buttons at the bottom are Cancel and OK.	The Interfaces tab includes sections for Camera, SSH, VNC, SPI, I2C, Serial Port, Serial Console, 1-Wire, and Remote GPIO. Each section has Enabled or Disabled radio buttons. Buttons at the bottom are Cancel and OK.	The Performance tab shows Overclock (Not available) and GPU Memory (128). Buttons at the bottom are Cancel and OK.	The Localisation tab includes Locale, Timezone, Keyboard, and WiFi Country. Buttons at the bottom are Cancel and OK.
1. Change User Password 2. Network Options 3. Boot Options 7. Advanced Options 의 - A2. Overscan - A5. Resolution - A6. Pixel Doubling	5. Interfacing Options	6. Overclock	4. Localisation Options

각 메뉴가 무엇을 어떻게 설정하는가에 대해서는 "2.3 raspi-config 를 이용한 시스템 기본 설정" 단원을 참고하자.

## 2.4.2. WiFi 설정

WiFi 설정은 Console 에서 "/etc/dhcpcd.conf" 파일과 "/etc/wpa\_supplicant/wpa\_supplicant.conf" 파일을 편집해야 하지만 GUI 환경으로 시작할 경우 좀 더 손쉽게 설정할 수 있다. 부트옵션이 CLI 인 경우 아래와 같이 입력하여 GUI 환경( X-Windows )을 시작한다.

```
pi@raspberrypi:~$ startx
```

WiFi 설정을 하기 전에 "/etc/wpa\_supplicant/wpa\_supplicant.conf" 파일의 내용을 미리 확인해보자. 터미널을 열어 nano( 또는 vi 등 ) 에디터로 "/etc/wpa\_supplicant/wpa\_supplicant.conf" 파일을 연다.

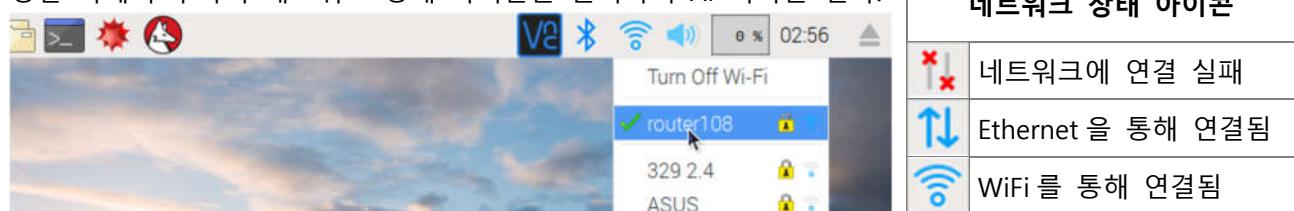
```
pi@raspberrypi:~$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

```
GNU nano 2.2.6          File: wpa_supplicant.conf
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

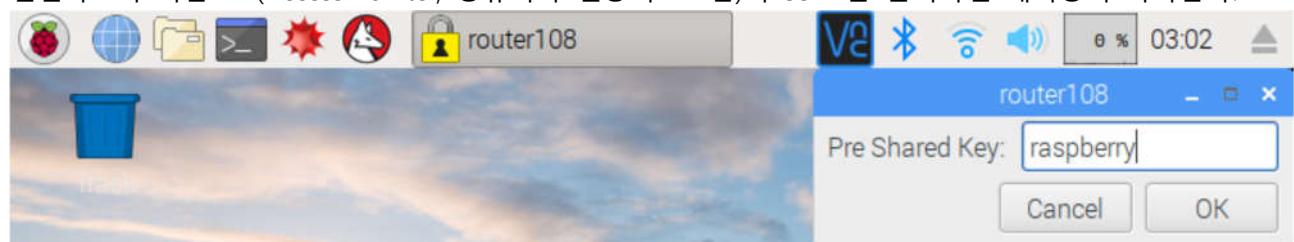
[Read 4 lines]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Cut Text  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U Uncut Text ^T To Spell
```

어떤 AP 사용을 위한 설정도 되어있지 않음을 확인 후, Ctrl+X, n, Enter 를 입력하여 nano 를 종료한다.

상단 바에서 우측의 네트워크 상태 아이콘을 클릭하여 AP 목록을 연다.



연결하고자 하는 AP(Access Point, 공유기가 일종의 AP 임)의 SSID 를 클릭하면 대화창이 나타난다.



선택한 AP(공유기)의 암호를 입력하고 "OK" 버튼을 클릭한다. 얼마 지나지 않아 네트워크 상태 아이콘이 WiFi 모양으로 고정되면 네트워크에 연결된 것이다.

그럼, 연결 전 확인 했었던 /etc/wpa\_supplicant/wpa\_supplicant.conf 파일의 내용을 다시 확인해 보자.

```
pi@raspberrypi:~$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

```
GNU nano 2.7.4          File: /etc/wpa_supplicant/wpa_supplicant.conf
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=GB

network={
    ssid="Router4Pi"
```

```

        psk="raspberry"
        key_mgmt=WPA-PSK
    }

[Read 4 lines]
^G Get Help   ^O WriteOut   ^R Read File   ^Y Cut Text   ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is    ^V Next Page  ^U Uncut Text  ^T To Spell

```

Wi-Fi 연결정보가 추가되어 있는 것을 확인할 수 있다.

201x-xx-xx-raspbian-stretch-lite 이미지를 사용할 경우 raspi-config 의 2 Network Options - N2 Wi-fi 에서 무선 공유기(Wireless AP)의 SSID, Passphrase 를 입력하거나 위 정보를 직접 /etc/wpa\_supplicant/wpa\_supplicant.conf 파일을 편집하여 추가하여야 한다.

이미 등록된 공유기( 무선라우터, 무선AP )외에 또 다른 장치에 연결할 경우 그 해당 연결정보( SSID, Passphrase )도 같은 방법으로 추가할 수 있다. 이런 경우 라즈베리파이가 구동되면 주변 AP를 검색하여 그 정보와 대조하여 같은 것이 검색되면 해당 AP( Acess Pointer )에 연결한다. ( 2가지 이상이 동시 검색될 경우는 이 파일에 기록된 순서에 의해 연결됨 )

**raspi-config** 의 **2 Network Options - N2 Wi-fi** 를 선택하면 라즈베리파이를 사용하게 될 국가를 선택하는 과정이 나타나는데 여기서 선택한 국가코드가 녹색으로 표시한 "country=GB"에 표시된다. 하지만 앞서 언급했듯이 이 값을 KR 로 변경할 경우 Wi-Fi 설정이 제대로 반영되지 않는 버그가 있으므로 변경하지 않는다. ( 변경하지 않아도 사용상에 어떤 문제도 없다. )

### 2.4.3. Bluetooth 설정

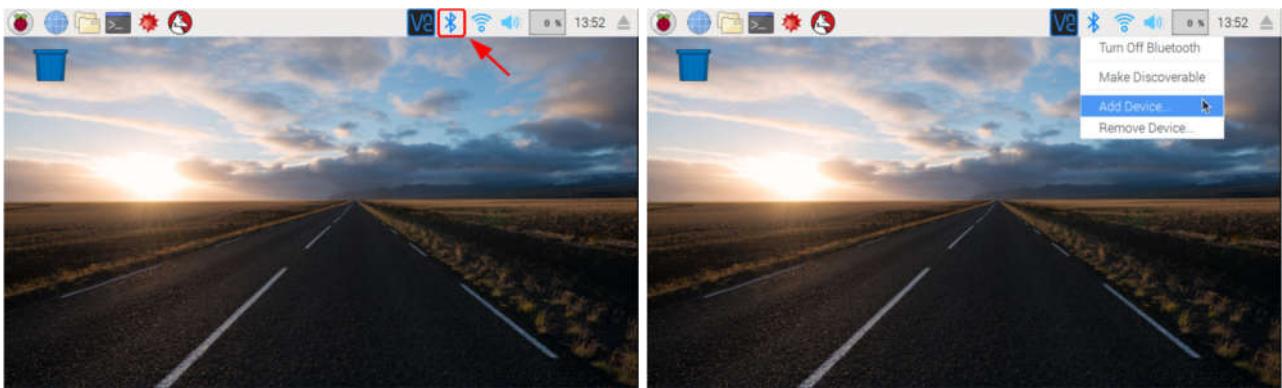
블루투스 장치를 연결하기 위해서는 몇 가지 단계를 거쳐야 한다. 아래 5 단계 중 4 번째 페어링 단계에서 장치에 따라 단말기에서 PIN Code 를 입력하거나, 장치에서 PIN Code 를 입력하거나, 또는 PIN Code 입력 없이 진행될 수 있다.

- Step 1.** 블루투스 장치의 전원 On
- Step 2.** 블루투스 장치를 검색( 페어링 )모드로 변경
- Step 3.** 단말기에서 장치 검색
- Step 4.** 단말기와 검색된 장치 페어링
- Step 5.** 단말기와 페어링된 장치 연결

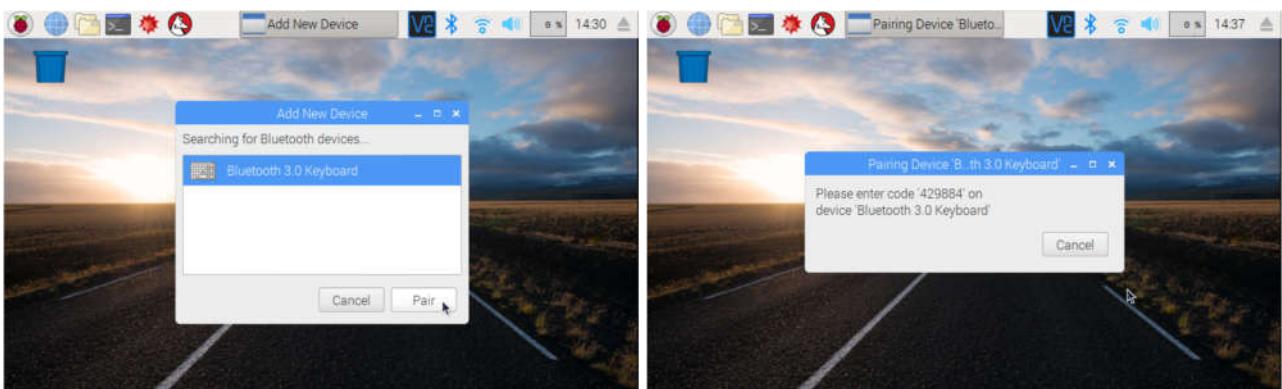
#### 1. 1. 블루투스 키보드 연결

우선 블루투스 장치( 키보드 )의 전원을 On 시키고, 페어링 모드로 진입한다. ( 장치의 페어링 모드 진입 방법은 장치 매뉴얼을 참조한다. )

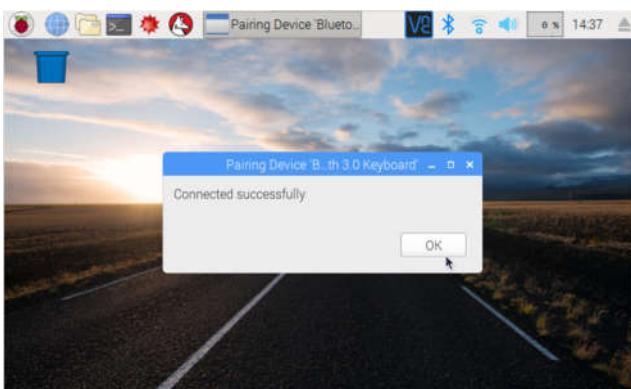
1. 블루투스 장치관리자 아이콘 클릭 클릭
2. "Add Device..." 메뉴 선택



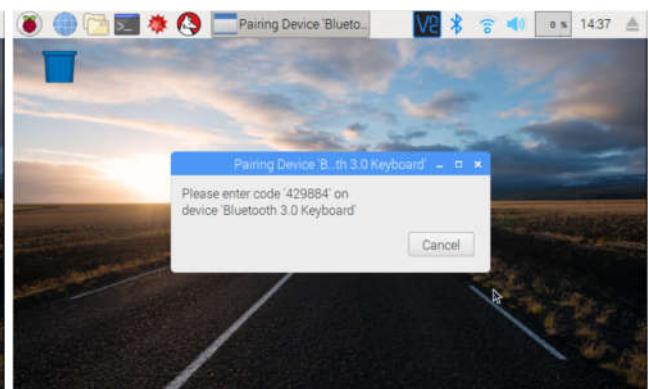
### 3. 연결할 장치(키보드) 선택 후 "Pair" 클릭



### 5. 연결완료. "OK" 클릭



### 4. 연결할 장치(키보드)에서 팝업창의 PIN 입력



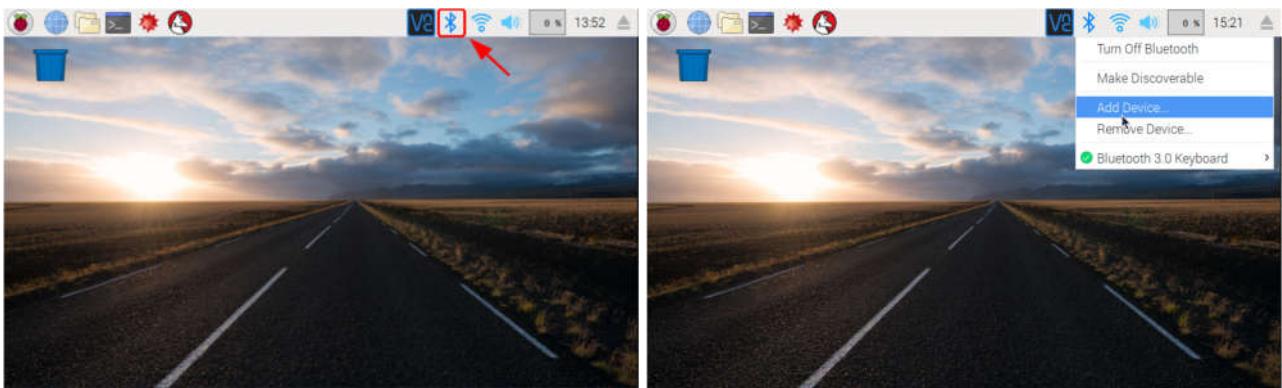
### 6. 연결한 블루투스 키보드를 테스트한다.

## 2. 2. 블루투스 마우스 연결

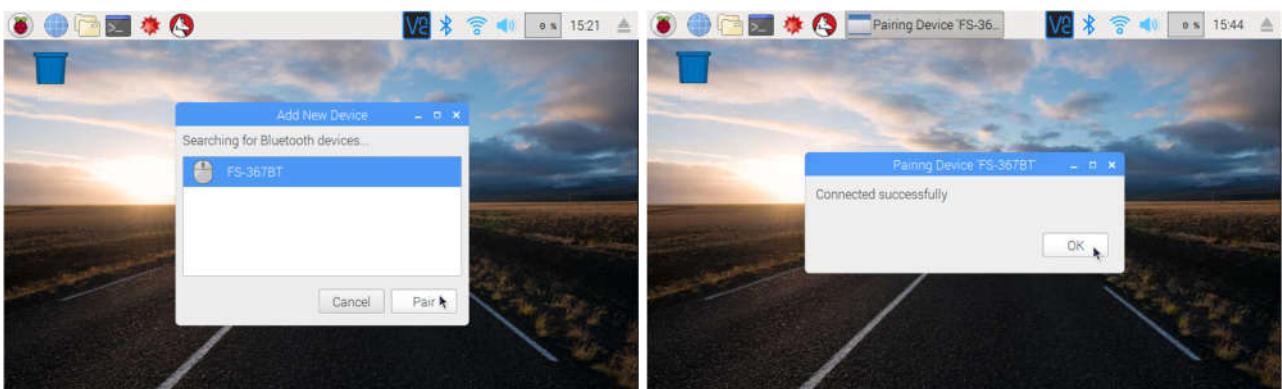
우선 블루투스 장치( 마우스 )의 전원을 On 시키고, 페어링 모드로 진입한다. ( 장치의 페어링 모드 진입 방법은 장치 매뉴얼을 참조한다. )

### 1. 블루투스 장치관리자 아이콘 클릭 클릭

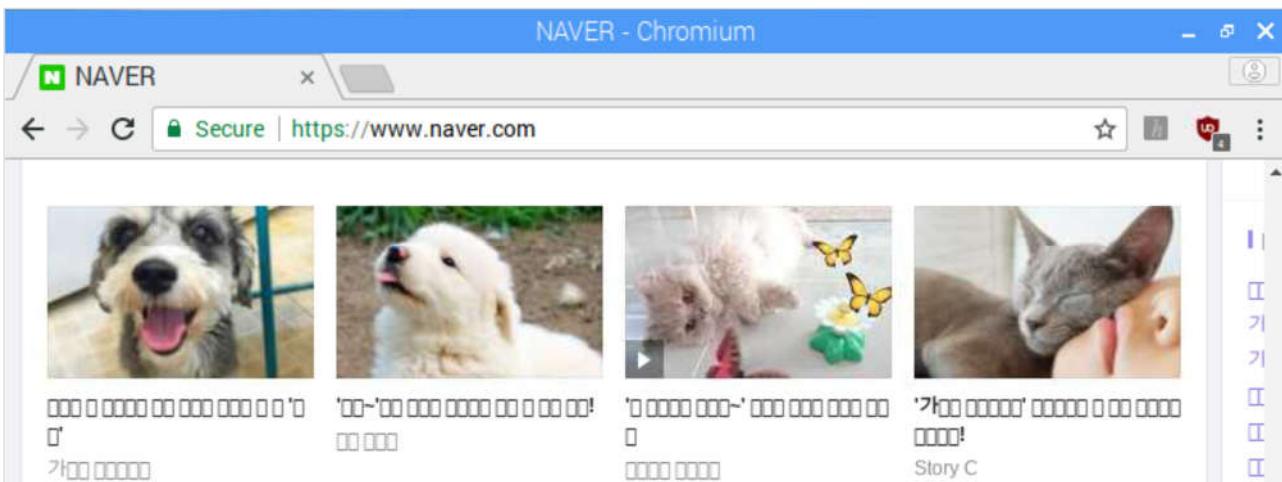
### 2. "Add Device..." 메뉴 선택



### 3. 연결할 장치(마우스) 선택 후 "Pair" 클릭



### 5. 연결한 블루투스 마우스의 동작을 테스트한다.



## 2.4.4. 한글 사용을 위한 설정

웹브라우저에서 한글이 사용되는 웹페이지를 열어보자.

한글폰트가 설치되어 있지 않아 한글이 제대로 표시되지 않는다. 한글이 제대로 표시되려면 한글 폰트를 설치해야 한다. 터미널 실행 후, 우선 update, upgrade를 실행한다.

## 1. 1. 한글 폰트 설치

먼저 업데이트를 실행한다.

```
$ sudo apt-get update
```

업데이트가 끝나면 업그레이드를 실행한다.

```
$ sudo apt-get upgrade
```

한글 폰트를 설치한다.

```
$ sudo apt-get install fonts-unfonts-core -y
```

X-Windows 가 설치된 한글 폰트를 적용하도록 시스템을 재시작 시킨다.

```
$ sudo reboot
```

시스템 리부팅 후, 웹브라우저를 구동하여 한글이 사용된 웹페이지를 열어 확인한다. ( 아래 그림은 네이버: <https://naver.com> 에서 테스트한 화면이다. )



## 2. 2. 한글 입력기 설치

아직은 문서편집기나 네이버 검색창에 한글입력이 불가능하다. 한글 입력을 지원하는 입력기( Input Method )가 설치된 것이 없기 때문이다. `

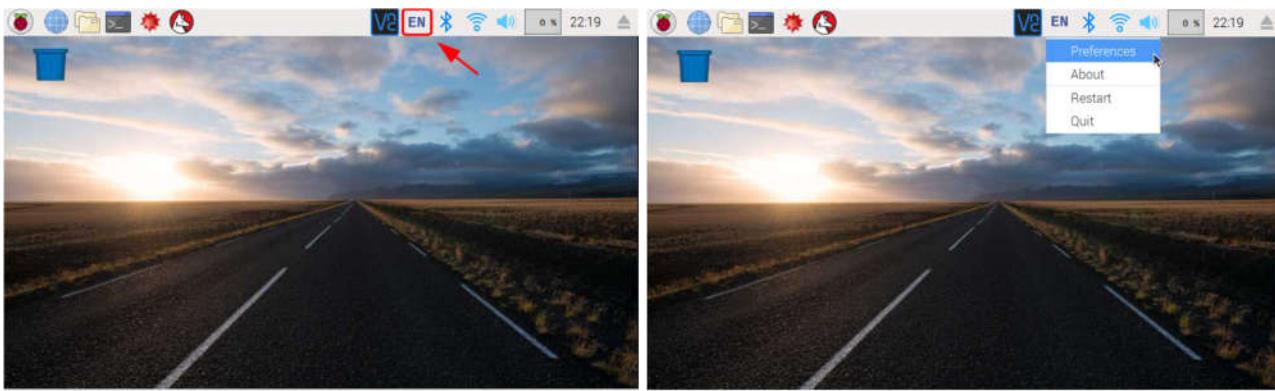
한글 입력기들 중 '**ibus-hangul**'을 설치하자.

```
$ sudo apt-get install ibus-hangul -y
```

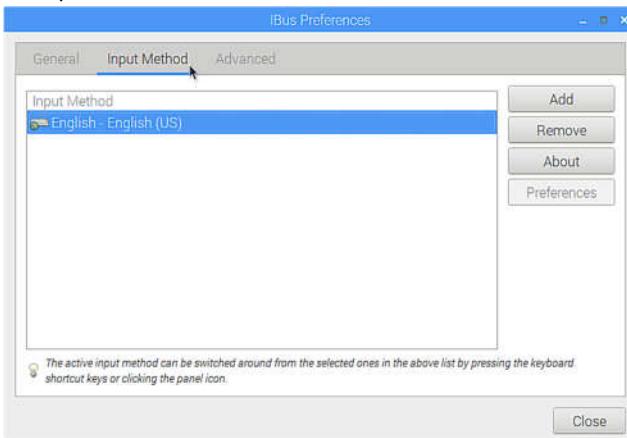
설치가 완료되면 역시 설치된 한글 입력기를 X-Windows 가 Load 하도록 시스템을 재시작 한다.

```
$ sudo reboot
```

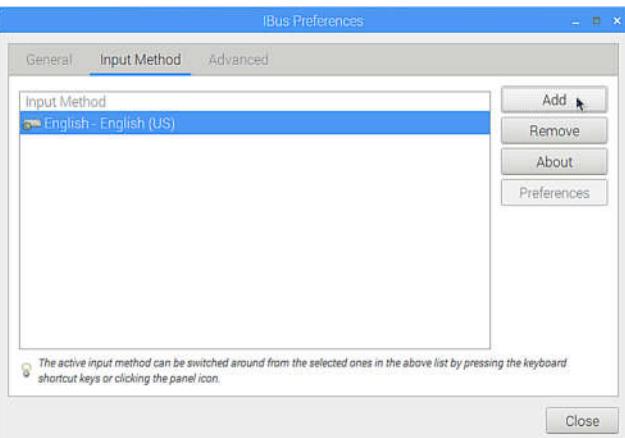
1. 입력기 아이콘을 마우스 우측 버튼으로 클릭
2. Preference 메뉴 선택



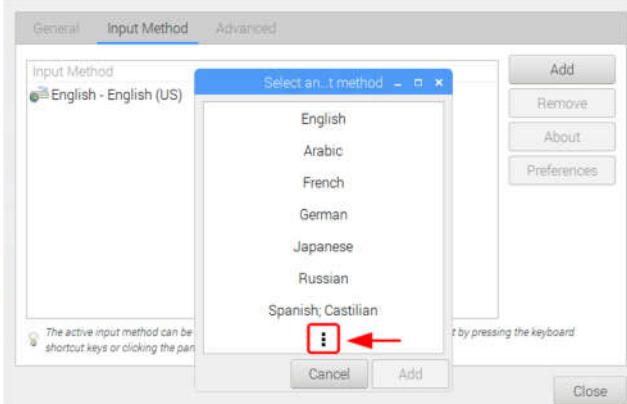
### 3. Input Method 탭 선택



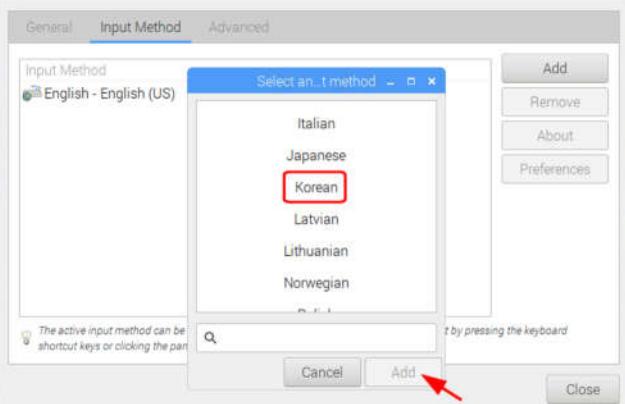
### 4. Add 버튼 클릭



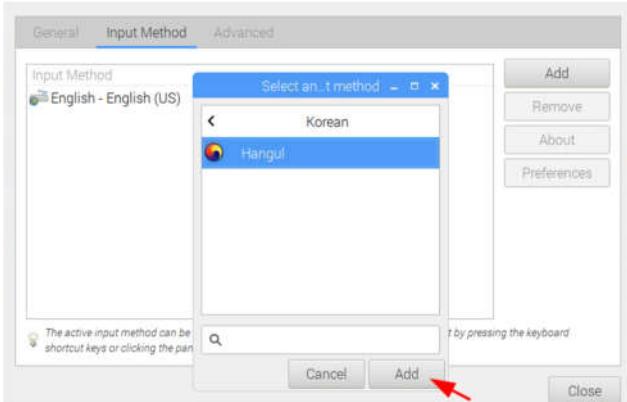
### 5. Korean 선택을 위해 아래 표시된 부분 클릭



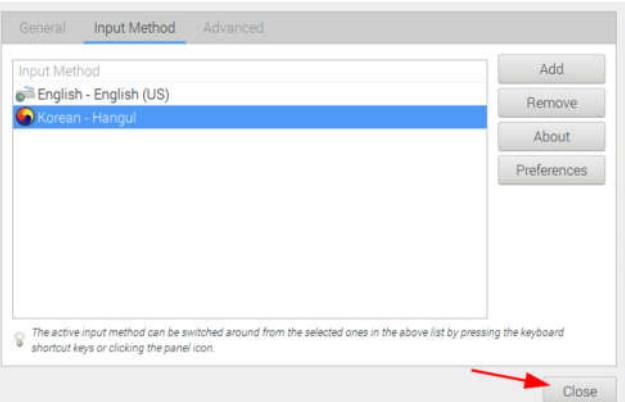
### 6. Korean



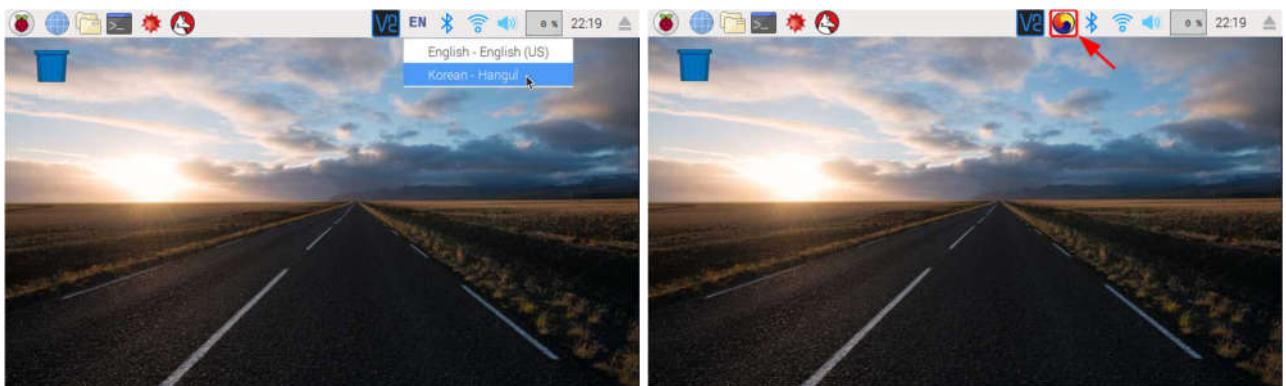
### 7. Hangul 선택 후, Add 버튼 클릭



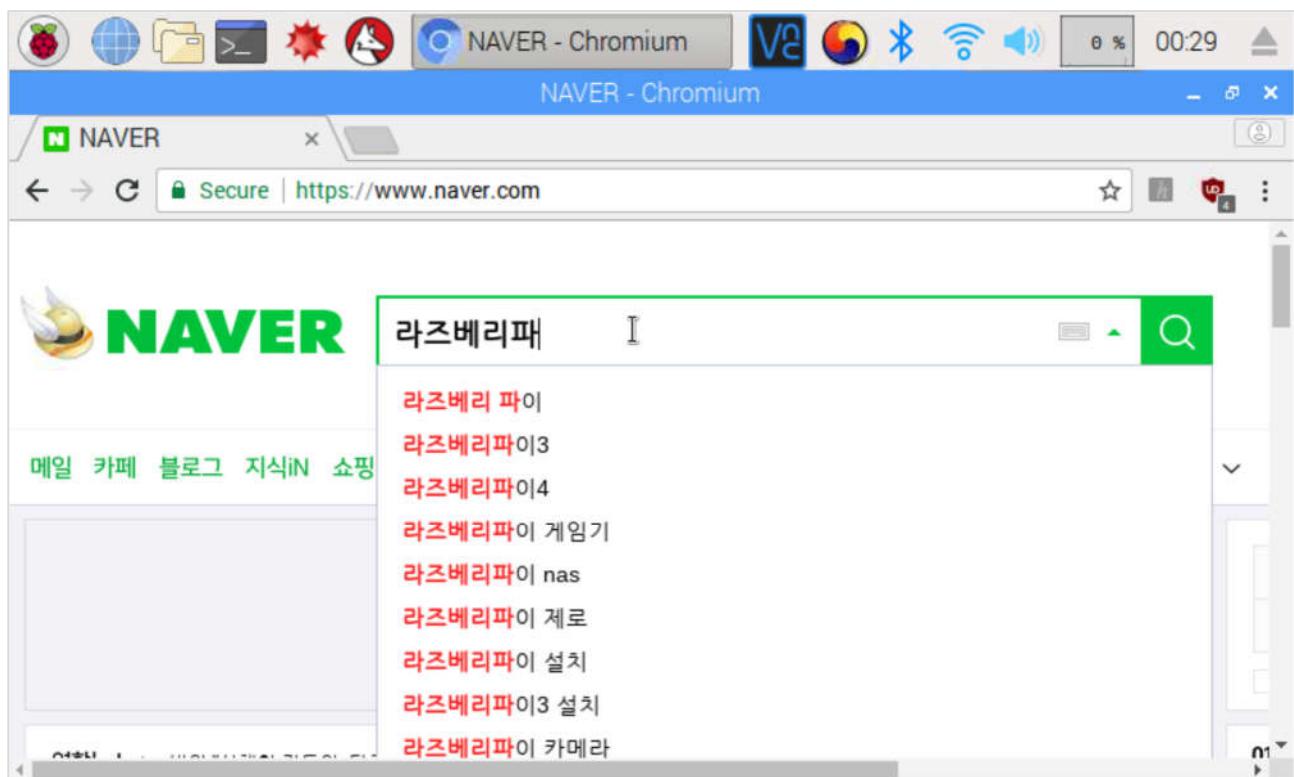
### 8. Korean - Hangul 선택 후, Close 버튼 클릭



9. 입력기 아이콘을 클릭하여 Korean-Hangul 선택 10. 입력기 아이콘이 태극문양으로 변경된 것을 확인



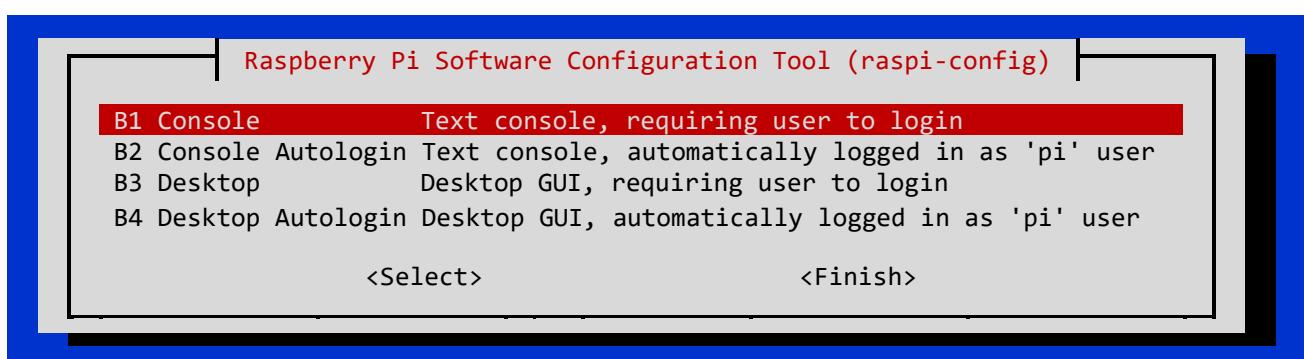
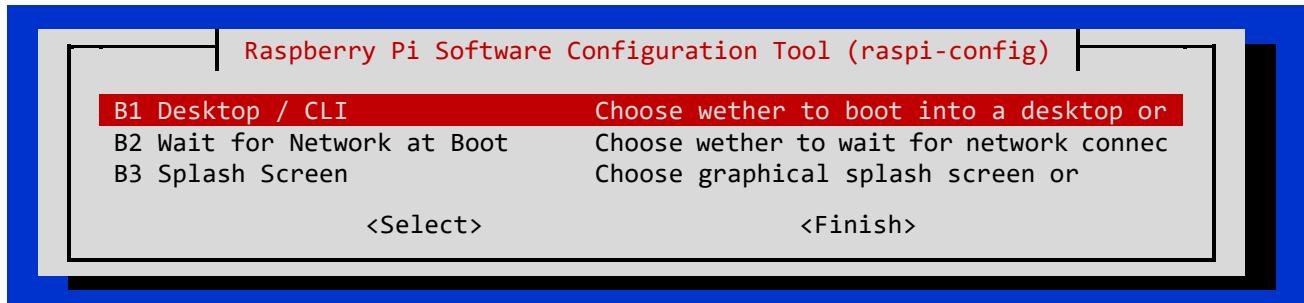
이제 문자 입력 시, **왼쪽 Shift** 와 **Space Bar** 조합으로 한글/영문 입력모드를 전환하여 한글 입력이 가능하다.



## 3. LINUX 기본

### 3.1. 기초 리눅스 명령어 및 용어

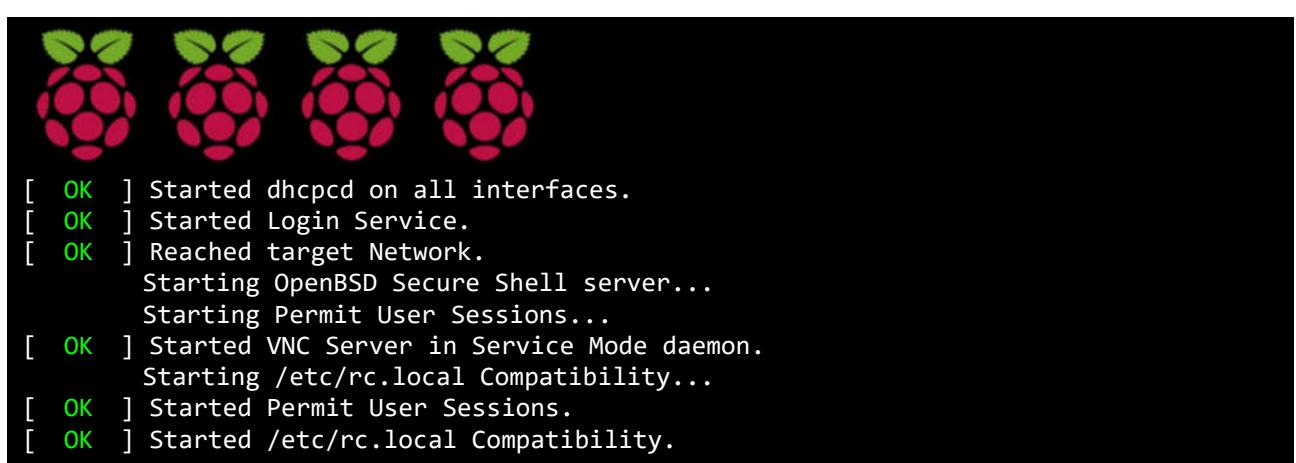
개념의 이해와 실습을 위해 라즈베리파이에 키보드 모니터를 연결하고, 전원을 투입한다. 부팅이 완료되면 터미널을 열어 raspi-config 를 실행하여 Boot Option 을 첫 번째 옵션인 'B1. Console Text console, requiring user to login' 을 선택하고 빠져 나와 라즈베리를 리부팅 시킨다.



```
$ sudo reboot
```

#### 3.1.1. 가상 터미널

아래는 CLI( Command Line Interface ) 모드에서, 부팅 직 후의 라즈베리파이 화면에 표시된 내용이다.



```
Starting Hold until boot process finishes up...
      Starting Terminate Plymouth Boot Screen...

Raspbian GNU/LINUX 9 raspberrypi tty1
Raspberrypi login: pi
Password:

Last login: Tue Jan 9 20:44:33 KST 2018 on tty1
Linux raspberrypi 4.9.59-v7+ #1047 SMP Sun Oct 29 12:19:23 GMT 2017 armv7l
The Program included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTLY NO WARRANTY, to the extent
Permitted by applicable law.
pi@raspberryPi:~ $
```

노란색으로 표시한 문자열은

“**Raspbian GNU/LINUX 9**” : 구동된 리눅스의 종류 및 버전  
“**raspberrypi tty1**” : 현재 Login 에 사용된 단말( 터미널 **tty1: teletypewriter 1** )

결국 “Raspbian GNU/LINUX 9( Raspbian Stretch 9 )” 리눅스가 “raspberrypi tty1” 단말에서 구동 중이라는 의미이며, 그 아랫줄에는 로그인 계정과 패스워드를 입력받아 해당단말에서 라즈베리파이에 로그인하는 과정인 것이다. 결국 마지막 행의 프롬프트가 나타난 것으로 보아 정상적으로 로그인 된 것을 알 수 있다.

현재 상태에서 다른 터미널을 통해 라즈베리파이에 로그인하기 위해 키보드의 “Alt + F2” 를 눌러보면 다음과 같은 화면이 나타난다.

```
Raspbian GNU/LINUX 9 raspberrypi tty2
Raspberrypi login: root
Password:
Last login: Tue Jan 8 22:10:32 KST 2018 on tty2
Linux raspberrypi 4.9.59-v7+ #1047 SMP Sun Oct 29 12:19:23 GMT 2017 armv7l

The Program included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTLY NO WARRANTY, to the extent
Permitted by applicable law.
root@raspberrypi:/root #
```

역시 이 번에도 노란색으로 표시한 문자열을 살펴 보면, 이번에는 “Raspbian GNU/LINUX 9” 리눅스가 “raspberrypi **tty2**” 단말에서 구동 중이라는 의미라는 것을 알 수 있을 것이다.

먼저 로그인한 “raspberrypi tty1” 단말로 이동하려면 “Alt + F1” 을 누르면 된다.

별도의 작업 없이 열 수 있는 가상 터미널은 모두 5 개로 “Alt + Fn” 으로 ttyn 가상터미널을 열 수 있다.  
( n = 2, 3, 4, 5, 6 )

tty1 단말은 가상터미널이 아니다. 직접 연결된 키보드, 모니터 콘솔이 tty1 단말이 된다.( GUI 환경에서 GUI 가 구동 중인 단말은 tty7 이다. )

### 3.1.2. 사용자 계정 관련 명령어

**사용자 계정( User Account )**은 유닉스나 리눅스 시스템에서 컴퓨터 사용을 위해 관리자로부터 발급받는 사용자 인증을 위한 Login Name 과 이 때 동시에 생성되는 디스크 상의 작업공간을 의미한다.

리눅스와 유닉스는 멀티유저( Multi-User ) 네트워크 운영체제( Network OS )이기 때문에 이 계정( account )은 매우 중요한 개념이다. 계정은 크게 일반 사용자 계정과 관리자 계정으로 나누어 볼 수 있다. “라즈베리파이” 의 경우 기본적으로 **기본 사용자 계정 ‘pi’ 와 관리자 계정 ‘root’가 만들어진 상태로 제공된다.**

#### **prompt**

CLI( Command Line Interface )에서, 또는 터미널 실행 시 아래와 같은 문자열을 만나게 되는 데, 이것을 **프롬프트( Prompt )**라고 한다.

```
pi@raspberrypi:~ $
```

기본적으로 프롬프트가 화면에 표시되었다는 것은 새로 수행할 명령을 입력 받을 준비가 되어 있음을 나타낸다. 새로운 명령을 입력하면, 명령 수행이 완료 되어야만 다시 프롬프트가 나타난다. ( 명령 수행이 성공하면, 새 프롬프트가, 실패하면 오류 메시지와 새 프롬프트가 표시된다. )

또, 프롬프트를 살펴보면 알 수 있는 몇가지 정보가 있는데 위 프롬프트를 예로들면

1. 사용자 명( **username** ) “pi” 로 로그인( login ) 하였으며.,
2. 로그인 한 컴퓨터 명( **hostname** )은 “raspberryPi” 이다.
3. 사용하고 있는 쉘( **shell** )은 ‘bash’ 이다.

라는 것을 알 수 있다.

#### **who**

호스트 컴퓨터에 로그인되어 있는 모든 사용자 정보를 출력한다. 아래의 예를 살펴보면

```
pi@raspberryPi:~ $ who
pi      tty1      2018-01-10 03:58
root    tty2      2018-01-10 04:01
pi@raspberryPi:~ $
```

tty1 터미널(단말기)에는 사용자명 'pi'와, tty2 터미널에는 사용자명 'root', 모두 2 명의 사용자가 현재 로그인하고 있음을 보여준다.

#### **whoami**

현재 ‘어떤 계정으로 로그인하고 있는가?’를 알려준다.

```
pi@raspberryPi:~ $ whoami  
pi  
pi@raspberryPi:~ $
```

키보드에서 "Alt + F2" 를 눌러 tty2 가상단말 화면으로 전환 후, "whoami" 명령을 실행해보자.

호스트 컴퓨터에 로그인되어 있는 모든 사용자 정보를 출력한다. 아래의 예를 살펴보면

```
root@raspberrypi:/root # whoami  
root  
root@raspberrypi:/root #
```

다음은 root 의 패스워드를 변경하는 과정이다.

```
pi@raspberrypi:~ $ sudo passwd root  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully  
pi@raspberrypi:~ $
```

모든 LINUX 또는 UNIX 컴퓨터의 관리자 계정명이 "root" 이므로 "pi"는 일반 사용자 계정이라는 것도 유추할 수 있다. 패스워드 변경 명령 "passwd"를 사용하여 "root" 계정의 패스워드를 변경해 보자.

```
pi@raspberryPi:~ $ passwd root  
passwd: You may not view or modify password information for root.  
pi@raspberryPi:~ $
```

"당신은 'root' 계정의 패스워드 정보를 열람하거나, 변경할 수 없다." 라는 일종의 permission 에러가 발생했다. 이 문제를 해결하는 방법 중 한 가지가 "sudo" 명령을 사용하는 것이다.

**sudo - sudo( substitution do ) : ( 관리자를 ) 대신하여 실행.**

'sudo'는 단독으로 실행되는 명령이 아니다. 관리자( root ) 권한이 필요한 명령을 실행해야 할 경우 명령 앞에 덧붙여 사용한다. ( 해당 계정이 sudo 그룹에 속해 있어야 사용할 수 있다. )

```
pi@raspberryPi:~ $ sudo passwd root  
Enter new UNIX password:
```

에러 메시지 대신 새 패스워드를 입력하라는 메시지가 나타났다. 변경할 패스워드와 Enter 키를 입력하면

```
pi@raspberryPi:~ $ sudo passwd root  
Enter new UNIX password:  
Retype new UNIX password:
```

한 번 더 변경할 패스워드 입력을 요구하는 메시지가 나타난다. 새로 사용할 패스워드와 Enter 를 다시 한 번 입력하면, 패스워드 업데이트( 변경이 )가 성공했다는 메시지가 출력되고, 프롬프트가 나타난다.

```
passwd: password updated successfully  
pi@raspberryPi:~ $
```

### **su**

root 계정의 패스워드를 변경하는 또 다른 한 가지 방법은 "su" 명령을 이용하는 방법이다. "su" 명령을 이용하면 일반 사용자로 로그인하여, 로그아웃 없이 잠시 동안 슈퍼유저( Super User – root ) 권한의 명령어를 실행할 수 있다. 그러면 root 계정의 패스워드를 변경해 보자.

실행하면 다음과 같이 패스워드( root 계정의 패스워드 ) 입력을 요구한다.

```
pi@raspberryPi:~ $ su  
Password:
```

root 의 패스워드와 Enter 를 입력하면 아래와 같이 root 로 로그인한 프롬프트가 나타난다.

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

"passwd" 를 입력하고 Enter 를 입력하면, 아래와 같이 신규 패스워드 입력 요구 메시지가 나타난다. 새 패스워드와 Enter 를 입력한다.

```
root@raspberrypi:/home/pi# passwd  
Enter new UNIX password:
```

패스워드와 Enter 키를 입력하면 한 번 더 변경할 패스워드 입력을 요구하는 메시지가 나타난다.

```
Retype new UNIX password:
```

다시 한 번 root 의 패스워드와 Enter 를 입력하면 아래와 같이 root 로 로그인한 프롬프트가 나타난다.

```
passwd: password updated successfully  
root@raspberrypi:/home/pi#
```

앞서 "su" 명령을 실행하기 전의 사용자 "pi" 로 로그인한 상태로 되돌아가기 위해 "exit" 명령과 Enter 를 입력한다.

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

"exit" 메시지가 출력되고, "pi" 로그인한 프롬프트가 나타난다.

```
exit  
pi@raspberrypi:~ $
```

### **logout**

네트워크 운영체제인 유닉스나 리눅스에서는 컴퓨터를 사용하기 위해 반드시 login 과정을 거쳐야만 한다. 이 로그인 과정에서 빠져 나오는 명령이 바로 logout 이다. ( putty 를 통한 원격 로그인의 경우 logout 대신 exit 를 사용한다. )

### **passwd**

사용자의 login password 를 변경하는 명령이다. 다음처럼 단독으로 사용할 경우, 현재 로그인 되어 있는 사용자의 패스워드를 변경하게 된다.

```
pi@raspberryPi:~ $ passwd  
Changing password for pi.  
(current) UNIX password
```

"pi"의 패스워드를 변경한다는 메시지와, 현재 사용중인 패스워드 입력을 요구하는 메시지가 나타난다. 여기서 새로 사용할 패스워드와 Enter를 입력하면 다음과 같이 새로 사용할 패스워드 입력을 요구한다.

```
Enter new UNIX password:
```

새 패스워드와 Enter를 입력하면, 신규 패스워드의 재입력을 요구하는 메시지가 나타난다.

```
Retype new UNIX password:
```

입력한 신규 패스워드와 Enter를 한 번 더 입력하면 패스워드 업데이트(변경이)가 성공했다는 메시지가 출력되고, 다시 프롬프트가 나타난다.

```
passwd: password updated successfully  
pi@raspberryPi:~ $
```

### 3.1.3. 파일 및 디렉토리 관련 명령어

디렉토리(directory)는 MS-Windows에서의 폴더(folder)와 동일한 개념으로 어떤 기준에 의해 파일(file)들을 모아둔 파일 시스템 상의 논리적 공간이다.

**pwd**(print working directory) 현재 사용자의 경로를 화면에 출력한다.

```
pi@raspberryPi:~ $ pwd  
/home/pi  
pi@raspberryPi:~ $
```

**cd**(change directory) 디렉토리 경로 변경.

```
pi@raspberryPi:~ $ cd /  
pi@raspberryPi:/ $ ls  
bin dev home lost+found mnt proc run srv tmp var  
boot etc lib media opt root sbin sys usr  
pi@raspberryPi:/ $ cd dev  
pi@raspberryPi:/dev $ cd /home/pi  
pi@raspberryPi:~ $ cd ..  
pi@raspberryPi:/home $ cd ..  
pi@raspberryPi:/ $ ls  
bin dev home lost+found mnt proc run srv tmp var  
boot etc lib media opt root sbin sys usr  
pi@raspberryPi:/ $ cd  
pi@raspberryPi:~ $
```

**ls** 파일 및 디렉토리 목록 출력 명령

단독으로 사용할 경우 현재 위치의 파일 및 디렉토리 목록을 화면 출력. 뒤에 특정 디렉토리 위치를 적어 줄 경우 해당 디렉토리의 파일 및 디렉토리 목록을 화면 출력.

```
pi@raspberryPi:~ $ ls
2018-01-04-135058.png 2018-01-04-144701.png Desktop Music python_games
2018-01-04-135155.png a1234.txt Documents Pictures Templates
2018-01-04-142041.png a123.txt Downloads Public Videos
pi@raspberryPi:~ $
```

-a 적용

```
pi@raspberryPi:~ $ ls -a
. .bash_logout .local Templates
.. .bashrc Music .themes
2018-01-04-135058.png .cache .nano .thumbnails
2018-01-04-135155.png .config Pictures Videos
2018-01-04-142041.png .dbus .pki .vnc
2018-01-04-144701.png Desktop .profile .Xauthority
a1234.txt Documents Public .xsession-errors
a123.txt Downloads python_games .xsession-errors.old
.bash_history .gnupg .ssh
pi@raspberryPi:~ $
```

-l 적용

```
pi@raspberryPi:~ $ ls -l
total 2012
-rw-r--r-- 1 pi pi 551391 Jan  4 13:50 2018-01-04-135058.png
-rw-r--r-- 1 pi pi 551196 Jan  4 13:51 2018-01-04-135155.png
-rw-r--r-- 1 pi pi 424738 Jan  4 14:20 2018-01-04-142041.png
-rw-r--r-- 1 pi pi 481145 Jan  4 14:47 2018-01-04-144701.png
-rw-r--r-- 1 pi pi     62 Jan 11 14:08 a1234.txt
-rw-r--r-- 1 pi pi     89 Jan 11 14:10 a123.txt
drwxr-xr-x 2 pi pi 4096 Jan 11 00:22 Desktop
drwxr-xr-x 5 pi pi 4096 Nov 29 11:22 Documents
drwxr-xr-x 4 pi pi 4096 Jan 11 00:22 Downloads
drwxr-xr-x 2 pi pi 4096 Nov 29 11:56 Music
drwxr-xr-x 2 pi pi 4096 Nov 29 11:56 Pictures
drwxr-xr-x 2 pi pi 4096 Nov 29 11:56 Public
drwxr-xr-x 2 pi pi 4096 Nov 29 11:22 python_games
drwxr-xr-x 2 pi pi 4096 Nov 29 11:56 Templates
drwxr-xr-x 2 pi pi 4096 Nov 29 11:56 Videos
pi@raspberryPi:~ $
```

- al 적용

```
pi@raspberryPi:~ $ ls -al
total 2168
drwxr-xr-x 22 pi pi 4096 Jan 11 14:10 .
```

```

drwxr-xr-x  3 root root  4096 Nov 29 10:22 ..
-rw-r--r--  1 pi  pi  551391 Jan  4 13:50 2018-01-04-135058.png
-rw-r--r--  1 pi  pi  551196 Jan  4 13:51 2018-01-04-135155.png
-rw-r--r--  1 pi  pi  424738 Jan  4 14:20 2018-01-04-142041.png
-rw-r--r--  1 pi  pi  481145 Jan  4 14:47 2018-01-04-144701.png
-rw-r--r--  1 pi  pi      62 Jan 11 14:08 a1234.txt
-rw-r--r--  1 pi  pi      89 Jan 11 14:10 a123.txt
-rw-----  1 pi  pi    3689 Jan 11 12:28 .bash_history
-rw-r--r--  1 pi  pi    220 Nov 29 10:22 .bash_logout
-rw-r--r--  1 pi  pi    3523 Nov 29 10:22 .bashrc
drwxr-xr-x  8 pi  pi  4096 Jan  4 20:38 .cache
drwx----- 15 pi  pi  4096 Jan 11 00:22 .config
drwx-----  3 pi  pi  4096 Nov 29 12:01 .dbus
drwxr-xr-x  2 pi  pi  4096 Jan 11 00:22 Desktop
drwxr-xr-x  5 pi  pi  4096 Nov 29 11:22 Documents
drwxr-xr-x  4 pi  pi  4096 Jan 11 00:22 Downloads
drwx-----  3 pi  pi  4096 Nov 29 11:56 .gnupg
drwxr-xr-x  3 pi  pi  4096 Nov 29 11:22 .local
drwxr-xr-x  2 pi  pi  4096 Nov 29 11:56 Music
drwxr-xr-x  2 pi  pi  4096 Jan 10 05:28 .nano
drwxr-xr-x  2 pi  pi  4096 Nov 29 11:56 Pictures
drwx-----  3 pi  pi  4096 Jan  4 18:54 .pki
-rw-r--r--  1 pi  pi     675 Nov 29 10:22 .profile
drwxr-xr-x  2 pi  pi  4096 Nov 29 11:56 Public
drwxr-xr-x  2 pi  pi  4096 Nov 29 11:22 python_games
drwx-----  2 pi  pi  4096 Jan  5 04:45 .ssh
drwxr-xr-x  2 pi  pi  4096 Nov 29 11:56 Templates
drwxr-xr-x  3 pi  pi  4096 Nov 29 11:56 .themes
drwx-----  4 pi  pi  4096 Jan  5 19:15 .thumbnails
drwxr-xr-x  2 pi  pi  4096 Nov 29 11:56 Videos
drwx-----  3 pi  pi  4096 Nov 29 12:05 .vnc
-rw-----  1 pi  pi     161 Jan 11 12:28 .Xauthority
-rw-----  1 pi  pi    4332 Jan 11 12:28 .xsession-errors
-rw-----  1 pi  pi   71708 Jan 11 09:22 .xsession-errors.old
pi@raspberryPi:~ $

```

### **mkdir( make directory ) 디렉토리 생성**

```

pi@raspberryPi:~ $ ls
2018-01-04-135058.png  2018-01-04-144701.png  Desktop      Music      python_games
2018-01-04-135155.png  a1234.txt                Documents    Pictures   Templates
2018-01-04-142041.png  a123.txt                Downloads   Public    Videos
pi@raspberryPi:~ $ mkdir example
pi@raspberryPi:~ $ ls
2018-01-04-135058.png  a1234.txt  Downloads  Public
2018-01-04-135155.png  a123.txt   example   python_games
2018-01-04-142041.png  Desktop    Music    Templates

```

```
2018-01-04-144701.png  Documents  Pictures  Videos  
pi@raspberryPi:~ $
```

```
pi@raspberryPi:~ $ cd example  
pi@raspberryPi:~/example $ mkdir subdir1  
pi@raspberryPi:~/example $ mkdir subdir2  
pi@raspberryPi:~/example $ mkdir subdir3  
pi@raspberryPi:~/example $ ls  
subdir1  subdir2  subdir3  
pi@raspberryPi:~/example $
```

**rmdir( remove directory )** 디렉토리 삭제.

```
pi@raspberryPi:~/example $ rm dir subdir1  
pi@raspberryPi:~/example $ ls  
subdir2  subdir3  
pi@raspberryPi:~/example $
```

**cp( copy )** 파일 복사.

```
pi@raspberryPi:~/example $ cd  
pi@raspberryPi:~ $ ls  
2018-01-04-135058.png  a1234.txt  Downloads  Public  
2018-01-04-135155.png  a123.txt   example    python_games  
2018-01-04-142041.png  Desktop    Music     Templates  
2018-01-04-144701.png  Documents  Pictures  Videos  
pi@raspberryPi:~ $ cp ./a1234.txt ./example/subdir2  
pi@raspberryPi:~ $ ls ./example/subdir2  
a1234.txt  
pi@raspberryPi:~ $ cp ./a123.txt ./example/subdir2  
pi@raspberryPi:~ $ ls ./example/subdir2  
a1234.txt  a123.txt  
pi@raspberryPi:~ $
```

**mv( move )** 파일 이동 명령.

```
pi@raspberryPi:~ $ mv *.png ./example/subdir3  
pi@raspberryPi:~ $ ls  
a1234.txt  Desktop    Downloads  Music      Public       Templates  
a123.txt   Documents  example   Pictures  python_games  Videos  
pi@raspberryPi:~ $ ls ./example/subdir3  
2018-01-04-135058.png  2018-01-04-142041.png  
2018-01-04-135155.png  2018-01-04-144701.png  
pi@raspberryPi:~ $
```

**rm( remove )** 파일 삭제

```

pi@raspberryPi:~ $ rm *.txt
pi@raspberryPi:~ $ ls
Desktop Downloads Music Public Templates Documents
example Pictures python_games Videos
pi@raspberryPi:~ $ rm /home/pi/example/subdir3/*.png
pi@raspberryPi:~ $ ls /home/pi/example/subdir3
pi@raspberryPi:~ $

```

**touch** 크기가 0인 파일 생성. 파일이 존재할 경우 파일 수정 일시를 명령이 실행된 일시로 변경.

```

pi@raspberryPi:~ $ touch test_touch01.txt
pi@raspberryPi:~ $ touch test_touch02.txt
pi@raspberryPi:~ $ ls -al
total 2168
drwxr-xr-x 22 pi pi 4096 Jan 11 14:10 .
drwxr-xr-x  3 root root 4096 Nov 29 10:22 ..
-rw-r--r--  1 pi pi 0 Jan 11 14:08 test_touch01.txt
-rw-r--r--  1 pi pi 0 Jan 11 14:10 test_touch02.txt
-rw-------  1 pi pi 3689 Jan 11 12:28 .bash_history
-rw-r--r--  1 pi pi 220 Nov 29 10:22 .bash_logout
-rw-r--r--  1 pi pi 3523 Nov 29 10:22 .bashrc
drwxr-xr-x  8 pi pi 4096 Jan  4 20:38 .cache
drwx----- 15 pi pi 4096 Jan 11 00:22 .config
.
.
.
drwx----- 3 pi pi 4096 Nov 29 12:05 .vnc
-rw------- 1 pi pi 161 Jan 11 12:28 .Xauthority
-rw------- 1 pi pi 4332 Jan 11 12:28 .xsession-errors
-rw------- 1 pi pi 71708 Jan 11 09:22 .xsession-errors.old
pi@raspberryPi:~ $

```

**chmod( change mode )** 파일 권한 변경 명령

다음은 hello.py 파일의 권한을 알아보기 위해 'ls -al hello.py' 명령을 수행한 결과다.

```

pi@raspberrypi:~ $ ls -al hello.py
-rw-r--r-- 1 pi pi 21 Jul 11 14:10 hello.py
pi@raspberrypi:~ $

```

모든 권한이 부여될 경우 빨간색으로 표시한 부분은 rwxrwxrwx 로 표시되고,

1. 첫번째 rwx 는 이 파일의 소유자의 권한을, 에게 읽기(r), 쓰기(w), 실행(x) 권한이 있다는 의미이고,
2. 두번째 rwx 는 이 파일의 소유자와 같은 그룹 사용자의 권한을,
3. 세번째 rwx 는 이 파일의 소유자도, 같은 그룹도 아닌 사용자의 권한을 의미하며,

rwx 의 'r'은 읽기, 'w'는 쓰기, 'x'는 실행 권한을 나타낸다.

이 때 rwx 의 각 자리를 이진수 3 자리로 나타내면 rwx = 111 = 7 에 해당한다. 따라서 위 'hello.py'의 경우 rw- r-- r-- 이므로 이는 110 100 100, 즉 644 로 나타낼 수 있다. 모든 사용자에게 실행 속성을 부여해보자.

```
pi@raspberrypi:~ $ chmod 755 hello.py  
pi@raspberrypi:~ $ ls -al hello.py  
-rwxr-xr-x 1 pi pi 21 Jul 11 14:10 hello.py
```

이번엔 다른 방법으로 모든 사용자에게 부여한 실행 권한을 제거해 보자.

```
pi@raspberrypi:~ $ chmod -x hello.py  
pi@raspberrypi:~ $ ls -al hello.py  
-rw-r--r-- 1 pi pi 21 Jul 11 14:10 hello.py
```

### cat 파일 내용 화면 출력 명령

텍스트 형식의 파일 내용을 표준출력장치( 일반적으로 화면 )로 출력.

```
pi@raspberrypi:~ $ cat hello.py  
print "Hello, World~"  
pi@raspberrypi:~ $
```

## 3.1.4. 시스템 종료 명령

시스템 종료 시에는 'shutdown' 명령을 통해 안전하게 시스템을 종료해야 한다. 또한 서버 용 운영체제인 리눅스의 성격 상, 다수의 사용자가 작업 중인 환경으로 간주 할 때, 이 명령은 관리자만이 실행할 수 있다.

### shutdown

sudo shutdown –옵션( -h: halt 시스템 종료, -r: reboot 시스템 재시작 ) 시간( 분 단위 정수 )

```
pi@raspberryPi:~ $ sudo shutdown -r 30
```

( 30 분 후 시스템 재시작. 로그온 되어 있는 모든 사용자와 단말에 그 안내 메시지가 매 5 분마다 출력된다. )

**halt** sudo halt ( sudo shutdown -h now or sudo shutdown -h 0 )

**reboot** sudo reboot ( sudo shutdown -r now or sudo shutdown -r 0 )

## 3.2. 리눅스 네트워크 기초

### 3.2.1. 기본 개념 및 용어

아래는 현재 시스템의 네트워크 구성을 살펴 보기 위해 ifconfig 명령을 실행한 결과이다.

```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether b8:27:eb:4f:6c:aa txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1 (Local Loopback)
            RX packets 4 bytes 240 (240.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 4 bytes 240 (240.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.9 netmask 255.255.255.0 broadcast 192.168.0.255
        inet6 fe80::96e4:fe98:4f6f:cba8 prefixlen 64 scopeid 0x20<link>
            ether b8:27:eb:1a:39:ff txqueuelen 1000 (Ethernet)
            RX packets 39516 bytes 50238915 (47.9 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 18027 bytes 1937981 (1.8 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
pi@raspberrypi:~ $
```

## 1. 네트워크 인터페이스

위의 ifconfig 실행 결과에 굵게 표시된 eth0, lo, wlan0 등이 네트워크 인터페이스이다.

### eth0

첫 번째 유선 네트워크 카드( Ethernet Card )를 나타낸다. 만일 이더넷 어댑터를 추가 장착한다면, eth1 이 추가된 이더넷 어댑터의 인터페이스 이름이 된다.

### lo

localhost( 127.0.0.1 )의 인터페이스 이름이다. 로컬호스트( localhost )는 네트워크에서 시스템 자신을 가리키는 논리적인 인터페이스를 의미한다.

### wlan0

첫 번째 무선 네트워크 카드( Wireless Network Card )를 나타낸다. 만일 무선 어댑터를 추가 장착한다면, wlan1 이 추가된 무선 네트워크 인터페이스 이름이 된다.

## 2. 네트워크 용어

**ip**( internet protocol ) **address**

인터넷에서 연결된 컴퓨터의 식별을 위해 할당하는 주소로서 현재는 IPv4( 32bit )와 IPv6( 128bit ) 주소체계가 혼용되는 과도기이다.

#### **network mask or subnet mask**

네트워크의 규모( 범위 )를 한정하는 네트워크 파라메터.

어떤 호스트의 IP주소가 192.168.123.45 이고, 네트워크 마스크가 255.255.255.0 이라면 이 호스트가 속한 네트워크의 규모( 범위 )는 192.168.123.1~254 이다.

네트워크 마스크가 255.255.0.0 일 경우에는 192.168.0.1~192.168.254.254 이다.

#### **gateway**

임의의 네트워크에서 다른 네트워크로 연결하기 위해 거쳐야만 하는 네트워크 단말장치.

#### **DNS( domain name service or domain name server )**

인터넷 상의 컴퓨터에 접근하려면 IP 주소가 필요하다. 하지만 구글 검색을 하기 위해 웹 브라우저에 구글의 IP 주소를 직접 입력하는 일은 없다. 보통 "http://google.com/"이라고 입력할 것이다. 이는 네트워크 상에서 "google.com"이라는 도메인을 해당 IP 주소로 변환을 해주었기 때문에 가능한 것이다. 이처럼 도메인을 IP 주소로, IP 주소를 도메인으로 변환 해주는 서비스를 DNS( domain name service )라고 하고, 네트워크( 인터넷 )에서 이 DNS 서비스를 제공하는 컴퓨터를 DNS( domain name server )라고 한다.

### **3.2.2. 네트워크 관련 설정파일**

리눅스의 네트워크 관련 설정이 저장되어 있는 파일들에 대해 알아보자.

#### **/etc/dhcpcd.conf**

기본적인 네트워크설정 정보가 기록되어 있다.

#### **/etc/network/interfaces**

GUI 가 포함되지 않은 raspbian-xxxx-lite 버전의 기본적인 네트워크설정 정보가 기록되어 있다.

#### **/etc/wpa\_supplicant/wpa\_supplicant.conf**

Wi-Fi 연결 시 사용할 AP 의 SSID 와 패스워드 정보가 기록되어 있다.

#### **/etc/resolv.conf**

'도메인이름 - IP 주소' 변환을 해 줄 Domain Name Server( DNS ) 주소가 기록되어 있다.

#### **/etc/hostname**

호스트 네임이 기록되어있다.

#### **/etc/hosts**

호스트 네임과 localhost 관련 정보가 기록되어있다.

# 4. 원격접속(Remote Access) 환경 설정

## 4.1. SSH(Secure Shell) 원격접속

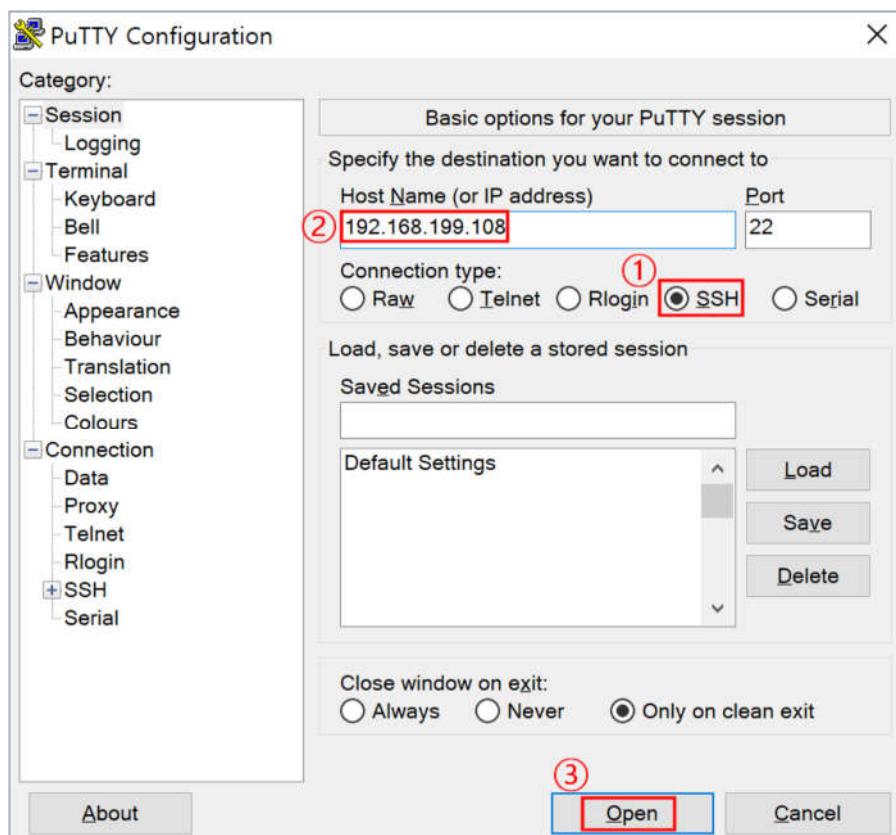
대표적인 원격접속 프로토콜인 Telnet( Tele-Networking )의 보안문제를 보완한 프로토콜이 SSH 프로토콜이다. 라즈베리파이에는 기본적으로 SSH 서버 데몬( Server Demon )이 설치되어 있다.

SSH를 이용한 원격접속을 사용하려면, "raspi-config" 의 "5. interfacing option"에서 "P2. SSH"를 활성화 시킨다. 이는 SSH Server Demon 프로그램을 부팅 시 실행한다는 것이다. 따라서 SSH Client 프로그램을 통해 SSH 원격 접속이 가능하다. 물론 이 때 **Server( 라즈베리파이 )와 Client( PC, 스마트폰, Tablet 등 )**는 같은 네트워크에 연결되어 있어야 하며, 클라이언트 기기에서 SSH 클라이언트 SW를 통해 접속한다. 리눅스나 유닉스의 경우, 터미널에서 아래와 같이 ssh 명령으로 접속 가능하다.

```
pi@raspberryPi:~ $ ssh pi@192.168.0.108
The authenticity of host '192.168.0.108 (192.168.0.108)' can't be established.
ECDSA key fingerprint is SHA256:kji7ycqS0qlJy85Px7/Bsw6L+07DynHlrs0r8dIDDd0.
Are you sure you want to continue connecting (yes/no)? yes
```

MS-Windows 계열에서 사용 가능한 대표적인 SSH Client 프로그램으로는 'Putty'가 있다.

Download Page : <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>



### ① Connection type

- SSH 선택

### ② Host Name

- 라즈베리파이의 IP 주소 입력

### ③ Open

\* Saved Sessions 메뉴를 이용하여 현재 SSH 접속환경을 저장해두고 사용 할 수 있다.

\* Category 의 Window 항목 하위 Appearance 설정에서 화면 폰트 관련 설정을 변경할 수 있다.

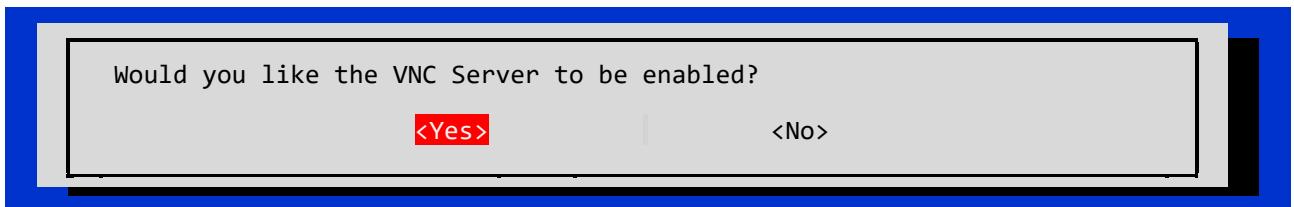
\* Saved Sessions 메뉴를 이용하여 현재의 접속환경을 저장해두고 사용할 수 있다.

## 4.2. GUI 원격접속 by VNC(Virtual Network Computing)

Raspbian-Stretch 는 Real VNC 서버가 지원된다. 따라서 다른 기기에서 Real VNC 클라이언트 프로그램을 이용한 GUI 원격 접속을 할 수 있다.

### \* 라즈베리파이 설정

VNC 를 이용한 GUI 원격제어를 사용하려면 라즈베리파이의 설정을 변경해 주어야 할 것이 있다. 기 위한 라즈베리파이 설정은 raspi-config 메뉴의 5. Interfacing options 의 P3. VNC 항목에서 VNC Server 활성화 여부 질문에서 <Yes>를 선택하는 것이다.



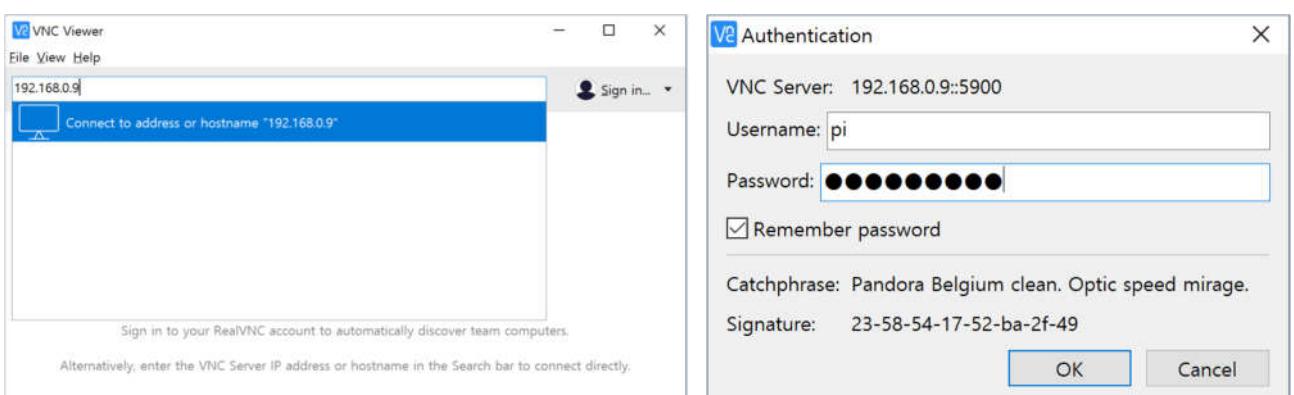
### \* 클라이언트 설정

1. 웹브라우저에서 <https://www.realvnc.com/en/connect/download/vnc/> 페이지를 연다.

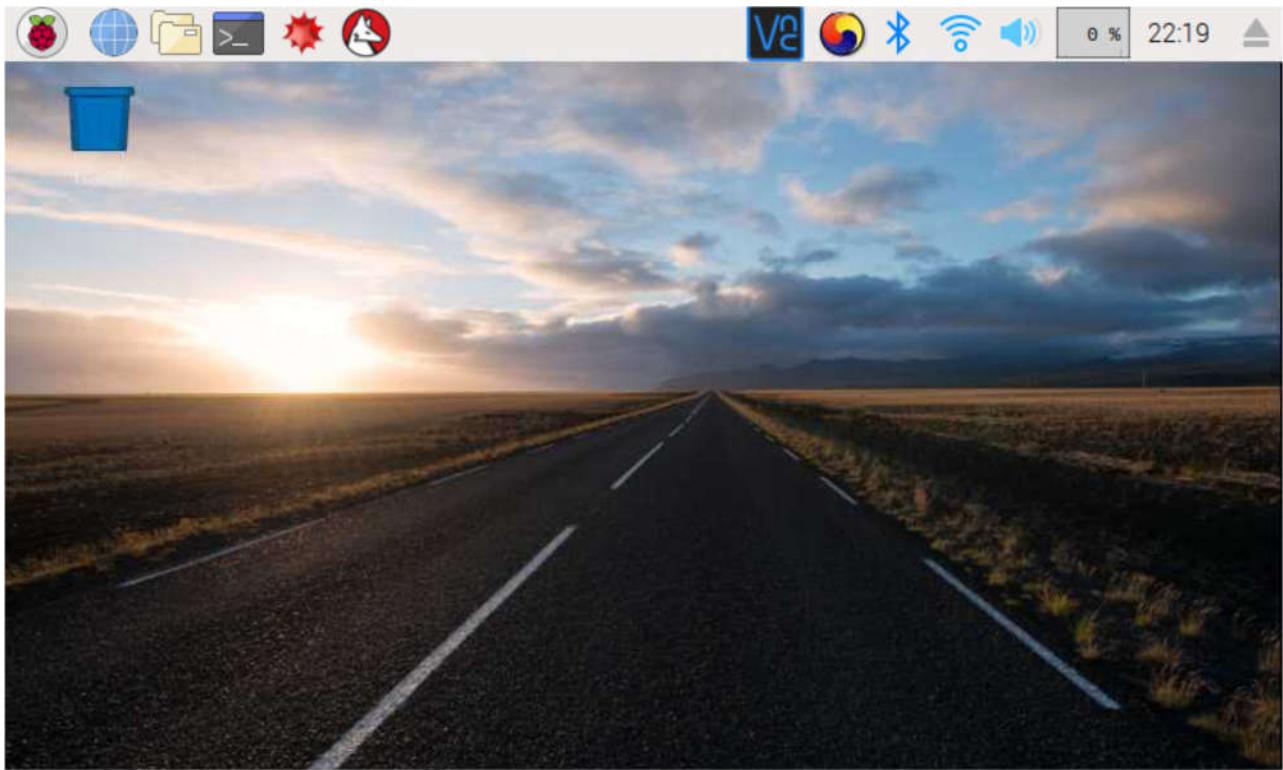


2. 사용 하는 운영체제를 선택 후, "DOWNLOAD VNC VIEWER" 클릭하여 "VNC VIEWER" 다운로드.

3. 다운로드한 VNC VIEWR 를 실행



4. 원격 연결할 라즈베리파이의 IP 주소 입력 후 Enter.
5. 팝업창의 Username에 "pi", Password에 "pi"의 패스워드 입력 후, "OK" 버튼 클릭.

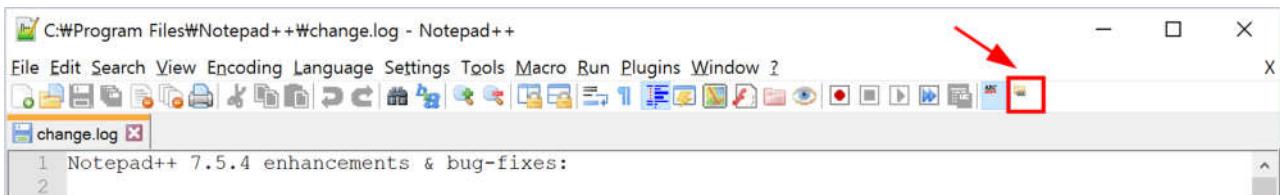


### 4.3. “Notepad++”를 이용한 원격 개발환경

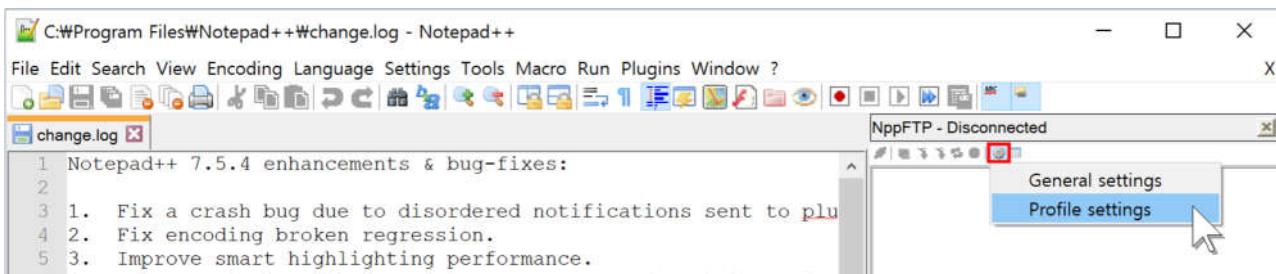
“Notepad++” 에디터는 sftp (SSH Transfer Protocol)를 지원하는 MS 윈도우용 free source code editor이다.

1. 다운로드( <https://notepad-plus-plus.org/repository/6.x/6.7.5/npp.6.7.5.Installer.exe> ) 및 설치, 실행  
( 7.x 이 후 버전은 NppFTP Plugin을 사용자가 따로 추가하고, 설정해 주어야 한다. )
2. NppFTP Plugin을 이용한 개발환경 설정

2-1 “Show NppFTP Window” 아이콘 클릭.

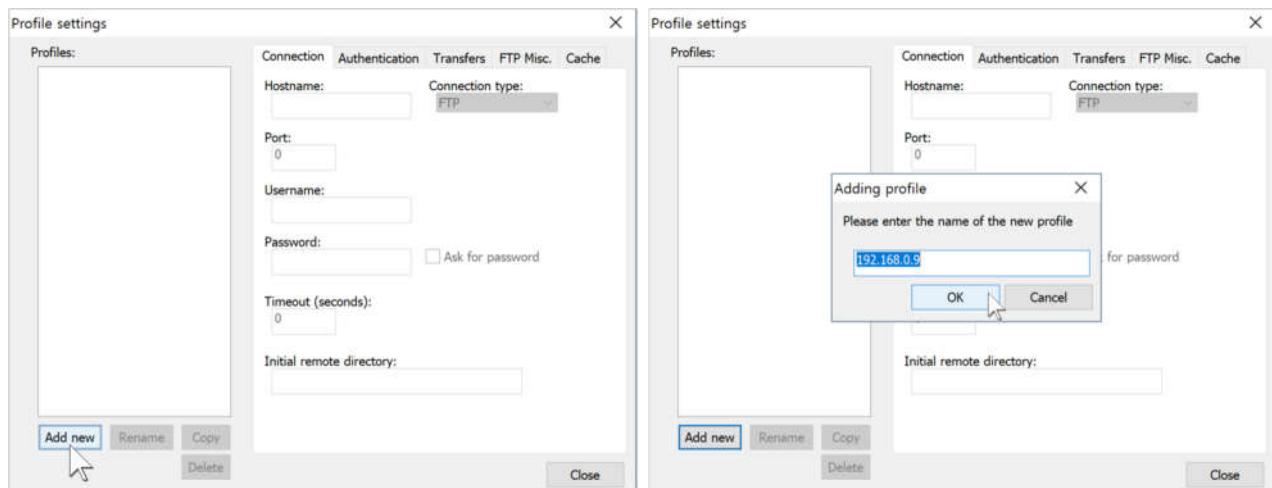


2-2 NppFTP Window의 “Setting” 아이콘(톱니바퀴 모양의 아이콘) 클릭 후, “Profile settings” 선택.



2-3 "Profile settings" 창에서 "Add new" 버튼 클릭.

2-4 "Adding profile" 팝업에서 이 연결의 이름 입력 후 "OK" 버튼 클릭.



2-5 아래의 Profile 세부 항목 작성 후 "Close" 버튼을 클릭한다.

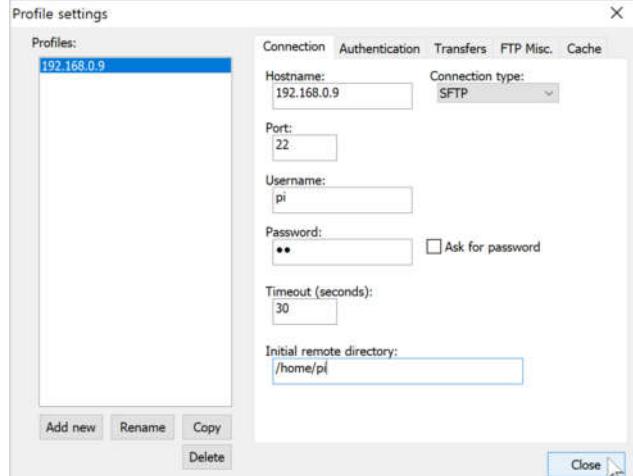
**Hostname:** 라즈베리파이의 IP 주소

**Connection type:** SFTP

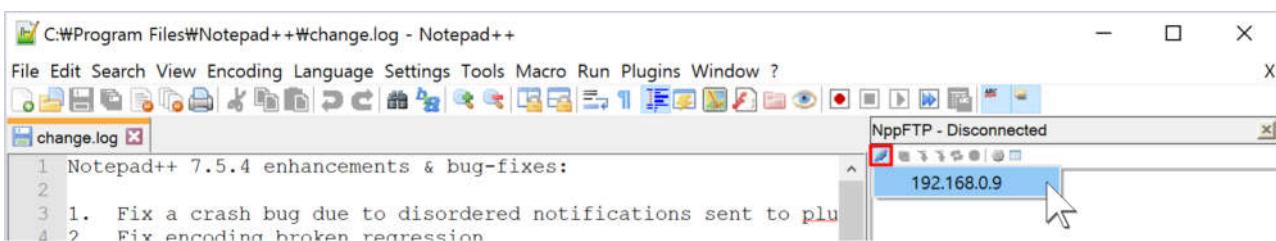
**Username:** pi

**Password:** 라즈베리파이 로그인 패스워드

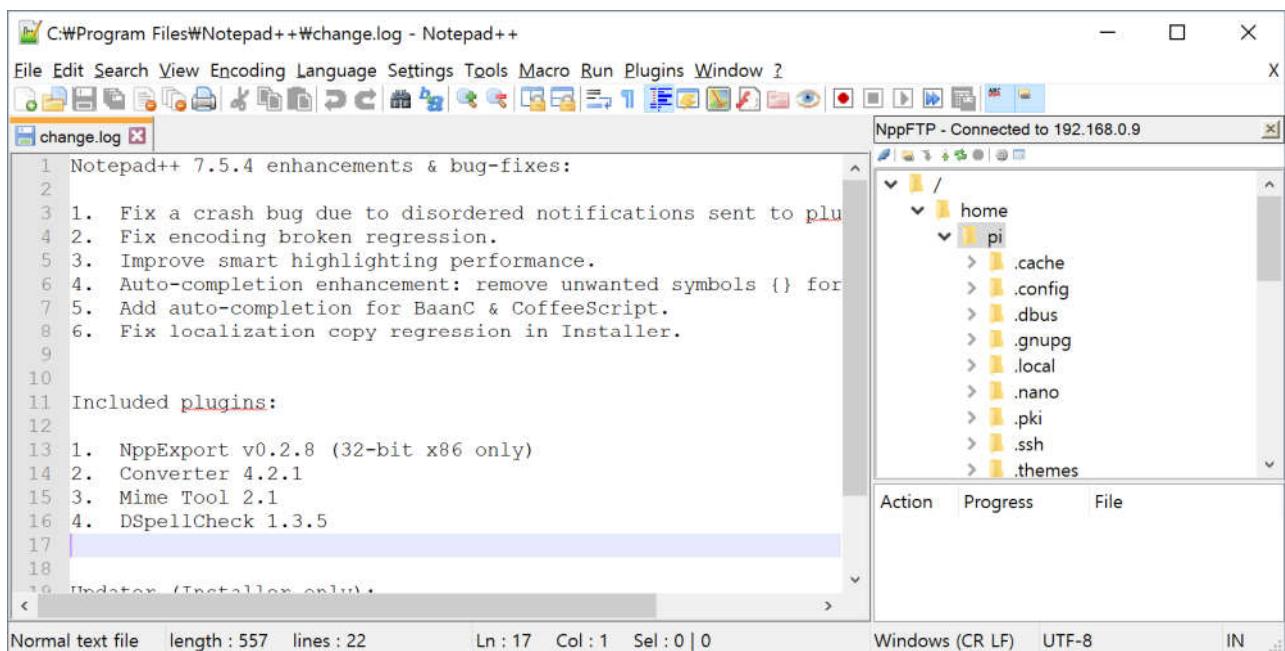
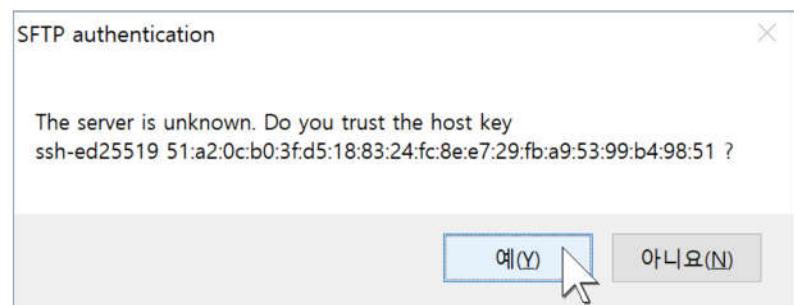
**Initial remote directory:** /home/pi



2-6 NppFTP Window 의 "(Dis)Connect" 아이콘 클릭 후, 추가한 profile 선택.



2-7 "SFTP Authentication" 팝업에서 "예(Y)" 버튼 클릭.

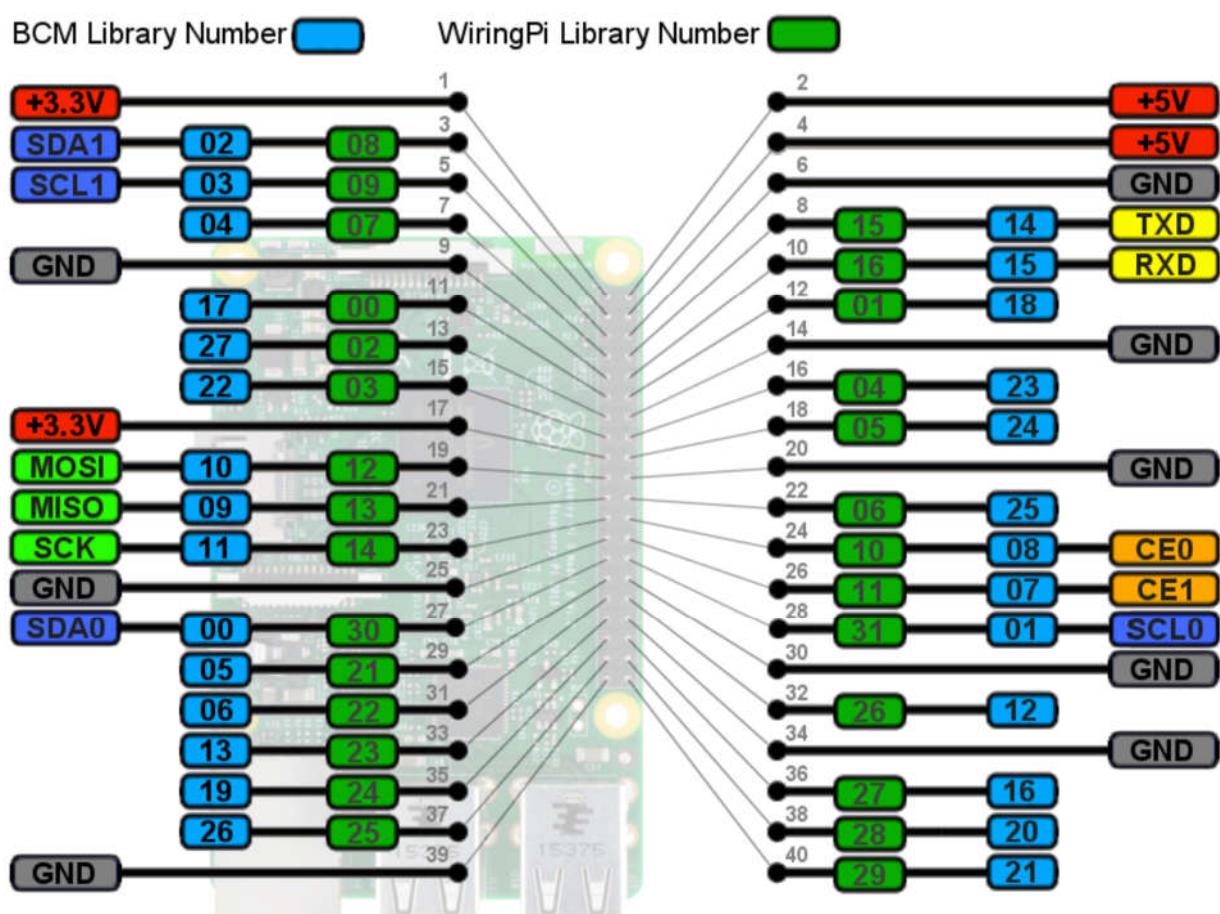


# 5. GPIO(General Purpose I/O) 포트 제어

GPIO( General Purpose Input Output ) Port 라는 용어에서 일단 Port 는 어떤 신호가 드나드는 지점을 말한다. 1 또는 0 에 해당하는 논리 레벨의 전기신호( Logical Level Signal )가 출력되거나 연결된 장치( 스위치 등 )로부터 입력되는 경우 입, 출력 포트, 통신을 하기 위한 신호가 드나들면 통신포트라고 한다. GPIO 포트는 입력이나, 출력, 또는 다른 부가기능( Alternative Function I/O )으로 사용될 수 있는 포트로서, 사용자가 작성한 프로그램으로 제어할 수 있는 포트를 말한다.

## 5.1. 라즈베리파이의 GPIO 포트

다음 그림은 라즈베리파이 40 핀 헤더에 할당된 GPIO 및 AFIO( Alternative Function I/O )기능이다.



- 에 표시된 번호는 WiringPi 라이브러리에서 사용하는 GPIO 핀 번호( WiringPi 라이브러리에서만 사용 )
- 에 표시된 번호는 Python, Nodejs 등의 WiringPi 라이브러리를 제외한 대부분의 라이브러리에서 사용된다.

위 그림을 보면 라즈베리파이의 GPIO 40 핀 헤더에서 전원에 6 개( +5V: 2, +3.3V: 2, GND: 2 )의 핀이 할당되어 남은 34 핀을 GPIO로 활용할 수 있지만, 남은 34 핀 중 UART( Universial Asynchronous Reciever & Transciever )의 TXD, RXD 2 개, I<sup>2</sup>C<sub>0</sub>( Inter Inter Connection 0 )의 SCL<sub>0</sub>, SDA<sub>0</sub> 2 개, I<sup>2</sup>C<sub>1</sub>( Inter Inter

Connection 1 )의 SCL<sub>1</sub>, SDA<sub>1</sub> 2 개, SPI( Serial Peripheral Interface )의 MOSI, MISO, SCLK, CE<sub>0</sub>, CE<sub>1</sub> 5 개 모두 11 개의 핀은 AFIO 로 전환 사용될 수 있다.

## 5.2. GPIO 제어 프로그래밍

매우 다양한 프로그래밍 언어가 라즈베리파이의 GPIO 제어 프로그래밍을 위한 라이브러리를 제공한다. Python 의 경우, 라즈베리파이에서 제공하는 GPIO 라이브러리와 카메라( 라즈베리파이 전용 카메라 ) 라이브러리가 가장 널리 사용되고, C 언어의 경우, 아두이노( Arduino )에서 사용하는 API 와 같은 사용법을 제공하는 wiringPi( <https://github.com/WiringPi/WiringPi> ) 라이브러리가 주로 사용된다.

그 외에도 pigpio( <http://abyz.me.uk/rpi/pigpio/> ), BCM2835( <https://www.airspayce.com/mikem/bcm2835/> ), 그리고 Python, PHP, Ruby, Perl, Node.js 를 위한 wiringPi( <https://github.com/WiringPi/> ) 라이브러리 등이 있다.

'wiringPi' 라이브러리 설치( 2018-04-18 이 후, raspbian-with-desktop 사용할 경우 이미 설치되어 있음 )

raspbian-wheezy-2018-04-18 이 후, 'raspbian with desktop' 배포판에는 wiringPi 가 설치된 시스템 디스크 이미지가 제공되므로 설치할 필요가 없지만, 그 이전 배포판이나, 'raspbian lite' 버전을 사용할 경우에는 다음과 같이 설치한다.

1. 'wiringPi' 라이브러리 소스 코드를 'git' 저장소에서 라즈베리파이로 복사.

```
pi@raspberrypi:~ $ git clone git://git.drogon.net/wiringPi
```

2. 'git clone' 결과 확인 후, '~/wiringPi'로 경로 변경

```
pi@raspberrypi:~ $ ls
Downloads Pictures Scratch wiringPi Public Desktop
Documents python_games
pi@raspberrypi:~ cd wiringPi
pi@raspberrypi:~/wiringPi $
```

3. 빌드

```
pi@raspberrypi:~ /wiringpi $ ./build
```

4. 확인 ( gpio -v, gpio read all )

```
pi@raspberrypi:~ /wiringpi $ cd
pi@raspberrypi:~ $ gpio -v
gpio version: 2.44
Copyright (c) 2012-2017 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty
```

Raspberry Pi Details:

```
Type: Pi 3, Revision: 02, Memory: 1024MB, Maker: Embest
```

```
* Device tree is enabled.
--> Raspberry Pi 3 Model B Rev 1.2
* This Raspberry Pi supports user-level GPIO access.
pi@raspberrypi:~ $
```

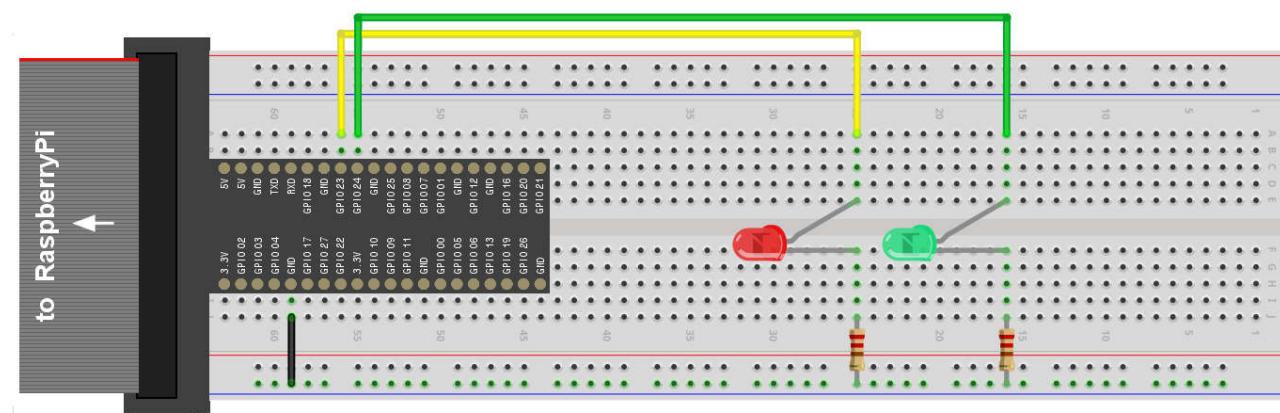
```
pi@raspberrypi:~ $ gpio readall
```

Pi 3											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
		3.3v			1	2					
2	8	SDA.1	IN	1	3	4					
3	9	SCL.1	IN	1	5	6					
4	7	GPIO. 7	IN	1	7	8	0	IN	TxD	15	14
		0v			9	10	1	IN	RxD	16	15
17	0	GPIO. 0	IN	0	11	12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	0	13	14			0v		
22	3	GPIO. 3	IN	0	15	16	0	IN	GPIO. 4	4	23
		3.3v			17	18	0	IN	GPIO. 5	5	24
10	12	MOSI	IN	0	19	20			0v		
9	13	MISO	IN	0	21	22	0	IN	GPIO. 6	6	25
11	14	SCLK	IN	0	23	24	1	IN	CE0	10	8
		0v			25	26	1	IN	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	1	31	32	0	IN	GPIO.26	26	12
13	23	GPIO.23	IN	0	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

```
pi@raspberrypi:~ $
```

## 5.2.1. GPIO 출력

GPIO 출력 코드를 동작 시켜볼 테스트 회로를 다음 그림과 같이 구성한다.



( GPIO23 - LED(적) 1 - 저항(220Ω) - GND, GPIO23 - LED(녹) 1 - 저항(220Ω) - GND )

## 1. Python

GPIO 출력으로 적색과 녹색 LED 를 점멸 시키는 파이썬 코드(gpioex01.py)를 작성하여 앞서 꾸민 회로를 동작시켜보자.

01	import RPi.GPIO as GPIO	# RPi.GPIO를 import하고 네임스페이스는 GPIO를 사용
02	import time	# time에 정의된 기능을 사용( time.sleep )
03		
04	GPIO.setmode(GPIO.BCM)	# GPIO 이름은 BCM 명칭 사용
05	GPIO.setup(23, GPIO.OUT)	# GPIO 23 출력으로 설정
06	GPIO.setup(24, GPIO.OUT)	# GPIO 24 출력으로 설정
07	print "GPIO Test~, press Ctrl+C to quit"	# ""안의 문자열 화면 출력
08		
09	try:	# 'try:' + 'except..:' 에러처리
10	while True:	# 무한 반복 'while'문( C언어 - 'while(1)' )
11	GPIO.output(23, True)	# GPIO 23에 HIGH 출력( 적색 LED 점등 )
12	GPIO.output(24, True)	# GPIO 24에 HIGH 출력( 녹색 LED 점등 )
13	time.sleep(0.5)	# 0.5초 동안 대기
14		
15	GPIO.output(23, False)	# GPIO 23에 LOW 출력( 적색 LED 소등 )
16	GPIO.output(24, False)	# GPIO 24에 LOW 출력( 녹색 LED 소등 )
17	time.sleep(0.5)	# 0.5초 동안 대기 ( 여기까지 반복 )
18		
19	except KeyboardInterrupt:	# Ctrl-C 입력 발생 시
20	GPIO.cleanup()	# GPIO 관련설정 Clear
21	print "bye~"	# 프로그램 종료 메세지 화면 출력

작성한 코드를 'gpioex1.py'로 저장 후, 다음 명령으로 코드를 실행한다.

```
pi@raspberrypi:~ $ python gpioex1.py
```

브래드 보드에 구성한 회로의 동작(0.5초간 모든 LED 점등, 0.5초간 모든 LED 소등을 무한 반복 )을 확인한다.

## 2. C 언어( WiringPi )

GPIO 출력으로 적색과 녹색 LED 를 점멸 시키는 C 코드( gpioex01.c )를 작성하여 앞서 꾸민 회로를 동작시켜보자

01	#include <stdio.h>	// stdio.h 파일 포함
02	#include <wiringPi.h>	// wiringPi.h 파일 포함
03		
04	#define LED1 4	// 4 번핀(GPIO 23) 대신 LED1 사용을 위한 정의
05	#define LED2 5	// 5 번핀(GPIO 24) 대신 LED2 사용을 위한 정의
06		
07	int main(void)	// main 함수
08	{	// { main() 시작
09	printf("Control GPIO by wiringPi\n");	// 메시지 화면 출력

```

10 wiringPiSetup(); // wiringPi 라이브러리 설정
11
12 pinMode (LED1, OUTPUT); // 4 번핀(GPIO 23) 출력 설정
13 pinMode (LED2, OUTPUT); // 5 번핀(GPIO 24) 출력 설정
14
15 while(1) // 무한 반복 구간
16 {
17     digitalWrite(LED1, HIGH); // 4 번핀(GPIO 23) HIGH 출력( 적색 LED 점등 )
18     digitalWrite(LED2, HIGH); // 5 번핀(GPIO 24) HIGH 출력( 녹색 LED 점등 )
19     delay(500); // 500ms (0.5 초) 동안 대기
20     digitalWrite(LED1, LOW ); // 4 번핀(GPIO 23) LOW 출력( 적색 LED 소등 )
21     digitalWrite(LED2, LOW ); // 5 번핀(GPIO 24) LOW 출력( 녹색 LED 소등 )
22     delay(500); // 500ms (0.5 초) 동안 대기
23 }
24 return 0; // 0 리턴(int main()에 대한 형식적인 값 반환)
25 }

```

작성한 코드를 'gpioex1.c'로 저장 후, 다음 명령으로 컴파일한다.

```
pi@raspberrypi:~ $ gcc -o gpioex1 gpioex1.c -lwiringPi
```

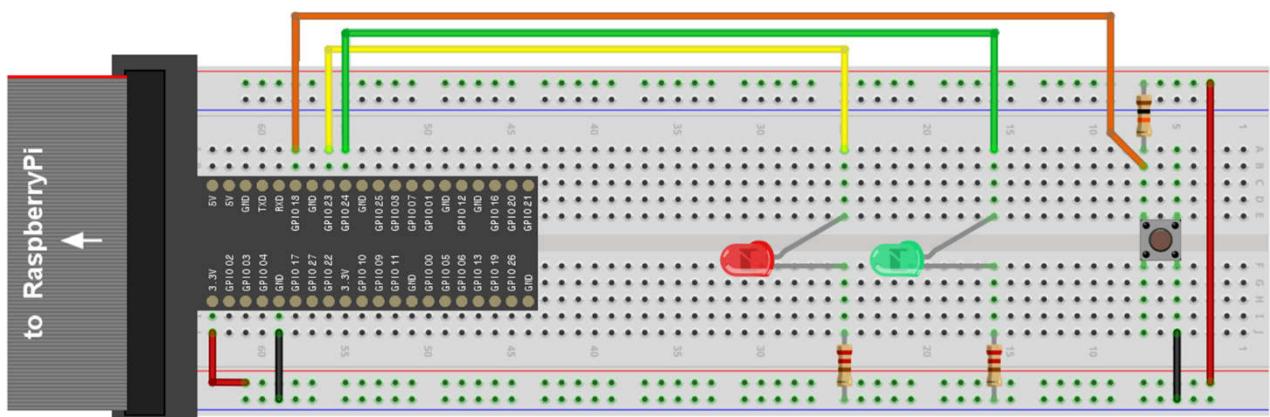
gcc : GNU C Compiler 실행  
 -o : object 파일을 생성하라는 컴파일 옵션  
 gpioex : 컴파일 결과로 생성될 실행 파일명 지정  
 gpioex1.c : 컴파일할 C 소스 파일명  
 -l : 추가로 사용될 라이브러리(표준 라이브러리 외에) 옵션  
 wiringPi : 추가 사용 라이브러리명

생성된 실행파일을 구동하여 브래드 보드에 구성한 회로의 동작( 0.5 초간 모든 LED 점등, 0.5 초간 모든 LED 소등을 무한 반복 )을 확인한다.

```
pi@raspberrypi:~ $ ./gpioex1
```

## 5.2.2. GPIO 입력

GPIO 출력 코드를 동작 시켜볼 테스트 회로를 다음 그림과 같이 구성한다.



01	<code>import RPi.GPIO as GPIO</code>	# RPi.GPIO 를 import 하고, 네임스페이스 RPi.GPIO 대신 GPIO 를 사용
02		# GPIO 이름은 BCM 명칭 사용
03	<code>GPIO.setmode(GPIO.BCM)</code>	# GPIO 23 출력으로 설정
04	<code>GPIO.setup(23, GPIO.OUT)</code>	# GPIO 24 출력으로 설정
05	<code>GPIO.setup(24, GPIO.OUT)</code>	# GPIO 18 입력으로 설정
06	<code>GPIO.setup(18, GPIO.IN)</code>	
07		
08	<code>print "Press SW or input Ctrl+C to quit"</code>	# 메세지 화면 출력
09		# try:행과 아래 except KeyboardInterrupt:
10	<code>try:</code>	# 이하는 생략가능( GPIO warnning 방지 )
11	<code>while True:</code>	# 무한 반복문 - C 언어의 while(1)에 해당
12	<code>GPIO.output(23, False)</code>	# GPIO 23 에 LOW 출력( 적색 LED 소등 )
13	<code>GPIO.output(24, False)</code>	# GPIO 24 에 LOW 출력( 녹색 LED 소등 )
14		
15	<code>while GPIO.input(18) == 0:</code>	# SW 가 ON 인 동안 반복
16	<code>GPIO.output(23, True)</code>	# GPIO 23 에 HIGH 출력( 적색 LED 점등 )
17	<code>GPIO.output(24, True)</code>	# GPIO 24 에 HIGH 출력( 녹색 LED 점등 )
18		
19	<code>except KeyboardInterrupt:</code>	# Ctrl-C 입력 시
20	<code>GPIO.cleanup()</code>	# GPIO 관련설정 Clear
21	<code>print "bye~"</code>	# 프로그램 종료 메세지 화면 출력

작성한 코드를 'gpioex2.py'로 저장 후, 다음 명령으로 코드를 실행한다.

```
pi@raspberrypi:~ $ python gpioex2.py
```

브래드 보드에 구성한 회로의 동작( 스위치를 누르면 점등, 떼면 소등 )을 확인한다.

## 2. C 언어( WiringPi )

풀업( Pull-up )된 스위치 신호를 GPIO 18 로 입력받아, 스위치가 눌려 있는 동안만 LED 가 켜지는 코드( gpioex2.c )를 작성하고, 구성한 회로를 동작시켜 보자.

01	<code>#include &lt;stdio.h&gt;</code>	// stdio.h 파일 포함( printf() 사용을 위해 )
02	<code>#include &lt;wiringPi.h&gt;</code>	// wiringPi.h 파일 포함
03		
04	<code>#define SW 1</code>	// 1 번핀(GPIO 18) 대신 SW 사용을 위한 정의
05	<code>#define LED1 4</code>	// 4 번핀(GPIO 23) 대신 LED1 사용을 위한 정의
06	<code>#define LED2 5</code>	// 5 번핀(GPIO 24) 대신 LED2 사용을 위한 정의
07		
08	<code>int main(void)</code>	// main 함수
09	{	// {

```

10 printf("Control GPIO by wiringPi\n"); // 메시지 화면 출력
11
12 wiringPiSetup(); // wiringPi 라이브러리 설정
13 // ( pinMode(), digitalWrite() 등의 사용을 위해 )
14 pinMode(SW , INPUT ); // 1번핀(GPIO 18) 입력 설정
15 pinMode(LED1, OUTPUT); // 4번핀(GPIO 23) 출력 설정
16 pinMode(LED2, OUTPUT); // 5번핀(GPIO 24) 출력 설정
17
18 while(1) // 무한 반복 구간
19 {
20     digitalWrite(LED1, LOW ); // 4번핀(GPIO 23) LOW 출력( 적색 LED 소등 )
21     digitalWrite(LED2, LOW ); // 5번핀(GPIO 24) LOW 출력( 녹색 LED 소등 )
22     while( digitalRead(SW) == 0 ) // 스위치가 ON인 동안 반복구간
23     {
24         digitalWrite(LED1, HIGH); // 4번핀(GPIO 23) HIGH 출력( 적색 LED 점등 )
25         digitalWrite(LED2, HIGH); // 5번핀(GPIO 24) HIGH 출력( 녹색 LED 점등 )
26     }
27 }
28 return 0; // int main()의 리턴값-의미는 없지만 문법상 필요
29 }

```

작성한 코드를 'gpioex1.c'로 저장 후, 다음 명령으로 컴파일한다.

```
pi@raspberrypi:~ $ gcc -o gpioex2 gpioex2.c -lwiringPi
```

코드를 실행하고, 꾸며놓은 회로의 동작을 확인한다.

```
pi@raspberrypi:~ $ ./gpioex2
```

### 3. 스위치를 이용한 shutdown 구현( btn4halt.py )

```

01 #!/usr/bin/python
02 # line 1 is not comment. that's called 'Shebang' which specifies the interpreter location of
03 # scripts.
04 import RPi.GPIO as GPIO
05 from os import system
06
07 GPIO.setmode(GPIO.BCM) # Use the Broadcom SOC Pin numbers
08 GPIO.setup(18, GPIO.IN) # Setup the Pin input
09
10 # Do next
11 try:
12     print "### Shutdown button is enabled. ###"
13     # define function 'shutdown()' which is called back when event is occurred
14     def shutdown(channel):
15         system("sudo shutdown -h now")
16
17     # detect falling edge from GPIO 18

```

```

18     GPIO.add_event_detect(18, GPIO.FALLING, callback = shutdown, bouncetime = 100)
19
20     # Just wait for switch pressed
21     while True:
22         pass
23
24     # when K/B interrupt occurred
25 except KeyboardInterrupt:
26     print "\n### Shutdown button is disabled. ###"
27     GPIO.cleanup()

```

작성한 코드를 테스트해보자. 구동 후 스위치를 눌러 shutdown process 가 실행되는 것을 확인한다.

```

pi@raspberrypi:~ $ python btn4halt.py
### Shutdown button is enabled. ###
system is going down...
[ 2577.994766] reboot: Power down

```

이 번에는 이 코드에 실행 속성을 부여 후 실행해보자.

```

pi@raspberrypi:~ $ chmod +x btn4halt.py
pi@raspberrypi:~ $ ls *.py
btn4halt.py  gpioex1.py  gpioex2.py
pi@raspberrypi:~ $ ./btn4halt.py
### Shutdown button is enabled. ###
^C
### Shutdown button is disabled. ###
pi@raspberrypi:~ $

```

이렇게 실행속성을 주고 바로 실행하는 것이 가능하려면 코드가 **Shebang( '#!/' )**으로 시작해야 가능하다. 리눅스( 또는 유닉스 )에서 셜뱅이 있는 스크립트는 프로그램으로서 실행되며, 프로그램 로더가 스크립트의 첫 줄의 나머지 부분을 인터프리터 지시자(interpreter directive)로 구문 분석한다. 즉, 지정된 인터프리터 프로그램이 대신 실행되어 스크립트의 실행을 시도할 때 처음 사용되었던 경로를 인수로서 넘겨주게 된다.

이 때 실행이 안되고 에러가 나는 경우, 오타거나 코드의 오류가 아니라면 MS 윈도우에서 원격으로 작업한 경우이다. 이 경우 다음 같은 오류 메시지가 출력된다.

```

pi@raspberrypi:~ $ ./btn4halt.py
-bash: ./btn4halt.py: /usr/bin/python^M: bad interpreter: 그런 파일이나 디렉터리가 없습니다
pi@raspberrypi:~ $

```

MS 윈도우에서는 EOL( End of Line )로 CR( '\r' ) + LF( '\n' )를 사용하고, 리눅스( 또는 유닉스 )에서는 LF( '\n' )만 사용한다. 리눅스는 MS 윈도우 형식의 파일이 저장될 때 행의 마지막에 ^M( Ctrl + M )을 추가하여 CR + LF 가 단지 라인의 끝이라는 것을 표시한다.

이 '^M'이 '#!/usr/bin/python' 뒤에 붙어 있다보니, '#!/usr/bin/python^M' 까지가 스크립트 해석기의 이름으로 전달되어 '그런 이름의 해석기가 없다'( 'bad interpreter: ...' )와 같은 오류가 발생한 것이다. 바로잡기 위해서 편집기에서 이 제어문자가 표시되도록 설정하여 일일이 지워주는 방법도 있지만, 'sed( Stream EDitor )' 명령을 사용하여 처리하도록 하자.

```
pi@raspberrypi:~ $ sed -i "s/\r//" ./btn4halt.py
-bash: ./btn4halt.py: /usr/bin/python^M: bad interpreter: 그런 파일이나 디렉터리가 없습니다
pi@raspberrypi:~ $
```

현재 경로의 'btn4halt.py' 파일에서 '\r' 을 찾아 '/' 와 '/' 사이의 내용으로 '\r' 이 있던 그 위치에 치환하라는 명령이지만, '/' 와 '/' 사이에 아무 내용이 없으므로 결국 삭제하라는 뜻이 된다.

이 명령 수행 후, 다시 실행하면 오류없이 실행되는 것을 확인할 수 있다.

이제 이 코드를 시스템이 시작할 때 자동으로 실행되어, 언제든지 이 스위치로 시스템을 종료할 수 있도록 설정해보자. 리눅스에서 자동 실행을 등록하는 가장 간편한 방법은 '/etc/rc.local' 파일에 등록해 주는 방법이다.

```
pi@raspberrypi:~ $ sudo nano /etc/rc.local
```

```
GNU  nano 2.2.6          File: wpa_supplicant.conf

#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi
/home/pi(btn4halt.py & # add here!!!
exit 0

[Read 20 lines]
^G Get Help   ^O WriteOut   ^R Read File   ^Y Cut Text   ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is    ^V Next Page  ^U Uncut Text ^T To Spell
```

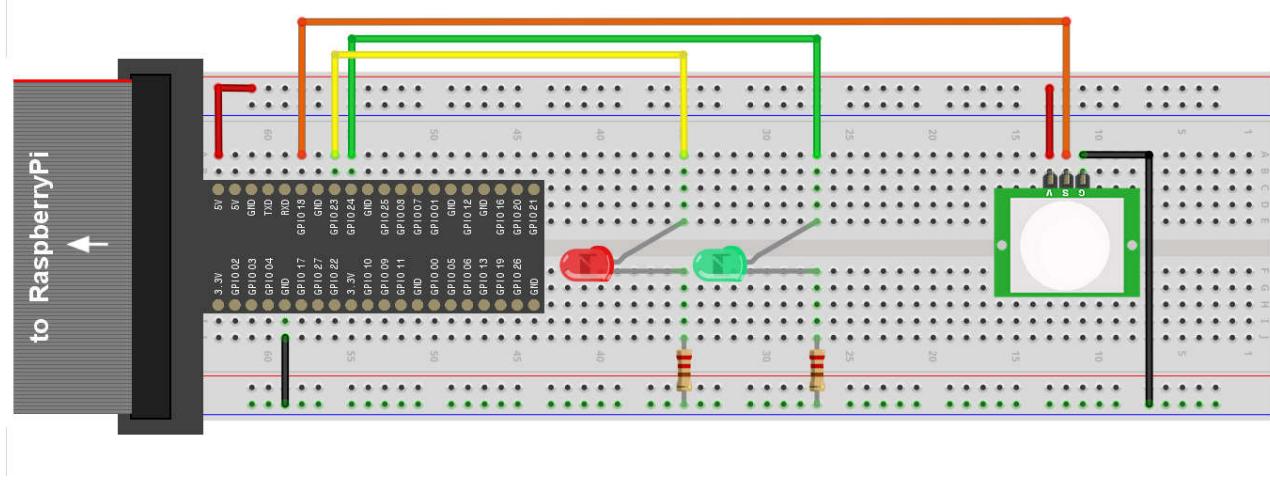
'fi'와 'exit 0' 사이에 '/home/pi(btn4halt.py &'를 삽입하고 'Ctrl+X'(종료), 'y'(저장), 'Enter'(현재 파일명으로 저장) 순으로 입력하면 변경내용이 저장된다. 추가한 행 마지막의 '&'는 자동실행 시 background에서 실행 하라는 명령으로, '&'를 빼뜨리면 부팅이 진행되지 않을 수도 있으므로(그런 상황이 발생하면 시스템 디스크를 다시 만들어야 할 수도 있다.) 주의하자.

이제 시스템을 재시작한다.

```
pi@raspberrypi:~ $ sudo reboot
```

부팅이 완료되면 스위치를 눌렀을 때 shutdown process 가 진행되는 것을 확인 한다.

### 5.2.3. 동작 검지 센서



적외선 인체 검지 센서 ( PIR 센서: Passive InfraRed Sensor )라고도 불린다. 열을 가진 물체는 열에너지를 복사( 적외선으로 방사 )한다. PIR 센서는 초전소자의 초전효과를 이용하여 주변과의 온도차를 검지한다.

위 회로를 이용하여 동작 검지에 의한 센서 출력이 유지되는 동안은 적색 LED 를, 출력이 없을 경우 녹색 LED 를 점등시키는 코드를 만들어보자.

#### 1. Python 코드

01	import RPi.GPIO as GPIO	# RPi.GPIO 를 import 하고, 네임스페이스 RPi.GPIO 대신 GPIO 를 사용
02	GPIO.setmode(GPIO.BCM)	# GPIO 이름은 BCM 명칭 사용
03		
04	GPIO.setup(23, GPIO.OUT)	# GPIO 23 출력으로 설정
05	GPIO.setup(24, GPIO.OUT)	# GPIO 24 출력으로 설정
06	GPIO.setup(18, GPIO.IN )	# GPIO 18 입력으로 설정
07		
08		
09	GPIO.output(23, False)	# GPIO 23 에 LOW 출력( 적색 LED 소등 )
10	GPIO.output(24, False)	# GPIO 24 에 LOW 출력( 녹색 LED 소등 )
11	print "PIR Sensor Test"	# 메세지 화면 출력
12		
13	try:	# try: except KeyboardInterrupt: 예러처리
14	while True:	# 무한 반복 구간 - C 언어의 while(1)에 해당
15	if GPIO.input(18) == 0:	# GPIO 18 입력(센서 출력)이 LOW 이면
16	GPIO.output(23, False)	# GPIO 23 에 LOW 출력( 적색 LED 소등 )
17	GPIO.output(24, True )	# GPIO 24 에 HIGH 출력( 녹색 LED 점등 )
18		
19	else:	# GPIO 18 입력(센서 출력)이 HIGH 이면
20	GPIO.output(23, False)	# GPIO 23 에 HIGH 출력( 적색 LED 점등 )
21	GPIO.output(24, True )	# GPIO 24 에 LOW 출력( 녹색 LED 소등 )
22		

23	<code>except KeyboardInterrupt:</code>	# Ctrl-C 입력 시
24	<code>    GPIO.cleanup()</code>	# GPIO 관련설정 Clear

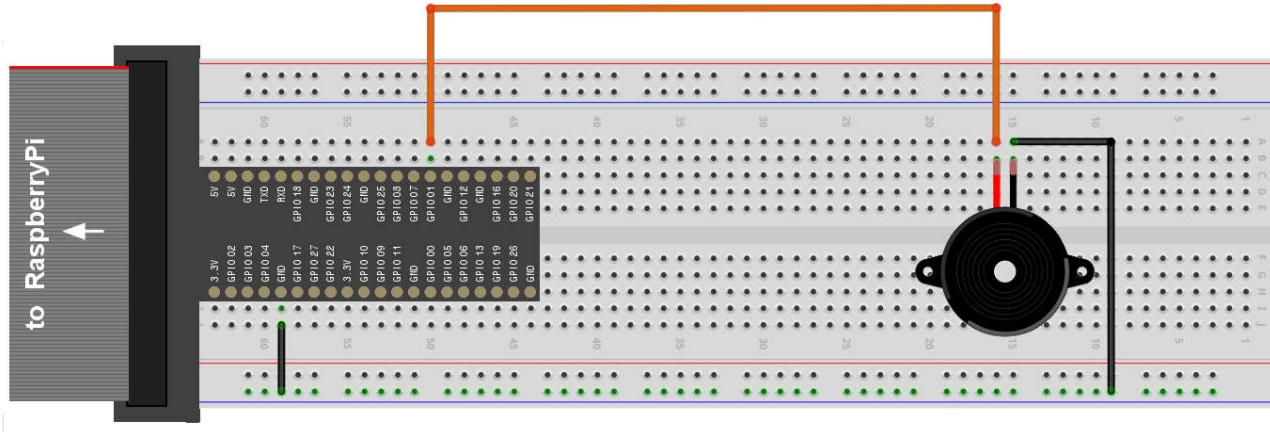
## 2. C 언어( WiringPi )

```

01 #include <stdio.h>                                // stdio.h 파일 포함( printf() 사용하기 위해 )
02 #include <wiringPi.h>                             // wiringPi.h 파일 포함
03
04 #define PIR  1                                     // 1번핀(GPIO 18) 대신 PIR 사용을 위한 정의
05 #define LED1 4                                    // 4번핀(GPIO 23) 대신 LED1 사용을 위한 정의
06 #define LED2 5                                    // 5번핀(GPIO 24) 대신 LED2 사용을 위한 정의
07
08 int main(void)                                   // main 함수
09 {
10     printf("PIR Sensor Test\n");                  // 메시지 화면 출력
11     wiringPiSetup();                            // wiringPi 라이브러리 사용설정
12
13     pinMode(PIR, INPUT );                      // 1번핀(GPIO 18) 입력 설정
14     pinMode(LED1, OUTPUT);                     // 4번핀(GPIO 23) 출력 설정
15     pinMode(LED2, OUTPUT);                     // 5번핀(GPIO 24) 출력 설정
16
17     while(1)                                    // 무한 반복 구간
18     {
19         if( digitalRead(PIN) == 0 )             // 1번핀(GPIO 18) 입력이 LOW 이면 - 센서출력이 LOW
20         {
21             digitalWrite(4, 0);                 // 4(23)번핀에 LOW 출력( 적색 LED 소등 )
22             digitalWrite(5, 1);                 // 5(24)번핀에 HIGH 출력( 녹색 LED 점등 )
23         }
24         else
25         {
26             digitalWrite(4, 1);                 // 4(23)번핀에 HIGH 출력( 적색 LED 점등 )
27             digitalWrite(5, 0);                 // 5(24)번핀에 LOW 출력( 녹색 LED 소등 )
28         }
29     }
30     return 0;                                  // int main()의 반환값
31 }
```

### 5.2.4. 버저( Buzzer )

다음 회로를 구성하고 Buzzer를 이용해 '도', '레', '미', '파', '솔', '라', '시', '도' 음계를 출력해 보자.



## 1. Python

```

01 import RPi.GPIO as GPIO          # RPi.GPIO 클래스 import & 네임스페이스 GPIO 사용
02 import time                     # time 모듈
03
04 GPIO.setmode(GPIO.BCM)          # GPIO 이름은 BCM 명칭 사용
05
06 buzz = 1                        # 핀 번호 1 대신 buzz 명칭사용을 위해 치환
07 GPIO.setup(buzz, GPIO.OUT)       # GPIO buzz 핀(1)을 출력으로 설정
08 freq = [523,587,659,698,784,880,988,1047] # freq 리스트 ( 음계 주파수 리스트 )
09
10 def makeTone(freq):            # 매개변수로 freq를 받는 makeTone 함수 정의 시작
11     scale = GPIO.PWM(buzz, freq) # buzz 핀으로 freq(Hz) PWM 파형을 생성 scale 정의
12     scale.start(10)             # scale 시작
13     time.sleep(0.5)             # 0.5 초 대기
14     scale.stop()               # scale 정지 ( makeTone 함수 정의 끝 )
15
16 try:                           # try: 21 행 except KeyboardInterrupt: 와 에러처리
17     for hz in freq:             # freq 리스트 개수 만큼 반복 시작
18         makeTone(hz)           # makeTone 함수에 freq
19     GPIO.cleanup()              # GPIO 관련설정 Clear
20
21 except KeyboardInterrupt:      # Ctrl-C 입력 시
22     GPIO.cleanup()              # GPIO 관련설정 Clear

```

## 2. C 언어( WiringPi )

```

01 #include <stdio.h>                // stdio.h 파일 포함
02 #include <wiringPi.h>              // wiringPi.h 파일 포함
03 #include <softTone.h>              // softTone.h 파일 포함
04
05 const int buzz = 31;                // 상수형 변수 buzz에 31 치환

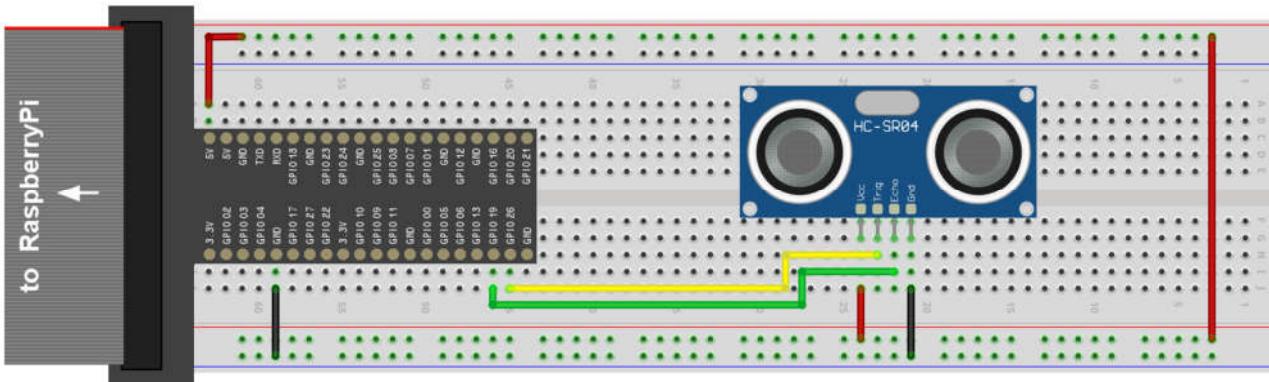
```

```

06
07 int freq[8]={523,587,659,698,784,880,988,1047}; // ( 버저가 연결된 GPIO 번호 – BCM: GPIO 1 )
08
09 void main() // 음계 주파수 배열 freq 선언 및 초기화
10 { // main 함수 시작
11     int i; // 정수형 변수 i 선언
12     wiringPiSetup(); // wiringPi 라이브러리 사용설정
13     softToneCreate(buzz); // GPIO 31을 통한 softTone 출력 설정
14     for (i = 0; i < 8; i++) // i 값을 0부터 7 까지 for 문에 의한 반복
15     {
16         printf ("%4d\n", i); // freq 리스트 만큼 반복 시작
17         softToneWrite(buzz, freq[i]); // buzz 핀으로 freq[i] 주파수 음계 출력
18         delay(200); // 200ms 딜레이
19     } // }
20 } // }

```

### 5.2.5. 초음파 센서( HC-SR04 )



초음파 거리측정 센서 HC-SR04 를 이용한 거리 측정 코드를 작성 후, 임의의 대상까지의 거리를 측정하여 보자.

이 센서는 Trig 핀으로 폭 10us 미만의 펄스를 가하면 위 센서 그림의 좌측 초음파 발신부(T)로부터 8 개의 초음파 펄스가 발사된다. 이 때 Echo 핀에서는 LOW 출력이 읽히지만, 측정대상 Object로부터 반사된 초음파가 센서 우측 초음파 수신부(R)에 수신되기 시작한 순간부터 마지막 반사파가 수신되기까지 HIGH 출력이 읽힌다. 이 Echo 핀이 HIGH 를 유지한 시간이 바로 측정대상까지 초음파가 왕복하는데 걸린 시간이다. 초음파 역시 소리이므로 속도는 음속 340(m/s)에 해당한다. 따라서 센서의 Echo 핀으로부터 HIGH 가 출력되는 순간을 검지하여, 그 출력이 LOW 로 떨어지는 순간까지 걸린 시간을 측정하는 코드를 작성하여 구하고자 하는 거리와 측정된 시간과 음속과의 관계식을 세운다.

## 1. Python ([https://github.com/greattoe/raspberrypi/tree/master/01\\_ctrlGPIO/06\\_hcsr04/picamex1.py](https://github.com/greattoe/raspberrypi/tree/master/01_ctrlGPIO/06_hcsr04/picamex1.py))

01	import RPi.GPIO as GPIO	# RPi.GPIO클래스 import & 네임스페이스 GPIO사용
02	import time	# time 클래스 import
03		
04	TRIG = 19	# TRIG 변수에 핀번호 4 치환
05	ECHO = 26	# ECHO 변수에 핀번호 17 치환
06		
07	GPIO.setmode(GPIO.BCM)	# GPIO 명칭은 BCM SOC 핀 번호 사용
08		
09	GPIO.setup(TRIG,GPIO.OUT)	# TRIG(GPIO19) 출력으로 설정
10	GPIO.setup(ECHO,GPIO.IN)	# ECHO(GPIO26) 입력으로 설정
11	GPIO.output(TRIG, False)	# TRIG(GPIO19)로 LOW 출력
12	time.sleep(1)	# 1초 동안 대기
13		
14	print ">>> Measure Distance with HC-SR04 <<<"	# 프로그램 메시지 화면 출력
15		
16	def trig():	# 팔스폭 10us 팔스 출력 함수 trig()정의
17	GPIO.output(TRIG, True)	# TRIG(GPIO19)로 HIGH 출력
18	time.sleep(0.000010)	# 10us 대기
19	GPIO.output(TRIG, False)	# TRIG(GPIO19)로 LOW 출력, trig()정의 끝
20		
21	def echo():	# Echo 입력 HIGH 유지시간 반환함수 echo() 정의
22	while GPIO.input(ECHO) is 0:	# Echo 입력이 LOW 인동안
23	echo_start = time.time()	# echo_start에 현재 시간 저장
24		
25	while GPIO.input(ECHO) is 1:	# Echo 입력이 HIGH 인동안
26	echo_end = time.time()	# echo_end에 현재 시간 저장
27		
28	echo_us = (echo_end - echo_start) * 1000000	# echo_end 값과 echo_start 값의 차를 us 초단위로
29		# echo_us에 저장
30	if(echo_us >= 240 and echo_us <= 23000):	# echo_us가 240 ~ 23000 범위에 있으면
31	return echo_us	# echo_us 값 반환
32		
33	else:	# 그렇지 않으면
34	return 0	# 0 반환, echo()정의 끝.
35		
36	try:	# try: ~ except KeyboardInterrupt: 예외처리
37	while True:	# 무한루프 시작
38	trig()	# trig()함수 호출
39	echo_time = echo()	# echo_time에 echo()함수 반환값 저장
40		
41	if echo_time is not 0:	# echo_time 0이 0이 아니면
42	dist = 17 * echo_time / 100	# 측정거리 계산하여 dist에 저장
43	dist = round(dist, 2)	# dist 값 반올림하여 소수점이하 두자리까지 표시

44	<code>print "Distance = ",dist,"(mm)"</code>	# dist 값 화면 출력
45		
46	<code>else:</code>	# echo_time이 0이면
47	<code>    print "Out of range~"</code>	# 에러메세지 화면 출력
48		
49	<code>time.sleep(1);</code>	# 1초동안 대기
50		
51	<code>except KeyboardInterrupt:</code>	# Ctrl+C 입력 예외 발생시
52	<code>    GPIO.cleanup()</code>	# GPIO.cleanup() 함수 호출

## 2. C 언어( WiringPi )

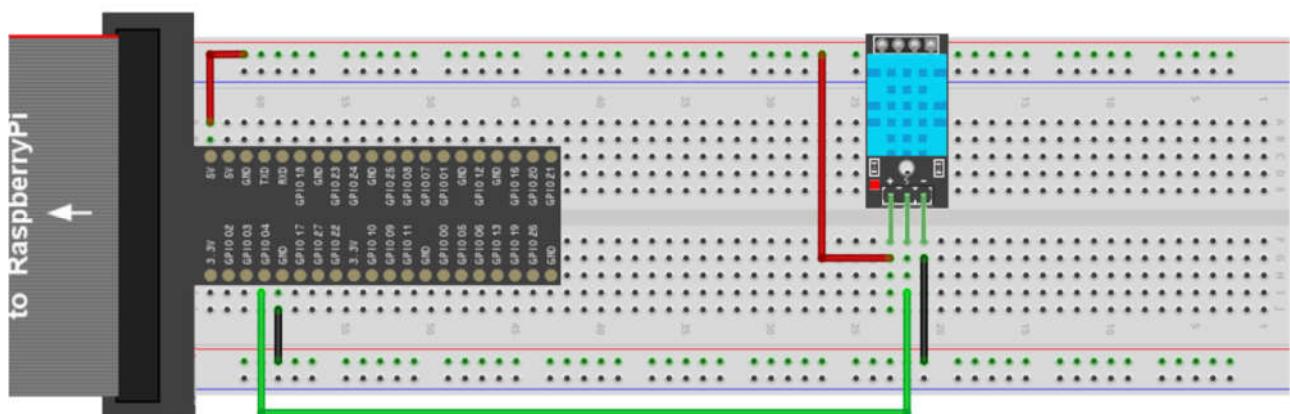
01	<code>#include &lt;stdio.h&gt;</code>	// stdio.h 파일 포함
02	<code>#include &lt;wiringPi.h&gt;</code>	// wiringPi.h 파일 포함
03		
04	<code>#define TRIG 23</code>	// Trig 신호가 연결된 GPIO23을 TRIG로 정의
05	<code>#define ECHO 24</code>	// Echo 신호가 연결된 GPIO24를 ECHO로 정의
06		
07	<code>void trig()</code>	// 10us 펄스 출력함수 trig() 정의
08	{	// {
09	<code>    digitalWrite( TRIG, HIGH );</code>	// TRIG(GPIO23)핀으로 HIGH 출력
10	<code>    delayMicroseconds( 10 );</code>	// 10us 대기
11	<code>    digitalWrite( TRIG, LOW );</code>	// TRIG(GPIO23)핀으로 HIGH 출력
12	}	// }
13		
14	<code>unsigned long echo(void)</code>	// echo()함수 정의
15	{	// (Echo 신호의 HIGH 출력 유지시간 리턴 함수)
16	<code>    unsigned long time = 0;</code>	// 부호없는 배정도 정수형 변수 time 정의
17		//
18	<code>    while( digitalRead( ECHO ) == LOW );</code>	// Echo 입력이 LOW인 동안 대기
19	<code>    time = micros();</code>	// Echo 입력이 HIGH로 변한 순간의 시간 저장
20	<code>    while( digitalRead( ECHO ) == HIGH );</code>	// Echo 입력이 HIGH인 동안 대기
21	<code>    time = micros() - time;</code>	// Echo 입력 HIGH 유지시간을 time에 저장
22		// 4.1cm ~ 3.9m
23	<code>    if( time &gt;= 240 &amp;&amp; time &lt;= 23000 )</code>	// time이 240 ~ 2300 범위 내에 있으면
24	<code>        return time;</code>	// time에 저장된 값 반환
25		
26	<code>    else</code>	// 그렇지 않으면 0 반환
27	}	
28		
29	<code>void main()</code>	// main()함수 정의
30	{	
31	<code>    unsigned long echo_time = 0;</code>	// 변수 echo_time 정의
32	<code>    unsigned long dist = 0;</code>	// 변수 dist 정의
33		

```

34 wiringPiSetup(); // wiringPi 라이브러리 사용설정
35
36 pinMode( ECHO, INPUT ); // ECHO(GPIO23) GPIO 입력으로 설정
37 pinMode( TRIG, OUTPUT ); // TRIG(GPIO24) GPIO 출력으로 설정
38 digitalWrite( TRIG, LOW ); // TRIG(GPIO24)로 LOW 출력
39
40 printf ("\nHC-SR04 Test~\n"); // 프로그램 메시지 화면출력
41
42 while(1) // 무한루프 시작
43 {
44     trig(); // trig() 함수 호출
45     echo_time = echo(); // 변수 echo_time에 echo()함수 반환값 저장
46
47     if( echo_time != 0 ) // echo_time이 0이 아니면
48     {
49         // 340000mm / 1000000us = 2 * dist / echo_time // 음속 = 340(m/sec) = 340000(mm)/1000,000(us)
50         // 34000 * echo_time = 2 * dist * 1000000
51         // dist = 34000 * echo_time / 2000000
52         // dist = 17 * echo_time / 100
53         dist = 17 * echo_time / 100; // dist 변수에 mm 단위 측정거리 저장
54         printf("distance = %4d (mm)\n",dist); // 측정거리 화면출력
55     }
56     else // echo_time이 0이면
57         printf("out of range!!!\n",dist); // 에러 메시지 화면출력
58
59     delay(1000); // 1초 동안 대기
60 }
61 }

```

## 5.2.6. 온습도 센서( DHT11 )



## 1. Python

DHT11 온, 습도 센서 라이브러리 설치

```
pi@raspberrypi:~ $ git clone https://github.com/szazo/DHT11_Python
```

설치한 DHT11 라이브러리를 이용한 온, 습도 측정 python 코드( dht11.py ) 작성

01	<code>import RPi.GPIO as GPIO</code>	# RPi.GPIO import & 네임스페이스 GPIO 사용
02	<code>from time import sleep</code>	# 'time' 클래스 멤버 중 'sleep' import
03	<code>import DHT11_Python.dht11 as dht11</code>	# './DHT11_Python/dht11.py'를 import 하고, # 네임스페이스로 'dht11'을 사용.
04		# GPIO 명칭은 BCM SOC 핀 번호를 사용
05	<code>GPIO.setmode(GPIO.BCM)</code>	
06		
07	<code>print "Get temperature &amp; humidity from dht11"</code>	# 프로그램 메시지 화면출력
08	<code>dht = dht11.DHT11(pin=4)</code>	# dht11 인스턴스 생성
09		
10	<code>try:</code>	# try: ~ except...: 예외처리
11	<code>    while True:</code>	# 무한 반복 시작
12	<code>        result = dht.read()</code>	# result에 dht11 센서 측정값 저장
13		
14	<code>    if result.is_valid():</code>	# result에 저장된 측정값이 유효한 값이면
15	<code>        print("T = %-3.1f C" % result.temperature)</code>	# 측정 온도 화면출력
16	<code>        print("H = %-3.1f %%" % result.humidity)</code>	# 측정 습도 화면출력
17		
18	<code>        sleep(6)</code>	# 6 초 대기
19		
20	<code>except KeyboardInterrupt:</code>	# Ctrl+C 가 입력되면
21	<code>    GPIO.cleanup()</code>	# GPIO.cleanup() 호출

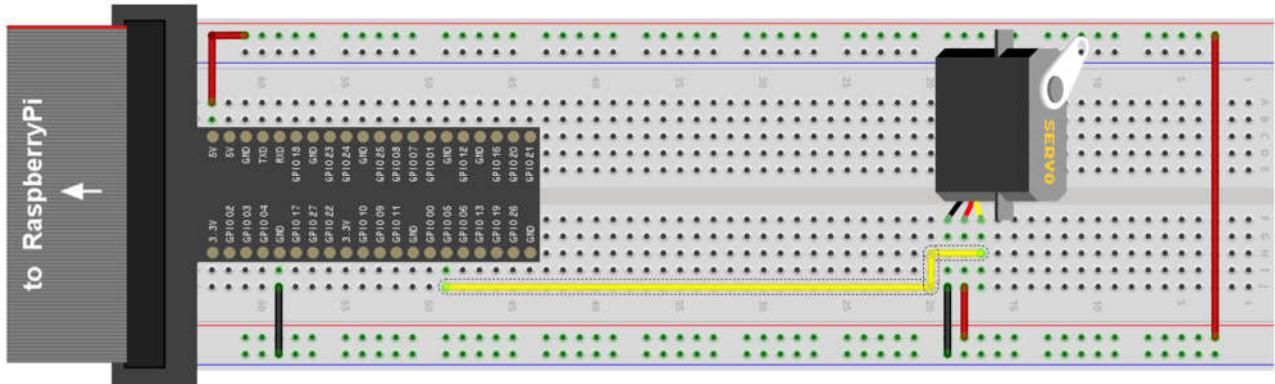
## 2. C 언어( WiringPi )

01	<code>#include &lt;stdio.h&gt;</code>	
02	<code>#include &lt;wiringPi.h&gt;</code>	
03		
04	<code>#define MAXTIMINGS 85</code>	
05	<code>#define DHTPIN 7</code>	
06		
07	<code>int dht11_dat[5] = { 0, 0, 0, 0, 0 };</code>	
08		
09	<code>void read_dht11_dat()</code>	
10	<code>{</code>	
11	<code>    unsigned char laststate = HIGH;</code>	
12	<code>    unsigned char counter = 0;</code>	
13	<code>    unsigned char j = 0, i;</code>	
14	<code>    float f;</code>	
15		
16	<code>    dht11_dat[0] = dht11_dat[1] = dht11_dat[2] = dht11_dat[3] = dht11_dat[4] = 0;</code>	
17		

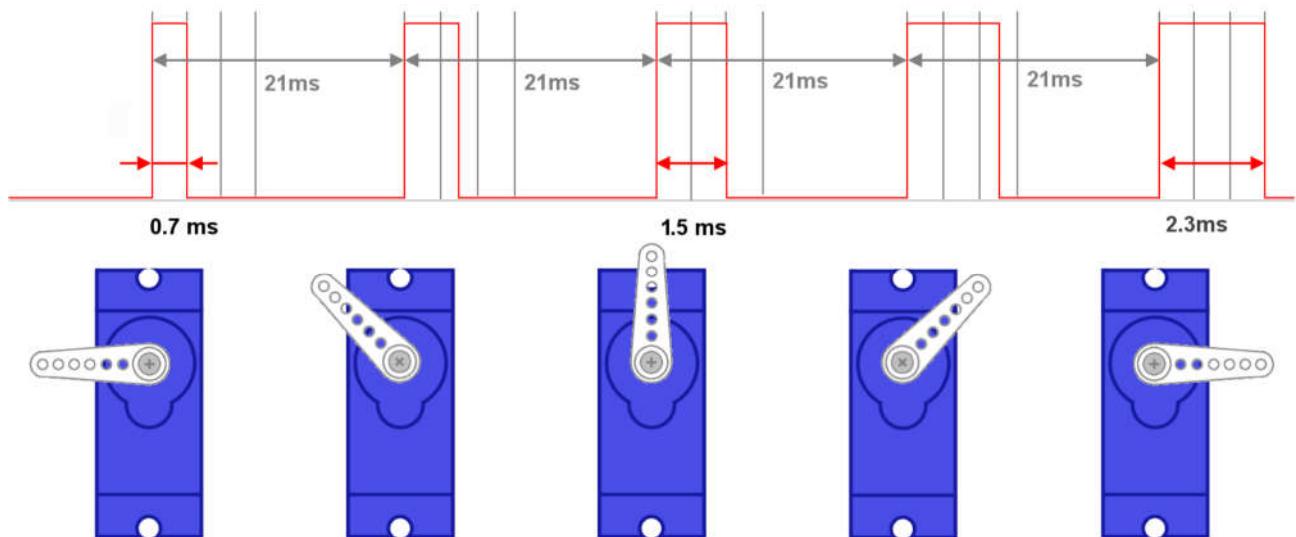
```

18 pinMode(DHTPIN, OUTPUT);
19 digitalWrite(DHTPIN, LOW);
20 delay(18);
21
22 digitalWrite(DHTPIN, HIGH);
23 delayMicroseconds(40);
24
25 pinMode(DHTPIN, INPUT);
26 for(i = 0; i < MAXTIMINGS; i++)
27 {
28     counter = 0;
29     while(digitalRead(DHTPIN) == laststate)
30     {
31         counter++;
32         delayMicroseconds(1);
33         if(counter == 255) break;
34     }
35     laststate = digitalRead(DHTPIN);
36     if(counter == 255) break;
37     if((i >= 4) && (i % 2 == 0))
38     {
39         dht11_dat[j / 8] <= 1;
40         if(counter > 50 /*16*/)
41             dht11_dat[j / 8] |= 1;
42         j++;
43     }
44 }
45 if((j >= 40) && (dht11_dat[4] == ((dht11_dat[0] + dht11_dat[1] + dht11_dat[2] + dht11_dat[3]) & 0xFF)))
46 {
47     // f = dht11_dat[2] * 9. / 5. + 32;
48     printf(" T = %.2f *C, H = %.2f %%\n", dht11_dat[2], dht11_dat[3], dht11_dat[0], dht11_dat[1]);
49 }
50 else printf("data read error, skip\n");
51 }
52
53 void main()
54 {
55     wiringPiSetup();
56
57     printf("Get Temperature & Humidity from DHT11~\n" );
58     while(1)
59     {
60         read_dht11_dat();
61         delay(2000);
62     }
63 }
```

## 5.2.7. 서보( Servo ) 모터 제어하기



서보모터는 주기가 21ms 인 PWM 파형의 진폭이 1.5ms 일 때 중앙 위치로 회전하고 여기서 0.7 ~ 0.8ms 만큼 진폭이 줄어들면 CCW(반시계-Counter Clock Wise)방향으로 -90 도, 늘어나면 CW(시계-Clock Wise)반향으로 90 도 회전한다.



### 1. Python

서보모터 테스트 python 코드(servo.py) 작성

01	<code>import RPi.GPIO as GPIO</code>	# RPi.GPIO import & 네임스페이스 GPIO 사용
02	<code>from time import sleep</code>	# 'time' 클래스 멤버 중 'sleep' import
03	<code>import sys, tty, termios</code>	# getch()에 필요한 모듈들 import
04		
05	<code>servo = 6</code>	# 서보가 연결된 GPIO 번호 servo에 저장
06	<code>pos = 7.5</code>	# 서보 시작위치 pos에 저장
07	<code>offset = 0.5</code>	# pos 증감량 offset에 저장
08	<code>MIN = 5.0</code>	# 서보 위치 최소값 설정
09	<code>MAX = 12.5</code>	# 서보 위치 최대값 설정
10		
11	<code>GPIO.setmode(GPIO.BCM)</code>	# BCM SOC 번호를 GPIO 번호로 사용
12	<code>GPIO.setup(servo, GPIO.OUT)</code>	# 서보가 연결된 GPIO 출력으로 설정

```

13
14 def getch():
15     fd = sys.stdin.fileno()
16     old_settings = termios.tcgetattr(fd)
17
18     try:
19         tty.setraw(sys.stdin.fileno())
20         ch = sys.stdin.read(1)
21
22     finally:
23         termios.tcsetattr(fd, termios.TCSADRAIN,
24                           old_settings)
25
26     return ch
27
28 pulse = GPIO.PWM(servo, 50)
29 pulse.start(pos)
30
31 print "Servo Control Test~"
32 print "Control Input: "
33 ch = getch()
34 print ch
35
36 try:
37     while ch != 'Q':
38
39         print "Control Input: "
40         ch = getch()
41         print ch
42
43         if ch == '.': # '>'
44             if pos <= MAX - offset:
45                 pos = pos + offset
46             else:
47                 pos = MAX
48
49         elif ch == ',': # '<'
50             if pos >= MIN + offset:
51                 pos = pos - offset
52             else:
53                 pos = MIN
54
55         elif ch == '1':
56             pos = 2.5
57
58         elif ch == '2':
59             pos = 5.0
60
61         pulse.ChangeDutyCycle(pos)
62
63         print "Current Position: ", pos
64
65     print "Program Terminated."
66
67 except KeyboardInterrupt:
68     print "Keyboard Interrupt Detected. Stopping Servo."
69
70 finally:
71     pulse.stop()
72     GPIO.cleanup()
73
74     print "All resources released and cleanup completed."
```

# 키보드에서 한글자 입력받는 getch() 정의  
# 표준 입력장치로부터 1바이트를 읽어 ch에 저장  
# 입력받은 1문자 반환  
# 50Hz pwm 출력 객체 pulse 정의  
# pulse 객체 기동  
# 프로그램 메시지 출력  
# 키입력 메시지 출력  
# 키입력을 변수 ch에 저장  
# 입력받은 문자 출력  
# try: except KeyboardInterrupt:와 예외처리  
# 'Q'가 입력되지 않는 동안 이하 반복 수행  
# 제어문자 입력 요청 메시지 화면 출력  
# 키 입력 한 글자를 ch에 치환  
# ch 화면출력  
# 입력 문자가 '.'이면  
# MIN에 offset을 빼준 것보다 pos가 작으면  
# pos를 offset 만큼 증가 시키고  
# 그렇지 않으면  
# pos는 MAX 값을 유지한다.  
# 입력 문자가 ','이면  
# MIN에 offset을 더해준 것보다 pos가 크면  
# pos를 offset 만큼 감소 시키고  
# 그렇지 않으면  
# pos는 MIN 값을 유지한다.  
# 입력 문자가 '1'이면  
# pos 값에 2.5 치환  
# 입력 문자가 '2'이면  
# pos 값에 5.0 치환

60	<code>elif ch == '3':</code>	# 입력 문자가 '3'이면
61	<code>    pos = 7.5</code>	# pos 에 7.5 치환
62		
63	<code>elif ch == '4':</code>	# 입력 문자가 '4'이면
64	<code>    pos = 10.0</code>	# pos 에 10.0 치환
65		
66	<code>elif ch == '5':</code>	# 입력 문자가 '5'이면
67	<code>    pos = 12.5</code>	# pos 에 12.5 치환
68		
69	<code>else:</code>	# 앞의 어떤 경우에도 해당되지 않을 경우
70	<code>    pass</code>	# 아무것도 하지 않는다.
71		
72	<code>pulse.ChangeDutyCycle(pos)</code>	# pos 값으로 Servo 회전
73	<code>sleep(0.5)</code>	# 0.5 초 대기
74		
75	<code>print "servo positon = ", pos</code>	# 현재 Servo 위치 값 화면출력
76		
77	<code>except KeyboardInterrupt:</code>	# 'Ctrl+C'입력 예외 발생 시 아래 코드 실행
78	<code>    pulse.stop()</code>	# Servo 제어 PWM Generation 중지
79	<code>    GPIO.cleanup()</code>	# GPIO 입출력 설정 Clear

## 2. C 언어( wiringPi )

서보모터 테스트 C 코드(servo.c) 작성

01	<code>#include &lt;stdio.h&gt;</code>	
02	<code>#include &lt;wiringPi.h&gt;</code>	
03	<code>#include &lt;softPwm.h&gt;</code>	
04		
05	<code>#define SERVO 21 // BCM 5</code>	
06	<code>#define MAX 23</code>	
07	<code>#define MIN 7</code>	
08		
09	<code>int ch2no(unsigned char ch);</code>	
10		
11	<code>int main()</code>	
12	<code>{</code>	
13	<code>    unsigned char ch;</code>	
14	<code>    unsigned int offset = 1;</code>	
15	<code>    unsigned int pos = 15;</code>	
16	<code>    unsigned int preset[5] = { 7, 11, 15, 19, 23 };</code>	
17		
18	<code>    if(wiringPiSetup() == -1) return 1;</code>	
19		
20	<code>    softPwmCreate(SERVO, 0, 200);</code>	
21		
22	<code>    printf("select preset position(1 ~ 5) ");</code>	
23	<code>    printf("or '&gt;' for cw, '&lt;' for ccw move.\n");</code>	
24		

```
25  while(ch != 'Q')
26  {
27      printf(">"); ch = getchar();
28
29      if( ch != '\n' && ch != '\r' )
30      {
31          if      (ch == '1')
32          {
33              softPwmWrite(SERVO0, preset[ch2no(ch)-1]);
34              pos   = preset[ch2no(ch)-1];
35          }
36          else if(ch == '2')
37          {
38              softPwmWrite(SERVO0, preset[ch2no(ch)-1]);
39              pos   = preset[ch2no(ch)-1];
40          }
41          else if(ch == '3')
42          {
43              softPwmWrite(SERVO0, preset[ch2no(ch)-1]);
44              pos   = preset[ch2no(ch)-1];
45          }
46          else if(ch == '4')
47          {
48              softPwmWrite(SERVO0, preset[ch2no(ch)-1]);
49              pos   = preset[ch2no(ch)-1];
50          }
51          else if(ch == '5')
52          {
53              softPwmWrite(SERVO0, preset[ch2no(ch)-1]);
54              pos   = preset[ch2no(ch)-1];
55          }
56          else if(ch == '.')
57          {
58              if(pos <= MAX-offset) pos += offset;
59              else                  pos = MAX;
60              softPwmWrite(SERVO0, pos);
61          }
62          else if(ch == ',')
63          {
64              if(pos >= MIN+offset) pos -= offset;
65              else                  pos = MIN;
66              softPwmWrite(SERVO0, pos);
67          }
68          else;
69          printf("SERVO Position = %2d\n", pos);
70      }
71      else;
72  }
```

```
73 }
74
75 int ch2no(unsigned char ch)
76 {
77     int res = ch - '0'; // '1' - '0' = 49 - 48 = 1
78     return res;         // '2' - '0' = 49 - 48 = 2
79 }
```

# 6. Web 을 통한 GPIO 제어

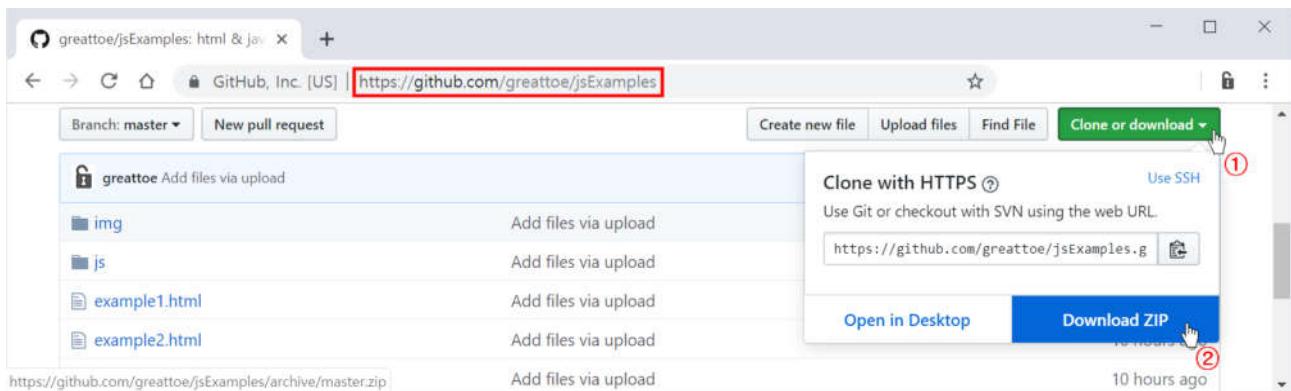
## 6.1. HTML, Javascript, Node.js

GPIO 입, 출력을 Web 을 통해 제어하는 웹서버를 Node.js 를 이용해서 구현해보자.

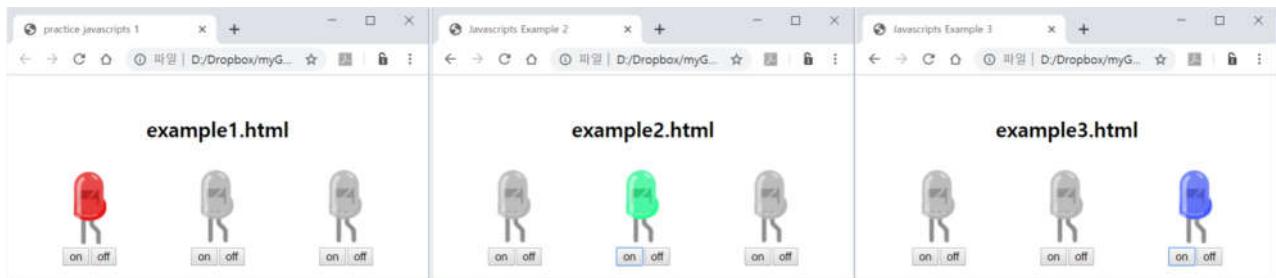
### 6.1.1. 간단히 살펴보는 HTML 과 Javascript

기본적으로 웹 페이지는 HTML로 구현된다. HTML( Hyper Text Markup Language )은 일종의 인터프리터 언어로서, HTML 의 인터프리터는 바로 웹 브라우저다. HTML 로 코드를 작성하는 것은 웹브라우저에 보여질 컨텐츠를 만드는 작업이다. 하지만 HTML 이 갖고 있는 한계로 인해 HTML 만 사용해서는 경우 정적인 컨텐츠밖에 표현할 수가 없다. 마우스 포인터가 하이퍼링크위로 이동하면 하이퍼링크 텍스트 색상이 바뀌거나, 서브메뉴가 열리는 등의 효과는 HTML 과 함께 사용되는 PHP 나 Javascript 같은 스크립트를 함께 사용해야 가능하다.

웹브라우저에서 <https://github.com/greattoe/jsExamples> 를 열고, 샘플 코드를 다운로드한다.



적당한 위치에 다운받은 zip 파일의 압축을 풀고 example1, 2, 3.html 을 편집기( 'NotePad++' 같은 소스 코드를 편집할 수 있는 ) 와 웹브라우저로 열어 둔다.



3 개의 파일 모두 'on', 'off' 버튼을 클릭하여 해당 LED 를 켜고 끄는 웹페이지이다. 물론 아직은 그림만 바뀔 뿐이다. 역시 3 개의 파일 모두 HTML 부분은 거의 같고( example2, 3 은 아예 똑같다. ), 버튼 클릭 시 LED 이미지를 교체하는 Javascript 로 구현된 부분이 조금 다르다.

예제( example1, 2, 3.html ) 코드들은 크게 HTML 부와 스크립트부로 나누어 볼 수 있다. 그 중 HTML 부는, 헤더와 본문으로 구성된다.

01	<html> <!-- ----- html 문서의 시작 태그 -->	HTML 코드는 '<'와 '>'로 둘러싸인 태그(Tag)로 표현된다. 태그는 보통 시작태그<tag>와 종료태그</tag>로 쌍을 이루지만 단독으로 쓰이는 태그도 있다. 태그에는 tag 이름 이외에도 해당 태그에서 지원하는 속성을 표현할 수도 있다.
02	<head> <!-- ----- header 의 시작 태그 -->	<tag 명 속성명="속성값"> 내용 </tag 명>
03	head 의 내용( 헤더 )	모든 HTML 문서는 <html> 태그로 시작해서 </html>로 끝나며, 크게 <head>와 </head>로 둘러싸인 헤더부와 <body>와 </body>로 둘러싸인 본문으로 이루어진다.
04	<head> <!-- ----- header 끝 태그 -->	
07	<body> <!-- ----- 본문의 시작 태그 -->	
.	body 의 내용( 본문 )	
.	body 의 내용( 본문 )	
57	</body> <!-- ----- 본문의 끝 태그 -->	
58	</html> <!-- ----- html 문서의 끝 태그 -->	

스크립트부는 <script>태그안에 작성하며, 그 범위에서의 문법은 해당 script 언어의 문법을 따른다.

32	<script> /* ----- script 시작 태그 */	<script> tag 이 후 이므로 Javascript 문법의 주석 처리문 사용
33	.	.
34	.	.
.	script 코드	javascript 코드
.	.	.
.	.	.
55	.	.
56	</script> <!-- ----- script 끝 태그 -->	</script> 스크립트 종료 tag 이 후 html 문법

## <table>태그 사용법

1	<table border="1">	좌측 HTML 코드로 만들어진 table						
2	<tr>							
3	<td>1 행 1 열</td><td>1 행 2 열</td><td>1 행 3 열</td>	<table border="1"><tr><td>1 행 1 열</td><td>1 행 2 열</td><td>1 행 3 열</td></tr><tr><td>2 행 1 열</td><td>2 행 2 열</td><td>2 행 3 열</td></tr></table>	1 행 1 열	1 행 2 열	1 행 3 열	2 행 1 열	2 행 2 열	2 행 3 열
1 행 1 열	1 행 2 열	1 행 3 열						
2 행 1 열	2 행 2 열	2 행 3 열						
4	</tr>							
5	<tr>							
6	<td>2 행 1 열</td><td>2 행 2 열</td><td>2 행 3 열</td>							
7	</tr>							
8	</table>							

좌측 HTML 코드로 만들어진 table

1 행 1 열	1 행 2 열	1 행 3 열
2 행 1 열	2 행 2 열	2 행 3 열

<tr> 행의 시작 </tr> 행의 끝

<td> 칸의 시작 </td> 칸의 끝

<td colspan="2"> 우측 칸과 두 칸 합치기

<td rowspan="2"> 아래 칸과 두 칸 합치기

## example1.html

01	<html>	
02	<head>	
03	<title>Javascript Example 1</title>	
04	</head>	
05	<body>	
06	<p> &nbsp; </p>	
07		
08		
09	<table align="center" border="0" background="white">	
10		
11	<tr align="center">	
12	<td colspan="6" width="150"><h2> example2.html </h2></td>	

```

13    </tr>
14
15    <tr align="center">
16        <td colspan="2" width="150"></td>
17        <td colspan="2" width="150"></td>
18        <td colspan="2" width="150"></td>
19    </tr>
20
21    <tr align="center">
22        <td align="right"><input type="button" value="on" onClick="on_red()" ></td>
23        <td align="left" ><input type="button" value="off" onClick="off_red()" ></td>
24        <td align="right"><input type="button" value="on" onClick="on_green()" ></td>
25        <td align="left" ><input type="button" value="off" onClick="off_green()"></td>
26        <td align="right"><input type="button" value="on" onClick="on_blue()" ></td>
27        <td align="left" ><input type="button" value="off" onClick="off_blue()" ></td>
28    </tr>
29
30 </table>
31
32 <script>
33     function on_red(){
34         document.images[0].src = "./img/onRed.gif";
35     }
36
37     function off_red(){
38         document.images[0].src = "./img/offRed.gif";
39     }
40
41     function on_green(){
42         document.images[1].src = "./img/onGreen.gif";
43     }
44
45     function off_green(){
46         document.images[1].src = "./img/offGreen.gif";
47     }
48
49     function on_blue(){
50         document.images[2].src = "./img/onBlue.gif";
51     }
52
53     function off_blue(){
54         document.images[2].src = "./img/offBlue.gif";
55     }
56 </script>
57 </body>
58 <html>
```

example2.html

```

01 <html>
02   <head>
03     <title>Javascript Example 2</title>
04   </head>
05
06   <body>
07     <p> &nbsp; </p>
08
09     <table align="center" border="0" background="white">
10
11       <tr align="center">
12         <td colspan="6" width="150"><h2>example2.html</h2></td>
13       </tr>
14
15       <tr align="center">
16         <td colspan="2" width="150"></td>
17         <td colspan="2" width="150"></td>
18         <td colspan="2" width="150"></td>
19       </tr>
20
21       <tr align="center">
22         <td align="right"><input type="button" value="on" onClick="on_red()" ></td>
23         <td align="left" ><input type="button" value="off" onClick="off_red()" ></td>
24         <td align="right"><input type="button" value="on" onClick="on_green()" ></td>
25         <td align="left" ><input type="button" value="off" onClick="off_green()" ></td>
26         <td align="right"><input type="button" value="on" onClick="on_blue()" ></td>
27         <td align="left" ><input type="button" value="off" onClick="off_blue()" ></td>
28       </tr>
29
30     </table>
31
32     <script>
33       function on_red(){
34         document.getElementById("rd").src = ".img/onRed.gif";
35       }
36
37       function off_red(){
38         document.getElementById("rd").src = ".img/offRed.gif";
39       }
40
41       function on_green(){
42         document.getElementById("grn").src = ".img/onGreen.gif";
43       }
44
45       function off_green(){
46         document.getElementById("grn").src = ".img/offGreen.gif";
47       }
48

```

```

49     function on_blue(){
50         document.getElementById("blu").src = "./img/onBlue.gif";
51     }
52
53     function off_blue(){
54         document.getElementById("blu").src = "./img/offBlue.gif";
55     }
56     </script>
57 </body>
58 <html>

```

### example3.html

```

01 <html>
02   <head>
03     <title>Javascript Example 3</title>
04   </head>
05
06   <body>
07     <p> &nbsp; </p>
08
09     <table align="center" border="0" background="white">
10
11       <tr align="center">
12         <td colspan="6" width="150"><h2>example2.html</h2></td>
13       </tr>
14
15       <tr align="center">
16         <td colspan="2" width="150"></td>
17         <td colspan="2" width="150"></td>
18         <td colspan="2" width="150"></td>
19       </tr>
20
21       <tr align="center">
22         <td align="right"><input type="button" value="on" onClick="on_red()" ></td>
23         <td align="left" ><input type="button" value="off" onClick="off_red()" ></td>
24         <td align="right"><input type="button" value="on" onClick="on_green()" ></td>
25         <td align="left" ><input type="button" value="off" onClick="off_green()" ></td>
26         <td align="right"><input type="button" value="on" onClick="on_blue()" ></td>
27         <td align="left" ><input type="button" value="off" onClick="off_blue()" ></td>
28       </tr>
29
30     </table>
31
32     <script src="http://code.jquery.com/jquery-latest.min.js"></script>
33     <script>
34       function on_red(){
35         $("#rd").attr('src','./img/onRed.gif');
36     }

```

```

37
38     function off_red(){
39         $("#rd").attr('src','./img/offRed.gif');
40     }
41
42     function on_green(){
43         $("#grn").attr('src','./img/onGreen.gif');
44     }
45
46     function off_green(){
47         $("#grn").attr('src','./img/offGreen.gif');
48     }
49
50     function on_blue(){
51         document.getElementById("blu").src = "./img/onBlue.gif";
52     }
53
54     function off_blue(){
55         document.getElementById("blu").src = "./img/offBlue.gif";
56     }
57     </script>
58 </body>
<html>

```

## 1. "example1.html"에서의 스크립트 처리

16 행: ``

- "example2.html"이 들어 있는 폴더의 "src"폴더 내에 있는 "offRed.gif"파일을 화면에 표시하는 `<img>`태그.

22 행: `<input type="button" value="on" onClick="on_red()">`

- "type"속성값이 "button", "value"속성값이 "on", "onClick"속성값이 "red\_on()"인 `<input>`태그로 여러 형태로 사용자 입력을 받을 수 있는 `<input>`태그 중 button이고, button에는 "on"을 표기하고, 버튼이 클릭되면 스크립트에서 "on\_red()"함수를 불러 실행한다.

23 행: `<input type="button" value="off" onClick="on_red()">`

- "type"속성값이 "button", "value"속성값이 "off", "onClick"속성값이 "red\_on()"인 `<input>`태그로 여러 형태로 사용자 입력을 받을 수 있는 `<input>`태그 중 button이고, button에는 "off"를 표기하고, 버튼이 클릭되면 스크립트에서 "off\_red()"함수를 불러 실행한다.

34 행: `document.images[0].src = "./img/onRed.gif";`

- javascript에서는 사용된 HTML 태그마다 해당 태그배열이 장동으로 만들어진다. 함수 `on_red()`는 이 HTML 문서에 사용된 `<img>`태그 배열 "images[]"의 첫번째 요소( `images[0]` )의 속성값을 "./img/onRed.gif"로 변경한다.

38 행: `document.getElementById("rd").src = "./img/offRed.gif";`

- 역시 `<img>`태그 배열 "images[]"의 첫번째 요소( `images[0]` )의 "src" 속성값을 "./img/offRed.gif"로 변경한다.

## 2. "example2.html"에서의 스크립트 처리

16 행: ``

- "example2.html"이 들어 있는 폴더의 "src"폴더 내에 있는 "offRed.gif"파일을 화면에 표시하는 "id"속성 값이 "rd"인 `<img>`태그

22 행: `<input type="button" value="on" onClick="on_red()">`

- "type"속성값이 "button", "value"속성값이 "on", "onClick"속성값이 "red\_on()"인 `<input>`태그로 여러 형태로 사용자 입력을 받을 수 있는 `<input>`태그 중 button이고, button에는 "on"을 표기하고, 버튼이 클릭되면 스크립트에서 "on\_red()"함수를 불러 실행한다.

23 행: `<input type="button" value="off" onClick="off_red()">`

- "type"속성값이 "button", "value"속성값이 "off", "onClick"속성값이 "red\_on()"인 `<input>`태그로 여러 형태로 사용자 입력을 받을 수 있는 `<input>`태그 중 button이고, button에는 "off"을 표기하고, 버튼이 클릭되면 스크립트에서 "off\_red()"함수를 불러 실행한다.

34 행: `document.getElementById("rd").src = "./img/onRed.gif";`

- 함수 `on_red()`는 이 HTML 문서에 사용된 태그 중 "id" 속성값이 "rd"인 태그의 "src" 속성값을 "./img /onRed.gif"로 변경한다.

38 행: `document.getElementById("rd").src = "./img/offRed.gif";`

- 함수 `off_red()`는 이 HTML 문서에 사용된 태그 중 "id" 속성값이 "rd"인 태그의 "src" 속성값을 "./img /offRed.gif"로 변경한다.

## 3. "example3.html"에서의 스크립트 처리

16 행: ``

- "example2.html"이 있는 폴더의 "img"폴더안의 "offRed.gif"파일을 화면에 표시하는 "id"속성값이 "rd"인 `<img>`태그의 "src" 속성값을 "./img /onRed.gif"로 변경한다.

22 행: `<input type="button" value="on" onClick="on_red()">`

- "type"속성값이 "button", "value"속성값이 "on", "onClick"속성값이 "red\_on()"인 `<input>`태그로 여러 형태로 사용자 입력을 받을 수 있는 `<input>`태그 중 button이고, button에는 "on"을 표기하고, 버튼이 클릭되면 스크립트에서 "on\_red()"함수를 불러 실행한다.

23 행: `<input type="button" value="off" onClick="off_red()">`

- "type"속성값이 "button", "value"속성값이 "off", "onClick"속성값이 "red\_on()"인 `<input>`태그로 여러 형태로 사용자 입력을 받을 수 있는 `<input>`태그 중 button이고, button에는 "off"을 표기하고, 버튼이 클릭되면 스크립트에서 "off\_red()"함수를 불러 실행한다.

32 행: `<script src="http://code.jquery.com/jquery-latest.min.js"></script>`

- 웹 개발을 빠르고 쉽게 할 수 있도록 해주는 자바스크립트 기반의 프레임워크인 "jQuery" 사용을 위한 CDN( Content Delivery Network ) 주소.

34 행: `$("#rd").attr('src','./img/onRed.gif');`

- "\$"는 "jQuery"문자열을 대신한다("\$\$"대신 "jQuery"라고 코드에서 사용해도 된다). "#"은 "id"를 의미한다. "id" 속성값이 "rd"인 태그를 찾아 "src" 속성값을 "./img/onRed.gif"로 변경한다.

38 행: `$("#rd").attr('src','./img/offRed.gif');`

- 역시 마찬가지로 "id" 속성값이 "rd"인 태그를 찾아 "src" 속성값을 "./img/offRed.gif"로 변경한다.

## 6.1.2. Node.js

Javascript 는 원래 스크립트 언어이다. 독립적으로 동작하지 않고, 웹브라우저와 함께 기능하도록 만들어졌다. 이것이 가능하다는 것은 웹브라우저에서는 보이지 않는 뒷편에 이렇게 기능하도록 설계된 무언가가 동작하고 있기 때문이다.

여기서 우리가 볼 수 있는 웹브라우저와 함께 기능하는 부분을 front-end, 그 뒤에서 보이지 않게 동작하는 부분을 back-end 라고 하는데, 예전에 Java, C++(.net)등으로 구현했던 back-end 부분을 back-end 용 Javascript 가 나타나 대신하면서( front-end, back-end 모두를 Javascript 만으로 구현하면서 ) 더 이상 웹브라우저에 의존적이지 않은 Javascript 가 탄생한다. 이 것이 Node.js 이다.

### 1. Node.js

웹에서 라즈베리파이의 GPIO를 통해 출력을 내보내거나, 신호를 입력 받으려면 일단 웹서버가 필요하다. 라즈베리파이에서 웹서버를 구동하기 위해 node.js 와 express.js 를 설치한다.

2019-07-10-raspbian-buster-full 사용자의 경우는 이미 설치되어 있어, Node.js 를 설치할 필요 없다. 하지만, npm( Node Package Manager )이 설치되어 있지 않으므로 다음 명령으로 npm 만 설치해주면 된다.

```
pi@raspberrypi:~ $ sudo apt-get install npm
```

#### node.js 설치

node.js 바이너리 저장소(repository) 등록 및 반영(update).

```
pi@raspberrypi:~ $ curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
```

#### node.js 설치.

```
pi@raspberrypi:~ $ sudo apt-get install nodejs -y
```

#### node.js 설치 확인.

```
pi@raspberrypi:~ $ node -v  
v8.16.0
```

#### npm( Node Package Manager ) 설치 확인.

```
pi@raspberrypi:~ $ npm -v  
6.4.1
```

### 2. Express.js

Node.js 를 이용한 웹 개발 Framework 가 바로 "Express.js"이다.

#### Express.js 설치

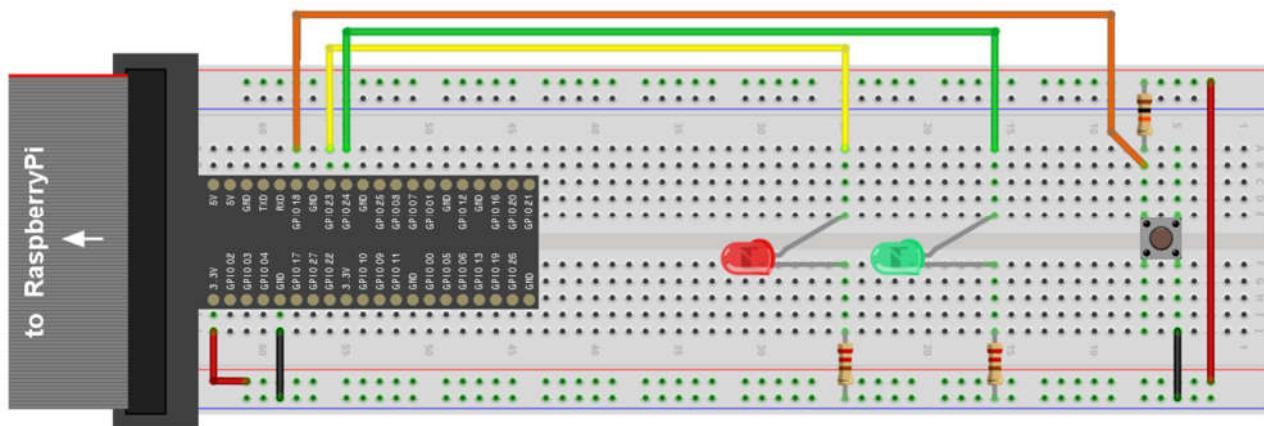
```
pi@raspberrypi:~ $ sudo npm install -g express
```

```
pi@raspberrypi:~ $ sudo npm install -g express-generator
```

## 6.2. GPIO 제어 웹서버 구현

### 6.2.1. LED 제어 웹 서버

앞서 꾸며 놓은 GPIO 입출력 제어에 사용한 회로의 LED를 웹을 통해 제어해 보자.



express.js로 LED 제어 웹 어플리케이션 'led' 생성

```
pi@raspberrypi:~ $ express led
```

기본 의존성 설치

```
pi@raspberrypi:~ $ cd led  
pi@raspberrypi:~/led $ npm install
```

추가 의존성("onoff" node-package) 설치

```
pi@raspberrypi:~/led $ npm install onoff --save
```

NotePad++의 ftp 플러그인을 이용하여 "~/led/app.js" 파일 편집

```
new 1 - Notepad++  
파일(F) 편집(E) 찾기(S) 보기(V) 인코딩(I) 언어(L) 설정(I) 도구(Q) 매크로 실행 플러그인 창 관리 ?  
new 1  
1  
NppFTP - Connected to pi4lesson  
> routes  
> views  
app.js  
Ac... Pro... File  
NppFTP  
11:47:14 [NppFTP] Everything initialized  
18:04:44 Connecting  
18:04:46 [SFTP] Host key accepted  
Normal text file length : 0 lines : 1 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 IN  
01 const express = require('express');  
02 const app = express();
```

```

03 const Gpio = require('onoff').Gpio;
04 const red = new Gpio(23, 'out');
05 const grn = new Gpio(24, 'out');
06
07 const port = 8000;
08 app.use(express.static(__dirname + '/public'));
09
10 app.get('/red_on', function(req, res) {
11   red.writeSync(1);
12   console.log('send: led red on');
13   res.writeHead(200, {'Content-Type': 'text/plain'});
14   res.end('LED Red On\n');
15 });
16
17 app.get('/red_off', function(req, res) {
18   red.writeSync(0);
19   console.log('send: led red off');
20   res.writeHead(200, {'Content-Type': 'text/plain'});
21   res.end('LED Red Off\n');
22 });
23
24 app.get('/grn_on', function(req, res) {
25   grn.writeSync(1);
26   console.log('send: led green on');
27   res.writeHead(200, {'Content-Type': 'text/plain'});
28   res.end('LED Green On\n');
29 });
30
31 app.get('/grn_off', function(req, res) {
32   grn.writeSync(0);
33   console.log('send: led green off');
34   res.writeHead(200, {'Content-Type': 'text/plain'});
35   res.end('LED Green Off\n');
36 });
37
38 app.listen(port, function(err) {
39   console.log('Connected port: ' + port);
40   if (err) {
41     return console.log('Found error - ', err);
42   }
43 });

```

## 1. index.html(Ver. 1.0)

"~/led/public/index.html"파일 생성

```

pi@raspberrypi:~/led $ cd public
pi@raspberrypi:~/led/public $ touch index.html

```

NotePad++에서 "~/led/public/index.html"파일 편집

```

01 <!DOCTYPE html>
02 <html lang=ko>
03   <head>
04     <title>Control LED on RPi</title>
05   </head>
06
07   <body>
08     <h2>Control LED on RPi</h2>
09
10     <a href="/red_on" target="response"><h3>Turn on red LED</h3></a>
11     <a href="/red_off" target="response"><h3>Turn off red LED</h3></a>
12
13     <a href="/green_on" target="response"><h3>Turn on green LED</h3></a>
14     <a href="/green_off" target="response"><h3>Turn off green LED</h3></a>
15
16   <table>
17     <tr height="30">
18       <td>Status Message: <br> &ampnbsp<br> &ampnbsp</td>
19       <td>
20         <iframe src="about:blank" width="300" height="30" frameborder="0"
21             marginwidth="0" marginheight="0" name="response"></iframe>
22         <p></p>
23       </td>
24     </tr>
25   </table>
26 </body>
27 </html>

```

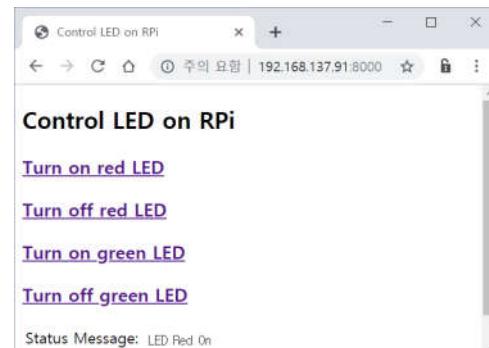
## LED 제어 웹서버 구동

```

pi@raspberrypi:~ $ cd led
pi@raspberrypi:~/led $ node app.js

```

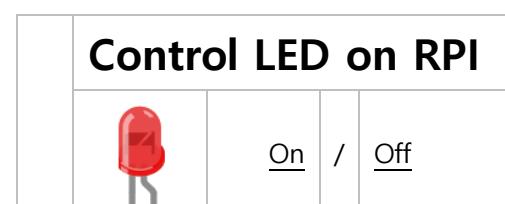
PC를 라즈베리파이와 같은 네트워크에 연결하고, 웹브라우저에서 URL "http://라즈베리파이의 IP 주소:8000"을 열어 오른쪽 그림과 같은 웹서버 구동화면을 확인한다.  
하이퍼링크를 클릭하여 LED 동작을 확인한다



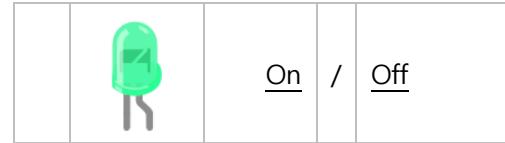
## 2. index.html(Ver. 2.0)

### 웹서버 구동화면에 LED 이미지 표시를 위한 레이아웃 디자인

파일 편집 전에 화면에 보여지게 될 레이아웃을 그려 보자. 텍스트와 이미지를 원하는 형태로 배치하기 위한 표(table)를 오른쪽에 그렸다. 왼쪽에는 마진을 주기위한 빈칸을 만들고, 제목을 표시하기 위한 행을 위에 배치했다.



아래 좌측에 적색 LED 이미지를, 우측에 On / Off 컨트롤을 배치하고, 그 아래 좌측에 녹색 LED 이미지를, 우측에 On / Off 컨트롤을 배치했다.



### "~/led/public/index.html" 파일 편집

이제 설계한 레이아웃을 반영하여 "~/led/public/index.html" 파일을 편집한다.

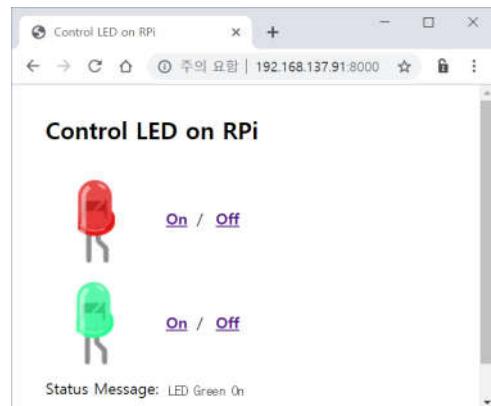
```
01 <!DOCTYPE html>
02 <html lang=ko>
03   <head>
04     <title>Control LED on RPi</title>
05   </head>
06
07   <body>
08
09     <table border="0">
10       <tr>
11         <td rowspan="3" width="20" height="50"></td>
12         <td colspan="4"><h2>Control LED on RPi</h2></td>
13       </tr>
14       <tr>
15         <td align="center"></td>
16         <td><a href="/red_on" target="response" align="right"><h4>On </h4></a></td>
17         <td>&nbsp;/ </td>
18         <td><a href="/red_off" target="response"><h4>Off</h4></a></td>
19       </tr>
20       <tr>
21         <td align="center"></td>
22         <td><a href="/green_on" target="response" align="right"><h4>On </h4></a></td>
23         <td>&nbsp;/ </td>
24         <td><a href="/green_off" target="response"><h4>Off</h4></a></td>
25       </tr>
26     </table>
27
28     <table>
29       <tr height="30">
30         <td width="20"></td>
31         <td>Status Message: <br> &nbsp;<br> &nbsp;</td>
32         <td>
33           <iframe src="about:blank" width="300" height="30" frameborder="0"
34             marginwidth="0" marginheight="0" name="response"></iframe>
35           <p></p>
36           </td>
37         </tr>
38     </table>
39
40   </body>
41 </html>
```

편집한 HTML 파일을 저장하고, 웹브라우저에서 확인하면 오른쪽 그림과 같은 화면을 볼 수 있다.

On / Off 링크를 통해 같은 색의 LED 가 켜지고 꺼지는 것은 확인할 수 있지만 화면에 표시된 LED 의 색은 변하지 않는다.

실제 LED 동작에 맞춰서 화면의 이미지도 변화시키기 위해 다시 index.html 파일을 편집한다.

지금까지는 HTML 만 가지고 구현이 가능하였지만, 바로 이런 기능을 구현해야 할 때 스크립트가 필요하다.

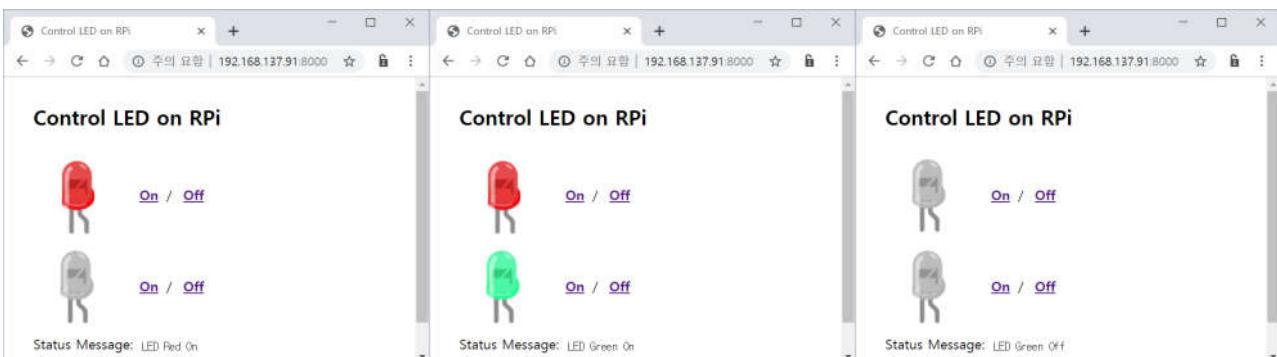


### 3. index.html(Ver. 3.0)

index.html 파일에 LED 동작에 따른 이미지변경 스크립트 추가

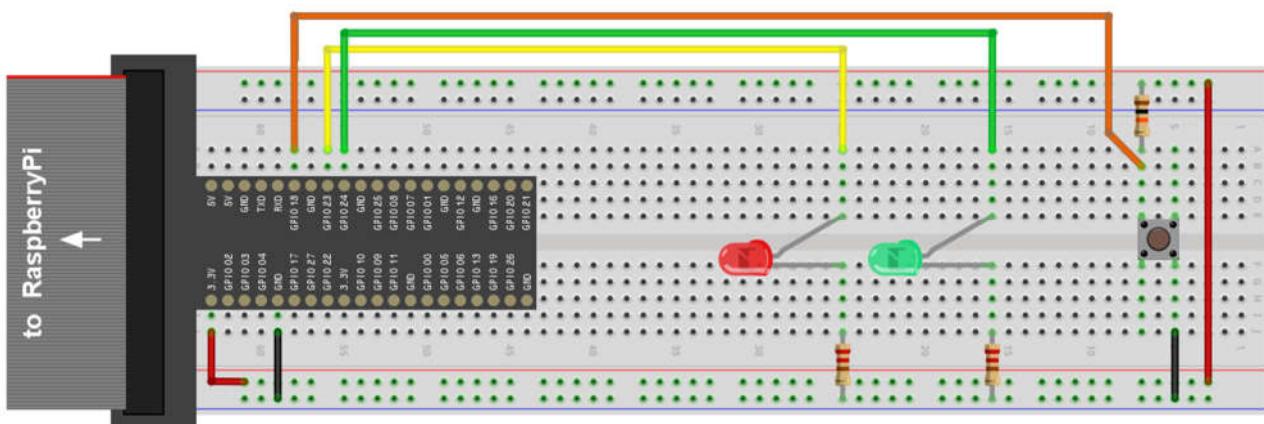
```
01 <!DOCTYPE html>
02 <html lang=ko>
03   <head>
04     <title>Control LED on RPi</title>
05   </head>
06   <body>
07     <table border="0">
08       <tr>
09         <td rowspan="3" width="20" height="50"></td>
10         <td colspan="4"><h2>Control LED on RPi</h2></td>
11       </tr>
12       <tr>
13         <td align="center"></td>
14         <td>
15           <a href="/red_on" target="response" align="right">
16             <h4><p onClick="red_on()">On </p></h4>
17           </a>
18         </td>
19         <td>&ampnbsp/ </td>
20         <td>
21           <a href="/red_off" target="response">
22             <h4><p onClick="red_off()">Off</p></h4>
23           </a>
24         </td>
25       </tr>
26       <tr>
27         <td align="center"></td>
28         <td>
29           <a href="/green_on" target="response" align="right">
30             <h4><p onClick="green_on()">On </p></h4>
31           </a>
32         </td>
33         <td>&ampnbsp/ </td>
```

```
34     <td>
35         <a href="/green_off" target="response">
36             <h4><p onClick="green_off()">Off</p></h4>
37         </a>
38     </td>
39     </tr>
40 </table>
41
42 <table>
43     <tr height="30">
44         <td width="20"></td>
45         <td>Status Message: <br> &ampnbsp<br> &ampnbsp</td>
46         <td>
47             <iframe src="about:blank" width="300" height="30" frameborder="0"
48                 marginwidth="0" marginheight="0" name="response"></iframe>
49             <p></p>
50         </td>
51     </tr>
52 </table>
53
54 <script>
55     function red_on(){
56         document.getElementById("rd").src = "./images/onRed.gif";
57     }
58
59     function red_off(){
60         document.getElementById("rd").src = "./images/offRed.gif";
61     }
62
63     function green_on(){
64         document.getElementById("grn").src = "./images/onGreen.gif";
65     }
66
67     function green_off(){
68         document.getElementById("grn").src = "./images/offGreen.gif";
69     }
70 </script>
71 </body>
72 </html>
```



## 6.2.2. 스위치 입력에 의한 웹페이지 변경

앞서 꾸며 놓은 GPIO 입출력 제어에 사용한 회로의 스위치 입력으로 웹페이지를 제어해 보자.



Express.js로 웹 어플리케이션 "changePg" 생성

```
pi@raspberrypi:~ $ express changePg
```

기본 의존성 설치

```
pi@raspberrypi:~ $ cd changePg
pi@raspberrypi:~/changePg $ npm install
```

추가 의존성 설치 1. ("onoff" node package 설치)

```
pi@raspberrypi:~/changePg $ npm install onoff --save
```

추가 의존성 설치 2. ("socket.io" node package 설치)

```
pi@raspberrypi:~/changePg $ npm install socket.io --save
```

NotePad++의 ftp 플러그인을 이용하여 "~/changePg/app.js" 파일 편집

```
01 const express = require('express');
02 const app = express();
03 const http = require('http').Server(app);
04 const io = require('socket.io')(http);
```

```

05 const Gpio = require('onoff').Gpio;
06
07 const sw = new Gpio(18, 'in', 'falling', {debounceTimeout: 100});
08
09 app.use(express.static(__dirname + '/public'));
10 const port = 3000;
11
12 var page = [ 'index.html', 'page2.html', 'page3.html' ];
13 var count = 1;
14 var index = 0;
15
16 sw.watch( function (err, val) {
17     count++;
18     if( conut == 999 ) count = 0;
19
20     index = count % page.length;
21     console.log('load ' + page[index]);
22
23     io.emit('page', page[index]);
24 });
25
26 http.listen(port, function() {
27     console.log('listening on *:' + port);
28 });

```

"~/changePg/public/index.html" 파일 생성

```

pi@raspberrypi:~/changePg $ cd public
pi@raspberrypi:~/changePg/public $ touch index.html

```

NotePad++의 ftp 플러그인을 이용하여 "~/changePg/public/index.html"파일 편집

```

01 <!DOCTYPE html>
02 <html lang=ko>
03     <head>
04         <meta charset="utf-8">
05         <title>1st page</title>
06     </head>
07
08     <body>
09         <p><h2> page1 </h2></p>
10         <hr>
11         <p><h3> Change web page by switch input </h3></p>
12         <script src="https://code.jquery.com/jquery-2.2.1.min.js"></script>
13         <!--script src=".//javascripts/jquery.min.js"></script-->
14         <script src="socket.io/socket.io.js"></script>
15         <script>
16             $(function(){
17                 var socket = io();
18                 socket.on('page', function(pgName){

```

```

19         window.location.replace(pgName);
20     });
21   });
22 </script>
23 </body>
24 </html>

```

"~/changePg/public/index.html"파일 편집

```

01 <!DOCTYPE html>
02 <html lang=ko>
03   <head>
04     <meta charset="utf-8">
05     <title>2nd page</title>
06   </head>
07
08 <body bgcolor="#c0c0c0">
09   <p><h2> page2 </h2></p>
10   <hr>
11   <p><h3> Change web page by switch input </h3></p>
12   <script src="https://code.jquery.com/jquery-2.2.1.min.js"></script>
13   <!--script src=".//javascripts/jquery.min.js"></script-->
14   <script src="socket.io/socket.io.js"></script>
15   <script>
16     $(function(){
17       var socket = io();
18       socket.on('page', function(pgName){
19         window.location.replace(pgName);
20       });
21     });
22   </script>
23 </body>
24 </html>

```

"~/changePg/public/page3.html"파일 편집

```

01 <!DOCTYPE html>
02 <html lang=ko>
03   <head>
04     <meta charset="utf-8">
05     <title>3rd page</title>
06   </head>
07
08 <body bgcolor="#a0a0ff">
09   <p><h2> page3 </h2></p>
10   <hr>
11   <p><h3> Change web page by switch input </h3></p>
12   <script src="https://code.jquery.com/jquery-2.2.1.min.js"></script>
13   <!--script src=".//javascripts/jquery.min.js"></script-->
14   <script src="socket.io/socket.io.js"></script>

```

```

15 <script>
16   $(function(){
17     var socket = io();
18     socket.on('page', function(pgName){
19       window.location.replace(pgName);
20     });
21   });
22 </script>
23 </body>
24 </html>

```

라즈베리파이에서 changePg 웹서버 구동하고, GPIO 18에 연결한 스위치를 누를 때마다 웹브라우저에 표시되는 페이지가 바뀌는 것을 확인한다. ( 웹서버 URL: "http://라즈베리파이 IP 주소:3000" )

The image shows three separate browser windows, each displaying a different web page. The top window is titled '1st page' and contains the text 'page1'. The middle window is titled '2nd page' and contains the text 'page2'. The bottom window is titled '3rd page' and contains the text 'page3'. All three windows have the same URL bar showing '192.168.137.91:3000'. Below each window, there is a small text area that says 'Change web page by switch input'.

## 6.3. GPIO 음성제어

앞서 구현한 "LED 웹 제어"와, "스위치를 이용한 웹 페이지 변경"을 이번에는 음성인식 기능을 이용하여 구현해 보자.

### 6.3.1. 음성 입력장치 설정

음성을 통한 GPIO 제어 웹서버를 구현을 위해서는 음성을 입력받기 위해 마이크가 필요하다. USB 포트에 마이크를 연결하고 터미널 창에서 다음 명령을 실행한다.



```

pi@raspberrypi:~ $ arecord -l
**** List of CAPTURE Hardware Devices ****
card 1: Device [USB PnP Sound Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0

```

```
pi@raspberrypi:~ $
```

이 명령은 녹음장치 리스트를 보여주는 명령이다. 현재 1 개의 녹음장치를 찾아 보여주고 있으며 장치명은 "USB PnP Sound Device" 인 것을 알 수 있다.

이번에는 재생장치 리스트를 출력해보자.

```
pi@raspberrypi:~ $ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: ALSA [bcm2835 ALSA], device 0: bcm2835 ALSA [bcm2835 ALSA]
    Subdevices: 7/7
    Subdevice #0: subdevice #0
    Subdevice #1: subdevice #1
    Subdevice #2: subdevice #2
    Subdevice #3: subdevice #3
    Subdevice #4: subdevice #4
    Subdevice #5: subdevice #5
    Subdevice #6: subdevice #6
card 0: ALSA [bcm2835 ALSA], device 1: bcm2835 ALSA [bcm2835 IEC958/HDMI]
    Subdevices: 1/1
    Subdevice #0: subdevice #0
pi@pi4lesson:~$
```

이 정보들을 바탕으로 "~/.asoundrc" 파일을 다음과 같이 편집 후 저장한다.

```
pi@raspberrypi:~ $ nano ~/.asoundrc
```

```
GNU  nano 2.7.4                               File: ./asoundrc                         Modified
pcm.!default
{
    type asym

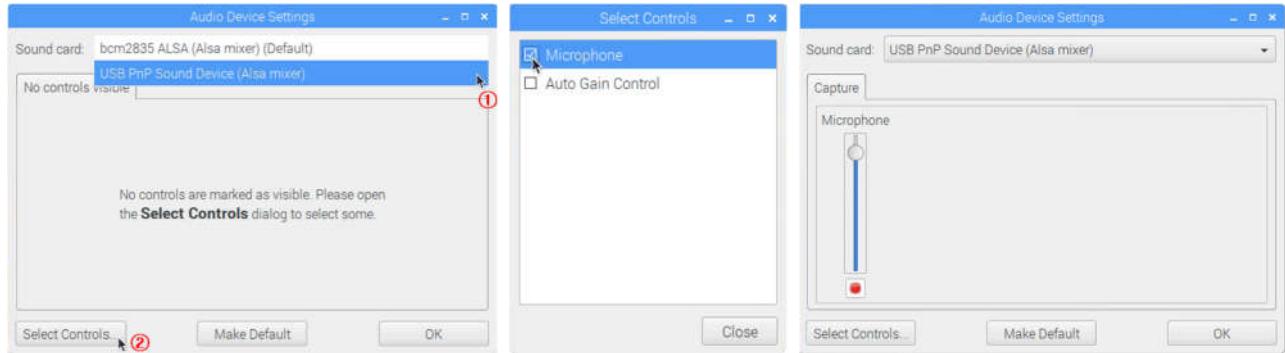
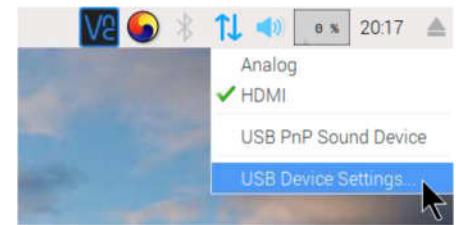
    # set audio output device
    playback.pcm {
        type plug
        # PLAYBACK Hardware Devices card 0, device 0 (figure out "aplay -l")
        slave.pcm "hw:0,0"
    }

    # set audio input device
    capture.pcm {
        type plug
        # CAPTURE Hardware Devices card 1, device 0 (figure out "arecord -l")
        slave.pcm "hw:1,0"
    }
}

^G Get Help   ^O WriteOut   ^R Read File   ^Y Cut Text   ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is    ^V Next Page  ^U Uncut Text ^T To Spell
```

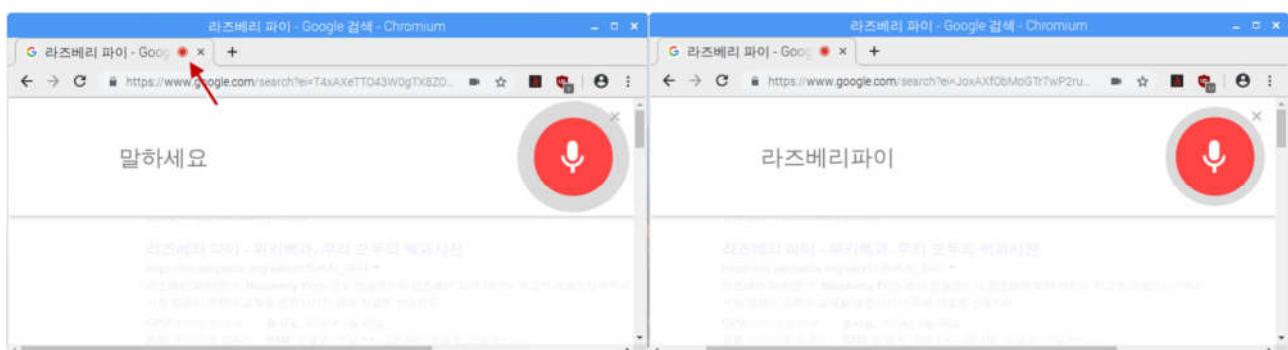
마이크의 녹음 음량을 조절하기 위해 오른 쪽 그림과 같이 우측 상단 스피커 아이콘을 마우스 오른쪽 버튼으로 클릭하여 "USB Device Setting" 을 선택한다.

"Audio Device Setting" 메뉴창이 열리면, 아래 그림처럼 Sound Card 항목을 "USB PnP Sound Device"로 변경 후 "Select Control" 버튼을 클릭한다.



"Select Control" 메뉴창이 열리면, 위 그림처럼 "Microphone" 을 선택하고, "Close" 버튼을 클릭하면, 마이크 볼륨조절 컨트롤이 나타난다. 마이크 볼륨을 충분히 높혀주고 "OK" 버튼을 클릭한다.

마이크 음성 입력을 테스트하기 위해 웹브라우저에서 "<http://google.com>"을 열어 오른 쪽 그림의 마이크 모양 음성검색 아이콘을 클릭한다. 그 때 아래 왼쪽 그림에 화살표로 표시한 마이크 입력 활성화를 표시하는 빨간 원이 표시되면 마이크에 검색어를 소리내어 검색할 단어를 말하고 아래와 같이 검색어로 입력되는지를 확인한다.



이상의 사항이 확인되었다면, 새로 만들 어플리케이션에서 마이크를 사용할 준비가 된 것이다.

### 6.3.2. 음성 명령으로 LED 제어하기

express.js로 LED 제어 웹 어플리케이션 'voiceLED' 생성

```
pi@raspberrypi:~ $ express voiceLED
```

기본 의존성 설치

```
pi@raspberrypi:~ $ cd voiceLED  
pi@raspberrypi:~/voiceLED $ npm install
```

추가 의존성( "onoff" node-package ) 설치

```
pi@raspberrypi:~/voiceLED $ npm install onoff --save
```

"~/voiceLED/app.js" 파일 편집 - 앞서 작성한 "~/led/app.js" 파일을 그대로 사용한다.

```
pi@raspberrypi:~/voiceLED $ cp ..//led/app.js ./  
pi@raspberrypi:~/voiceLED $
```

"~/voiceLED/public/index.html" 파일은 "~/led/public/index.html"의 내용에 스크립트만 추가하면 되므로 다음 명령으로 "~/led/public/index.html"을 복사해온다.

```
pi@raspberrypi:~/voiceLED $ cp ..//led/public/index.html ./public  
pi@raspberrypi:~/voiceLED $
```

"~/voiceLED/public/index.html" 편집

index.html 파일에 음성 인식 스크립트 추가

```
1 <!DOCTYPE html>  
2 <html lang=ko>  
3   <head>  
4     <title>Control LED on RPi</title>  
5   </head>  
6   <body>  
7     <table border="0">  
8       <tr>  
9         <td rowspan="5" width="20" height="50"></td>  
10        <td colspan="4"><h2>Control LED on RPi</h2></td>  
11      </tr>  
12      <tr>  
13        <td align="center"></td>  
14        <td>  
15          <a href="/red_on" target="response" align="right">  
16            <h4><p onClick="red_on()">On </p></h4>  
17          </a>  
18        </td>  
19        <td>&ampnbsp/ </td>  
20        <td>  
21          <a href="/red_off" target="response">  
22            <h4><p onClick="red_off()">Off</p></h4>  
23          </a>  
24        </td>  
25      </tr>  
26      <tr>  
27        <td align="center"></td>  
28        <td>  
29          <a href="/green_on" target="response" align="right">
```

```
30             <h4><p onClick="green_on()">On </p></h4>
31         </a>
32     </td>
33     <td>&nbsp;/ </td>
34     <td>
35         <a href="/green_off" target="response">
36             <h4><p onClick="green_off()">Off</p></h4>
37         </a>
38     </td>
39     </tr>
40 </table>
41
42 <table>
43     <tr height="30">
44         <td width="20"></td>
45         <td>Status Message: <br> &nbsp;<br> &nbsp;</td>
46         <td>
47             <iframe src="about:blank" width="300" height="30" frameborder="0"
48                 marginwidth="0" marginheight="0" name="response"></iframe>
49             <p></p>
50         </td>
51     </tr>
52 </table>
53
54 <script>
55 $(document).ready(function()
56 {
57     if (annyang)
58     {
59         var commands = {
60             '빨강 켜':function(){
61                 responsiveVoice.speak("빨강색 엘이디가 켜졌습니다.", "Korean Female");
62                 response.location.href=("http://localhost:3000/red_on");
63                 red_on();
64             },
65
66             '빨강 꺼':function(){
67                 responsiveVoice.speak("빨강색 엘이디가 꺼졌습니다.", "Korean Female");
68                 response.location.href=("http://localhost:3000/red_off");
69                 red_off();
70             },
71
72             '초록 켜':function(){
73                 responsiveVoice.speak("초록색 엘이디가 켜졌습니다.", "Korean Female");
74                 response.location.href=("http://localhost:3000/green_on");
75                 green_on();
76             },
77         }
78     }
79 }</script>
```

```

78     '초록 깨':function(){
79         responsiveVoice.speak("초록색 엘이디가 꺼졌습니다.", "Korean Female");
80         response.location.href=("http://localhost:3000/green_off");
81         green_off();
82     }
83 };
84 annyang.addCommands(commands);
85 annyang.removeCallback();
86 annyang.setLanguage('ko');
87 annyang.start({ autoRestart: true, continuous: false });
88 }
89 );
90
91 function red_on(){
92     document.getElementById("rd").src = "./images/onRed.gif";
93 }
94
95 function red_off(){
96     document.getElementById("rd").src = "./images/offRed.gif";
97 }
98
99 function green_on(){
100    document.getElementById("grn").src = "./images/onGreen.gif";
101 }
102
103 function green_off(){
104     document.getElementById("grn").src = "./images/offGreen.gif";
105 }
106 </script>
107 </body>
108 </html>

```

아래와 같이 웹서버를 구동한다.

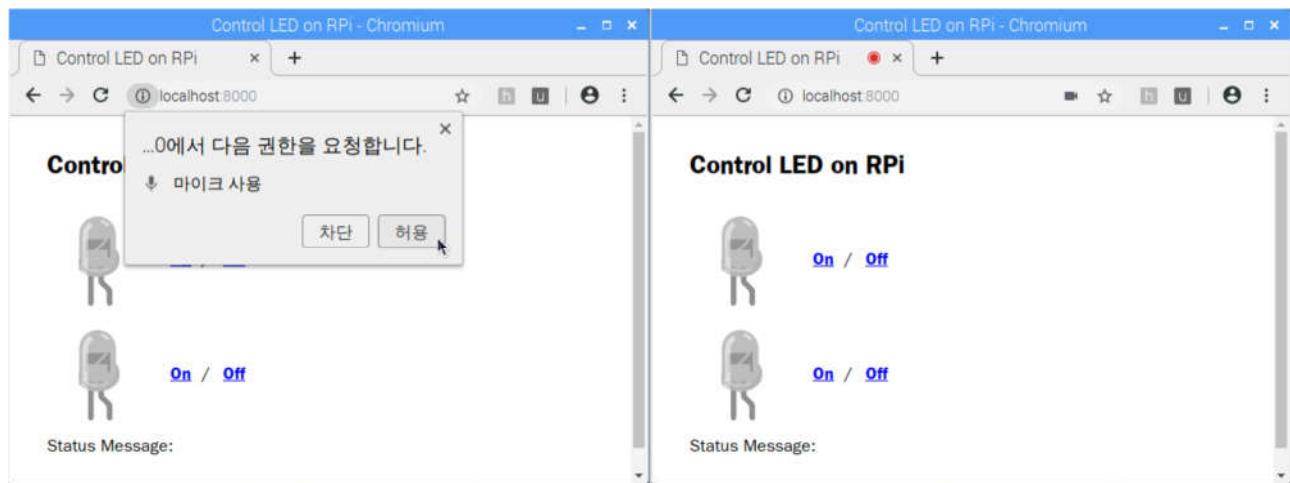
```

pi@raspberrypi:~/voiceLED $ node app.js
Connected port - 3000

```

음성인식에 사용할 마이크가 웹서버( 라즈베리파이 )에 연결되어 있으므로 PC 의 웹브라우저에서는 마이크 사용 권한을 얻을 수 없다.

라즈베리파이에서 직접 웹브라우저를 구동하고, URL "http://localhost:3000/" 을 연다. 마이크 사용권한 요청에 "허용"을 클릭하면 역시 마이크 입력 활성화를 나타내는 빨간 동그라미가 타이틀 옆에 잠시 깜빡이다 고정되는 것을 볼 수 있다. 음성 입력을 받을 준비가 되었다는 신호다. 이제, 음성명령("빨강 켜", "초록 켜",...)을 통해 LED 를 제어해보자.



## 7. Pi Camera 제어

라즈베리파이 카메라는 전용 라이브러리를 이용하면, 손쉽게 제어가 가능하다.

### 7.1. Pi Camera 연결

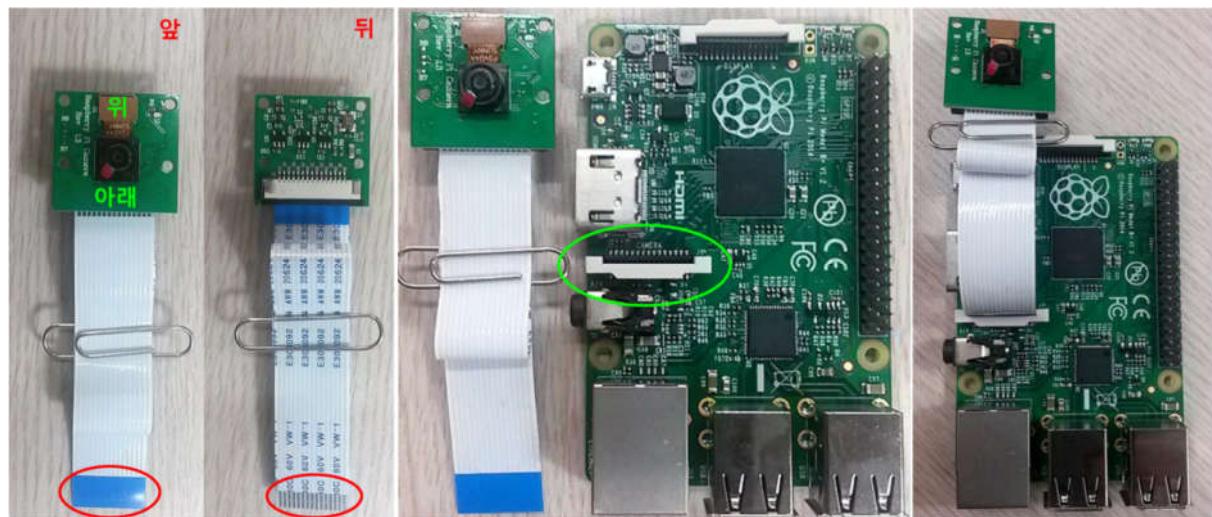
연결 커넥터의 위치

- HDMI 커넥터와 Ethernet 포트 사이( 아래 중앙 사진 참조)

연결 시 케이블 방향

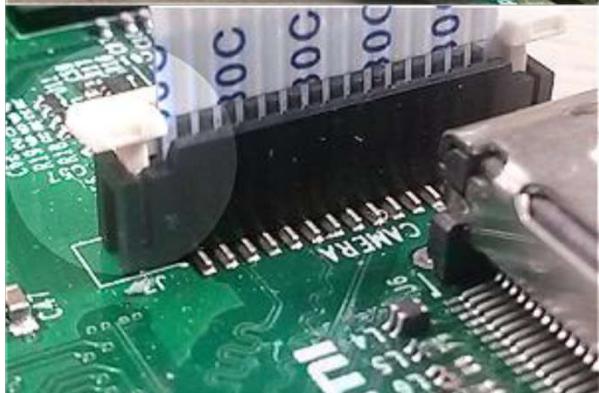
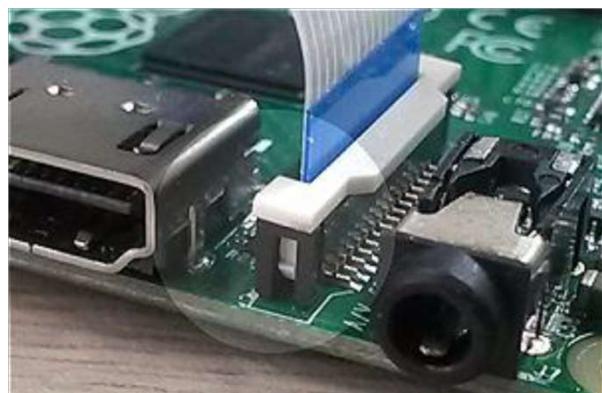
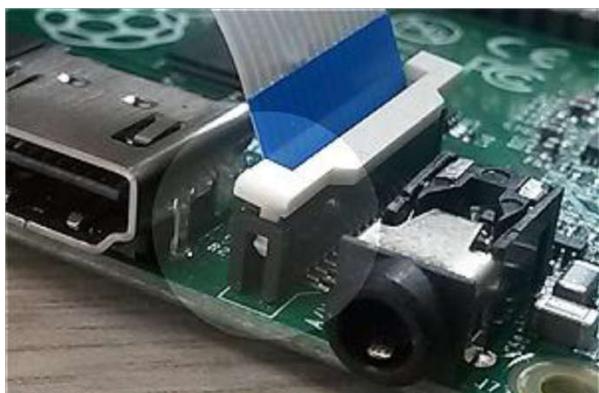
- 플렛케이블의 전극이 노출된 면이 HDMI 커넥터를 향하도록 연결

아래 사진들을 참고하여 견고히 연결한다.



Release

Lock

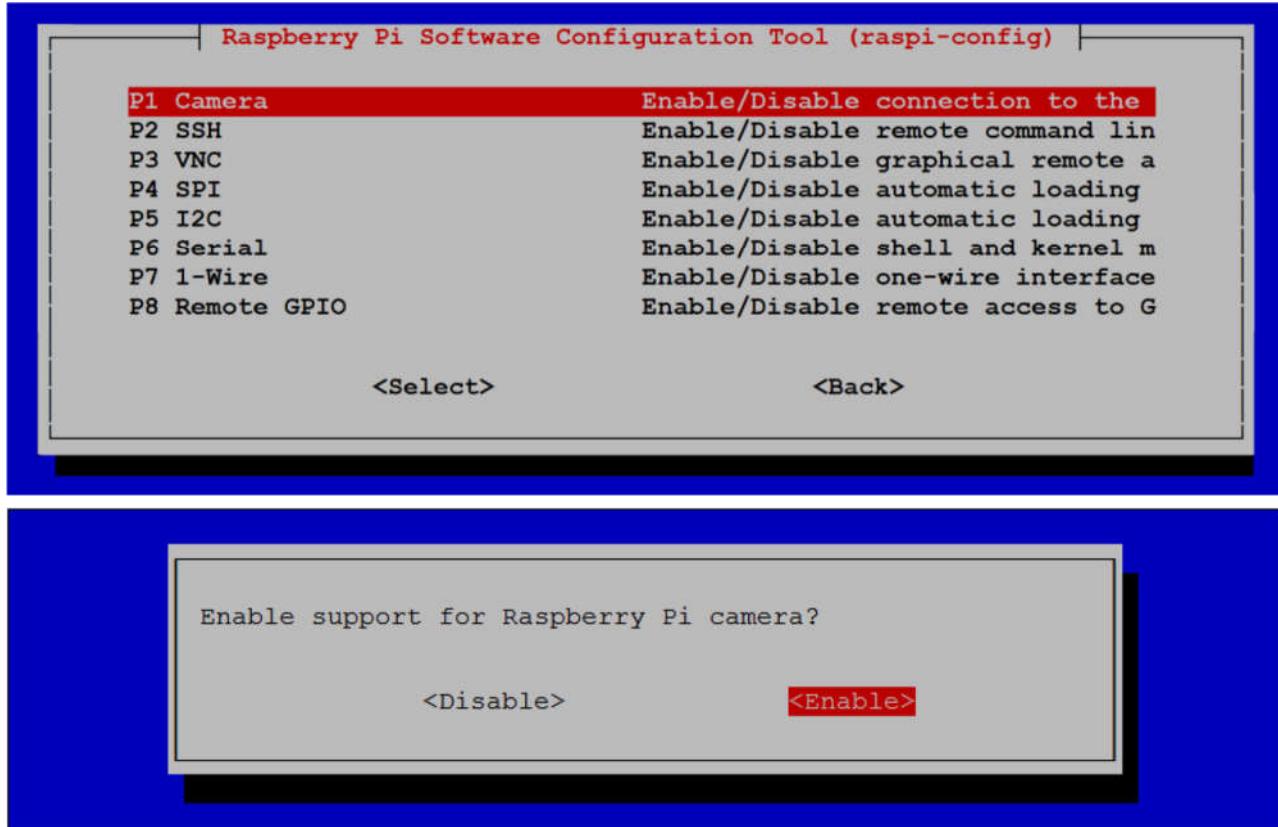


## 7.2. Pi Camera 인터페이스 활성화

raspi-config 실행

```
pi@raspberrypi:~ $ sudo raspi-config
```

raspi-config 의 메인 메뉴 중, “5. Interfaceing Options” 의 “P1. Enable camera” 에서 “<Enable>”선택



라즈베리파이가 리부팅 되고 나면, 카메라 인터페이스가 활성화 된다.

## 7.3. Pi Camera 동작 테스트

스틸 이미지 캡춰 테스트

```
pi@raspberrypi:~ $ raspistill -o img001.jpg
```

연결된 모니터에 5 초간 preview 후, 마지막 영상을  
스틸샷으로 캡춰 후, 파일명 "img001.jpg"로 저장.

파일명을 "~/Desktop/img001.jpg"와 같이 줄 경우  
바탕화면에 저장할 수도 있다.



## 동영상 Capture 테스트

```
pi@raspberrypi:~ $ raspivid -o ~/Desktop/video01.h264 -fps 30 -t 10000
```

초당 프레임수 30 프레임으로 10,000ms( 10 초 ) 동안 녹화하여 video01.h264 로 저장.

```
pi@raspberrypi:~ $ sudo apt-get install gpac
```

H264 형식을 mp4 형식으로 변환하기 위한 MP4Box 설치.

```
pi@raspberrypi:~ $ MP4Box -add ~/Desktop/video01.h264 ~/Desktop/video01.mp4
```

video01.h264 파일을 video01.mp4 로 형식변환.

## 7.4. Python 라이브러리를 이용한 Pi Camera 제어

### 7.4.1. 스틸 이미지 캡춰 파이썬 코드 1

모니터에 5 초간 카메라 프리뷰 후, 마지막 영상을 Capture 하여 바탕화면에 파일명 “image01.jpg”로 저장하는 코드 picamex01.py 작성.

- [https://github.com/greattoe/raspberrypi/tree/master/02\\_piCamera/picamex1.py](https://github.com/greattoe/raspberrypi/tree/master/02_piCamera/picamex1.py)

```
01 from picamera import PiCamera  
02 from time import sleep  
03  
04 camera = PiCamera()  
05  
06 camera.start_preview()  
07 sleep(5)  
08 camera.capture('/home/pi/Desktop/image01.jpg')  
09 camera.stop_preview()
```

### 7.4.2. 스틸 이미지 캡춰 파이썬 코드 2

GPIO( 스위치 ) 입력을 받아 이미지 캡처하는 파이썬 코드 picamex02.py 를 작성해보자

- GPIO 입력회로는 앞서 “GPIO 입력 테스트에서 사용한 스위치 연결” 을 사용하여 실습한다.

- [https://github.com/greattoe/raspberrypi/tree/master/02\\_piCamera/picamex2.py](https://github.com/greattoe/raspberrypi/tree/master/02_piCamera/picamex2.py)

```
01 import RPi.GPIO as GPIO  
02 from picamera import PiCamera  
03  
04 GPIO.setmode(GPIO.BCM)  
05 GPIO.setup(18, GPIO.IN)  
06  
07 camera = PiCamera()
```

```

08 try:
09     camera.start_preview()
10
11     GPIO.wait_for_edge(18, GPIO.RISING)
12
13     camera.capture('/home/pi/Desktop/image02.jpg')
14     camera.stop_preview()
15
16 except KeyboardInterrupt:
17     GPIO.cleanup()

```

### 7.4.3. 스틸 이미지 캡춰 파이썬 코드 3

GPIO( 스위치 ) 입력을 받으면 1 초간격으로 LED 를 점멸하다가, 5 초 후 카메라 이미지를 캡처하는 파이썬 코드

```

01 import time
02 import picamera
03 import RPi.GPIO as GPIO
04
05 GPIO.setmode(GPIO.BCM)
06 GPIO.setup(18, GPIO.IN)
07 GPIO.setup(23, GPIO.OUT)
08
09 print "Take picture by SW-input with 5 sec delay"
10
11 try:
12     # now use "camera" instead "picamera.PiCamera()"
13     with picamera.PiCamera() as camera:
14         camera.start_preview()
15         GPIO.wait_for_edge(18, GPIO.RISING)
16
17     for i in range(5): # for( i=0; i<5; i++ )
18         GPIO.output(23, True )
19         time.sleep(0.5)
20         GPIO.output(23, False)
21         time.sleep(0.5)
22
23     camera.capture('image.jpg')
24     camera.stop_preview()
25 except KeyboardInterrupt:
26     GPIO.cleanup()
27     print "\nProgram ended!"

```

## 7.5. MJPEG Streamer 를 이용한 네트워크 영상 스트리밍

라즈베리파이의 카메라 영상을 네트워크를 통해 스트리밍하는 방법은 여러가지가 있지만 가장 많은 오픈소스 코드를 찾아볼 수 있는 MJPEG-Streamer 를 이용하여 구현해보자

MJPEG Streamer 빌드에 필요한 의존성( git, cmake, libjpeg8-dev, imagemagick ) 설치

```
pi@raspberrypi:~ $ sudo apt-get install git cmake libjpeg8-dev imagemagick -y
```

videodev2.h 로 대체된 videodev.h 헤더 파일을 위한 sysbolic link 설정

```
pi@raspberrypi:~ $ sudo ln -s /usr/include/linux/videodev2.h /usr/include/linux/videodev.h
```

github에서 mjpg-streamer 소스코드 다운로드

```
pi@raspberrypi:~ $ git clone https://github.com/jacksonliam/mjpg-streamer
```

다운로드한 mjpg-streamer 소스코드 빌드 디렉토리로 경로 변경

```
pi@raspberrypi:~ $ cd ~/mjpg-streamer/mjpg-streamer-experimental
```

빌드 & 설치

```
pi@raspberrypi:~ $ make  
pi@raspberrypi:~ $ sudo make install
```

mjpg-streamer 실행 스크립트 작성

```
pi@raspberrypi:~ $ nano mjpg.sh
```

```
GNU  nano 2.7.4          File: mjpg.sh          Modified  
#!/bin/sh -e  
  
cd /home/pi/mjpg-streamer/mjpg-streamer-experimental/  
. ./mjpg_streamer -o "output_http.so -w ./www" -i "input_raspicam.so"  
  
^G  Get Help   ^O  WriteOut   ^R  Read File   ^Y  Cut Text   ^K  Cut Text   ^C  Cur Pos  
^X  Exit       ^J  Justify    ^W  Where Is    ^V  Next Page   ^U  Uncut Text  ^T  To Spell
```

작성한 스크립트 실행

```
pi@raspberrypi:~ $ nano mjpg.sh
```

같은 네트워크에 연결된 단말(PC, 스마트폰 등)에서 웹브라우저를 실행 후, <http://라즈베리파이의 IP 주소:8080> 를 연다.

