

라즈베리파이 종결 키트

Hands-on Experience with Raspberry Pi



[Free PDF Manual]

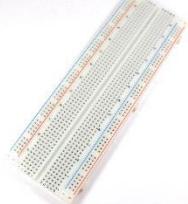
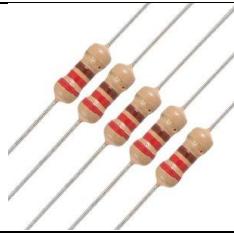
2015.04.29

라즈베리파이 종결 키트를 구성하며 ..

과거 1990년 초기에는 인텔 486 컴퓨터를 구입하기 위해 소형 자동차 가격에
버금가는 비용을 지불해야 했지만, 현재 비슷한 성능의 컴퓨터를 구입하기
위해서는 책 한권의 비용이면 충분할 정도로 마이크로 프로세서의 가격은
저렴해졌습니다. 라즈베리파이의 등장으로 인해서 과거의 커다란 컴퓨터 사이즈는
지갑만한 사이즈로 작아졌으며 누구나 쉽게 사용할 수 있는 사용자 중심의
인터페이스를 갖추고 있습니다. 메카솔루션에서는 누구나 쉽게 재미있게 따라하고,
배우고, 적용할 수 있는 플랫폼에 대한 연구와 컨텐츠 개발에 지금 이 시각에도
힘쓰고 있으며, 하드웨어와 접목된 프로그래밍을 배울 수 있는 좋은 플랫폼으로
라즈베리파이를 선정하여 키트를 준비하였습니다. 라즈베리파이 자체로도 리눅스
컴퓨팅과 프로그래밍 등 소프트웨어를 배울 수 있는 부분도 있지만, 다양한 센서와
카메라 모듈, 그리고 기초 전기전자 부품들을 포함함으로써 피지컬 컴퓨팅을
활용한 보다 인터랙티브한 교육을 제공하기 위해서 필수적이거나 많이 사용되는
부품들을 구성하였습니다. 지면상의 제약으로 인해 본 문서에서 다뤄지지 않은
부분들은 메카솔루션의 블로그, 기술상담 게시판 혹은 전화로 해결하실 수 있을
것이며, 전반적으로 라즈베리파이 활용에 대하여 주안점을 두었습니다. 거듭하면서
컨텐츠에 대해서는 업데이트가 될 예정이며, 무료로 다운로드 받아서 사용하실 수
있습니다.

- 메카솔루션 일동 드립 -

- 라즈베리파이 종결 키트 구성품 -

			
라즈베리파이 2	브레이드보드 830핀	2A 5V 어댑터	8GB MicroSD
			
코블러 브레이크 아웃보드	40핀 GPIO 리본케이블	64핀 브레이드보드용 점퍼와이어	Red LED x 5
			
Green LED x 5	Yellow LED x 5	220옴 저항 x 10	1K옴 저항 x 10
			
10K옴 저항 x 10	중형 푸쉬 버튼 x 5	HC-SR04 초음파 센서	DHT11 온습도 센서

			
TMP36 온도센서	HMC5883L 자기장 센서	SG90 미니 서보 모터	라즈베리파이 2 케이스
			
수분센서 (식물 재배용)	이더넷 네트워크 케이블	MPU6050 자이로센서	조도센서
			
MCP3008 ADC	3.3V to 5V 로직 컨버터 (옵션)	SD 메모리 카드 리더기 (옵션)	와이파이 동글 (옵션)
			
HDMI-DVI 케이블 (옵션)	HDMI-RCA 케이블 (옵션)	키보드 마우스 (옵션)	라즈베리카메라 (옵션)

옵션 아이템은 기존의 라즈베리파이 종결키트에 포함되어 있는 상품이 아니며,

메카솔루션 홈페이지 (<http://Mechasolution.com>)에서 별도로 구매가 가능합니다.



목 차

**1. 라즈베리파이의 기본**

A. 라즈베리파이 소개	P. 8
B. 하드웨어 준비 및 OS (Raspbian) 설치하기	P. 9
C. 와이파이 연결하기	P. 15
D. SSH를 통한 라즈베리파이 원격제어.....	P. 17
E. Xrdp를 통한 라즈베리파이 원격제어.....	P. 23
F. 라즈베리파이 환경설정하기.....	P. 25

2. 라즈베리파이와 리눅스

A. 리눅스 기본 명령어들	P. 28
B. 리눅스의 파일 에디터	P. 33
C. 리눅스로 프로그래밍하기	P. 37

3. 라즈베리파이와 Mathematica, Wolfram, Scratch

A. Mathematica로 프로그래밍하기	P. 39
B. Wolfram 커맨드 라인 사용하기	P. 41
C. 그래프 출력하기	P. 43
D. Scratch로 비쥬얼 프로그래밍하기	P. 44

4. 라즈베리파이와 파이썬

A. 파이썬 소개.....	P. 48
B. 파이썬 기초.....	P. 49
C. 연산자와 수식.....	P. 50
D. 반복 및 조건문.....	P. 51
E. 함수.....	P. 58
F. 모듈 및 패키지.....	P. 63

5. 라즈베리파이와 GPIO

A. 라즈베리파이 GPIO핀 기초.....	P. 65
B. PWM 서보모터 제어하기 (C 파이썬),..	P. 66
C. 초음파 거리센서 사용하기 (C 파이썬)	P. 73
D. DHT11(온습도 센서) 사용하기 (C)	P. 78
E. ADC를 이용해 조도센서 사용하기 (파이썬).....	P. 82
F. ADC를 이용해 온도센서 사용하기 (파이썬).....	P. 85

6. 라즈베리파이와 Node.js

A. Node.js란?	P. 88
B. LED 제어하기	P. 90
C. 웹페이지를 사용하여 Hello World 출력하기	P. 92
D. 웹페이지를 사용하여 LED 제어하기	P. 95
E. 시리얼 통신 사용하기	P. 100
F. Dynamixel 서모 모터 제어하기	P. 108

7. 라즈베리파이와 카메라 영상처리

A. 디지털 영상처리	P. 114
B. OpenCV와 Python	P. 115
C. OpenCV 설치	P. 116
D. 파이카메라를 활용한 실시간 스트리밍	P. 119
E. 영상처리 기본실습	P. 119

8. 참고문헌

P. 125

1. 라즈베리파이의 기본

A. 라즈베리파이 소개

라즈베리파이는 아두이노와 오픈소스계의 양대산맥이라 불리며 많은 개발자들 및 하비스트들에게 널리 알려져 있는 하드웨어 플랫폼입니다. 아두이노와 마찬가지로 손바닥만한 사이즈이지만, 이더넷 포트, USB 포트, HDMI 포트 등 일반 컴퓨터처럼 사용할 수 있음과 동시에 GPIO 핀들을 통해서 마이크로프로세서처럼 디지털 입출력을 지원함으로써, 사물인터넷 애플리케이션, 로보틱스, 스마트 홈 등 네트워크와 프로세싱이 필요로 하는 곳에서 광범위하게 사용되고 있습니다. 라즈베리파이의 역사에 대하여 간단히 설명을 하자면, 2006년에 영국에서 아트멜의 재단 이사인 에반 유톰이 학생들과 교육자들에게 컴퓨터에 대한 영감 및 프로그래밍 교육을 위해서 라즈베리파이 재단을 설립하였고, 2012년에 처음으로 판매가 되기 시작하였습니다. 이후, 프로세싱 및 인터페이스의 개선으로 현재 라즈베리파이 2가 판매되고 있습니다.



라즈베리파이는 피지컬 컴퓨팅 뿐만 아니라 훌륭한 그래픽 성능으로 게임개발 플랫폼으로도 사용될 수 있으며, XBMC라는 미디어센터로도 활용이 가능합니다. 라즈비안 (Raspbian)이라는 리눅스의 데비안을 기반으로 한 컴팩트하고 가벼운 운영체제를 무료로 설치할 수 있습니다.

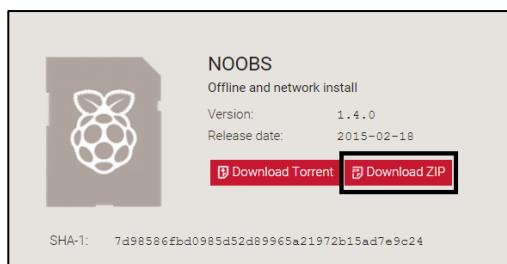
B. 하드웨어 준비 및 OS (Raspbian) 설치하기

일반적으로 CD를 사용하여 운영체제를 설치하는 고전적인 방법과는 달리, 라즈베리파이는 NOOBS라는 파일을 SD 카드 (A) 혹은 MicroSD 카드 (B+)에 다운로드 받은 후에 설치할 수 있습니다. NOOBS는 New Out Of the Box Software의 약자로서 쉽게 운영체제의 설치를 돋기 위해서 만들어진 소프트웨어입니다.

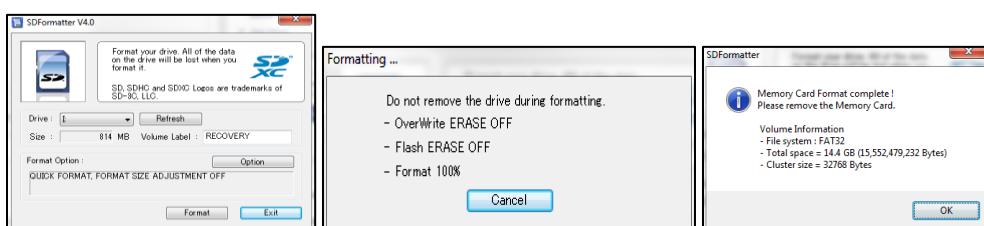
운영체제가 이미 SD카드에 저장되어 있는 것을 구입하여 라즈베리파이에 설치할 수도 있지만, 한국과 같이 초고속 인터넷이 가능한 곳에서 다운로드 받고 설치해보는 것도 도움이 될 것 같습니다. 이미 다운로드 받은 파일이 있다면 SD카드를 포맷한 후에 복사하여 사용하는 것도 좋은 방법입니다.

[NOOBS로 OS 설치하기]

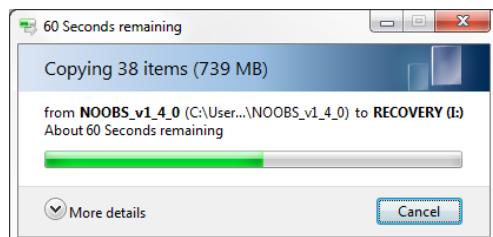
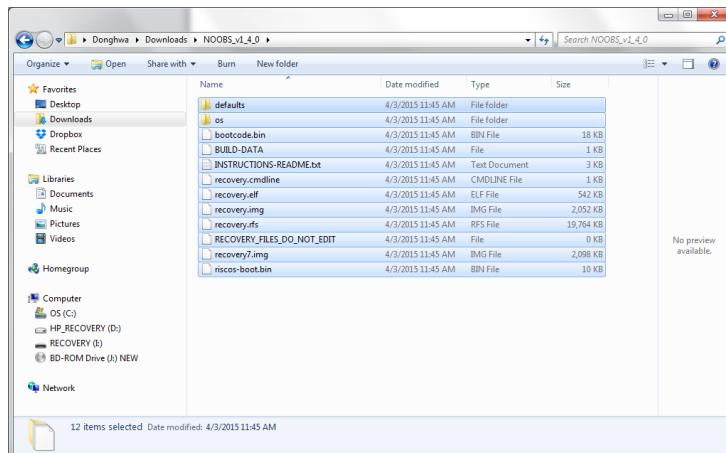
1. <http://www.raspberrypi.org/downloads/> 의 사이트에 접속하여 최신버전의 NOOBS.ZIP를 다운로드 받아줍니다.



2. 라즈베리파이에 연결해 사용할 Micro SD 카드를 포맷한 후, 다운로드 받은 NOOBS 압축 파일을 풀어 복사해 넣습니다.



[Micro SD Card 포맷]



[NOOBS 압축 파일을 풀어 Micro SD 카드로 복사하기]

3. NOOBS 파일들이 마이크로 SD카드에 복사가 완료되면, 라즈베리파이의 뒷면에 있는 마이크로SD카드 소켓에 다음과 같이 마이크로SD카드의 로고가 보이는 방향으로 끼워넣습니다 (딸깍 소리가 날 때까지).

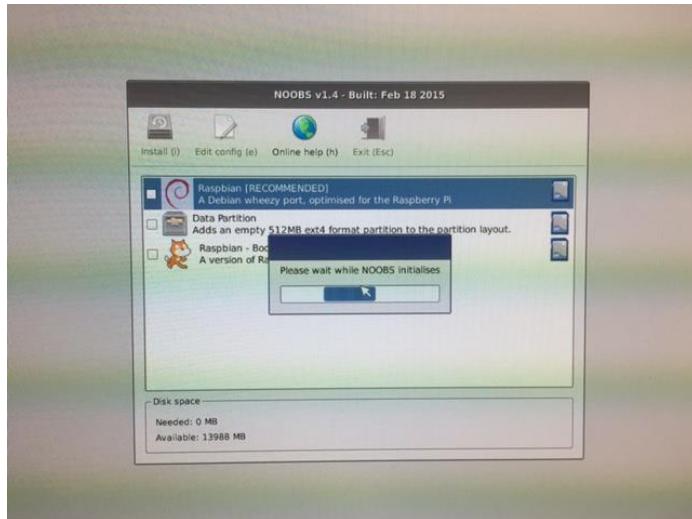


여기서 잠깐!

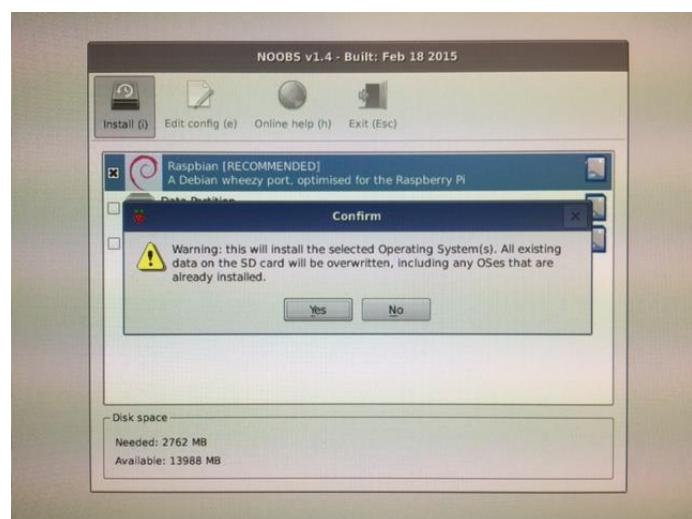


예고치 못한 정전기 및 물 엎지름등과 같은 이유로 보드에 치명적인 손상을 가할 수 있기 때문에, 케이스를 사용하는 것을 추천합니다. 구버전인 라즈베리파이 B형과 같은 경우는 USB 포트가 2개뿐이기 때문에, 라즈베리파이 B형 전용 케이스를 사용하시면 USB 포트 4개를 사용하는 라즈베리파이 B+ 및 2에 맞지 않습니다. 따라서 케이스는 라즈베리파이 B+ 혹은 2 전용을 사용해주십시오. 케이스에 안전히 들어간 라즈베리파이의 USB 단자에 키보드, 마우스를 연결하고, HDMI 케이블을 이용하여 모니터에 연결합니다. HDMI-HDMI를 사용하실 수도 있고, HDMI-DVI 케이블을 사용하여 모니터와 연결할 수 있습니다. 그리고, 마이크로USB 단자 (HDMI케이블 왼쪽)을 사용하여 5V 전원을 공급합니다. 라즈베리파이를 구동하기 위해서는 적어도 1A의 전류가 필요하므로, 5V 1A, 5V 1.5A, 혹은 5V 2A를 사용하시면 됩니다. 라즈베리파이는 별도의 전원 스위치가 없으므로, 5V 어댑터를 바로 연결하게 되면 전원이 들어오게 됩니다. 스위칭이 가능한 멀티탭을 이용하면 편리하게 라즈베리파이의 전원을 껐다 켰다할 수 있습니다.

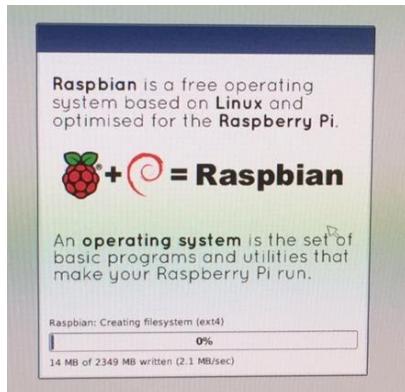
4. 라즈베리에 전원을 공급하게 되면, Micro SD 카드를 통해서 설치되어 있는 파일을 검색하며 다음의 창을 띄우게 됩니다.



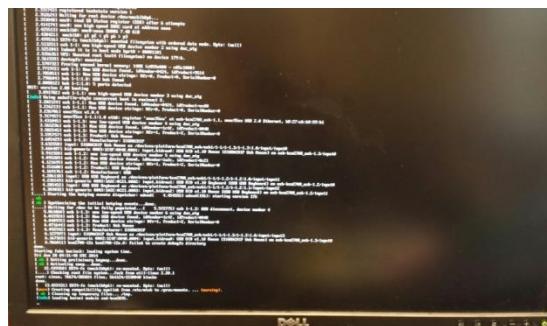
5. Micro SD 카드 내의 다양한 OS 설치 옵션이 있으며, 99%가 사용한다는 Raspbian을 설치해봅니다.



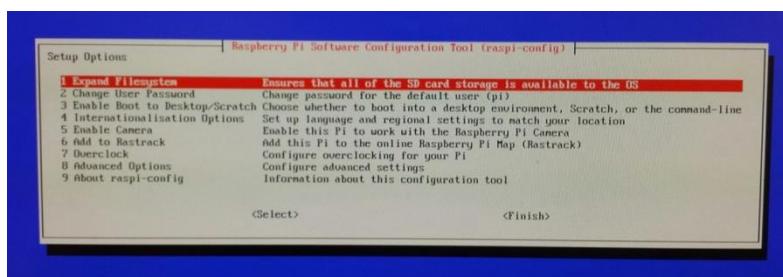
- 설치하게 되면, SD 카드의 데이터가 손실될 수 있다는 경고 메시지가 뜨지만, 처음 설치하는 것이기 때문에 Yes를 클릭하고 진행합니다.

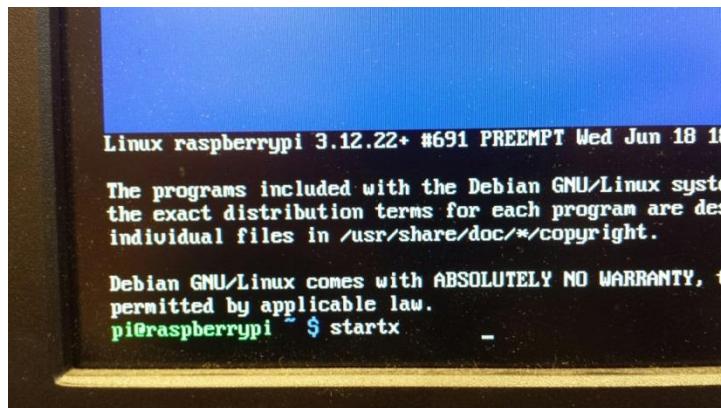


- 10~30분에 걸쳐서 설치가 진행됩니다.

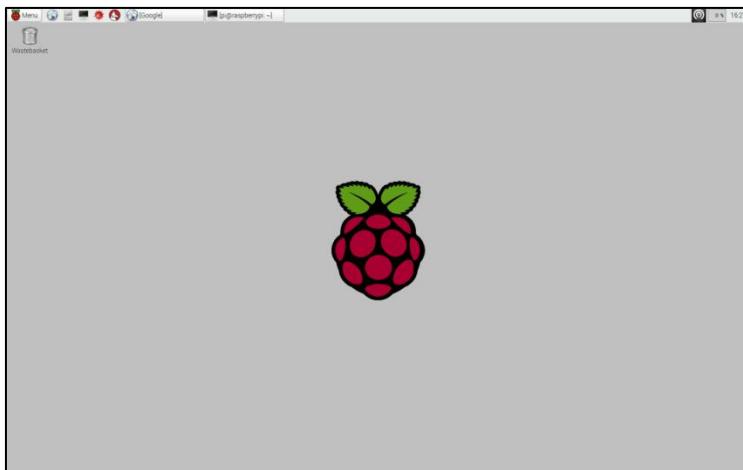


- 기본 설정창입니다 필요에 따라 설정해줄 것으로 지금은 <Finish> 를 눌러 줍니다. Tab키를 이용하여 이동이 가능합니다.





9. pi@raspberrypi ~\$와 같이 프롬프트가 뜨면, "startx" 입력하여 Raspbian이 제공하는 Graphical User Interface (GUI)를 실행하게 됩니다. 그러면 다음과 같은 Raspbian OS의 초기 데스크탑 화면이 보이게 됩니다.



10. Menu 버튼을 누르면, Programming, Internet, Games, Accessories, Help, Preferences, Run, Shutdown의 메뉴가 있으며 각각 서브 메뉴가 있습니다. Programming 메뉴에서는 수학관련 프로그래밍 인터페이스인 Mathematica와 파이썬 개발환경, 비쥬얼 프로그래밍 툴인 스크래치 (Scratch), 오디오 프로그래밍인 Sonic Pi, 그리고 컴퓨터이션 엔진으로 많이 사용되는 Wolfram이 있습니다.

C. 와이파이 연결하기

- Preferences 메뉴로 가면 WiFi Configuration이라는 서브 메뉴가 있는데, 이를 통해서 라즈베리파이를 와이파이에 연결할 수 있습니다. 먼저, USB 형태의 와이파이 동글을 라즈베리파이에 끼웁니다.



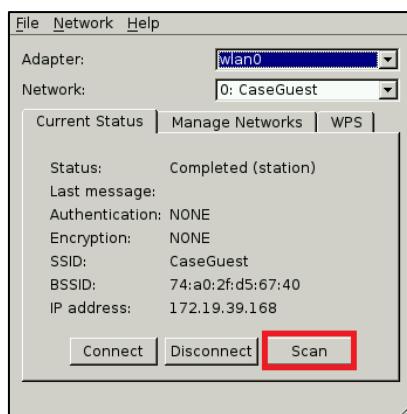
라즈베리파이 호환 WiFi 동글

상품 링크:

http://mechasolution.com/shop/goods/goods_view.php?goodsno=329&category=006005

- WiFi Config 아이콘  을 실행해주면,

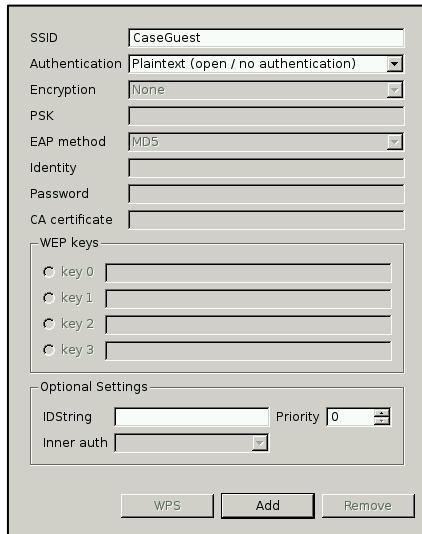
아래와 같은 와이파이 설정 윈도우를 확인할 수 있습니다.



- Scan 버튼을 누르게 되면 연결 가능한 와이파이 신호를 검색하게 됩니다. 그리고 사용할 와이파이를 선택한 후에, 필요에 따라 암호를 넣고 사용할 수 있습니다.

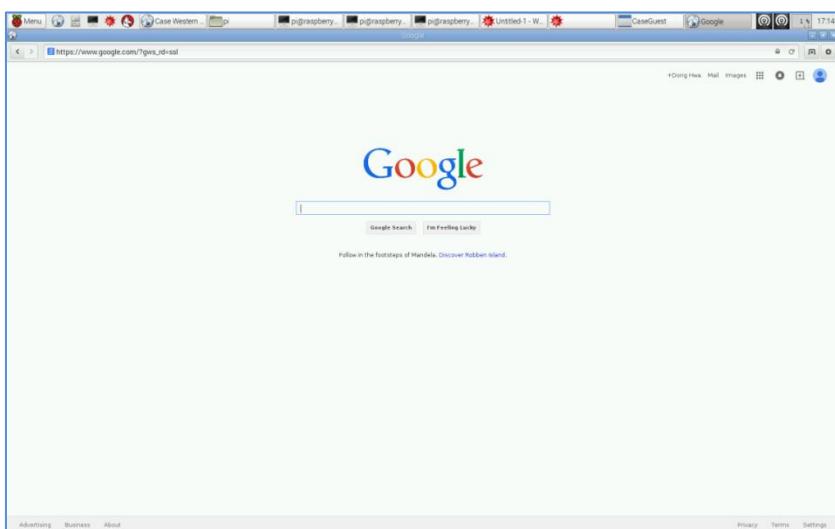
SSID	BSSID	frequency	signal	flag
HP784C16	02:bfb:3d:...	2457	-183 dBm	[IBS]
CaseWireless	74:a0:2f:d5:...	2462	-180 dBm	[WP]
CaseGuest	74:a0:2f:d5:...	2462	-181 dBm	[ES]

4. 암호가 필요없는 경우에는 다음과 같이 사용할 수 있으며, 해당되는 Authentication 방법에 따라서 암호를 입력해서 사용할 수 있습니다.



Add 버튼을 누르고 다시 Connect를 누르게 되면 초기 와이파이 구성 원도우의 Status에 Completed라는 것을 확인할 수 있으며, IP 주소가 업데이트된 것을 체크할 수 있습니다.

5. 와이파이 연결이 잘 되었는지 브라우저를 실행하여 확인해봅니다.



D. SSH를 통한 라즈베리파이 원격제어

기존에 사용하던 컴퓨터에서 라즈베리파이를 원격제어 하게 됨으로써, 키보드, 마우스, 모니터 등의 추가 구성요소 줄일 수 있습니다. 원격제어 설정 후 인터넷 (유선 또는 무선) 연결과 전원 공급만 해주면 되기 때문에 배선정리, 공간, 비용 등에 있어 효율적입니다. SSH (보안 셸 : Secure Shell)는 원격 컴퓨터에 연결하는 가장 일반적인 방법으로, 라즈베리파이에서 사용하는 모든 명령어가 SSH에서 가능하며, 통신을 암호화하기 때문에 보안이 뛰어납니다. 특성이자 단점이라면 SSH는 CLI 환경(텍스트 환경)이라는 점입니다.



이를 실행하기 위해서는 다음의 설정 단계를 거쳐야 합니다.

1. 라즈베리파이에서의 설정 (라즈베리파이에서 SSH 기능 사용을 가능하게 설정해주고 라즈베리파이의 IP를 찾아줍니다.)
2. 윈도우에서의 설정 (Putty라는 프로그램을 설치하여 라즈베리파이의 IP를 입력해줍니다.)

1. 라즈베리파이에서의 설정

(1) 라즈베리파이를 인터넷 (무선 or 유선) 에 연결해줍니다.

(2) LX Terminal 을 실행해준 후 다음 명령어를 입력합니다.

라즈베리파이 환경설정 창

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

(3) 8. Advanced Option → A4 SSH → Enable 를 선택해줍니다.
(최신 버전의 라즈비안에서는 SSH 가 자동으로 활성화 되어있습니다.)

(4) <Finish>를 눌러 다시 커맨드창으로 나온 후 다음 명령어를 입력합니다.

라즈베리파이 IP확인

```
pi@raspberrypi ~ $ ifconfig
```

유선인터넷이라면 eth0 부분의 inet addr 를,
무선인터넷이라면 wlan0 의 inet addr 을 체크합니다.

```
pi@raspberrypi: ~
eth0      Link encap:Ethernet HWaddr b8:27:eb:82:7a:c5
          UP BROADCAST MULTICAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)   TX bytes:0 (0.0 B)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)   TX bytes:0 (0.0 B)

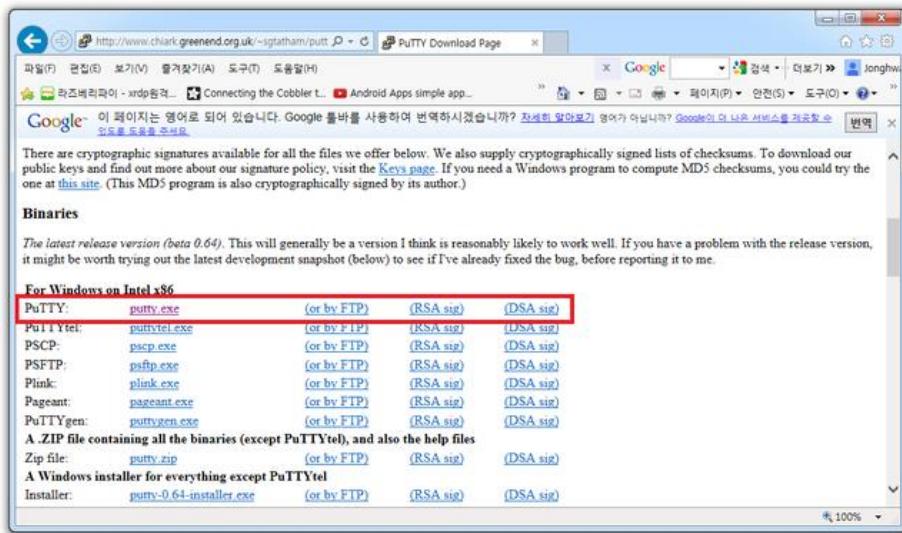
wlan0    Link encap:Ethernet HWaddr 00:0b:81:94:f7:a8
          inet addr:192.168.120.30 Bcast:192.168.120.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:98 errors:0 dropped:6 overruns:0 frame:0
          TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:173468 (169.4 KiB)   TX bytes:29300 (28.6 KiB)

pi@raspberrypi ~ $
```

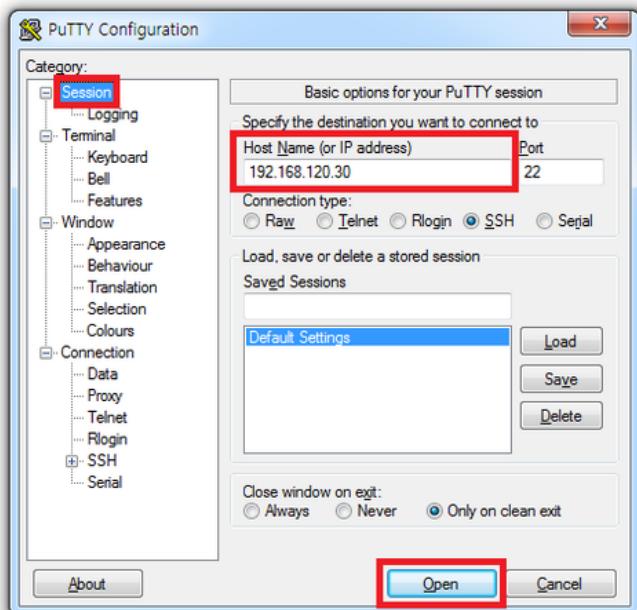
2. 윈도우에서의 설정

(1) 아래의 링크에서 PuTTY를 다운로드 받아 실행해줍니다.

<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>



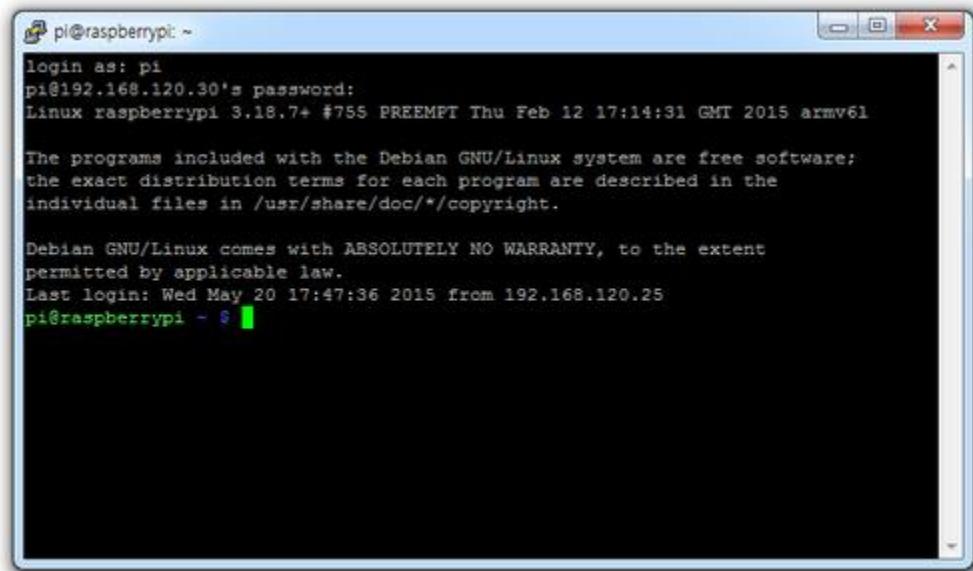
(2) PuTTY의 Session – Host Name(or IP address)에 체크해둔 라즈베리파이의 IP를 입력한 후 Open을 눌러줍니다.



- (3) 아래와 같은 프로토콜이 나타나면 라즈베리파이의 아이디를 입력, 그 후 패스워드를 입력해주도록 합니다.
(초기의 라즈베리파이 ID / Password : pi / raspberry)



- (4) 오류가 발생하지 않았다면 아래와 같은 화면이 등장하며, 연결이 성공됩니다.

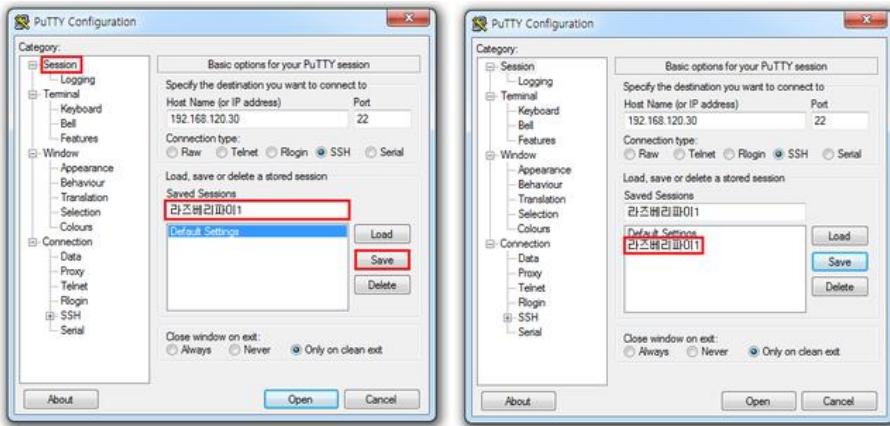


3. PuTTy 환경설정

PuTTy에는 Session, Terminal, Window, Connection 등의 여러가지 기본설정이 있습니다. 특별한 사항이 아니라면 초기 설정 그대로 사용하셔도 무방하지만, 편의를 위해 몇가지 설정에 대해 알아보도록 하겠습니다.

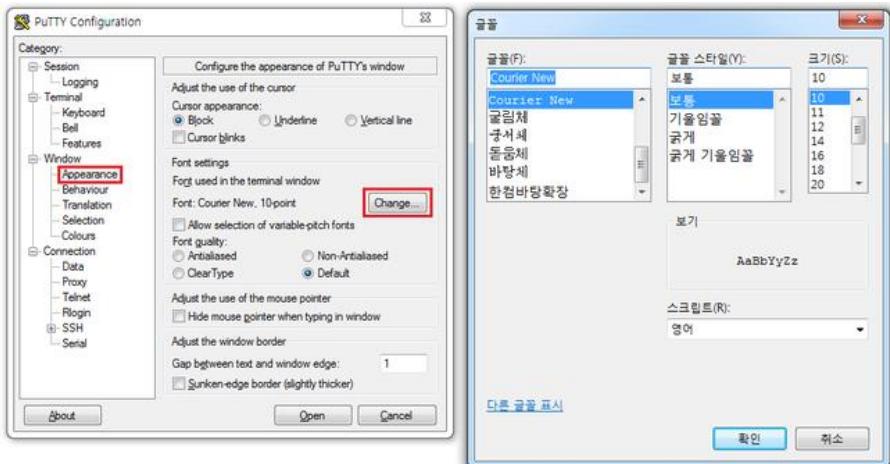
(1) 설정 저장하기

변경한 설정들을 저장하여, 불러올 수 있습니다.



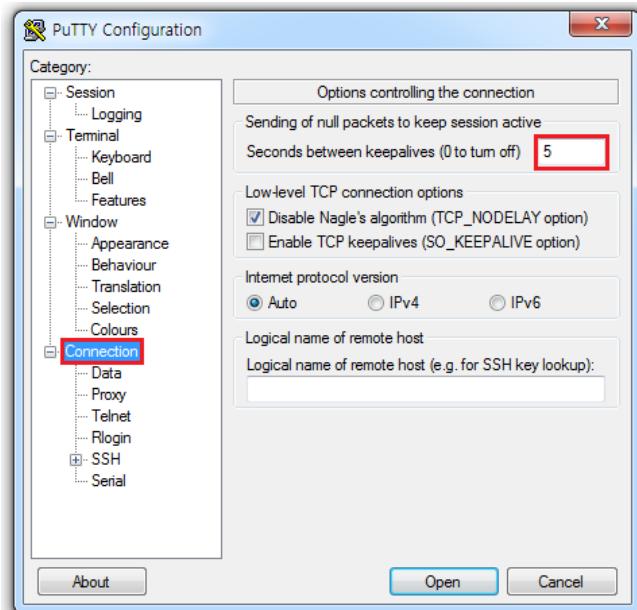
(2) 글씨체, 크기 변경하기

커맨드라인에 입력되는 글씨체나 크기를 변경할 수 있습니다.



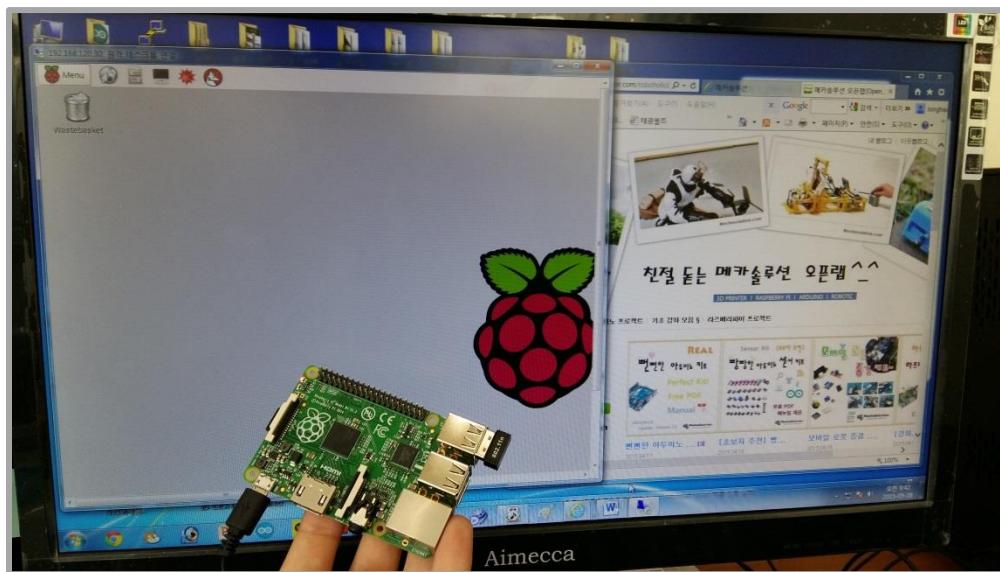
(3) 자동 연결 종료 시스템 Off하기

라즈베리파이를 연결한 후, 커맨드라인에 일정시간동안 명령어를 입력해주지 않으면 연결이 자동으로 종료됩니다. 설정을 하나 변경하여 자동 종료를 막아줍니다. (입력된 시간마다 null 패킷을 보냄)



E. Xrdp를 통한 라즈베리파이 원격제어.

Xrdp(X-Remote Desktop Protocol)은 커맨드 라인의 그래픽을 지원하는 SSH와 다르게 라즈베리파이의 GUI 그래픽 환경을 지원하는 원격제어 방법입니다. 간단하게 시도하여 연결할 수 있지만, 속도가 다소 느려진다는 단점이 있습니다.



이를 실행하기 위해서는 다음의 설정 단계를 거쳐야 합니다.

1. 라즈베리파이에서의 설정

(라즈베리파이의 IP를 찾아주고, xrdp를 설치 한 후 xrdp 데몬 실행)

2. 윈도우에서의 설정

(딱히 설정해줄 것은 없고, 프로그램을 실행하여 IP를 입력해주면 완료)

1. 라즈베리파이에서의 설정

(1) 라즈베리파이의 IP를 체크합니다. (매뉴얼 P. 18 참고)

(2) Xrdp를 설치합니다.

LX Terminal을 실행하여 아래의 명령어를 차례대로 입력합니다.

xrdp 설치하기

```
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get upgrade
pi@raspberrypi ~ $ sudo apt-get install xrdp
```

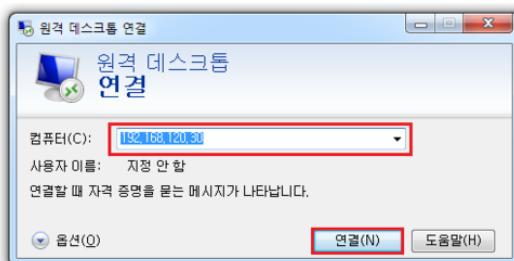
(3) 설치가 완료되면 xrdp 데몬을 실행해줍니다.

시작 – Run – 실행창에 mstsc 입력. (이 후 재부팅시 자동실행)

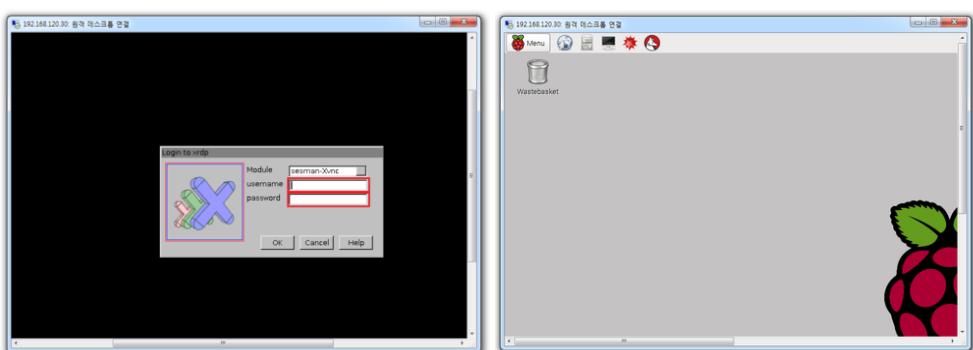
2. 윈도우에서의 설정

(1) 시작을 누른 후 검색창을 이용해 ‘원격 데스크톱 연결’을 실행해 준 후

표시창에 라즈베리파이의 IP주소를 입력, 연결 해줍니다.



(2) 라즈베리파이의 ID와 Password를 입력해주면 연결이 완료됩니다.



F. 라즈베리파이 환경설정하기

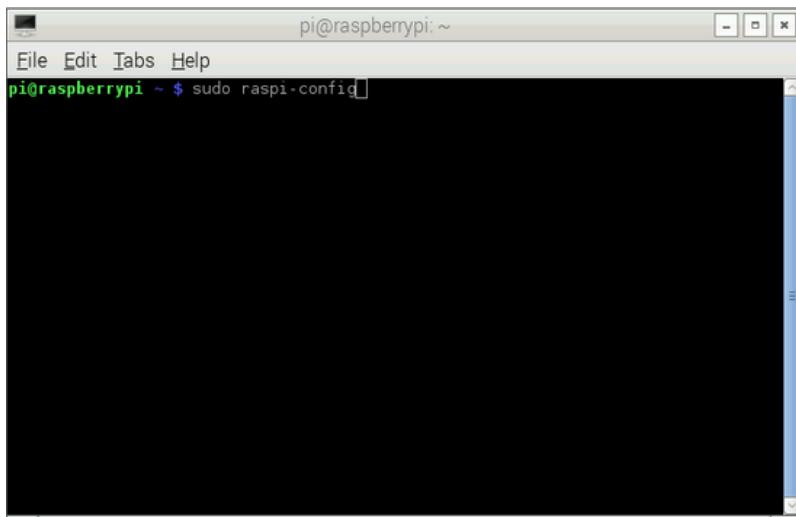
라즈베리파이를 자신의 입맛대로 사용하기 위해서는 환경설정창을 이용하여 설정을 변경해줘야 합니다.

- 터미널 (LXTerminal) 을 실행해주세요.

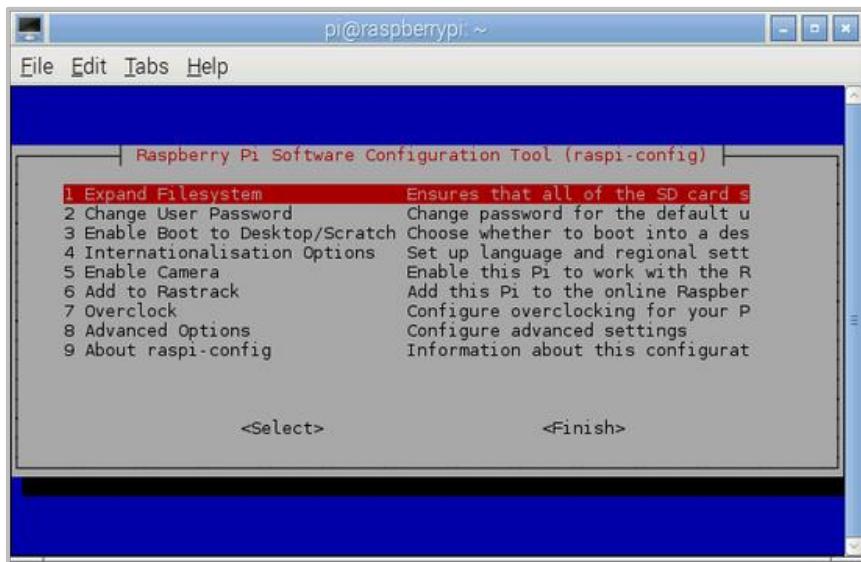


- 다음 명령어를 입력해주세요.

```
라즈베리파이 환경설정 창  
pi@raspberrypi ~ $ sudo raspi-config
```



3. 라즈베리파이 소프트웨어 환경 설정창이 등장합니다. 왼쪽은 설정창 네임, 오른쪽은 설정창에 대한 설명입니다. 순서대로 어떤 기능인지 알아보도록 하겠습니다.



(1) Expand filesystem

SD 카드 사용 영역을 설정하기 위한 탭입니다. SD 카드의 모든 영역을 사용하기 위해서 반드시 설정해줘야하며, 탭을 선택하여 OK를 눌러주신 후 재부팅을 하면 적용됩니다.

(2) Change User Password

라즈베리파이의 비밀번호 수정 탭입니다. 운영체제 설치시 초기 비밀번호는 raspberry이며, 탭을 선택한 후 변경해주시면 됩니다.

(3) Enable Boot to Desktop/Scratch

라즈베리파이를 켰을 때, 데스크탑, 커맨드 라인, 스크래치 화면 중 어느 화면으로 부팅할지 선택이 가능합니다. 데스크탑으로 부팅시 아이디/비밀번호 입력 과정이 생략되므로 설정을 해두시면 간편해집니다.

(4) Internationalisation Options

나라(국제화) 설정입니다. 1) 언어, 2) 시간, 3) 키보드 레이아웃 설정이 가능합니다. 언어를 한국어로 변환 할 경우 화면이 깨지거나 오번역이 있을 수 있습니다.

한국 언어 설정 방법

(1) Change Locale → ko_KR.UTF-8 UTF-8

한국 시간 설정 방법

(2) Change Timezone → Asia → Seoul

(5) Enable Camera

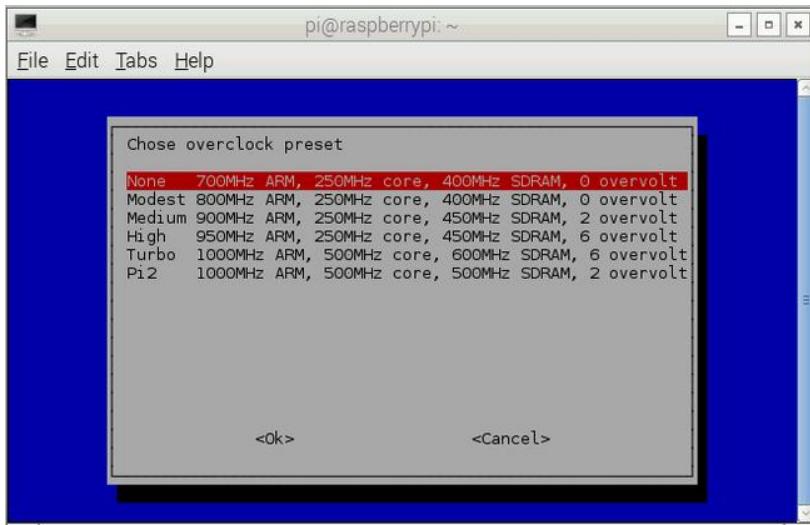
라즈베리파이 카메라 사용여부를 설정하는 탭입니다.

(6) Add to Rastrack

전 세계에 라즈베리파이 사용자의 분포를 알려주는 탭입니다.

(7) Overclock

말그대로 정해진 클럭을 오버 시키는 기능입니다. 사용하시면 속도가 빨라질 수는 있으나, 기기에 무리가 가기 때문에 발열이 생기고, 기기에 고장 가능성이 생깁니다. 방열판을 이용하시면 발열을 줄일 수 있습니다.



(8) Advanced Options : 고급설정 탭으로 10 가지의 설정이 가능합니다.

1. Overscan : 구형 디스플레이 장치에 맞는 화면을 설정해줍니다.
2. Hostname : 호스트네임을 변경해줍니다. 기본 호스트네임은 raspberrypi 입니다.
3. Memory split : GPU 메모리를 설정해줍니다.
4. SSH : SSH 사용에 관한 설정입니다. 원격제어에 유용합니다.
5. SPI : SPI 통신 사용여부를 설정해줍니다.
6. I2C : I2C 통신 사용여부를 설정해줍니다.
7. Serial : Serial 통신 사용여부를 설정해줍니다.
8. Audio : 사용할 오디오잭을 설정해줍니다.
9. Update : raspi-config 도구를 최신버전으로 업데이트 해줍니다.

(9) About raspi-config

raspi-config 툴에 대한 정보를 제공합니다.

4. 설정을 마친 후, TAB 키를 이용하여 Finish 를 눌러주면 설정이 종료됩니다.

2. 라즈베리파이와 리눅스

A. 리눅스 기본 명령어들

라즈베리파이에서 사용하는 Raspbian은 Debian을 기반으로 한 운영체제이기 때문에 리눅스 명령어를 알아두는 것이 라즈베리파이를 보다 효율적으로 사용할 수 있습니다. 모든 명령어를 외울 필요도, 외울 수도 없지만 자주 사용되는 명령어들을 익혀 두면 좋기 때문에 몇 가지를 다뤄보도록 하겠습니다.

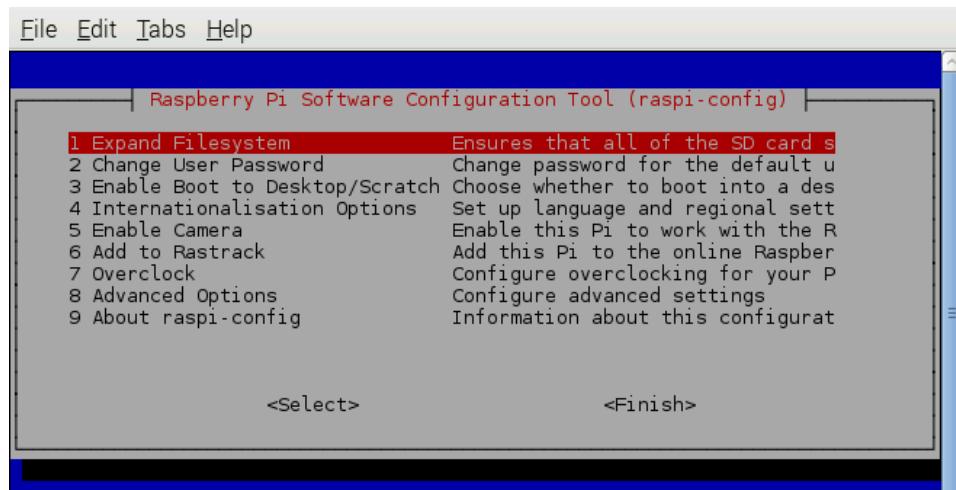
먼저, 운영체제를 처음에 설치할 때 나오지만, 비밀번호 변경, 라즈베리파이용 카메라의 설정, 언어 및 시간 설정 등을 필요할 때 다음과 같이 할 수 있습니다.



위의 모니터 모양의 아이콘을 클릭하면 검정색 바탕의 터미널이 열리게 되며, 커맨드(명령)를 입력하여 시스템 탐색, 시작, 설정 등을 처리할 수 있습니다.

다음과 같이 sudo raspi-config를 입력해서 시스템 설정 등을 변경할 수 있습니다.

라즈베리파이 소프트웨어 Configuration
pi@raspberrypi ~ \$ sudo raspi-config



여기서 sudo라는 것은 “substitute user do”의 약자로써, 라즈베리파이에서 최고 권한을 가지고 파일 인스톨, 실행 등을 할 수 있도록 하는 명령어입니다. 리눅스에서는 사용자의 권한에 따라서 가능한 것이 나뉘는데, sudo를 사용하게 되면 일종의 “슈퍼파워”를 가진 super user로 파일의 이동, 복사, 삭제 그리고 프로그램 설치 등이 가능하게 됩니다.

메뉴의 shutdown을 클릭해서 시스템을 재부팅하거나 종료할 수도 있지만, 다음 명령어로 OS를 재부팅하거나 종료 할 수 있습니다.

OS 재부팅

```
pi@raspberrypi ~ $ sudo reboot
pi@raspberrypi ~ $ sudo shutdown -r now
```

OS 종료

```
pi@raspberrypi ~ $ sudo halt
pi@raspberrypi ~ $ sudo shutdown -h now
```

이 때, Username과 Password를 묻게 되는데, 초기값은 다음과 같습니다.

Username: pi
Password: raspberry

리눅스를 사용하면서 가장 많이 사용하는 것 중에 하나가 프로그램 패키지 업데이트와 업그레이드입니다. 패키지들간의 의존성(Dependency)이 있어서 설치가 되지 않는 경우가 있기 때문에, 이를 해결하기 위해서 최선 버전으로 유지해줄 필요가 있습니다. `update`는 업데이트 내역을 확인하는 명령어이며 `upgrade`는 업데이트된 내 역을 설치합니다. `update`를 해서 내역을 확인하고 `upgrade`를 통해 다운받습니다.

프로그램 패키지 업데이트

```
pi@raspberrypi ~ $ sudo apt-get update
```

프로그램 패키지 업그레이드

```
pi@raspberrypi ~ $ sudo apt-get upgrade
```

만약, 프로그램 패키지를 설치하고 싶은 경우에는 해당되는 패키지 이름을 터미널에서 설치할 수 있습니다.

프로그램 패키지 설치

```
pi@raspberrypi ~ $ sudo apt-get install <패키지이름>
```

예를 들어, 스크린캡쳐를 할 수 있도록 `scrot`이라는 것을 설치하기 위해서는 `sudo apt-get install scrot`을 입력하면 설치가 됩니다. 그리고 터미널에서 `scrot`을 입력해서 현재 스크린을 캡쳐할 수 있고, 이는 홈 디렉토리인 `/home/pi`에 저장이 됩니다.

특정 원도우를 캡쳐하고 싶을 때에는 `scrot -cd 5 -u`를 사용하면 편리합니다.

여기서 `-u`는 활성화되어 있는 특정 원도우를 캡쳐하겠다는 파라미터이고, `-cd`는 카운트다운을 하겠다는 의미입니다.



File Manager를 이용하여 마우스로 검색 및 파일/폴더의 이동, 생성, 삭제 등을 할 수 있지만, 터미널을 사용하는 것이 점점 더 편리해질 수 있습니다.

아래는 폴더와 파일 등을 관리하는 리눅스 명령어입니다.
(디렉토리는 폴더와 같은 의미로 사용됩니다.)

현재의 디렉토리 위치 확인

```
pi@raspberrypi ~ $ pwd
```

현재 디렉토리 내용 확인

```
pi@raspberrypi ~ $ ls
```

디렉토리 만들기 (현재 폴더에 mechasolution이라는 디렉토리를 만듭니다.)

```
pi@raspberrypi ~ $ mkdir mechasolution
```

파일 및 디렉토리 이동하기 (mechasolution 디렉토리를 Desktop으로 이동합니다.)

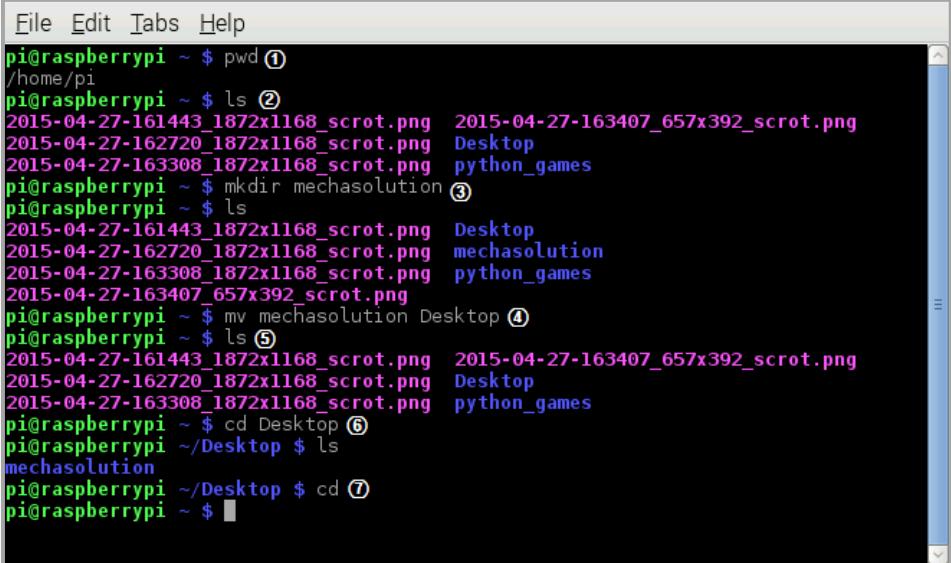
```
pi@raspberrypi ~ $ mv mechasolution Desktop
```

경로 변경하기

```
pi@raspberrypi ~ $ cd Desktop
```

홈 경로로 이동하기

```
pi@raspberrypi ~ $ cd
```



```

File Edit Tabs Help
pi@raspberrypi ~ $ pwd ①
/home/pi
pi@raspberrypi ~ $ ls ②
2015-04-27-161443_1872x1168_scrot.png 2015-04-27-163407_657x392_scrot.png
2015-04-27-162720_1872x1168_scrot.png Desktop
2015-04-27-163308_1872x1168_scrot.png python_games
pi@raspberrypi ~ $ mkdir mechasolution ③
pi@raspberrypi ~ $ ls
2015-04-27-161443_1872x1168_scrot.png Desktop
2015-04-27-162720_1872x1168_scrot.png mechasolution
2015-04-27-163308_1872x1168_scrot.png python_games
2015-04-27-163407_657x392_scrot.png
pi@raspberrypi ~ $ mv mechasolution Desktop ④
pi@raspberrypi ~ $ ls ⑤
2015-04-27-161443_1872x1168_scrot.png 2015-04-27-163407_657x392_scrot.png
2015-04-27-162720_1872x1168_scrot.png Desktop
2015-04-27-163308_1872x1168_scrot.png python_games
pi@raspberrypi ~ $ cd Desktop ⑥
pi@raspberrypi ~/Desktop $ ls
mechasolution
pi@raspberrypi ~/Desktop $ cd ⑦
pi@raspberrypi ~ $ █

```

위 사진에서 실행한 명령어를 살펴보도록 하겠습니다.

1. `pwd`를 통해 현재 작업중인 디렉토리를 확인.
2. `ls`를 통해 현재 디렉토리 안에 들어있는 내용을 확인.
3. `mkdir`을 사용하여 `mechasolution`이라는 디렉토리를 생성.
4. `mv`를 사용하여 `mechasolution` 디렉토리를 `Desktop` 디렉토리로 이동.
5. `ls`를 통해 현재의 디렉토리(`/home /pi`)에 `mechasolution` 디렉토리가 사라진것을 확인.
6. `cd Desktop`을 통해 `Desktop` 디렉토리로 이동 후, `ls`를 통해 `mechasolution` 디렉토리가 `Desktop` 디렉토리로 이동된 것을 확인.
7. `cd`를 통해 홈 경로로 이동.

이 밖에도 `cp`를 사용하여 복사하거나 `rm`을 통해서 삭제를 할 수 있습니다.

하지만 대부분의 경우는 윈도우 창을 통해 마우스 드래그 혹은 `Ctrl+C`, `Ctrl+V`로 작업할 수 있습니다. 터미널을 사용해야 하는 경우는 폴더나 파일에서 복사, 이동등의 권한이 없을 경우, `sudo mv`, `sudo cp`, `sudo rm` 등과 같이 슈퍼유저 권한으로 실행할 수 있습니다.

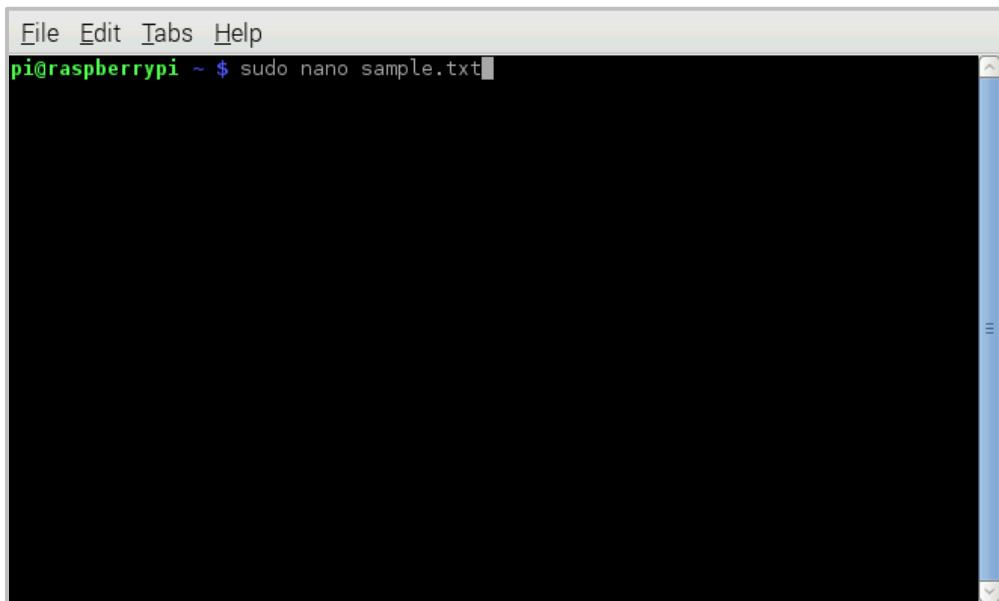
B. 리눅스의 파일 에디터

파일을 에디팅하기 위해서 리눅스에서는 VIM, NANO, VI, EMACS, GEDIT, PICO 등을 사용할 수 있습니다. 라즈베리파이에는 vi, nano, pico가 기본으로 설치되어 있습니다. 각각 텍스트 에디터와 같은 형태이기 때문에 vi [파일이름], nano [파일이름], pico [파일이름]을 터미널에서 입력해서 사용할 수 있습니다.

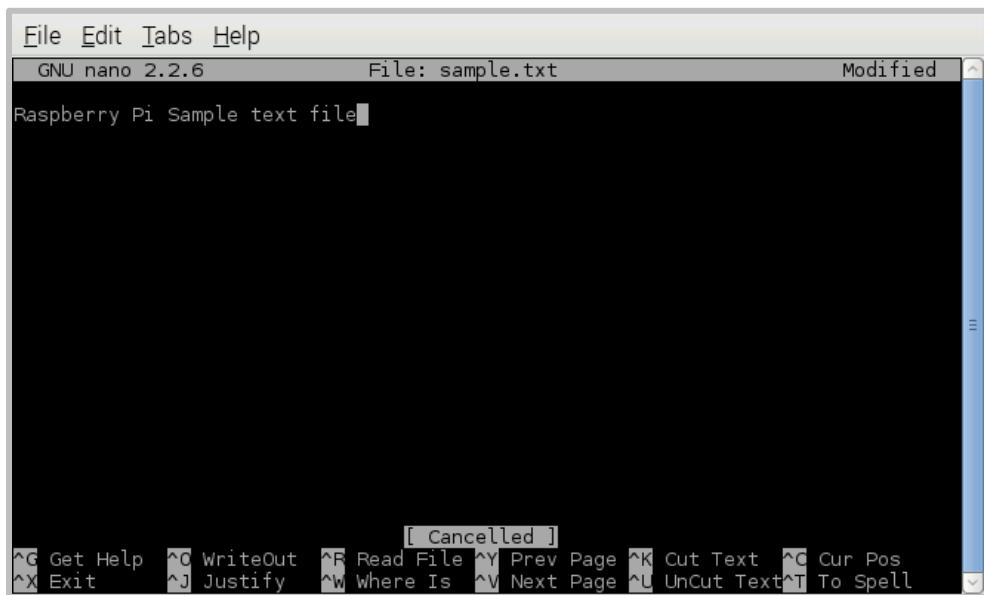
본 메뉴얼에서는 비교적 사용법이 간단한 nano를 사용해보도록 하겠습니다.

nano 에디터로 sample.txt 파일 만들기

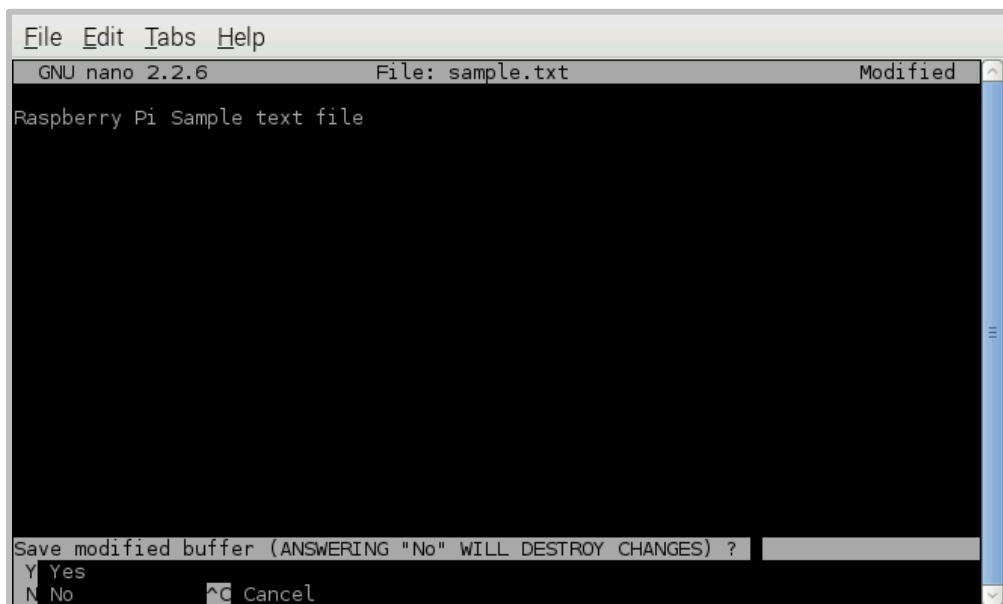
```
pi@raspberrypi ~ $ sudo nano sample.txt
```



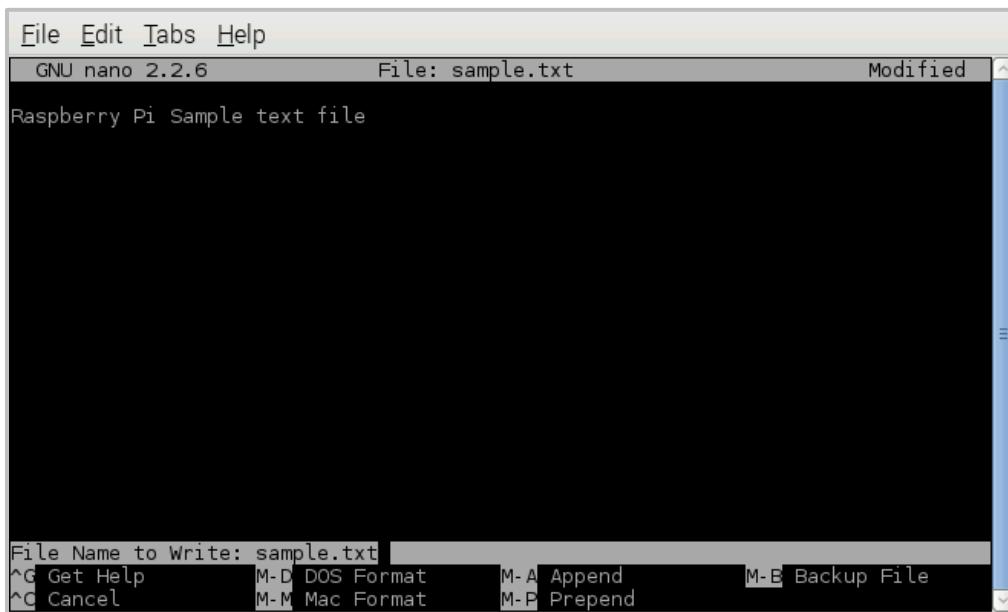
Sample.txt라는 이름의 텍스트파일을 생성하기 위해서 위와 같은 명령어를 터미널에서 입력합니다. 이미 같은 이름의 파일이 존재한다면, 기존의 파일이 열리게 됩니다.,



텍스트 파일 안에 내용을 넣고 Ctrl+X 를 누르면 파일을 저장하고 편집기를 닫을지, 저장하지 않고 닫을지를 묻습니다.



Y 를 타이핑하면 저장을 하고 편집기를 닫게 됩니다.



이 때, 파일명을 변경할 수도 있습니다. 마찬가지로 파일을 생성하고 변경할 때도 하단의 파라미터들을 사용하여 편집기를 사용할 수 있습니다.

키보드 설정 변경하기

타이핑을 하다보면, 화면에 보이는 키와 누른 키가 일치하지 않는 경우를 볼 수 있습니다. 예를 들어 “\”라든지 “@”와 같은 기호를 눌러도 실제로 보이는 값은 이 값이 아닙니다. 이는 Raspbian에서 설정되어 있는 기본 키보드로 인해서 발생한 이슈인데, 이를 해결하기 위해서는 다음과 같이 설정을 변경할 수 있습니다.

먼저, 다음과 같이 키보드 설정 화면으로 이동합니다.

키보드 설정 변경하기
pi@raspberrypi ~ \$ sudo nano /etc/default/keyboard

```

File Edit Tabs Help
GNU nano 2.2.6          File: /etc/default/keyboard
# KEYBOARD CONFIGURATION FILE
# Consult the keyboard(5) manual page.

XKBMODEL="pc105"
XKBLAYOUT="gb"
XKBVARIANT=""
XKBOPTIONS=""

BACKSPACE="guess"

[ Read 10 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell

```

위의 윈도우에서 XKBLAYOUT 이 gb 로 설정되어 있는데, 이를 us 로 변경하고 Ctrl+X 를 통해 저장 후 편집기를 닫습니다.

```

File Edit Tabs Help
GNU nano 2.2.6          File: /etc/default/keyboard          Modified
# KEYBOARD CONFIGURATION FILE
# Consult the keyboard(5) manual page.

XKBMODEL="pc105"
XKBLAYOUT="us"
XKBVARIANT=""
XKBOPTIONS=""

BACKSPACE="guess"

[ Read 10 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell

```

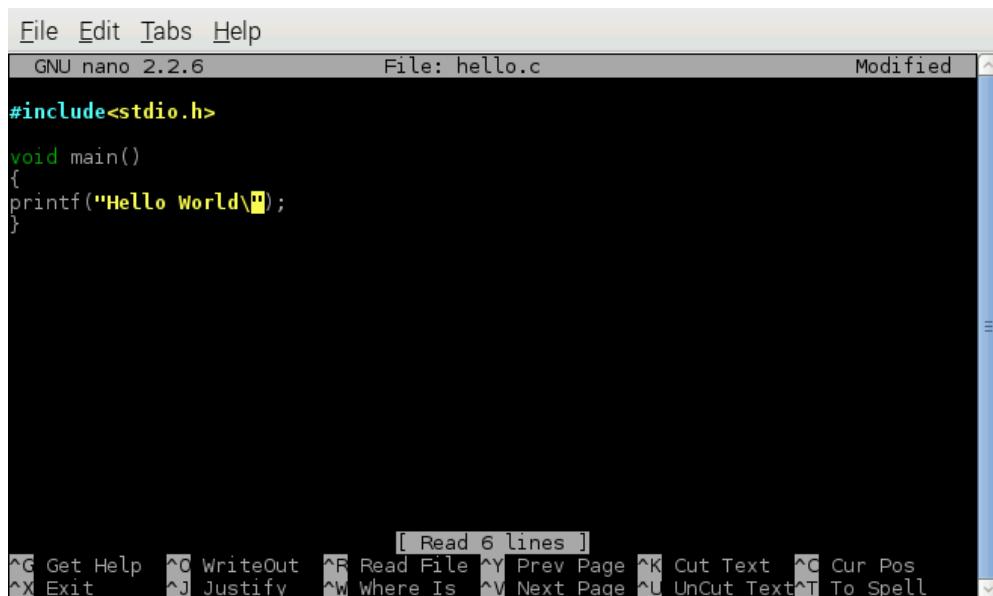
그린 다음 sudo reboot 를 통해 시스템 재부팅을 하면 키보드 입력과 화면 출력이 일치하는 것을 확인할 수 있습니다.

C. 리눅스로 프로그래밍하기

윈도우에서 프로그래밍을 해 본 경험이 있으시다면 Visual Studio 혹은 이클립스(Eclipse)와 같은 개발환경을 사용해보셨을지 모릅니다. 리눅스에서 프로그래밍을 하고 컴파일을 하는 경우는 기본적으로 GCC 컴파일러를 사용하는 경우가 많이 있습니다. 따라서 본 매뉴얼에서는 GCC 컴파일러를 사용하여 프로그래밍하는 방법에 대해서 알아보도록 하겠습니다. 파이썬과 C 프로그래밍을 다루게 될 것이며, 각각의 프로그램을 통해 Hello World를 콘솔창(검정색 터미널)에 프린트해보도록 하겠습니다.

C로 Hello World 프린트하기

```
pi@raspberrypi ~ $ sudo nano hello.c
```



The screenshot shows the nano text editor window. The menu bar includes File, Edit, Tabs, Help. The title bar displays "GNU nano 2.2.6" and "File: hello.c". The status bar shows "Modified". The main area contains the following C code:

```
#include<stdio.h>
void main()
{
printf("Hello World\n");
}
```

At the bottom of the screen, there is a command-line interface with various keyboard shortcuts:

- [Read 6 lines]
- Get Help** ^G **WriteOut** ^O **Read File** ^R **Prev Page** ^Y **Cut Text** ^K **Cur Pos** ^C
- Exit** ^X **Justify** ^J **Where Is** ^W **Next Page** ^V **UnCut Text** ^U **To Spell** ^T

GCC 사용하여 컴파일하고 실행하기

```
pi@raspberrypi ~ $ gcc -o hello hello.c
pi@raspberrypi ~ $ ./hello
```

Python으로 Hello World 프린트하기

```
pi@raspberrypi ~ $ sudo nano hello.py
```

The screenshot shows a terminal window titled 'File: hello.py' running the 'GNU nano 2.2.6' editor. The code 'print "Hello World"' is typed into the editor. The bottom status bar displays various keyboard shortcuts for file operations like 'Get Help', 'WriteOut', and 'Cut Text'. A scroll bar is visible on the right side of the editor window.

파이썬 실행하기

```
pi@raspberrypi ~ $ sudo python hello.py
```

The screenshot shows a terminal window with a black background and white text. It displays the command 'sudo python hello.py' being run, followed by the output 'Hello World'. The terminal window has a scroll bar on the right side.

3. 라즈베리파이와 Mathematica, Wolfram, Scratch

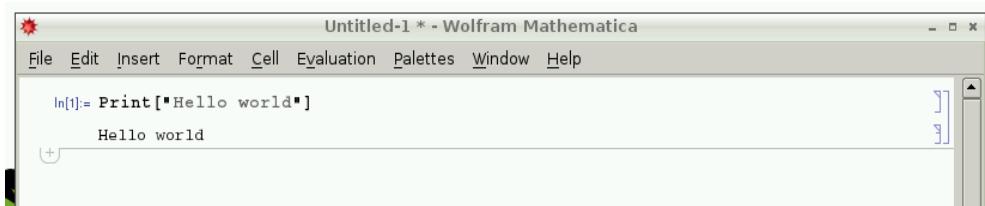
A. Mathematica로 프로그래밍하기

라즈베리파이의 왼쪽 상단에 빨간색 밤틀 모양의 Mathematica 아이콘이 있습니다.



Mathematica는 수학 전공 학생들이 많이 사용하는 프로그래밍 툴로써,
엔지니어들이 사용하는 Matlab과 비슷한 툴이라고 볼 수 있습니다.

위 아이콘을 클릭하여 실행한 후 다음과 같이 타이핑을 하면 Mathematica를
사용한 Hello World 출력을 할 수 있습니다.



Mathematica는 다음과 같은 간단한 산수부터 복잡한 수학 계산까지 훌륭하게 수행
할 수 있는 알고리즘이 내장되어 있습니다.

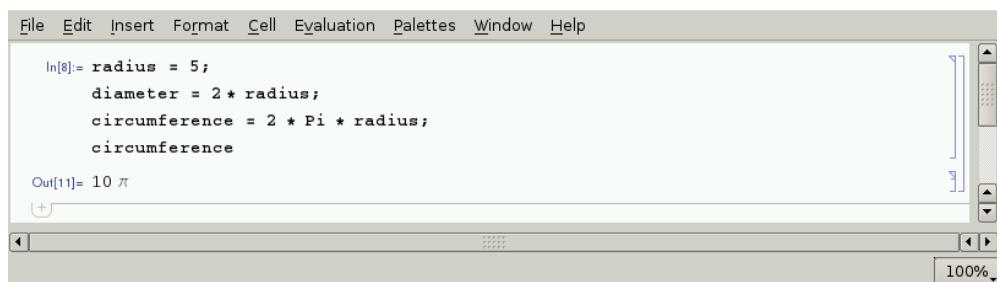
```
In[2]:= 2 + 2
Out[2]= 4
In[3]:= 16254 / 32
Out[3]= 8127 / 16
In[4]:= 1024 * 32
Out[4]= 32768
```

2 + 2를 입력한 후에, Shift + Enter를 치면 결과 (Out)이 출력됩니다.
또한, 변수를 저장하여 복잡한 프로그램을 수행할 수도 있습니다.

```
radius = 5;
diameter = 2 * radius;
circumference = 2 * Pi * radius;
```

변수를 지정할 때, 세미콜론(:;)이 없는 상태에서 Shift+Enter를 입력하면 그에 따르는 결과가 출력되지만 세미콜론이 있다면 출력되지 않습니다.

예를 들어 위에서 원주의 길이 (circumference)를 알고 싶다면 새로운 줄에서 circumference만 입력하고 (세미콜론 없이), Shift+Enter를 칩니다. 그럼 Out으로 원의 둘레 길이가 출력이 되는 것을 확인할 수 있습니다.



B. Wolfram 커맨드 라인 사용하기



Wolfram은 Mathematica와 달리 터미널을 기반으로 In/Out 스타일의 프로그래밍 환경을 제공합니다. 즉, 다음과 같이 사용할 수 있습니다.

A screenshot of the Wolfram Language interface. The window title is "File Edit Tabs Help". Inside the window, the text reads:

```
Wolfram Language (Raspberry Pi Pilot Release)
Copyright 1988-2015 Wolfram Research
Information & help: wolfram.com/raspi

In[1]:= radius = 3;
In[2]:= circumference = 2 * Pi * radius;
In[3]:= circumference
Out[3]= 6 Pi
In[4]:= 
```

The interface has a dark background with white text and a vertical scroll bar on the right side.

Wolfram을 사용하였을 때의 장점으로는 Mathematica가 GUI를 기반으로 하여 Graphics를 사용하는데 반해 터미널을 사용하기 때문에 프로세싱 스피드가 조금 더 빠르다는 것을 들 수 있습니다.

Wolfram에서는 스크립트 파일을 실행할 수도 있는데, .m 혹은 .wl과 같은 확장자의 파일을 실행할 수 있습니다.

테스트 스크립트 생성
pi@raspberrypi ~ \$ sudo nano test.m

Print["Hello World"]를 입력한 후에 Ctrl+X를 통해 파일을 저장합니다. 그리고 터미널에서 다음과 같이 입력하면 Hello World가 출력되는 것을 확인할 수 있습니다.

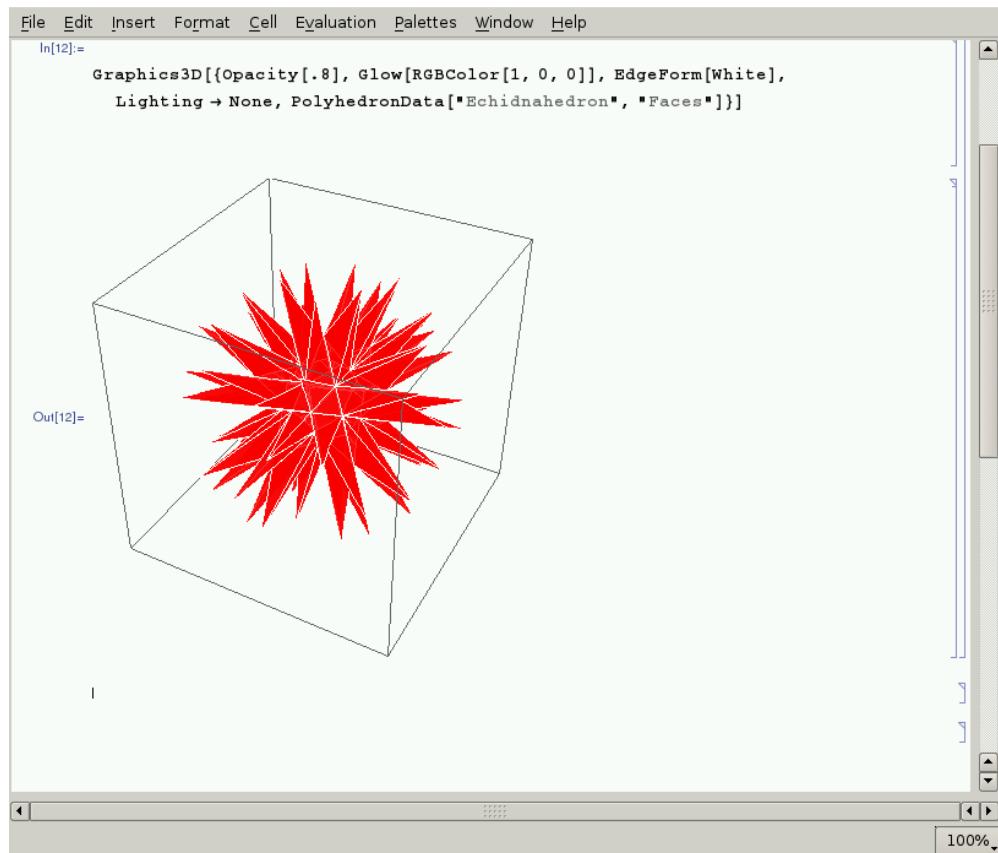
실행하기
pi@raspberrypi ~ \$ wolfram –script test.m

C. 그래프 출력하기

Mathematica를 사용하는 묘미를 이야기할 때, 수려한 그래프들을 이야기하지 않을 수 없습니다.

```
Graphics3D[{Opacity[.8], Glow[RGBColor[1, 0, 0]],  
EdgeForm[White], Lighting -> None,  
PolyhedronData["Echednahedron", "Faces"]}]
```

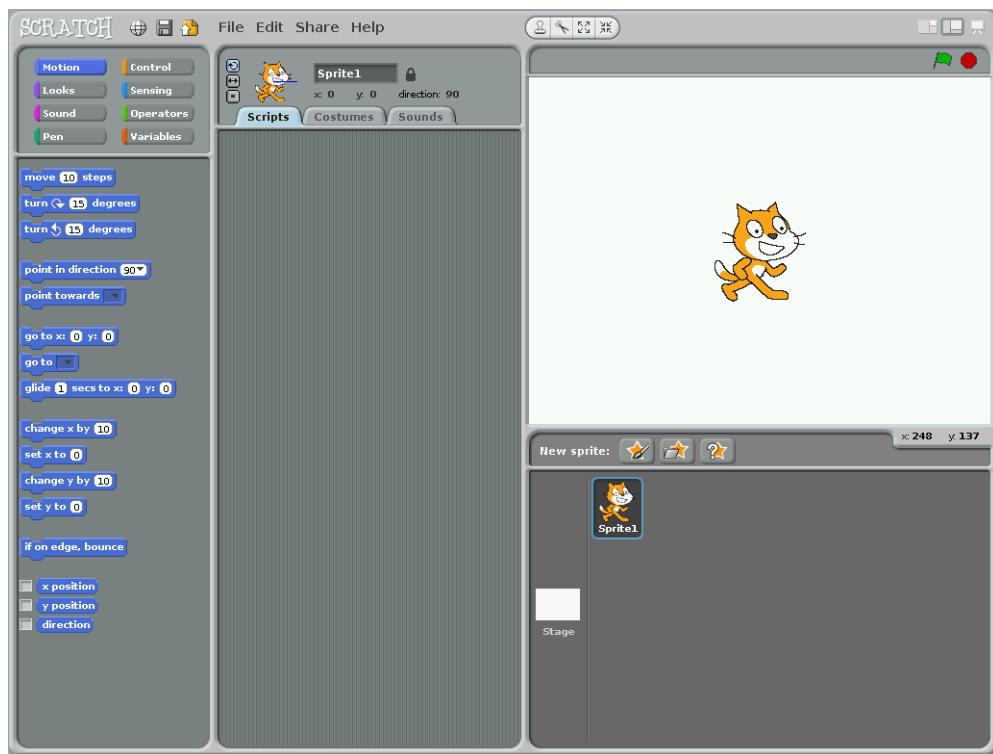
위의 함수를 Mathematica에 입력한 후에 Shift+Enter를 치면 Mathematica 아이콘과 비슷한 형태의 Echednahedron이 출력이 됩니다.



D. Scratch로 비쥬얼 프로그래밍하기

스크래치(Scratch)란 사용자로 하여금 애니메이션 혹은 게임을 간단한 사용법으로 구현할 수 있도록 개발된 비쥬얼 프로그래밍 (Visual programming) 환경입니다. 스크립트(Script)를 기반으로 하는 자바(Java), 파이썬(Python), C 등의 언어를 몰라도 게임을 개발하거나 애니메이션을 만들 수 있는 매력이 있습니다.

뿐만 아니라 Scratch GPIO를 사용하여 LED, 모터 등의 하드웨어를 라즈베리파이의 GPIO를 사용하여 보다 인터랙티브하게 사용할 수 있습니다.



스크래치의 초기화면은 위와 같으며 라즈베리파이의 왼쪽 상단에 있는 Menu – Programming을 통해서 열 수 있습니다.

왼쪽에 있는 퍼즐 형식의 모듈을 가운데 드래그하여서 프로그래밍을 할 수 있으며, 오른쪽 상단의 깃발과 빨간색 정지 버튼을 통해서 프로그램을 실행해볼 수 있습니다.

간단히, Sprite라는 고양이 아이콘을 사용하여 데모를 구현해보도록 하겠습니다.

먼저, 왼쪽의 팔레트에서 녹색 깃발이 눌리면 시작할 수 있도록

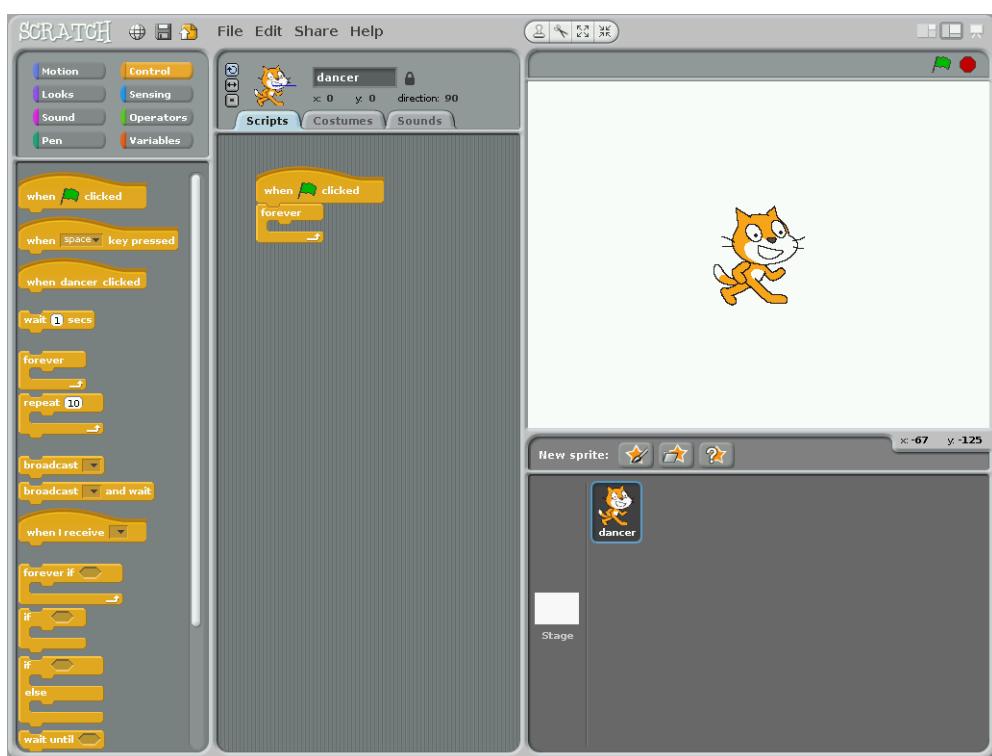


블럭을 가운데 Scripts로 드래그합니다.

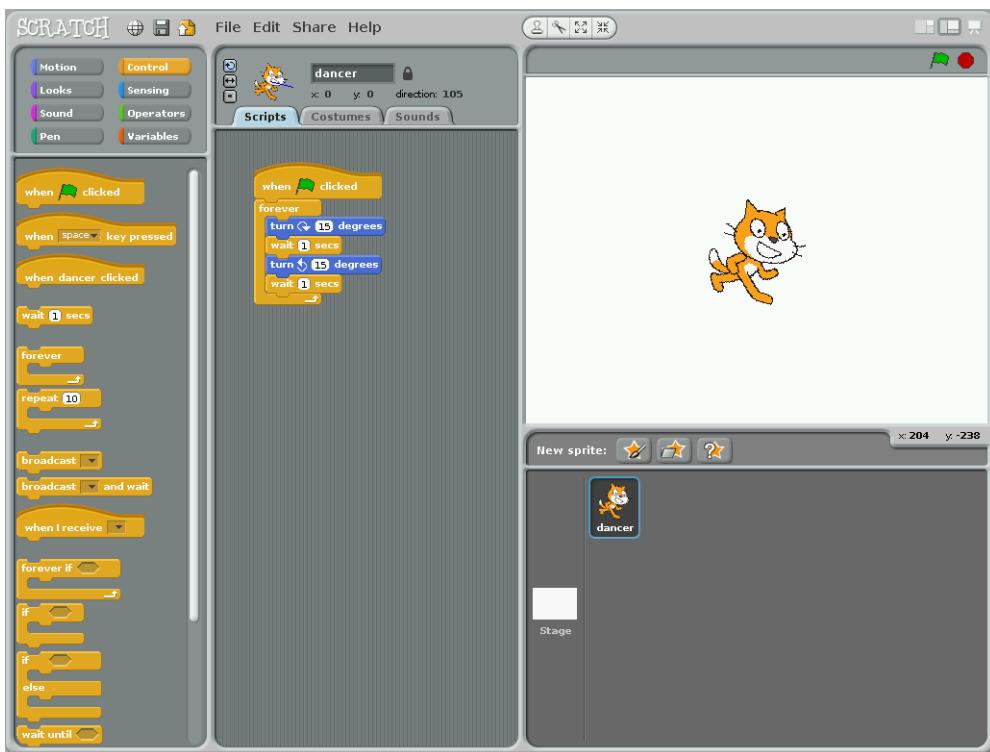
그리고 사용자가 빨간색 정지 버튼을 누르기 전까지 계속 반복되는



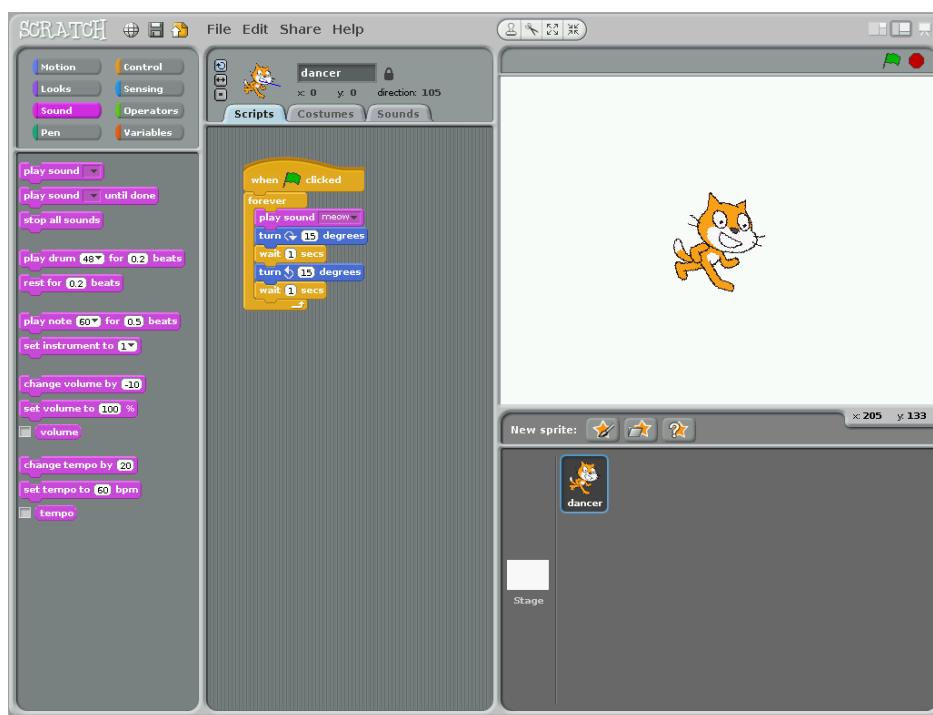
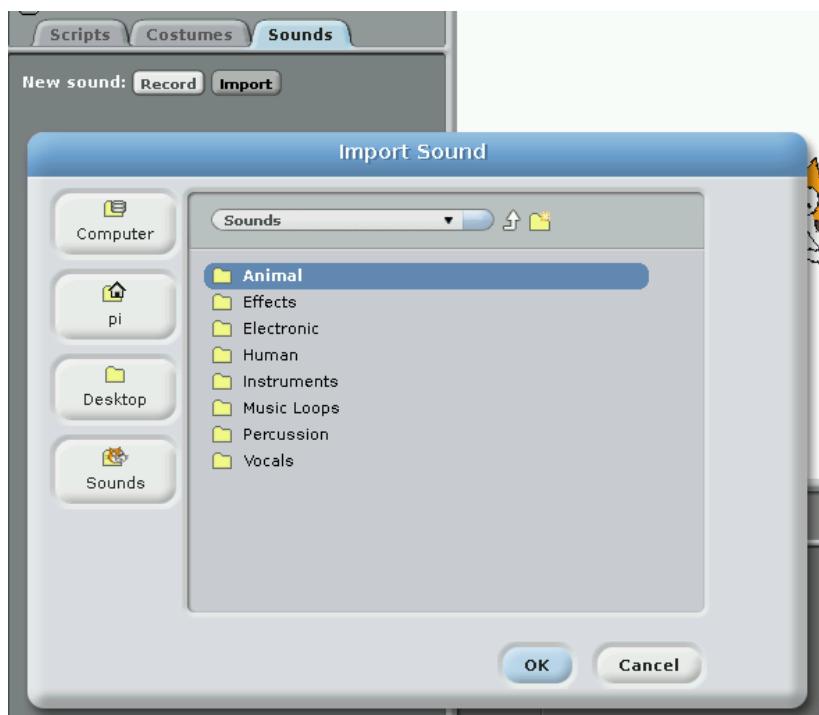
Forever 블럭을 드래그한 후 퍼즐처럼 맞춥니다.



그 다음, 왼쪽 팔레트의 Motion 버튼을 눌러서 시계방향으로 아이콘을 돌리는 블럭과 반시계 방향으로 아이콘을 돌리는 블럭을 드래그해서 forever 블럭 안쪽으로 끼워 넣습니다. 이 때, 오른쪽 상단의 녹색 깃발을 눌러보면 고양이가 너무 빨리 15도씩 왔다 갔다하는 것을 확인할 수 있습니다. 조금 천천히 움직임을 보기 위해서 다시 Control 버튼을 클릭한 후 wait 블럭을 다음과 같이 끼워 넣습니다. 그리고 녹색 깃발을 눌러 보면 고양이가 15도씩 약 1초간의 간격으로 회전하는 것을 확인할 수 있습니다.



스크래치에서는 이 뿐만 아니라 Sound와 같은 기능도 제공하고 있는데, 왼쪽의 팔레트에서 Sound 버튼을 클릭해봅니다. 그리고 play sound라는 블럭을 드래그해서 음원을 넣어서 재생하도록 합니다. 기본 음원을 사용할 수도 있고, 레코딩을 할 수도 있습니다. 또한 저장되어 있는 음원 파일을 import 하여 사용할 수도 있습니다. Sounds 버튼을 클릭한 후에 import 버튼을 누르게 되면 다양한 음원을 불러올 수 있습니다.



4. 라즈베리파이와 파이썬

A. 파이썬 소개

파이썬은 최근 많은 인기를 얻고 있는 프로그래밍 언어로써, 일상 언어에 가까운 가독성과 빠른 개발 기간으로 인해서 프로그래밍 교육 및 개발자들에게 인기를 얻고 있습니다. 보다 명확하고 표현력 있는 문법으로 인해서 초등학교 학생들도 파이썬을 배울 수 있으며 어린이들을 위한 파이썬 교재도 시중에 많이 판매되고 있습니다. 라즈베리 파이의 “파이”가 의미하는 것이 파이썬을 의미할 정도로 라즈베리파이와 파이썬을 뗄래야 뗄 수 없는 관계이며, 데크스탑 메뉴의 Programming에는 파이썬의 개발도구인 IDLE를 제공하고 있습니다.

파이썬의 특징으로는 1) 단순함, 2) 오픈소스, 3) 메모리 관리 불필요, 4) 높은 이식성, 5) 배우기 쉽다는 점이 있습니다. 예를 들어서 Hello World를 출력하기 위해서 print "Hello World"만으로 출력할 수 있고, 더 나은 파이썬을 만들고자 하는 사람들의 노력으로 인해 지속적으로 개선되고 있습니다. 그리고 메모리 관리 등과 같은 세부적인 노력이 필요없고, 공개된 무료 소프트웨어로 인해서 다른 OS 등에서 자유롭게 사용될 수 있는 높은 이식성을 가지고 있습니다. 단순하기 때문에 초심자들에게 배우기 쉽다는 이유도 파이썬의 사용자가 늘어나는 이유 중 하나입니다. 또한, 6) 인터프리터 언어라서 컴퓨터가 필요없으며, 7) 객체 지향 프로그램이라서 데이터와 기능이 결합된 객체라는 것을 사용하며 C++ 혹은 Java 보다 더 강력하고 쉬운 방법으로 객체 지향을 지원합니다.

B. 파이썬 기초

파이썬은 터미널을 사용할 수도 있고 편집기를 사용할 수도 있습니다. 즉, 라즈베리파이의 왼쪽 상단에 있는 프로그래밍에 보시면 Python 2 혹은 Python 3을 선택하셔서 Python Shell이라는 것을 통한 프로그래밍을 할 수도 있고 직접 검정색 바탕의 터미널에서 프로그래밍을 하실 수도 있습니다.

1. 주석

주석은 # 문자 뒤에 따라오는 짧은 문장입니다. 주로 소스 코드를 읽는 사람들을 위해 설명하기 위한 용도로 빈번하게 사용됩니다.

주석 예제

```
print 'hello world' # 주석 내용  
으로 (# 주석 내용)은 인식되지 않고, print 'hello world'만 인식됨.
```

2. 들여쓰기

파이썬에서 공백은 중요한 역할을 합니다. 정확하게 한 행의 앞에 붙어있는 공백이 정말 중요합니다. 이것을 _들여쓰기_라 부릅니다. 한 논리적 명령행의 앞에 붙어있는 공백(빈 칸 혹은 탭)은 논리적 명령행의 들여쓰기 단계를 의미하며, 이것은 한 명령의 범위를 구분하는 데 사용됩니다. 이것은 같은 범주 있는 명령들은 반드시 동일한 들여쓰기를 사용해야 함을 의미합니다. 이렇게 같은 들여쓰기를 사용하고 있는 명령들의 집합을 블록(block)이라고 부릅니다. ‘들여쓰기를 할 때에는 공백 4개를 이용하세요.’ 이것은 파이썬 언어에서 공식적으로 추천하는 방법입니다. 좋은 편집기들은 이 사항을 자동으로 준수합니다. 또, 들여쓰기를 할 때에는 항상 같은 개수의 공백을 사용해야 한다는 점에 유의하시기 바랍니다.

C. 연산자와 수식

연산자	연산자 설명	연산자	연산자 설명
+	덧셈 연산자	-	뺄셈 연산자
*	곱셈 연산자	**	거듭제곱 연산자
/	나눗셈 연산자	%	나머지 연산자
<<	왼쪽 시프트 연산자	>>	오른쪽 시프트 연산자
&	비트 연산자 (AND)		비트 연산자 (OR)
^	비트 연산자 (XOR)	~	비트 반전 연산자
<	작음	>	큼
<=	작거나 같음	>=	크거나 같음
==	같음	!=	같지 않음
not	불리언 NOT 연산자	and	불리언 AND 연산자
or	불리언 OR 연산자		

연산자와 수식을 이용한 파이썬 코드 작성 및 실행

소스코드 준비

```
pi@raspberrypi ~ $ sudo nano cal.py
```

cal.py

```
length = 5
width = 3
area = length * width
print 'Area is', area
print 'perimeter is', 2*(length+width)
```

실행하기

```
pi@raspberrypi ~ $ sudo python cal.py
Area is 15
Perimeter is 16
```

D. 반복 및 조건문

1. If

if 문은 조건을 판별할 때 사용됩니다. if (만약) 조건이 참이라면, _if 블록_의 명령문을 실행하며 else (아니면) _else 블록_을 실행합니다. 여기서 elif (아니면 만약)를 사용하면 바로 else로 가지 않고 여러 개의 조건문을 달아줄 수 있습니다. 이때 else는 if와 모든 elif가 거짓이여야 실행됩니다.

소스코드 준비

```
pi@raspberrypi ~ $ sudo nano If.py
```

If.py

```
number = 23
guess = int(raw_input('Enter an integer : '))

if guess == number:
    # New block starts here
    print 'Congratulations, you guessed it.'
    print '(but you do not win any prizes!)'
    # New block ends here
elif guess < number:
    # Another block
    print 'No, it is a little higher than that'
    # You can do whatever you want in a block ...
else:
    print 'No, it is a little lower than that'
    # you must have guessed > number to reach here
    print 'Done'
    # This last statement is always executed,
    # after the if statement is executed.
```

사용자로부터 숫자를 입력받아 기존의 숫자와 비교하는 프로그램입니다. 현재 기존 숫자는 23으로 선언되어 있습니다. `raw_input`이라는 함수는 숫자를 입력받는 함수입니다. 인자로 문자열을 넣어주면 입력받기 전에 해당 문자열을 출력하여 줍니다. 여기서 입력받은 숫자를 기존 숫자와 비교합니다. 먼저 `if`문이 실행되어 숫자가 같은지 비교합니다. 같다면 블록 안에 있는 `print` 명령어들이 실행되며 아니라면 다음 `elif`로 넘어가게 됩니다. `elif`에서는 입력받은 숫자가 더 작은지 판별합니다. 이까지 다 거짓이라면 최종적으로 `else`가 실행됩니다. 결론적으로 `else`의 실행 조건은 숫자가 '같지도 않고, 작지도 않을 때' 입니다.

조건문(`guess == number` 등)뒤에 보시면 :(콜론)이 찍혀있는 것을 볼 수 있습니다. 이는 블록의 시작을 나타내는 부분으로써 들여쓰기와 같이 시작과 끝을 나타내는데 사용됩니다. 콜론 뒤에는 들여쓰기를 하게 되며 이 들여쓰기가 끝나면 블록이 끝났다는 것을 의미합니다.

실행 결과

```
pi@raspberrypi ~ $ sudo python If.py
Enter an integer : 50
No, it is a little lower than that
Done
pi@raspberrypi ~ $ sudo python If.py
Enter an integer : 22
No, it is a little higher than that
Done
pi@raspberrypi ~ $ sudo python If.py
Enter an integer : 23
Congratulations, you guessed it.
(but you do not win any prizes!)
Done
```

2. While

while 문은 *반복문*의 한 예입니다. 조건문이 참일 경우 계속해서 블록의 명령문들을 실행합니다. 거짓일 경우 실행되는 else문도 사용이 가능합니다.

소스코드 준비

```
pi@raspberrypi ~ $ sudo nano While.py
```

While.py

```
number = 23
running = True
while running:
    guess = int(raw_input('Enter an integer : '))
    if guess == number:
        print 'Congratulations, you guessed it.'
        # this causes the while loop to stop
        running = False
    elif guess < number:
        print 'No, it is a little higher than that.'
    else:
        print 'No, it is a little lower than that.'
    else:
        print 'The while loop is over.'
    # Do anything else you want to do here
print 'Done'
```

이전 프로그램에서 while문이 추가되었습니다. 반복기능을 추가함으로써 계속 프로그램을 실행시키지 않아도 됩니다. 처음에 running이 참인지 판단하고 참이라면 구문을 실행합니다. 전부 다 실행됐다면 다시 running의 참 / 거짓여부를 판단하고 이를 반복 합니다. 거짓일 경우 while문을 탈출하고 else문을 실행합니다.

실행 결과

```
pi@raspberrypi ~ $ sudo python While.py
Enter an integer : 50
No, it is a little lower than that.
Enter an integer : 22
No, it is a little higher than that.
Enter an integer : 23
Congratulations, you guessed it.
The while loop is over.
Done
```

3. For 루프

for in 문은 객체의 열거형을 따라서 반복하여 실행되는 하나의 반복문입니다. 각 항목을 하나씩 거치며 실행됩니다. 여기서 열거형은 여러 항목이 나열된 목록집합이라고 보시면 됩니다.

소스코드 준비

```
pi@raspberrypi ~ $ sudo nano For.py
```

For.py

```
for i in range(1, 5):
    print i
else:
    print 'The for loop is over'
```

range(1, 5)에 해당하는 숫자를 출력하는 프로그램입니다. range는 첫번째 숫자 이상, 두번째 숫자 미만인 배열을 반환합니다. 세번째 숫자는 얼마씩 더할 것인지를 나타내며 기본 설정 값(디폴트 값)은 1입니다. while등과 마찬가지로 else문이 사용 가능합니다.

실행 결과

```
pi@raspberrypi ~ $ sudo python For.py
```

```
1
2
3
4
```

```
The for loop is over
```

4. Break

`break`문은 조건문에 관계없이 강제로 탈출하고 싶을 때 사용하는 구문입니다. `if`문과 같이 주로 사용합니다. 만약 `break`문을 사용해서 `for` 루프나 `while` 루프를 빠져나왔을 경우에는 조건문이 거짓일 때 실행되는 `else` 블록은 실행되지 않습니다.

소스코드 준비

```
pi@raspberrypi ~ $ sudo nano Break.py
```

Break.py

```
while True:
    s = raw_input('Enter something : ')
    if s == 'quit':
        break
    print 'Length of the string is', len(s)
print 'Done'
```

반복해서 입력을 받고 `len`이라는 함수를 이용하여 문자열의 길이를 출력합니다. 다만 입력받은 문자열이 ‘quit’일 경우 `break`문이 실행되어 `while`문을 탈출하게 됩니다. 그 외 `for`문에서도 `break`문이 사용 가능합니다.

실행 결과

```
pi@raspberrypi ~ $ sudo python Break.py
Enter something : Programming is fun
Length of the string is 18
Enter something : When the work is done
Length of the string is 21
Enter something : if you wanna make your work also fun:
Length of the string is 37
Enter something : use Python!
Length of the string is 11
Enter something : quit
Done
```

5. Continue

continue 문은 다음 루프 블록으로 넘어가도록 합니다. 이때 남아있는 명령문들은 실행되지 않습니다.

소스코드 준비

```
pi@raspberrypi ~ $ sudo nano Continue.py
```

Continue.py

```
while True:
    s = raw_input('Enter something : ')
    if s == 'quit':
        break
    if len(s) < 3:
        print 'Too small'
        continue
    print 'Input is of sufficient length'
    # Do other kinds of processing here...
```

if문이 하나 추가되었습니다. 입력받은 글자의 길이가 3미만일 경우 'Too small'을 출력하고 continue를 실행합니다. 이때 다음 루프로 넘어가므로 아래의 print 명령어는 실행되지 않습니다. while문에서는 다음 루프라는 느낌이 없으나 for문 같은 경우 다음 루프를 실행하기에 열거형 안의 변수 하나를 배제할 수 있어 예외처리에 종종 사용됩니다.

실행 결과

```
pi@raspberrypi ~ $ sudo python Continue.py
Enter something : a
Too small
Enter something : 12
Too small
Enter something : abc
Input is of sufficient length
Enter something : quit
```

E. 함수

함수는 재사용 가능한 프로그램의 조각을 말합니다. 명령어 덩어리를 묶어 하나의 블록으로 만들고 이를 프로그램 어디에서든 실행할 수 있게 합니다. 이때 실행하는 과정을 ‘함수를 호출한다’라고 합니다. 앞서 사용한 `len`이나 `range`도 함수의 일종입니다. 자주 사용되는 명령어들을 하나로 묶어 함수로 만듬으로써 프로그램의 수정과 관리가 용이하도록 해줍니다.

파이썬에서는 `def` 키워드를 이용하여 함수를 정의합니다. 뒤에 함수의 식별자 이름을 입력하고 소괄호를 이용하여 인자의 목록을 입력한 후 콜론으로 함수의 정의를 마무리합니다. 이후 함수에 포함될 명령어들을 입력하는데 `if`문 등과 비슷합니다. 들여쓰기를 통해 어디까지 함수에 포함되는지를 말해 줍니다.

소스코드 준비

```
pi@raspberrypi ~ $ sudo nano Function1.py
```

Function1.py

```
def say_hello():
    print 'hello world'
say_hello()
say_hello()
```

아무런 인자를 입력받지 않는 `say_hello`라는 함수를 정의하였습니다. 여기서 인자란 함수 호출시 넘겨주는 값들을 말합니다. 입력해줄 것이 없으므로 괄호내에는 아무것도 써주지 않습니다. 호출된 함수는 `print 'hello world'`라는 명령어를 실행합니다. 이는 여러 번 같은 구문을 입력하는 것 보다 더 효율적으로 프로그래밍을 할 수 있도록 해줍니다.

실행 결과

```
pi@raspberrypi ~ $ sudo python Function1.py
hello world
hello world
```

함수를 정의할 때 매개 변수를 지정해줄 수 있습니다. 매개 변수란 인자와 비슷하지만 차이점이 있습니다. 인자는 암수 호출시 넘겨주는 값, 매개 변수는 함수에서 받아들이는 값입니다. 매개 변수들은 변수와 거의 같이 취급되며 같은 함수가 호출될 때 넘겨받은 값으로 채워집니다. 각 매개 변수들은 쉼표로 구분하며 소괄호 안에 위치합니다.

소스코드 준비

```
pi@raspberrypi ~ $ sudo nano Param.py
```

Param.py

```
def print_max(a, b):
    if a > b:
        print a, 'is maximum'
    elif a == b:
        print a, 'is equal to', b
    else:
        print b, 'is maximum'

print_max(3, 4)

x = 5
y = 7

print_max(x, y)
```

여기서는 매개 변수 a, b를 가진 print_max라는 함수를 정의하였습니다. if구문을 이용하여 크기를 비교하고 둘중 더 큰값을 출력하는 함수입니다. 함수를 호출할 때 처음에는 3, 4라는 상수를 인자로 입력하였습니다. 이때 매개 변수 a에는 3이, b에는 4가 입력됩니다. 두번째에는 a에는 변수 x의 값이, b에는 변수 y의 값이 입력됩니다. 이때 x는 5, y는 7이므로 각각 5, 7이 입력된 꼴이 됩니다.

실행 결과

```
pi@raspberrypi ~ $ sudo python Param.py
4 is maximum
7 is maximum
```

정의한 함수 안에서 변수를 선언하고 사용할 경우, 함수 밖에 있는 같은 이름의 변수들과 함수 안에 있는 변수들과는 서로 연관이 없습니다. 이러한 변수들을 함수의 지역 (local) 변수라고 하며, 그 범위를 변수의 스코프(scope)라고 부릅니다. 모든 변수들은 변수가 정의되는 시점에 서의 블록을 스코프로 가지게 됩니다.

소스코드 준비

```
pi@raspberrypi ~ $ sudo nano FunctionLocal.py
```

FunctionLocal.py

```
x = 50
def func(x) :
    print 'x is', x
    x = 2
    print 'Changed local x to ', x
func(x)
print 'x is still', x
```

x라는 매개 변수를 가진 함수 func를 선언하였습니다. 이 함수는 입력받은 x를 한번 출력하고 x를 2로 바꿔서 다시 출력합니다. 그리고 마지막에 print로 x값을 출력해 보면 처음 대입된 50이라는 값은 전혀 바뀌지 않았습니다. func는 단지 매개 변수인 x의 값을 바꾸었을 뿐입니다. 그렇기에 인자로 사용되었던 x는 변함이 없었던 것입니다.

실행 결과

```
pi@raspberrypi ~ $ sudo python FunctionLocal.py
```

```
x is 50
```

```
Changed local x to 2
```

```
x is still 50
```

함수 안에서도 인자로 받은 변수를 바꿀 수 있는 방법이 있습니다. 바로 전역(global) 변수로 사용할 것이라고 알려주는 것이지요. 여기서 전역변수란 프로그램 전체에 영향을 기치는 변수로 프로그램 내라면 어디서든 사용이 가능합니다. 매우 편리한듯 싶지만 전역변수를 남발할 경우 프로그램이 커지면 커질수록 더 헷갈리게 되어 문제를 초래하는 경우가 많으니 사용에 주의가 필요합니다.

소스코드 준비

```
pi@raspberrypi ~ $ sudo nano FunctionGlobal.py
```

FunctionGlobal.py

```
x = 50
def func():
    global x
    print 'x is', x
    x = 2
    print 'Changed global x to', x
func()
print 'Value of x is', x
```

global 문을 이용하여 x가 전역변수임을 알려줍니다. 이후로 함수 안에서도 x에 값을 대입하면 메인 블록의 x 값 또한 바뀌게 됩니다. global 문은 일반 변수와 동일하게 여러 개의 변수를 한번에 지정해 줄 수도 있습니다. 예를 들어 global x, y, z 와 같이 해주시면 x, y, z가 전부 전역변수라는 의미입니다.

실행 결과

```
pi@raspberrypi ~ $ sudo python FunctionGlobal.py
```

```
x is 50
```

```
Changed global x to 2
```

```
Value of x is 2
```

return 문은 함수에서 되돌아(return) 나올 때 사용됩니다. 이때 return 값 형식으로 원하는 값을 반환해줄 수 있습니다. return 문을 사용하게 되면 함수는 종료됩니다.

소스코드 준비

```
pi@raspberrypi ~ $ sudo nano Return.py
```

Return.py

```
def maximum(x, y):
    if x > y:
        return x
    elif x == y:
        return 'The numbers are equal'
    else:
        return y

print maximum(2, 3)
```

maximum 함수는 매개 변수중 큰 값을 반환합니다. if..else 구문을 이용하여 비교하며 return을 이용하여 값을 반환하는데 같은 경우는 'The numbers are equal'이라는 문장을 반환합니다.

return 문 뒤에 아무런 값도 지정하지 않을 경우 이는 return None을 실행하는 것과 같습니다. None은 아무것도 없는 것을 의미합니다. 여러분이 return 문을 함수에 쓰지 않았더라도 암시적으로 return None이 호출되어 값을 반환하고 종료합니다.

실행 결과

```
pi@raspberrypi ~ $ sudo python Return.py
```

```
3
```

F. 모듈 및 패키지

함서 재사용에 목적을 둔 함수를 배웠습니다. 함수를 쓰는건 정말 편리한 일이지만 매번 함수를 정의해준다는 것은 정말 귀찮은 일입니다. 그렇기에 모듈을 이용하여 함수들을 미리 선언해두고 한번에 불러올 수 있습니다. 모듈을 만드는 방법으로는 첫번째, .py 확장자를 가진 파일을 하나 만들고 그 안에 함수들과 변수들을 정의하는 것입니다. 두번째 현재 사용중인 파이썬 인터프리터를 만드는데 사용된 프로그래밍 언어로 모듈을 작성하는 것입니다. 예를들어 표준 파이썬 인터프리터의 경우 c언어를 이용하여 모듈을 작성하고 컴파일하면 파이썬에서 이를 불러와 사용할 수 있습니다.

그러면 표준 라이브러리중 하나인 sys를 이용하여 기본 모듈이 어디에 있는지 확인해 보도록 하겠습니다.

소스코드 준비

```
pi@raspberrypi ~ $ sudo nano moduleusingsys.py
```

moduleusingsys.py

```
import sys
print('The command line arguments are:')
for i in sys.argv:
    print i
print '\n\nThe PYTHONPATH is', sys.path, '\n'
```

먼저 sys 모듈을 import 문을 이용하여 불러왔습니다. 이는 파이썬에게 이 모듈을 사용할 것이라고 알려주는 과정입니다. sys 모듈은 시스템의 구체적인 파라미터와 함수들을 알려주는 모듈입니다. import 를 하게되면 기본적으로 참조하는 경로에서 해당 모듈을 검색합니다. 만약 해당하는 모듈이 없다면 에러를 출력할 것입니다.

모듈을 무사이 찾았다면 내부의 명령들을 읽어들입니다. sys.argv는 sys 모듈 안의 argv 변수를 의미합니다. 이는 입력된 인자의 리스트를 뜻하는 것으로 제일 처음오는 프로그램의 이름이 argv[0], 그 이후로 argv[1], argv[2]... 순으로 나아갑니다. 실행시 프로그램 이름 뒤에 we are arguments라고 입력한 후 실행해 보겠습니다.

sys.path 변수는 모듈을 불러올 때 참조하는 경로들을 나타냅니다. 여기서 제일 처음오는 경로는 프로그램의 경로를 뜻하며 그 이후는 차이가 있을 수 있습니다. import를 이용하여 모듈 호출시 이 경로에서 불러오게 됩니다.

실행 결과

```
pi@raspberrypi ~ $ sudo python moduleusingsys.py we are arguments
```

The command line arguments are:

moduleusingsys.py

we

are

arguments

The PYTHONPATH is ['/home/pi', '/usr/ ...

파이썬에서 변수는 함수 내에 존재하며 함수와 전역변수는 모듈안에 존재합니다. 그리고 모듈은 패키지 안에 포함되어 있습니다. 패키지는 모듈을 엮어둔 것으로 하나의 디렉토리라고 할 수 있습니다. 만약 패키지 안에 있는 특정 모듈만 호출하고 싶다면 from 패키지명 import 모듈명 형식으로 불러올 수 있습니다.

5. 라즈베리파이와 GPIO

A. 라즈베리파이 GPIO핀 기초

라즈베리파이2에는 40여개의 GPIO핀이 있습니다. 이 핀을 이용하면 SPI, 시리얼포트, I2C 와 같은 연결 방법으로 외부기기를 제어, 동작 할 수 있습니다. 핀들은 범용 입/출력 핀으로 사용될 수 있으며, 3.3V를 지원하기 때문에 3.3V 이상을 공급했을 시 기기의 손상이 올 수 있으므로 주의해야합니다.

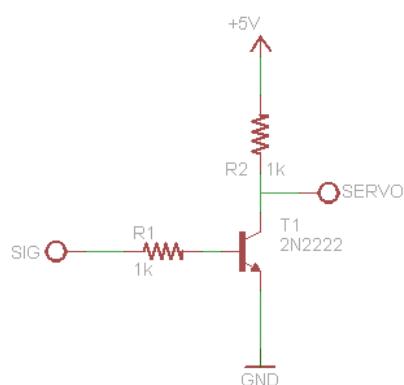
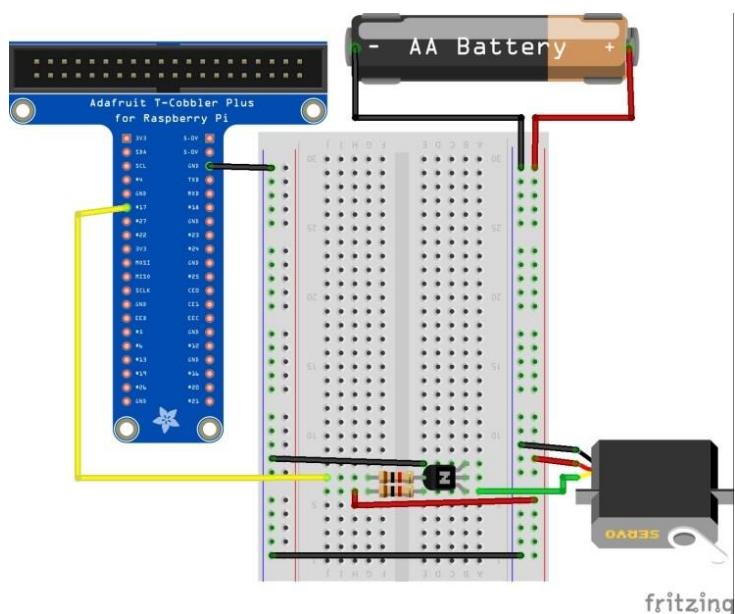


Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I2C)	DC Power 5v	04
05	GPIO03 (SCL1 , I2C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)	(I2C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

B. PWM 서보모터 제어하기

서보모터는 주로 일정한 각도를 가지고 움직이는 모터를 칭하며 약 180도정도의 작동 범위를 가지며 속도보다는 힘을 중시한 경우가 많습니다. 20ms주기로 동작하며 처음에 High 신호를 얼마나 주느냐에 따라 각도가 결정됩니다.

회로도



코블러 커넥터를 사용하였으며 BCM기준 17번(wPi기준 0번)에 신호선을 연결하여 주시면 됩니다. 관련된 사항은 wiringPi.h 설치부분에 구술되어 있습니다. 2222A에서 GND부분은 emitter, 가운데 신호선이 들어가는 부분은 base, 5V가 인가되는 부분은 collector입니다.

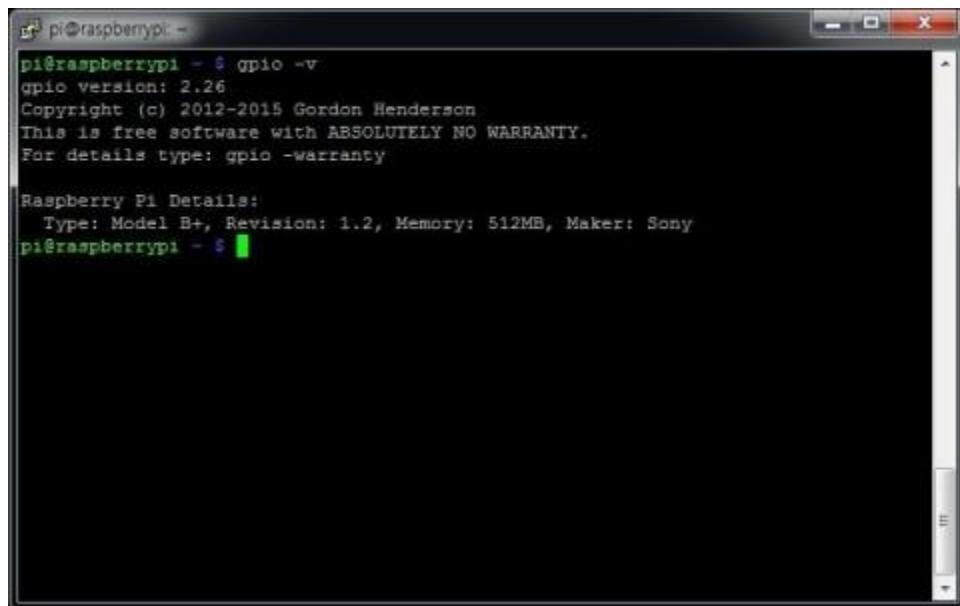
1. C 코딩

먼저, wiringPi.h를 설치해 보도록 하겠습니다.

wiringPi.h 설치

```
pi@raspberrypi ~ $ cd  
pi@raspberrypi ~ $ sudo apt-get update  
pi@raspberrypi ~ $ sudo apt-get upgrade  
pi@raspberrypi ~ $ sudo apt-get install git-core  
pi@raspberrypi ~ $ git clone git://git.drogon.net/wiringpi  
pi@raspberrypi ~ $ cd wiringpi  
pi@raspberrypi ~ $ ./build
```

설치가 완료되면 gpio -v를 입력해서 다음을 확인할 수 있습니다.



```
pi@raspberrypi ~  
pi@raspberrypi ~ $ gpio -v  
gpio version: 2.26  
Copyright (c) 2012-2015 Gordon Henderson  
This is free software with ABSOLUTELY NO WARRANTY.  
For details type: gpio -warranty  
  
Raspberry Pi Details:  
Type: Model B+, Revision: 1.2, Memory: 512MB, Maker: Sony  
pi@raspberrypi ~ $
```

GPIO 핀 맵 확인하기

pi@raspberrypi ~ \$ gpio readall

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

이 중, wPi라고 적힌 부분이 wiringPi의 핀 번호이며, BCM은 파이썬에서 사용하게 될 핀 번호입니다.

만약, 설치가 되지 않을 경우에는 수동으로 설치할 수 있습니다.

<https://git.drogon.net/?p=wiringPi;a=summary>에 접속해 최신로그의 snapshot을 클릭합니다.

shortlog

```

2015-03-08 Gordon Henderson Updated the board types to cope with an 0014 version... master
2015-02-03 Gordon Henderson OK. So the Pi v2 I have had older firmware and it wasn...
2015-01-31 Gordon Henderson OK, so an easier way to manage versions.
2015-01-30 Gordon Henderson Updates to the build process
2015-01-29 Gordon Henderson Updated a technicality in softPwm, and added a suggest...
2015-01-07 Gordon Henderson Updated versions for new maker - mbest
2014-11-10 Gordon Henderson Minor changes to the PiGlow code - got the orange ...
2014-07-17 Gordon Henderson Fixed a bug in the gpio readall command on model B...
2014-07-14 Gordon Henderson Updated mostly to the gpio readall command to support...
2014-06-27 Gordon Henderson Fixed a small bug in the ISR code where it was looking...
2014-06-24 Gordon Henderson Bumped version to 2.15
2014-06-24 Gordon Henderson Updates for the Raspberry Pi Compute Module - changes...
2014-05-20 Gordon Henderson changed to pin mode to support softPwm.
2013-08-03 Gordon Henderson Added some tweaks to gpio to set alt modes on pins...
2013-07-28 Gordon Henderson Bumped version
2013-07-28 Gordon Henderson It helps if you add the files into GIT...

```

heads

7 weeks ago [master](#) [shortlog](#) [log](#) [tree](#)

wiringPi - A 'wiring' like library for the Raspberry Pi

그러면 압축파일이 Downloads 폴더에 저장이 되며, 터미널을 사용하여 압축을 풀고 wiringPi를 설치할 수 있습니다.

압축풀기

```
pi@raspberrypi ~ $ cd downloads
pi@raspberrypi ~ $ ls
pi@raspberrypi ~ $ tar xfz wiringPi-5edd177.tar.gz
pi@raspberrypi ~ $ ls
pi@raspberrypi ~ $ cd wiringPi-5edd177
pi@raspberrypi ~ $ ./bulid
```

```
File Edit Tabs Help
pi@raspberrypi ~ $ cd Downloads
pi@raspberrypi ~/Downloads $ ls
wiringPi-5edd177.tar.gz
pi@raspberrypi ~/Downloads $ tar xfz wiringPi-5edd177.tar.gz
pi@raspberrypi ~/Downloads $ ls
wiringPi-5edd177 wiringPi-5edd177.tar.gz
pi@raspberrypi ~/Downloads $ cd wiringPi-5edd177
pi@raspberrypi ~/Downloads/wiringPi-5edd177 $ ./build
wiringPi Build script
=====
wiringPi Library
[UnInstall]
[Compile] wiringPi.c
[Compile] wiringSerial.c
[Compile] wiringShift.c
[Compile] piHiPri.c
[Compile] piThread.c
[Compile] wiringPiSPI.c
[Compile] wiringPiI2C.c
[Compile] softPwm.c
[Compile] softTone.c
[Compile] mcp23008.c
```

설치가 완료가 된 후에는 git을 사용했을 때와 마찬가지로 gpio -v와 gpio readall를 입력해서 GPIO가 잘 설치되었는지 확인할 수 있습니다.

프로그래밍을 위해 새로운 디렉토리와 파일을 생성해보겠습니다.

소스코드 준비

```
pi@raspberrypi ~ $ mkdir ~/c_sources
pi@raspberrypi ~ $ cd ~/c_sources
pi@raspberrypi ~ $ sudo nano servo.c
```

servo.c

```
#include <wiringPi.h> //wiringPi.h 선언
#include <stdio.h>
#include <stdlib.h>
#include <softPwm.h>

int main (int argc, char *argv[])
{
    int pos = 180 ;
    int dir = 1 ;
    if (wiringPiSetup() == -1) exit(1) ; //init wiringPi

    pinMode(0, OUTPUT) ; //set the 0 pin as OUTPUT
    digitalWrite(0, LOW) ; //0 pin output LOW voltage
    softPwmCreate(0, 0, 200) ; //pwm initialize HIGH time 0, LOW time 200ms

    while(1) {
        pos += dir ;
        if (pos < 180 || pos > 194) dir *= -1 ;
        softPwmWrite(0, pos) ;
        delay(50) ;
    }
    return 0 ;
}
```

컴파일 후 실행하기

```
pi@raspberrypi ~ /c_sources $ gcc servo.c -o servo -lwiringPi
pi@raspberrypi ~ /c_sources $ sudo ./servo
```

종료하기

Ctrl + C

설명

softPwm을 이용하여 pwm신호를 만들어 주었습니다. softPwmCreate를 이용하여 200ms의 주기를 만들어주고 softPwmWrite를 이용하여 값을 써줍니다. 180 혹은 194에 다다랐을 때 -1을 곱하여 역방향으로 돌아가게 합니다.

2. 파이썬 코딩

소스코드 준비

```
pi@raspberrypi ~ $ mkdir ~/py_sources
pi@raspberrypi ~ $ cd ~/py_sources
pi@raspberrypi ~ $ sudo nano servo.py
```

servo.py

```
import RPi.GPIO as gpio
import time

gpio.setmode(gpio.BCM)
gpio.setup(27, gpio.OUT)
val = 1
inc = 0.1
try :
    while True :
        gpio.output(27, False)
        time.sleep(val / 1000.0)
        gpio.output(27, True)
        time.sleep((20 - val) / 1000.0)
        val += inc
        time.sleep(0.05)
        if val > 2 or val < 0.6 :
            inc *= -1
except KeyboardInterrupt :
    gpio.cleanup()
```

실행하기

```
pi@raspberrypi ~ /py_sources $ sudo python servo.py
```

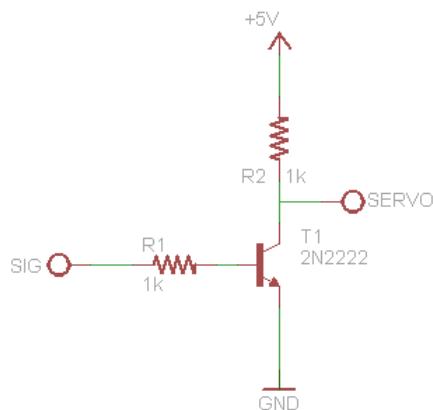
종료하기

```
Ctrl + C
```

설명

c에서 사용한 라이브러리 없이 `time.sleep`과 `gpio.output`만을 사용하였습니다. 주기는 똑같이 20ms이며 서보에 써주는 신호값이 2ms 혹은 0.6ms에 다다랐을 경우 방향을 바꿔 줍니다.

자세히 보시면 신호가 반대로 되어있는 것을 알 수 있습니다. `True`가 짧은 시간이 되어야 하고 `False`가 긴 시간이 되어야 하는데 위 코드에서는 반대로 되어 있습니다. 이는 왜일까요?

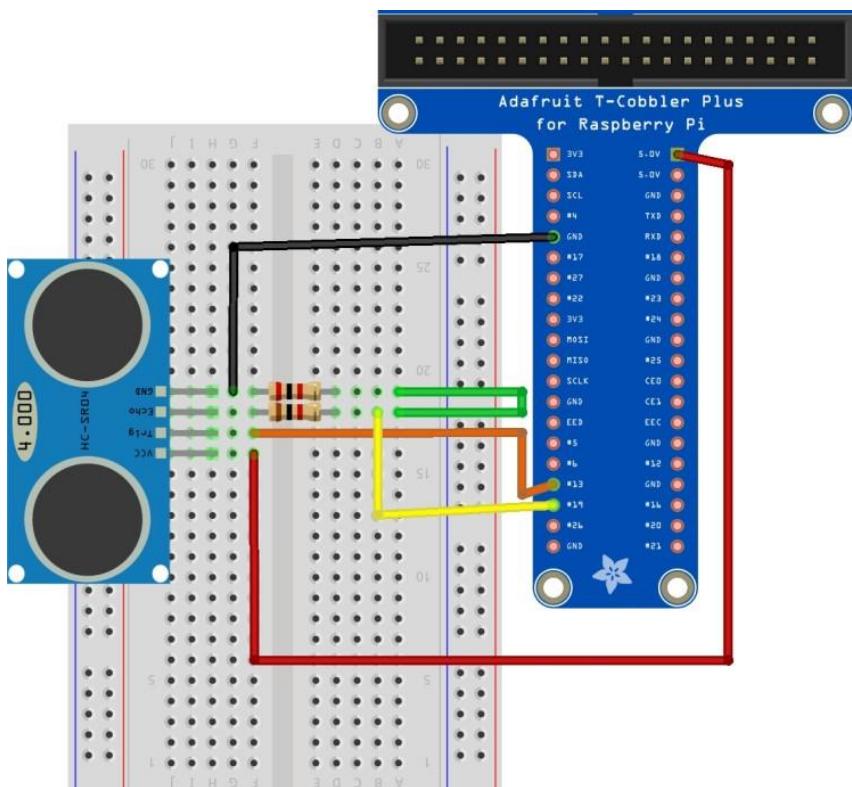


회로를 보면 SIG핀이 HIGH가 되었을 시 TR이 동작합니다. 그러면 5V에서 신호를 받던 SERVO부분이 LOW가 됩니다. 반대로 SIG핀이 LOW가 되면 TR이 동작하지 않습니다. 그러면 SERVO핀은 HIGH상태가 되겠지요? 그래서 `False`가 먼저오고 `True`가 나중에 오는 것입니다.

C. 초음파 거리센서 사용하기

디지털 신호를 사용하는 센서중 하나인 초음파 센서(HC-SR04)를 사용해 보도록 하겠습니다. 거리 측정으로 대표적인 센서이며 trig핀과 echo핀을 이용하여 사용합니다. 5V를 사용하는 센서이기에 회로 구성이 필요하나 적외선과 비교하여 주변 환경의 영향을 덜 받기 때문에 많이 사용되고 있습니다.

회로도



fritzing

trig핀은 wPi 기준 23번핀 (BCM기준 13번핀)에 연결해 줍니다.

echo핀은 wPi 기준 24번핀 (BCM기준 19번핀)에 연결해 줍니다.

1. C 코딩

소스코드 준비

```
pi@raspberrypi ~ $ cd ~/c_sources
pi@raspberrypi ~ $ sudo nano ultra.c
```

ultra.c

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
    int trig = 23 ;
    int echo = 24 ;
    int start_time, end_time ;
    float distance ;

    if (wiringPiSetup() == -1) exit(1) ;

    pinMode(trig, OUTPUT) ;
    pinMode(echo , INPUT) ;

    while(1) {
        digitalWrite(trig, LOW) ;
        delay(500) ;
        digitalWrite(trig, HIGH) ;
        delayMicroseconds(10) ;
        digitalWrite(trig, LOW) ;
        while (digitalRead(echo) == 0) ;
        start_time = micros() ;
        while (digitalRead(echo) == 1) ;
        end_time = micros() ;
        distance = (end_time - start_time) / 29. / 2. ;
        printf("distance %.2f cm\n", distance) ;
    }
    return 0 ;
}
```

컴파일 후 실행하기

```
$gcc ultra.c -o ultra -lwiringPi
$sudo ./ultra
```

설명

micros()라는 함수가 나왔습니다. 프로그램이 시작된 후 시간이 얼마나 흘렀는지를 리턴하는 함수입니다. 이를 이용하여 신호가 바뀌었을 때 시간을 저장하여 초음파 센서의 값을 읽어들입니다. 여기서 입력받은 시간값을 $/ 29 / 2$ 로 나눠서 cm단위로 변환해 줍니다. $/ 29 / 2$ 는 HC-SR04의 데이터시트에 보면 변환 공식이라고 친절하게 설명되어 있습니다. 이 값을 마지막에 $\%.2f$ 를 이용하여 값을 소수점 이하 2번째 자리까지 출력해 주면서 while문 1사이클이 종료됩니다.



```
pi@raspberrypi: ~/c_sources
pi@raspberrypi ~/c_sources $ gcc ultra.c -o ultra -lwiringPi
pi@raspberrypi ~/c_sources $ sudo ./ultra
distance 12.67 cm
distance 12.76 cm
distance 12.74 cm
distance 12.74 cm
distance 10.91 cm
distance 10.34 cm
distance 6.22 cm
distance 5.76 cm
distance 5.36 cm
distance 5.53 cm
distance 4.67 cm
distance 4.21 cm
distance 6.03 cm
```

2. 파이썬 코딩

소스코드 준비

```
pi@raspberrypi ~ $ cd ~/py_sources
pi@raspberrypi ~ $ sudo nano ultra.py
```

ultra.py

```
import RPi.GPIO as gpio
import time

gpio.setmode(gpio.BCM)

trig = 13
echo = 19

print "start"

gpio.setup(trig, gpio.OUT)
gpio.setup(echo, gpio.IN)

try :
    while True :
        gpio.output(trig, False)
        time.sleep(0.5)

        gpio.output(trig, True)
        time.sleep(0.00001)
        gpio.output(trig, False)

        while gpio.input(echo) == 0 :
            pulse_start = time.time()

        while gpio.input(echo) == 1 :
            pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start
        distance = pulse_duration * 17000
        distance = round(distance, 2)

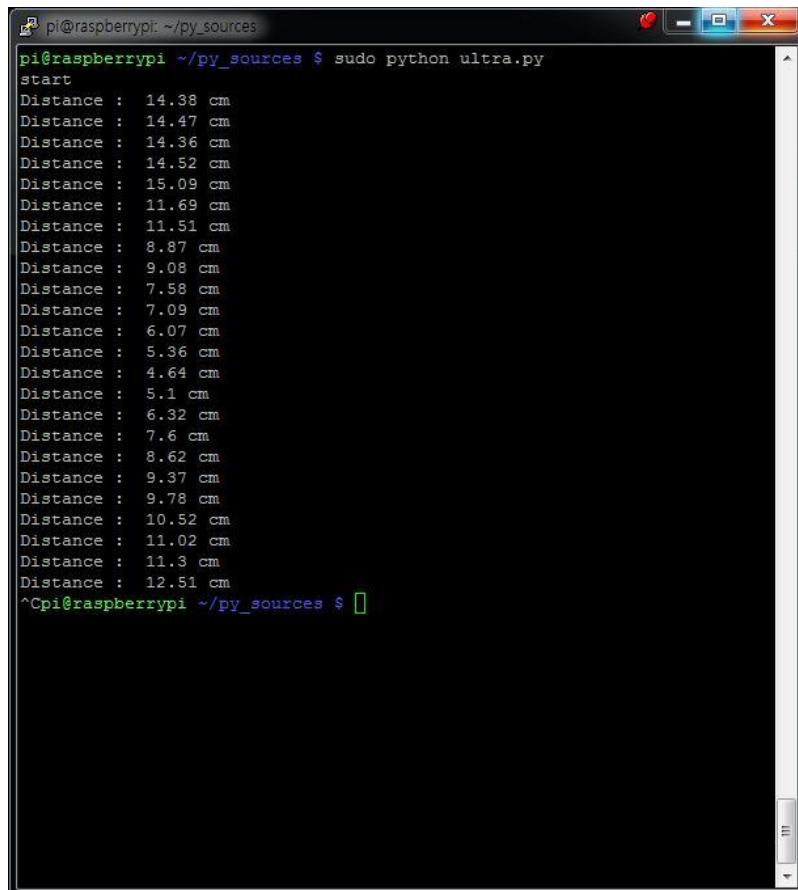
        print "Distance : ", distance, "cm"
except :
    gpio.cleanup()
```

실행하기

```
$sudo python ultra.py
```

설명

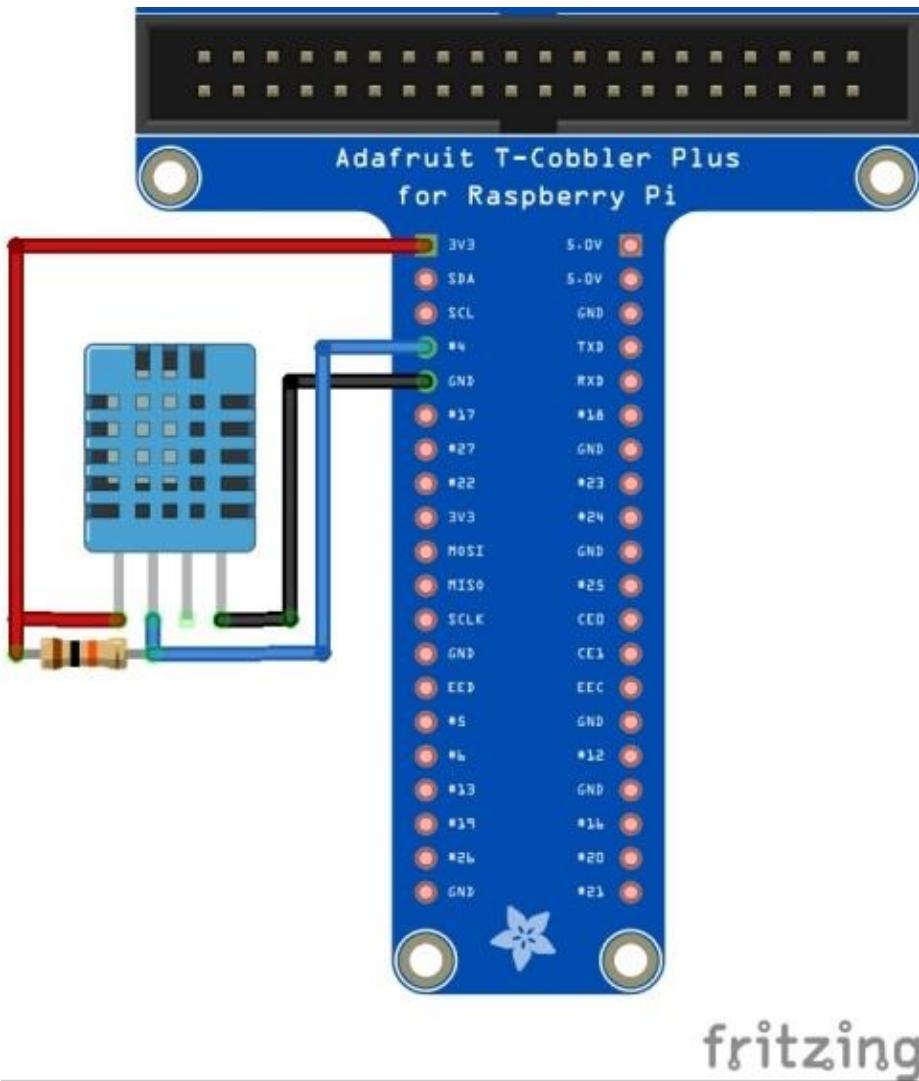
c 부분에서는 micros()라는 함수를 이용했으나 파이썬에서는 time.time()이라는 함수를 이용하였습니다. 1970년 1월 1일 이후 누적된 초를 float 단위로 리턴하여 줍니다. 시간값을 비교한다는 점은 c 와 같으나 차이점이 하나 있습니다. 값을 나눠주는 것이 아니라 17000이라는 값을 곱해줍니다. 이는 초음파의 속도가 약 340m/s 그리고 왕복시간이기에 /2 그리고 cm 단위로 바꿔주기에 17000이 된 것입니다. 이렇게 해서 바로 출력하면 소숫점 아래 약 10 자리정도가 나와 지저분해 보이므로 round 를 이용하여 소숫점 아래 3 째자리에서 반올림하여 줍니다.



```
pi@raspberrypi:~/py_sources $ sudo python ultra.py
start
Distance : 14.38 cm
Distance : 14.47 cm
Distance : 14.36 cm
Distance : 14.52 cm
Distance : 15.09 cm
Distance : 11.69 cm
Distance : 11.51 cm
Distance : 8.87 cm
Distance : 9.08 cm
Distance : 7.58 cm
Distance : 7.09 cm
Distance : 6.07 cm
Distance : 5.36 cm
Distance : 4.64 cm
Distance : 5.1 cm
Distance : 6.32 cm
Distance : 7.6 cm
Distance : 8.62 cm
Distance : 9.37 cm
Distance : 9.78 cm
Distance : 10.52 cm
Distance : 11.02 cm
Distance : 11.3 cm
Distance : 12.51 cm
^Cpi@raspberrypi:~/py_sources $
```

D. DHT11(온습도 센서) 사용하기

DHT11은 온도와 습도를 동시에 측정할 수 있는 센서로서, 디지털 방식을 채택하고 있습니다. 4개의 핀이 있으나 이중 3개만 사용합니다. 신호선은 풀업저항으로 상시 HIGH상태를 유지하고 있습니다.



신호선을 wPi 기준 7번핀 (BCM기준 4번핀)에 연결해 줍니다.

소스코드 준비

```
pi@raspberrypi ~ $ cd ~/c_sources
pi@raspberrypi ~ $ sudo nano dht.c
```

dht.c

```
#include <wiringPi.h>

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

#define MAXTIMINGS 83
#define DHTPIN 7

int dht11_dat[5] = {0, } ;

void read_dht11_dat()
{
    uint8_t laststate = HIGH ;
    uint8_t counter = 0 ;
    uint8_t j = 0, i ;
    uint8_t flag = HIGH ;
    uint8_t state = 0 ;
    float f ;

    dht11_dat[0] = dht11_dat[1] = dht11_dat[2] = dht11_dat[3] = dht11_dat[4] = 0 ;

    pinMode(DHTPIN, OUTPUT) ;
    digitalWrite(DHTPIN, LOW) ;
    delay(18) ;

    digitalWrite(DHTPIN, HIGH) ;
    delayMicroseconds(30) ;
    pinMode(DHTPIN, INPUT) ;

    for (i = 0; i < MAXTIMINGS; i++) {
        counter = 0 ;
        while ( digitalRead(DHTPIN) == laststate) {
            counter++ ;
            delayMicroseconds(1) ;
            if (counter == 200) break ;
        }
        laststate = digitalRead(DHTPIN) ;
        if (counter == 200) break ; // if while breaked by timer, break for
        if ((i >= 4) && (i % 2 == 0)) {
```

```

dht11_dat[j / 8] <<= 1 ;
if (counter > 20) dht11_dat[j / 8] |= 1 ;
j++ ;
}
}
if ((j >= 40) && (dht11_dat[4] == ((dht11_dat[0] + dht11_dat[1] + dht11_dat[2] +
dht11_dat[3]) & 0xff))) {
    printf("humidity = %d.%d %% Temperature = %d.%d *C \n", dht11_dat[0],
dht11_dat[1], dht11_dat[2], dht11_dat[3]) ;
}
else printf("Data get failed\n") ;

}

int main(void)
{
    printf("dht11 Raspberry pi\n");
    if (wiringPiSetup() == -1) exit(1);

    while (1) {
        read_dht11_dat();
        delay(1000);
    }
    return 0;
}

```

컴파일 후 실행하기

```
$gcc dht.c -o dht -lwiringPi
$sudo ./dht
```

이번 코드에서는 변수가 uint8_t라고 되어 있습니다. 이는 user-defined type이라고 부르며 unsigned 8 bit type 변수를 의미합니다.

DHT11은 여타 센서와 다르게 신호의 길이로 데이터를 내보냅니다. 처음에 신호선으로 LOW를 18ms동안, 그리고 HIGH 신호를 20 ~ 40us동안 주면 start 신호입니다.

여기서는 중간값인 30us만큼 HIGH신호를 주었습니다. 그러면 DHT11에서 신호를 내보내기 시작합니다. 총 83개의 신호가 오가는데 이는 처음 3개의 비트(Response, pullup ready, start to transmit bit) + 40개의 데이터비트 * 2(LOW, HIGH 반복 1번으로 1개의 신호를 나타내기 때문)로 이루어 져 있습니다.

if 연산식을 보면 처음 3개는 데이터와 관계가 없기에 $i \geq 4$ 를 써주었고 $i \% 2 == 0$ 을 해서 데이터를 나타내는 HIGH상태에서만 작동하도록 합니다. 데이터를 나타내는 HIGH신호는 26~28us면 0, 70us면 1을 의미합니다. 하나 연산시간이 있기에 여기서는 counter가 20이 넘어가는 것을 기준으로 하였습니다.

The screenshot shows a terminal window titled "pi@raspberrypi: ~/c_sources". The window contains the following text:

```
pi@raspberrypi:~/c_sources $ gcc dht.c -o dht -lwiringPi
pi@raspberrypi:~/c_sources $ sudo ./dht
dht11 Raspberry pi
Data get failed
humidity = 35.0 % Temperature = 24.0 *C
Data get failed
humidity = 34.0 % Temperature = 24.0 *C
humidity = 33.0 % Temperature = 24.0 *C
humidity = 33.0 % Temperature = 24.0 *C
Data get failed
humidity = 33.0 % Temperature = 24.0 *C
humidity = 32.0 % Temperature = 25.0 *C
humidity = 33.0 % Temperature = 24.0 *C
```

E. ADC를 이용해 조도센서 사용하기

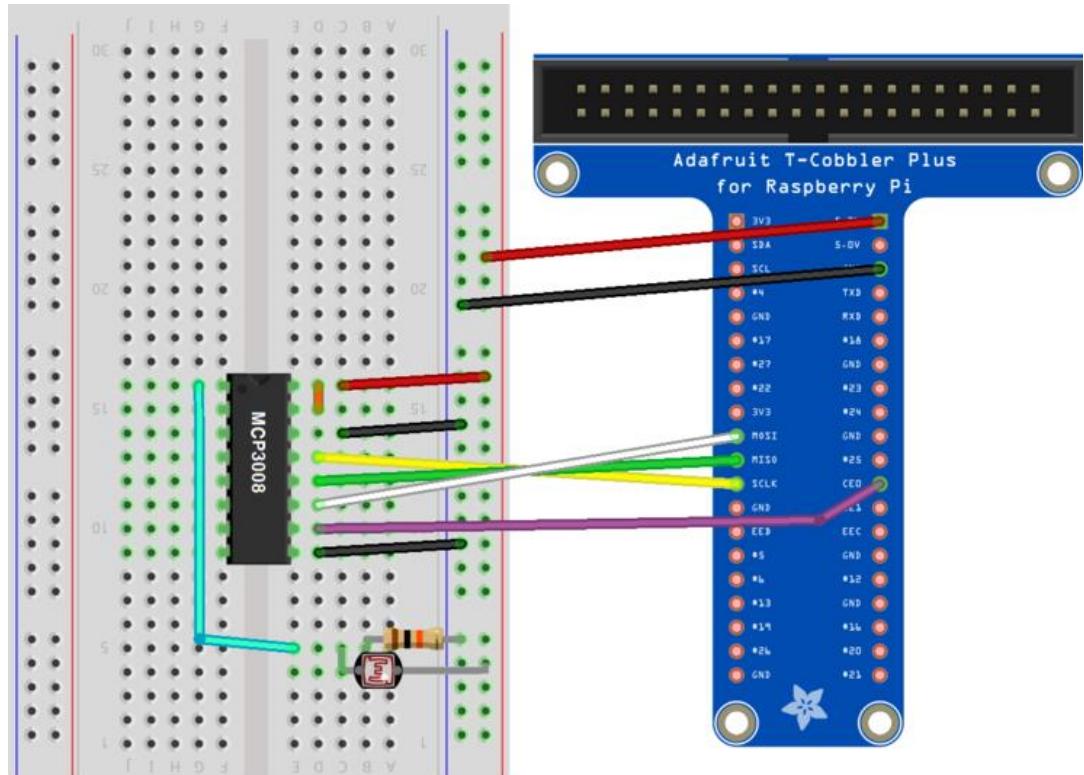
조도센서는 빛의 세기에 따라 저항값이 변하는 소자로써, 다음과 같이 사용하여 빛의 세기를 측정할 수 있습니다.

아두이노와 달리, 라즈베리파이에서는 아날로그 신호를 읽기 위해서 외부 ADC를 사용해야 합니다. ADC는 Analog to Digital Converter의 약자로 아날로그 신호를 디지털로 변환해 줍니다. 여기서는 8채널 입력, SPI출력을 가진 MCP3008 칩을 사용해 보도록 하겠습니다.

CH0	1	16	V _{DD}
CH1	2	15	V _{REF}
CH2	3	14	AGND
CH3	4	13	CLK
CH4	5	12	D _{OUT}
CH5	6	11	D _{IN}
CH6	7	10	CS/SHDN
CH7	8	9	DGND

MCP3008

1. 라즈베리파이 회로도 연결



2. 라즈베리파이 SPI 설정

ADC를 통해 변환된 아날로그 신호를 라즈베리파이에서 SPI통신 프로토콜을 사용하여 읽기 위해서는 라즈베리파이에서 SPI를 사용할 수 있게 해주어야 합니다.

SPI 사용 설정

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

→ 8. Advanced Options → A6 SPI → Yes and OK

```
pi@raspberrypi ~ $ sudo nano /etc/modules
```

파일 끝부분에 spidev 추가

```
pi@raspberrypi ~ $ cd ~
```

```
pi@raspberrypi ~ $ sudo apt-get install python-dev
```

```
pi@raspberrypi ~ $ git clone git://github.com/doceme/py-spidev
```

```
pi@raspberrypi ~ $ cd py-spidev/
```

```
pi@raspberrypi ~ $ sudo python setup.py install
```

```
pi@raspberrypi ~ $ sudo shutdown -r now
```

3. 라즈베리파이 코드 구성 및 실행

소스코드 준비

```
pi@raspberrypi ~ $ cd ~/py_sources
```

```
pi@raspberrypi ~ $ sudo nano MCP3008.py
```

MCP3008.py

```
import spidev, time

spi = spidev.SpiDev()
spi.open(0,0)

def analog_read(channel):
    r = spi.xfer2([1, (8 + channel) << 4, 0])
    adc_out = ((r[1]&3) << 8) + r[2]
    return adc_out

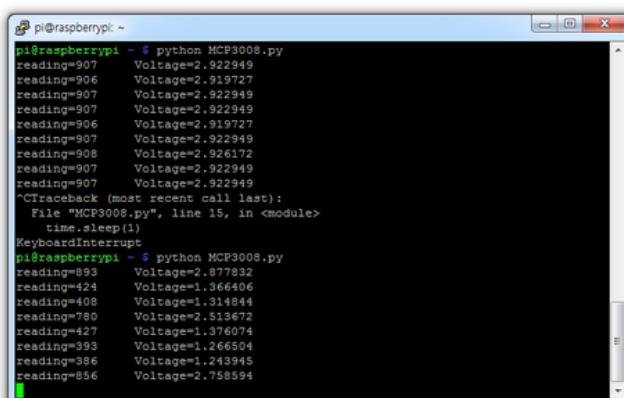
while True:
    reading = analog_read(0)
    voltage = reading * 3.3 / 1024
    print("Reading=%d\tVoltage=%f" % (reading, voltage))
    time.sleep(1)
```

실행하기

```
pi@raspberrypi ~ $ sudo python MCP3008.py
```

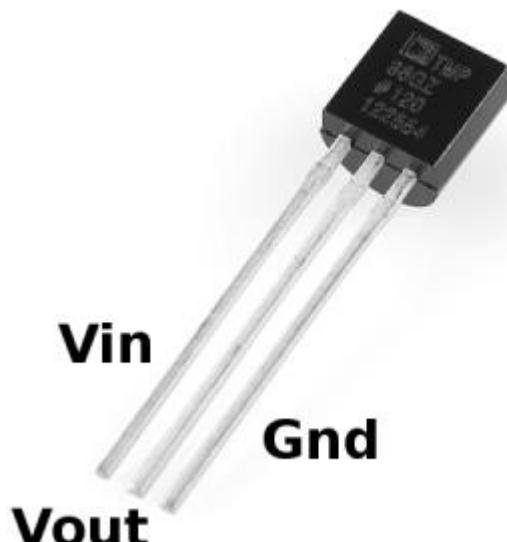
설명

`analog_read`라는 함수를 만들고 값을 읽어들입니다. `spi.xfer2`라는 함수는 안에 있는 데이터를 쓰고 리턴된 값을 반환합니다. 각각 시작비트, 몇번째 데이터를 불러올 것인지, 그리고 자릿수를 맞추기 위한 값입니다. 반환되는 `r`은 첫 8비트는 비어있으며 두번째 8비트중 1, 2번째 비트는 ADC 데이터의 9, 10번째 비트, 세번째는 하위 8비트를 의미합니다. 이를 더함으로써 총 10비트의 데이터가 완성되었습니다. 이 값에 3.3을 곱하고 1024로 나눠줌으로써 들어오는 전압이 약 몇V인지 계산하였습니다.



F. ADC를 이용해 온도센서 사용하기 (파이썬 코딩)

TMP36 온도센서는 3핀으로 구성된 아날로그 소자로써, 온도에 따라서 아날로그 값이 Vout핀을 통해서 출력됩니다. 위 예제를 이용하여 아날로그 값을 읽어들이며 공식을 이용하여 온도값도 출력합니다.



조도센서와 동일한 회로에 추가 연결한다면, MCP3008의 CH1에 TMP36 온도센서의 Vout을 연결하고, Vin에는 3.3V, GND는 라즈베리파이의 GND로 연결하면 온도 값을 확인할 수 있습니다.

소스코드 준비

```
pi@raspberrypi ~ $ cd ~/py_sources
pi@raspberrypi ~ $ sudo nano MCP3008_tmp36.py
```

MCP3008_tmp36.py

```
import spidev
import time
import os

spi = spidev.SpiDev()
```

```
spi.open(0,0)

def ReadChannel(channel):
    adc = spi.xfer2([1,(8+channel)<<4,0])
    data = ((adc[1]&3) << 8) + adc[2]
    return data

def ConvertVolts(data,places):
    volts = (data * 3.3) / float(1023)
    volts = round(volts,places)
    return volts

def ConvertTemp(data,places):

    # ADC Value
    # (approx) Temp Volts
    #   0      -50  0.00
    #  78      -25  0.25
    # 155       0  0.50
    # 233      25  0.75
    # 310      50  1.00
    # 465     100  1.50
    # 775     200  2.50
    # 1023    280  3.30

    temp = ((data * 330)/float(1023))-50
    temp = round(temp,places)
    return temp

light_channel = 0
temp_channel = 1

delay = 5

while True:

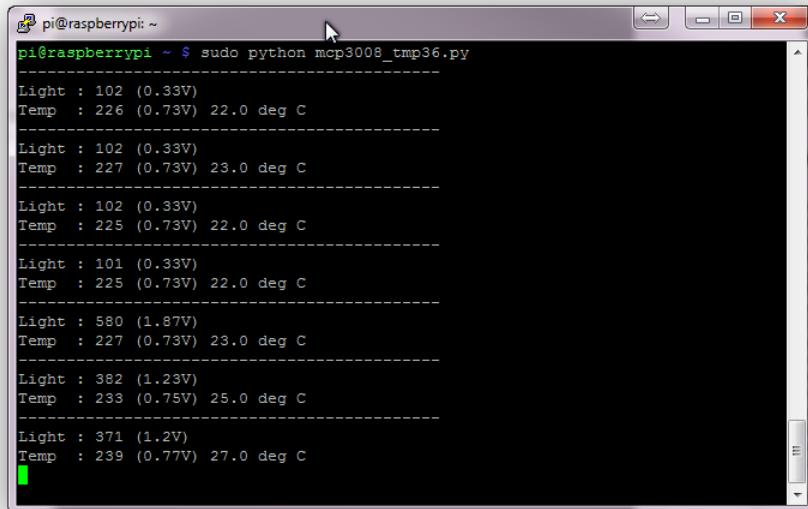
    light_level = ReadChannel(light_channel)
    light_volts = ConvertVolts(light_level,2)

    temp_level = ReadChannel(temp_channel)
    temp_volts = ConvertVolts(temp_level,2)
    temp      = ConvertTemp(temp_level,2)
```

```
print "-----"  
print("Light: {} ({}V)".format(light_level,light_volts))  
print("Temp : {} ({}V) {} deg C".format(temp_level,temp_volts,temp))  
  
time.sleep(delay)
```

결과 출력하기

```
pi@raspberrypi ~ $ sudo python MCP3008_tmp36.py
```



The screenshot shows a terminal window titled 'pi@raspberrypi: ~'. The command entered is 'sudo python MCP3008_tmp36.py'. The output displays five sets of sensor readings, each consisting of Light level and Temp (in degrees Celsius). The light levels range from 101 to 580, and the temperatures range from 22.0 to 27.0 degrees Celsius.

```
pi@raspberrypi ~ $ sudo python MCP3008_tmp36.py  
-----  
Light : 102 (0.33V)  
Temp : 226 (0.73V) 22.0 deg C  
-----  
Light : 102 (0.33V)  
Temp : 227 (0.73V) 23.0 deg C  
-----  
Light : 102 (0.33V)  
Temp : 225 (0.73V) 22.0 deg C  
-----  
Light : 101 (0.33V)  
Temp : 225 (0.73V) 22.0 deg C  
-----  
Light : 580 (1.87V)  
Temp : 227 (0.73V) 23.0 deg C  
-----  
Light : 382 (1.23V)  
Temp : 233 (0.75V) 25.0 deg C  
-----  
Light : 371 (1.2V)  
Temp : 239 (0.77V) 27.0 deg C
```

6. 라즈베리파이와 Node.js

A. Node.js란?

자바스크립트 엔진위에서 돌아가는 플랫폼으로 빠르고, 가볍이 용이한 프로그램입니다.
분산된 장치 사이에서 *event-driven*, *non-blocking* 방식을 가지며 가볍고, 능률적이며
실시간 자료 집중형에 완벽하게 만들어 줍습니다. – node.js 홈페이지 발췌 –

event-driven 이란 아두이노의 인터럽트를 생각하면 쉽습니다. 순차적으로 쭉 내려가는
방식이 아니라 발생하는 이벤트에 따라 작동하는 형태입니다.

non-blocking 이란 blocking 과 반대되는 개념입니다. 쉽게 설명하자면 delay 나 sleep 등
대기하는 함수를 쓰면 일정 시간동안 아무것도 못하고 멈춰있게 됩니다. 하지만 *non-*
blocking 의 경우 그냥 다음으로 넘어가버립니다. 그리고 함수 실행이 완료되면 다시
돌아와서 작업을 합니다.

자 그러면 라즈베리에 node.js 를 설치해 보도록 하겠습니다.

Node.js 설치하기

```
pi@raspberrypi ~ $ cd
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get upgrade
pi@raspberrypi ~ $ wget http://node-arm.herokuapp.com/node_latest_armhf.deb
pi@raspberrypi ~ $ sudo dpkg -i node_latest_armhf.deb
```

설치가 끝났습니다. 잘 설치 되었는지 확인하려면

```
$node -v
```

를 입력해주면 됩니다. 그러면 버전이 확인될 것입니다.

GPIO 접근하기 위한 셋업 및 npm 설치

```
pi@raspberrypi ~ $ git clone git://github.com/quick2wire/quick2wire-gpio-admin.git
~/gpio-admin
pi@raspberrypi ~ $ cd ~/gpio-admin
pi@raspberrypi ~ /gpio-admin $ make
pi@raspberrypi ~ /gpio-admin $ sudo make install
pi@raspberrypi ~ /gpio-admin $ sudo adduser pi gpio
pi@raspberrypi ~ /gpio-admin $ exec su -l pi
pi@raspberrypi ~ $ npm install onoff
```

여기까지 완료되었다면 이제 코딩을 할 시간입니다.

Hello World 로 테스트를 해보겠습니다.

GPIO 접근하기 위한 셋업 및 npm 설치

```
pi@raspberrypi ~ $ mkdir js  
pi@raspberrypi ~ $ cd js  
pi@raspberrypi ~ /js $ sudo nano hello.js
```

에디터 안에 다음을 입력한 후에 Ctrl+X 를 사용하여 저장하고 에디터를 닫습니다.

```
console.log('Hello World!') ;
```

그리고 실행합니다.

실행하기

```
pi@raspberrypi ~ /js $ node hello.js
```

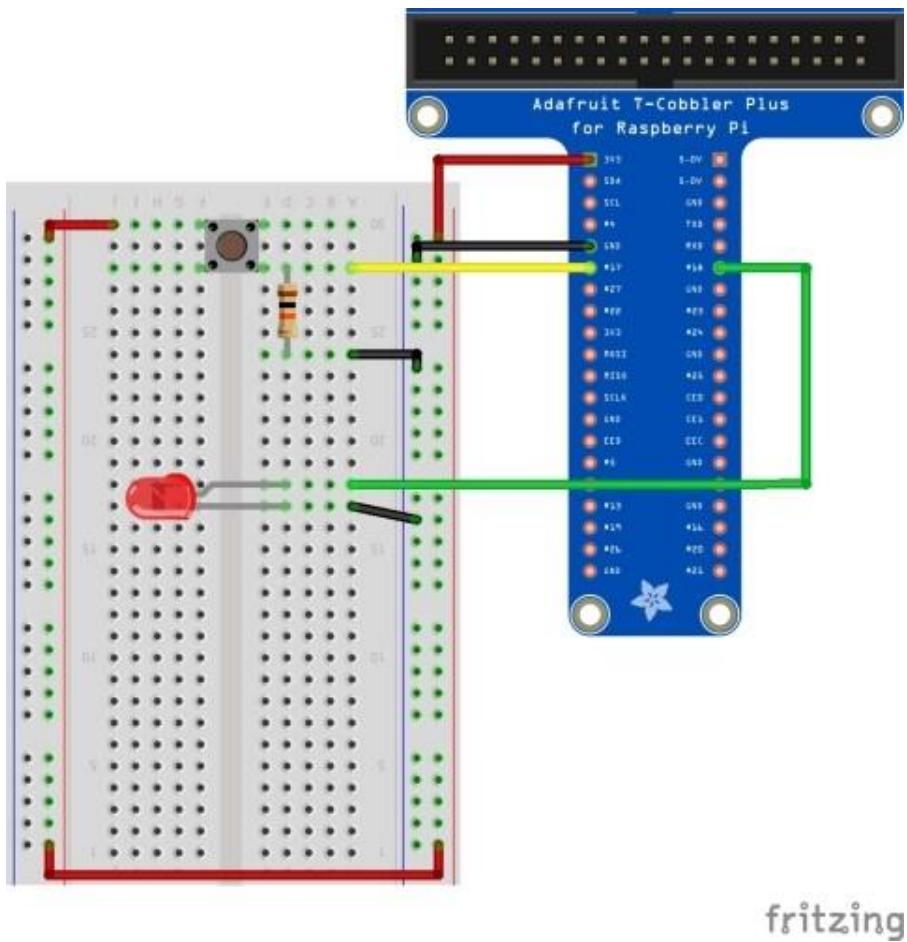
The screenshot shows a terminal window titled 'pi@raspberrypi: ~/js'. The window contains the following text:

```
pi@raspberrypi:~/js$ node hello.js
Hello World!
pi@raspberrypi:~/js$
```

B. LED 제어하기

Node.js를 사용하여 LED를 제어하는 데모를 구현해보도록 하겠습니다.

라즈베리파이 회로도 연결



17 번에 스위치가 연결되어 있으며 풀다운저항을 달았습니다.

눌리면 HIGH, 아니라면 LOW 입니다.

그리고 18 번핀에는 LED 가 연결되어 있습니다.

소스코드 led.js

```
var GPIO = require('onoff').Gpio,
    led = new GPIO(18, 'out'),
    button = new GPIO(17, 'in', 'both') ;

function light(err, state) {
  if (state == 1) {
    led.writeSync(1) ;
    console.log('on') ;
  }
  else {
    led.writeSync(0) ;
    console.log('off') ;
  }
}

console.log('start') ;
button.watch(light) ;
```

특별히 눈여겨 봐야할 것은 `button.watch(light)`입니다.

`watch`는 상태가 바뀌었을 때 실행되는 이벤트로 `button` 상태가 바뀌면 `light` 함수를 실행합니다.

인자는 `err`(에러코드)와 `state`(상태)로 이루어 져 있습니다.

이 값은 `watch`에 포함되어 있는 것으로 자세한 내용은 레퍼런스를 참조 바랍니다.

<https://github.com/fivdi/onoff#watchcallback>

`state == 1` 인 경우는 스위치가 눌렸을 때 즉 HIGH상태일때 입니다.

0인 경우는 그 반대입니다.

코드를 다 짰으니 이제 실행을 해보겠습니다.

실행하기

```
pi@raspberrypi ~ /js $ node led.js
```

C. 웹페이지를 사용하여 Hello World 출력하기

node.js 를 설치하여 간단한 테스트를 해봤으니 이번 시간에는 웹페이지를 띄워 보도록 하겠습니다.

방식은 간단합니다. node.js 에 특정한 요청이 오면 그냥 html 파일을 뿐려주는 형식입니다.

이게 기본이 되어 여러가지를 할 수 있기에 먼저 해보도록 하겠습니다.

먼저 express 를 설치해 주어야 합니다.

express 설치하기

```
pi@raspberrypi ~ /js $ npm install express
```

hello.js 소스코드

```
var express = require('express'),
    http = require('http'),
    app = express(),
    server = http.createServer(app) ;

app.get('/hello', function(req, res) {
    res.sendfile('hello.html', {root : __dirname }) ;
})

server.listen(8000, function() {
    console.log('express server listening on port ' + server.address().port
}) ;
```

hello.html 소스코드

```
<!DOCTYPE html>
<html>
<head>
<title>Hello</title>
</head>
<body>
<h1>Hello World!</h1>
</body>
</html>
```

require 의 경우 해당 경로의 모듈을 추출합니다.
그냥 가져와서 사용한다는 의미입니다.
express 를 가져와서 express 에 저장합니다.
http 도 마찬가지입니다.

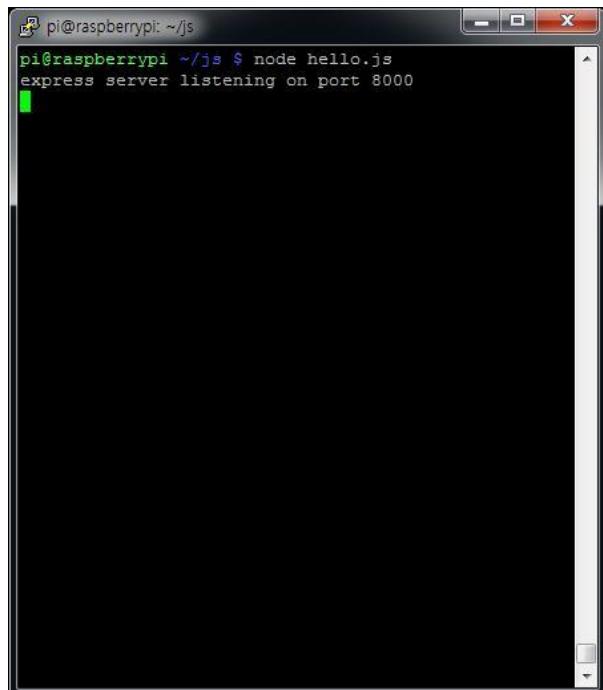
그리고 app 으로 express 를 실행합니다.

그 후 server 가 app 을 이용하여 서버를 생성합니다.
이제 서버를 쓸 수 있나구요?
아닙니다.

포트도 설정 안해줬고 뭘 할지도 안했으니까요.

```
server.listen(8000, function() {  
    console.log('express server listening on port ' + server.address().port  
}) ;
```

포트는 8000 번, 설정이 완료되면 express server listening on port 라는 문자를 출력하고 그 뒤에 포트번호를 출력합니다.



A screenshot of a terminal window titled "pi@raspberrypi: ~/.js". The window shows the command "node hello.js" being run and the output "express server listening on port 8000" displayed below it. The terminal has a standard black background with white text and a red title bar.

```
pi@raspberrypi: ~/.js  
pi@raspberrypi ~/js $ node hello.js  
express server listening on port 8000
```

그러면 포트가 설정되었습니다.

그리고 포트에 입력이 들어오면 무언가를 해주어야 합니다.

여기선 html 코드를 뿌려줄 것으로 res.sendfile 을 사용했습니다.

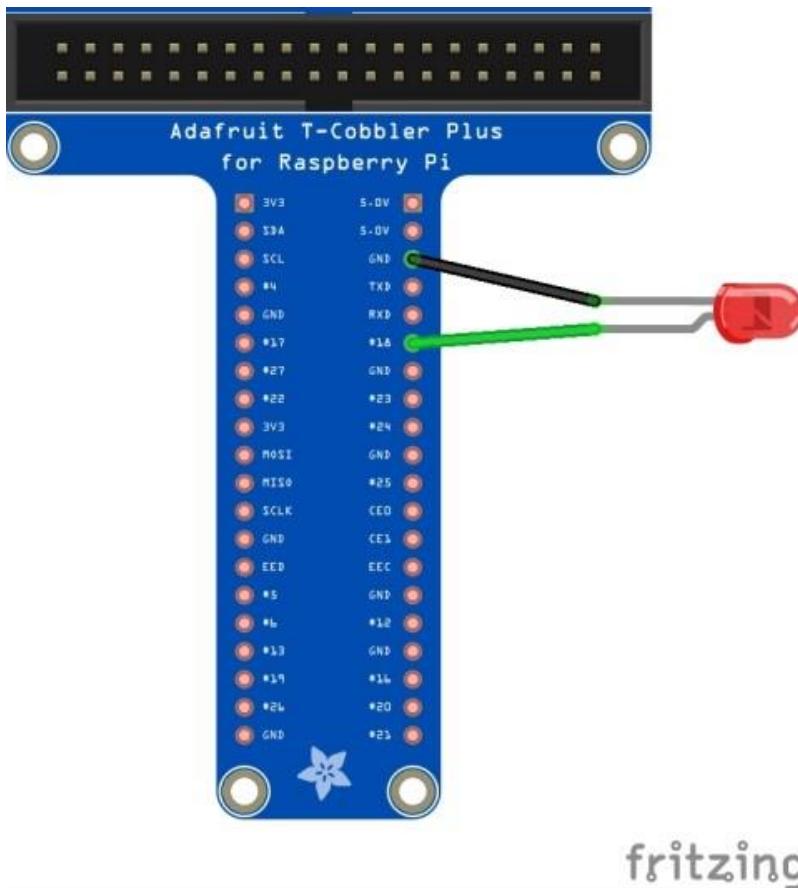
```
app.get('/hello', function(req, res) {  
    res.sendFile('hello.html', {root : __dirname }) ;  
}) ;
```

주소창에 /hello 라는 값을 받으면 hello.html 을 뿌려줍니다.

hello.html 에는 Hello World!가 출력되도록 되어 있으므로 접속해보면 다음과 같이 나올 것입니다.



D. 웹페이지를 사용하여 LED 제어하기



18 번핀에 LED 의 +, GND 에 LED 의 -를 연결해주시면 됩니다.

코딩에 앞서 저번시간에 express 를 설치해 주었던 것처럼 body-parser 를 설치해 줍니다.

body-parser 설치하기

```
pi@raspberrypi ~ /js $ npm install body-parser
```

소스코드 작성하기

```
pi@raspberrypi ~ /js $ sudo nano led_web.js
```

소스코드

```

var express = require('express'),
    http = require('http'),
    app = express(),
    server = http.createServer(app) ;
var bodyParser = require('body-parser') ;
var GPIO = require('onoff').Gpio,
    led = new GPIO(18, 'out') ;

app.use(bodyParser.json()) ;
app.use(bodyParser.urlencoded({ extended : false })) ;

app.get('/led', function(req, res) {
    res.sendFile('led_web.html', {root : __dirname }) ;
})

app.post('/data', function(req, res) {
    var state = req.body.led ;
    if (state == 'on') {
        led.writeSync(1) ;
    }
    else {
        led.writeSync(0) ;
    }
    console.log(state) ;
    res.sendFile('led_web.html', {root : __dirname }) ;
})

server.listen(8000, function() {
    console.log('Express server listening on port ' + server.address().port) ;
})

```

Html 파일 작성하기

```
pi@raspberrypi ~ /js $ sudo nano led_web.html
```

led_web.html

```

<!DOCTYPE html>
<html>
<head>
<title>led</title>
</head>
<body>
<h1>LED page</h1>
<form action = '/data' method = 'post'>
    <input type = 'submit' value = 'on' name = 'led' >
    <input type = 'submit' value = 'off' name = 'led' >
</form>
</body>
</html>

```

html은 그냥 버튼입력이 되면 값을 보내줍니다.

그럴기에 led_web.js부분이 중요한데 non-blocking 형식이라 조금 난해할 수 있습니다.
제일 먼저 변수선언 부분입니다.

```
var express = require('express'),
    http = require('http'),
    app = express(),
    server = http.createServer(app) ;
var bodyParser = require('body-parser') ;
var GPIO = require('onoff').Gpio,
    led = new GPIO(18, 'out') ;
```

첫번째 var의 경우 지난시간에 설명했으므로 넘어가도록 하겠습니다.

혹시 못보신 분들은 <http://blog.naver.com/roboholic84/220340563379> 를 참조해 주시기 바랍니다.

led는 GPIO의 18번핀을 출력으로 사용합니다.

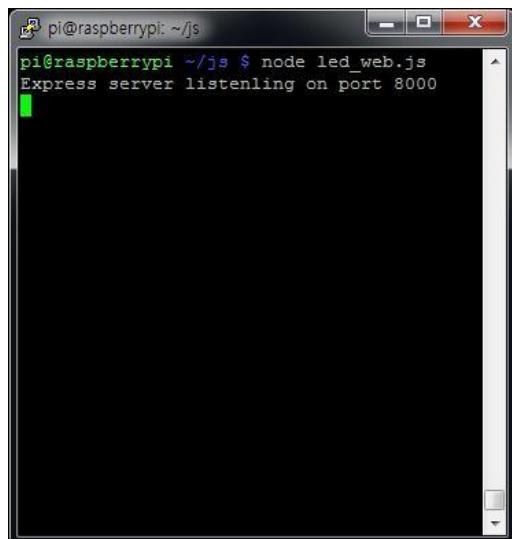
그리고 이벤트 부분입니다.

```
server.listen(8000, function() {
    console.log('Express server listening on port ' + server.address().port) ;
}) ;
```

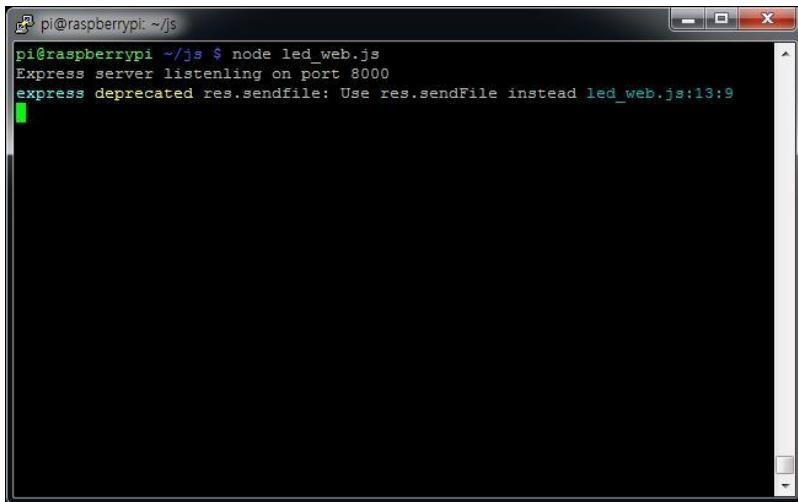
8000 번 포트로 서버를 실행합니다.

그리고 콘솔창에 Express~~~~, 여기에 서버 포트를 더하여(추가적으로) 출력합니다.

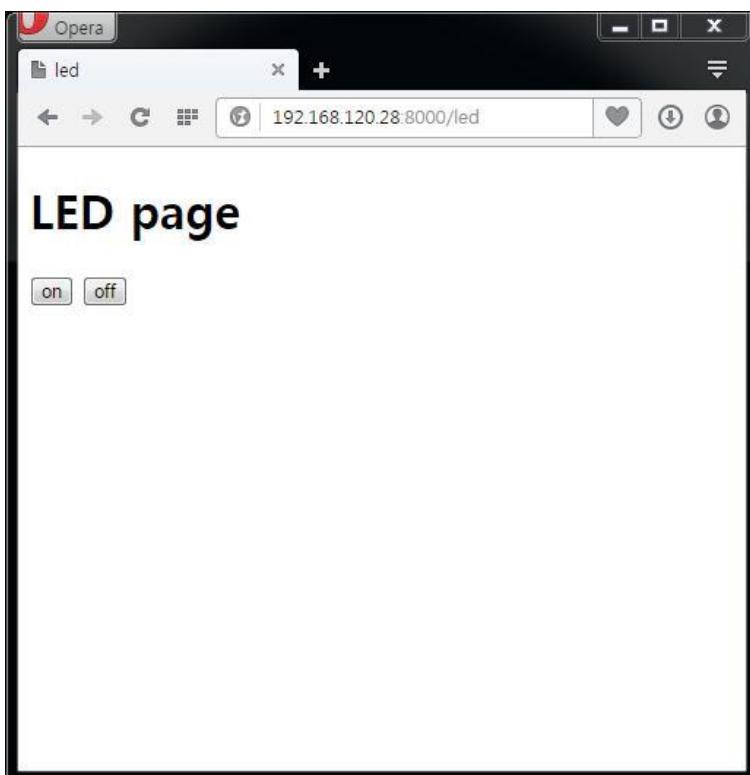
그러면 다음과 같이 나올 것입니다.



이후 8000 번 포트 그 안에 led 로 접속하게 되면 다음과 같은 창이 나옵니다.



```
pi@raspberrypi:~/js $ node led_web.js
Express server listening on port 8000
express deprecated res.sendfile: Use res.sendFile instead led_web.js:13:9
[green bar]
```



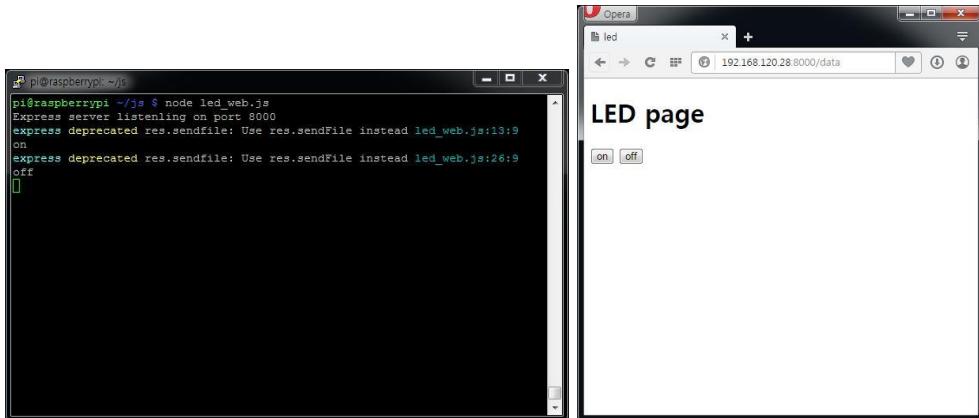
```
app.get('/led', function(req, res) {
    res.sendfile('led_web.html', {root : __dirname }) ;
}) ;
```

/led 라는 값을 받을 경우 실행됩니다.
웹페이지에는 led_web.html 을 뿌려줍니다.

```
app.post('/data', function(req, res) {
    var state = req.body.led ;
    if (state == 'on') {
        led.writeSync(1) ;
    }
    else {
        led.writeSync(0) ;
    }
    console.log(state) ;
    res.sendfile('led_web.html', {root : __dirname }) ;
}) ;
```

/data에 post로 값을 받을 경우 실행됩니다.
수신된 값중 body 부분, 그중 led 부분의 값을 변수 state에 저장한 후 처리합니다.
state 가 on 일 경우 led 를 켜고, off 일 경우 끕니다.
콘솔에는 state 를 표시하고 led_web.html 파일을 다시 뿌려줍니다.

실행 예제입니다.



on, off 를 각 한번씩 눌러보았습니다. 이때 LED 가 켜졌다 꺼졌다 합니다.

E. 시리얼 통신 사용하기

라즈베리 GPIO 의 대표적인 사용법중 하나죠

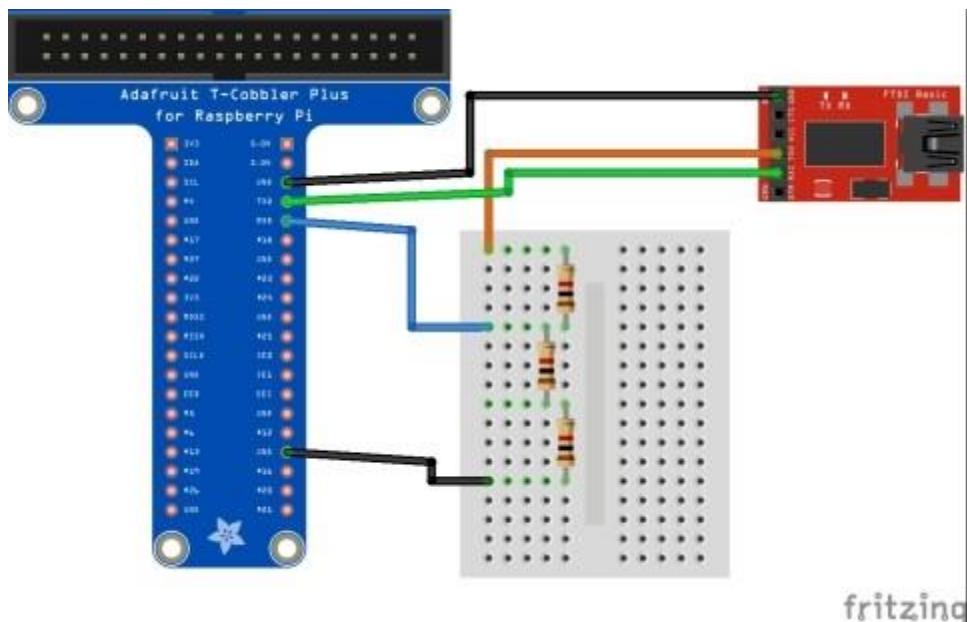
통신!

그중에서 블루투스등을 이용할 때 많이 사용되는 시리얼 통신을 사용해 보겠습니다.

usb to serial converter 를 이용하여 PC 와 통신해 보도록 하겠습니다.

컨버터가 5V 를 사용하기에 회로구성을 하나 해주어야 합니다.

회로도



간단하게 저항을 이용하였습니다.

원래는 컨버터를 사용해야 하지만 그냥 대략적인 비율을 맞춰 주었습니다.

1 종류의 저항으로도 위와같이 연결하면 충분히 5V 를 약 3.3V 로 맞추어 줄 수 있습니다.

이렇게 연결한 후 npm 중 serial 을 설치합니다.

Npm serialport 설치하기

```
pi@raspberrypi ~ $ cd ~/js
pi@raspberrypi ~ $ npm install serialport
```

이제 코딩을 해보겠습니다.

Serial.js

```
var express = require('express'),
    http = require('http'),
    app = express(),
    server = http.createServer(app) ;
var bodyParser = require('body-parser') ;
var SerialPort = require('serialport').SerialPort,
    serial = new SerialPort('/dev/ttyAMA0', {
        baudrate : 9600
    }) ;

app.use(bodyParser.json()) ;
app.use(bodyParser.urlencoded({ extended : false })) ;

app.get('/serial', function (req, res) {
    res.sendFile('serial.html', {root : __dirname }) ;
}) ;

app.post('/serial', function (req, res) {
    var val = req.body.send ;
    console.log(val) ;
    serial.write(val, function(err) {}) ;
    res.sendFile('serial.html', {root : __dirname }) ;
}) ;

serial.on('data', function(data) {
    console.log(data.toString()) ;
}) ;

server.listen(8000, function() {
    console.log('Express server listening on port ' + server.address().port) ;
}) ;
```

저번시간이랑 큰 차이가 없습니다.

바뀐 부분에 대해 설명하자면

```
var SerialPort = require('serialport').SerialPort,  
    serial = new SerialPort('/dev/ttyAMA0', {  
        baudrate : 9600  
    }) ;
```

/dev/ttyAMA0이라는 시리얼 포트를 baudrates 9600으로 사용한다는 말입니다.

그 외에도 데이터비트, 스탑비트, 패리티비트 등을 설정해주어야 하지만 디폴트값을 사용하기에 그냥 두었습니다.

여기서 디폴트값은 data = 8, stop = 1, parity - none입니다.

자세한 내용은 <https://github.com/voodootikigod/node-serialport>를 참조해주시기 바랍니다.

그 외 다음과 같은 코드가 있습니다.

```
serial.on('data', function(data) {  
    console.log(data.toString()) ;  
}) ;
```

시리얼 포트에 들어온 값이 있으면 출력하는 형태입니다.

다만 형식을 String으로 바꾸어 출력합니다.

그냥 출력할 경우 ASCII 코드가 나오기에 바꿔준 것입니다.

그러면 html 코드부분입니다.

[Serial.html](#)

```
<!DOCTYPE html>
<html>
<head>
<title>Serial node</title>
</head>
<body>
<form action = "/serial" method = "post">
<input type = "text" name = "send">
<input type = "submit" value = "send">
</form>
</body>
</html>
```

text형식으로 값을 받고 버튼을 누르면 post형식으로 값을 전송합니다.

그 외 큰 내용은 없습니다.

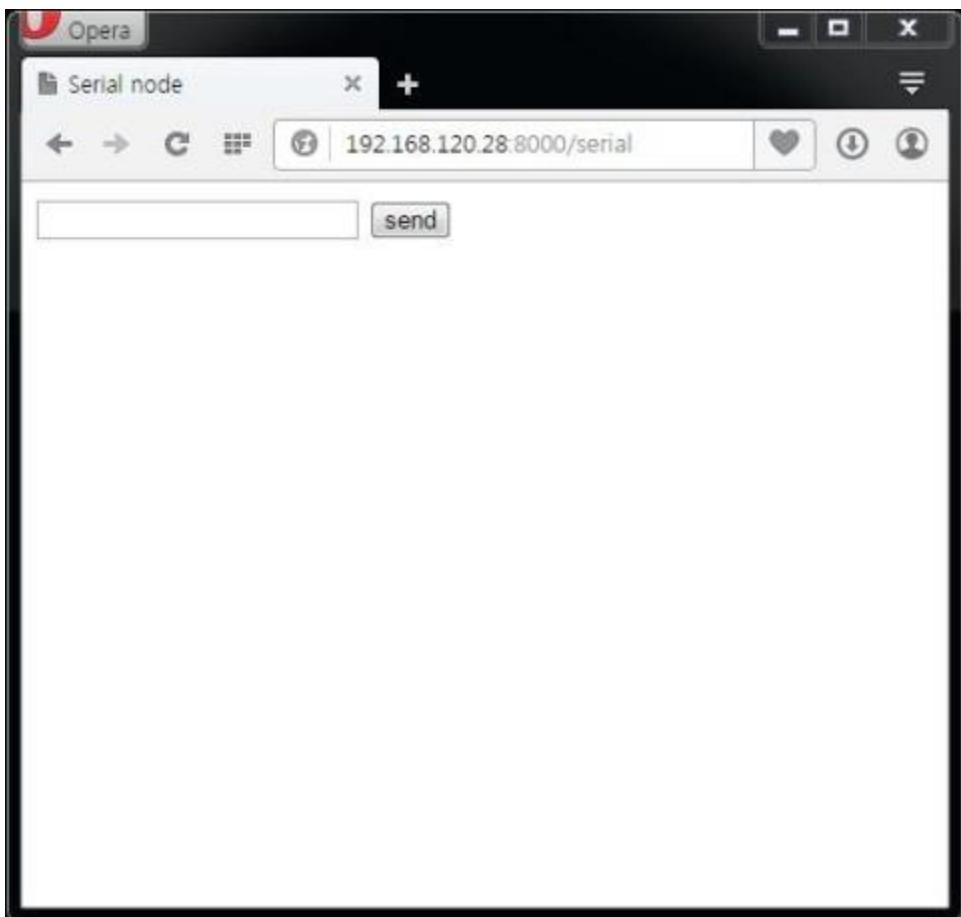
코딩이 끝났으니 사용해 보겠습니다.

시리얼 통신 프로그램은 아래 링크의 프로그램을 사용했습니다.

<http://blog.daum.net/pg365/276>

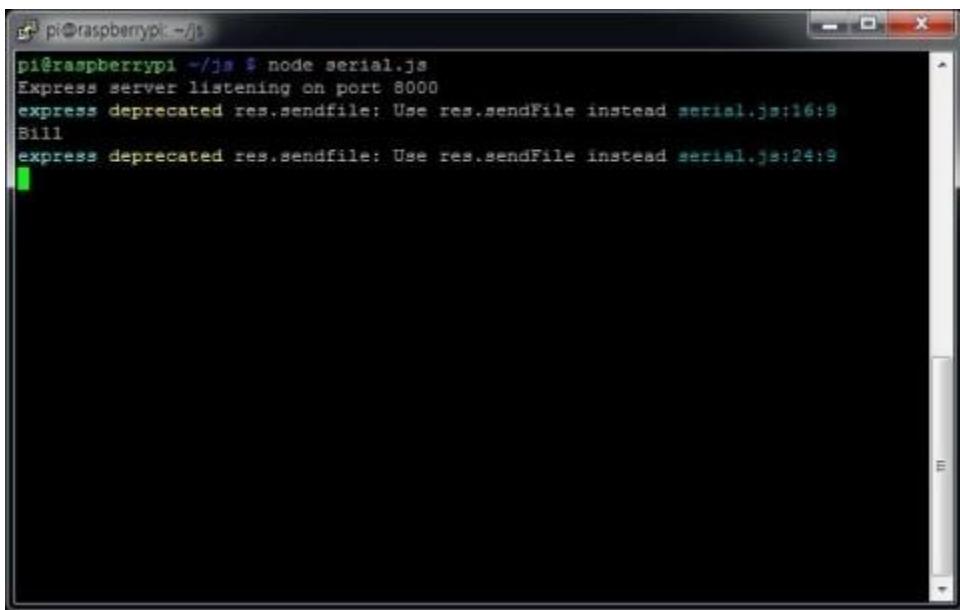
먼저 웹페이지에서 pi의 ip + :8000/serial로 접속합니다.

저는 pi의 ip가 192.168.120.28이므로 192.168.120.28:8000/serial로 접속해 보도록 하겠습니다.



그리고 텍스트박스에 Bill을 입력하고 send를 클릭합니다.

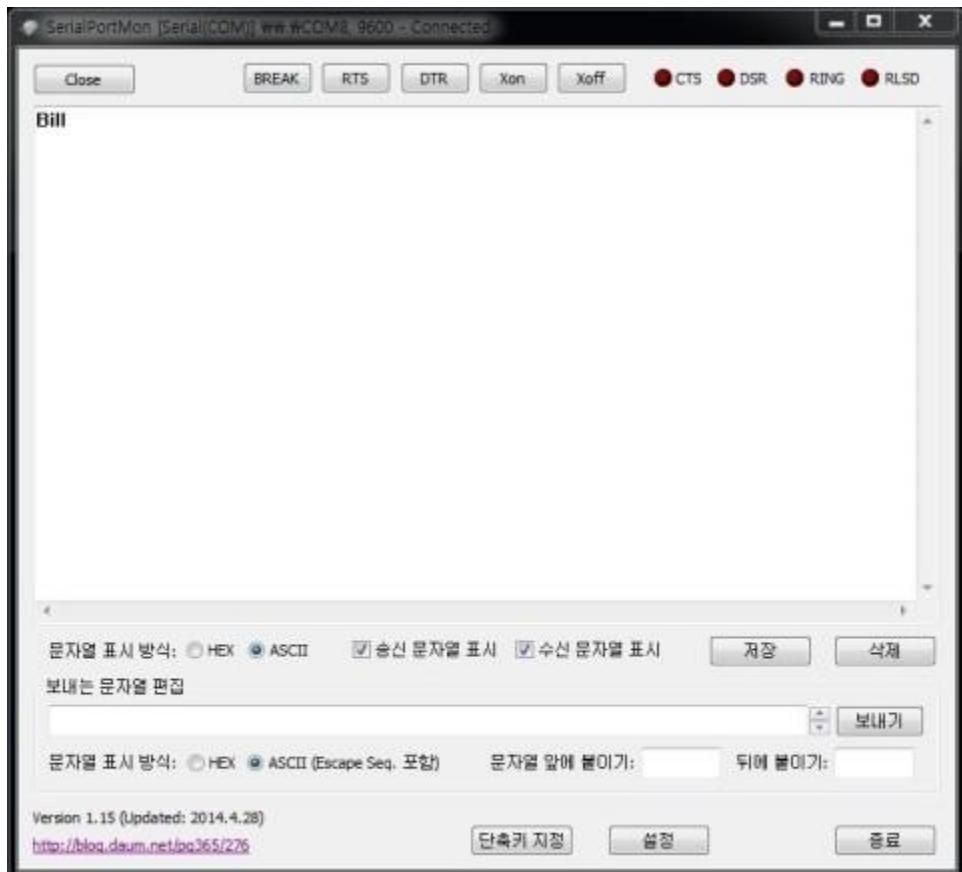
그러면 파일에서 다음과 같이 나올 것입니다.



```
pi@raspberrypi:~/js $ node serial.js
Express server listening on port 8000
express deprecated res.sendfile: Use res.sendFile instead serial.js:16:9
Bill
express deprecated res.sendfile: Use res.sendFile instead serial.js:24:9
```

정상적으로 나왔네요!

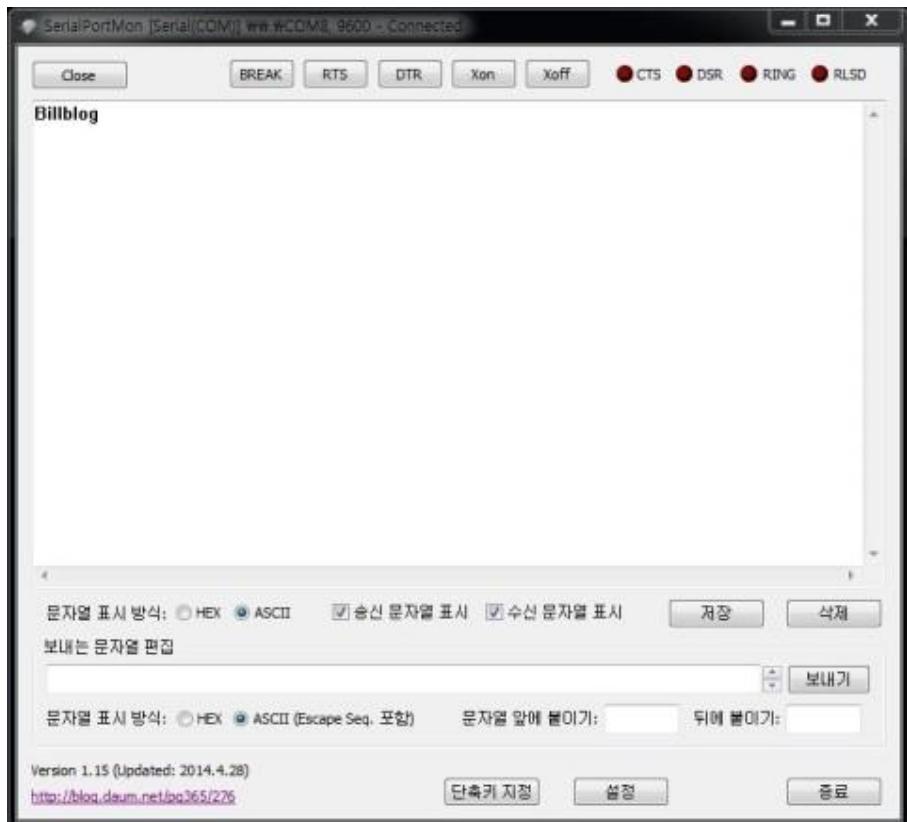
그리고 시리얼 모니터창을 보겠습니다.



역시 문제 없습니다.

그러면 시리얼창에서 값을 보내보도록 하겠습니다.

blog를 보내보겠습니다.



```
pi@raspberrypi:~/js$ node serial.js
Express server listening on port 8000
express deprecated res.sendFile: Use res.sendFile instead serial.js:16:9
Bill
express deprecated res.sendFile: Use res.sendFile instead serial.js:24:9
blog
blog
```

위와같이 나오면 성공입니다.

F. Dynamixel 서보 모터 제어하기

저번 시간에 Serial 통신을 했으니 이번에는 이를 이용하여 다이나믹셀을 구동해 보도록 하겠습니다.

다이나믹셀은 half-duplex 방식을 이용하고 있어 별도의 회로가 필요합니다.

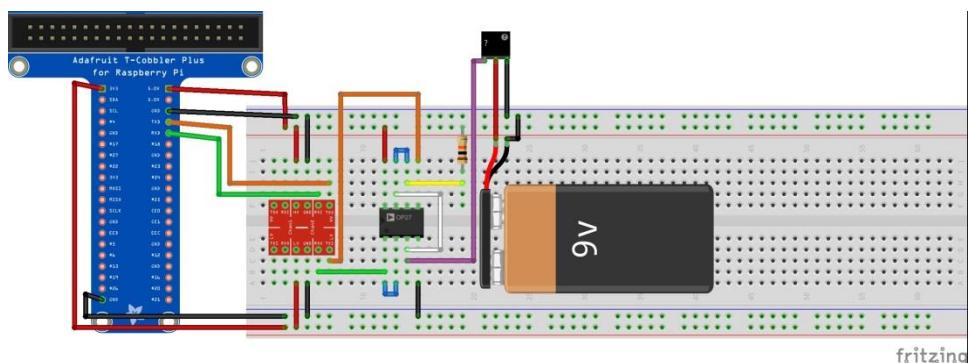
원래 3-state buffer 를 이용해야 하지만 가지고 있는게 없어 lm393(opamp)로 대체하였습니다.

opamp 의 – 입력부와 출력부를 연결해주면 버퍼와 같은 동작을 하게됩니다.

다만 3-state처럼 on / off 를 해줄 수 없는게 단점이나 동작에 큰 지장은 없습니다.

그리고 온전히 값을 전달해주기 위해 저번시간처럼 저항을 이용하지 않고 컨버터를 이용하였습니다.

회로도



조금 복잡하긴 합니다만 opamp 부분만 주의해 주시면 됩니다.

? 되어있는 부분은 다이나믹셀입니다.

전원은 외부에서 공급해주며 이때 GND는 통일해줍니다.

보라색 선은 half-duplex 로 변경된 통신선입니다.

다이나믹셀 1.0 프로토콜을 사용하였으며 이와 관련된 사항은 아래 링크를 참조 부탁드립니다.

http://support.robotis.com/ko/product/dynamixel/dxl_communication.htm

각 프로토콜의 명령어부분까지 찾지 못하여 OpenCM9.04 보드의 다이나믹셀 제어 함수들 값을 받아와서 사용하였습니다.

프로그래밍에 앞서 다이나믹셀의 baudrates를 설정해줄 필요가 있습니다.

다이나믹셀은 1Mbps를 사용하나 여기서 사용한 serialport 패키지는 지원하지 않습니다.

그런 이유로 라즈베리에서는 9600bps를, 다이나믹셀에서는 이와 제일 유사한 9615bps를 이용하였습니다.

baud 변경은 usb2dynamixel을 이용하거나 OpenCM9.04 혹은 CM계열 제어기들을 이용하면 변경이 가능합니다.

변경이 끝났으면 코딩을 해보도록 하겠습니다.

ax.js

```
var express = require('express'),  
    http = require('http'),  
    app = express(),  
    server = http.createServer(app) ;  
  
var bodyParser = require('body-parser') ;  
  
var SerialPort = require('serialport').SerialPort,
```

```
serial = new SerialPort('/dev/ttyAMA0', {  
    baudrate : 9600  
}) ;  
  
app.use(bodyParser.json()) ;  
app.use(bodyParser.urlencoded({ extended : false })) ;  
  
app.get('/ax', function (req, res) {  
    res.sendfile('ax.html', {root : __dirname }) ;  
}) ;  
  
app.post('/data', function (req, res) {  
    var val = req.body.ax ;  
    var data ;  
    if (val === 'cw') {  
        data = new Buffer([0xff, 0xff, 0x01, 0x05, 0x03, 0x20, 0x90, 0x01, 0x45]) ;  
    }  
    else if (val === 'ccw') {  
        data = new Buffer([0xff, 0xff, 0x01, 0x05, 0x03, 0x20, 0x90, 0x05, 0x41]) ;  
    }  
    else if (val === 'stop') {  
        data = new Buffer([0xff, 0xff, 0x01, 0x05, 0x03, 0x20, 0x00, 0x00, 0xd6]) ;  
    }  
})
```

```
serial.write(data, function(err) {});  
res.sendfile('ax.html', {root : __dirname});  
});  
  
serial.on('data', function(data) {  
    console.log(data);  
});  
  
server.listen(8000, function() {  
    console.log('Express server listening on port ' + server.address().port);  
});
```

저번시간이랑 큰 차이는 없습니다.

다만 `data = new Buffer([0xff, 0xff, 0x01, 0x05, 0x03, 0x20, 0x90, 0x01, 0x45]);`

이 부분을 유심히 봐야합니다.

0x는 16진수를 뜻하며 ff등은 숫자를 의미합니다.

16진수 ff는 255입니다. 나머지 숫자들도 이와 같습니다.

0xff, 0xff, 0x01, 0x05, 0x03, 0x20, 0x90, 0x01, 0x45 가 의미하는 것은 cw방향으로 400의 속도로 움직이라는 말입니다.

이는 프로토콜에 정의되어 있는 내용입니다.

들어오는 값에 따라 시리얼에 써줄 data값을 변경하여 쓰게되는 형태입니다.

그러면 html 코드입니다.

ax.html

```
<!DOCTYPE html>

<html>
  <head>
    <title>ax</title>
  </head>
  <body>
    <form action = '/data' method = 'post'>
      <input type = 'submit' value = 'cw' name = 'ax' >
      <input type = 'submit' value = 'ccw' name = 'ax' >
      <input type = 'submit' value = 'stop' name = 'ax' >
    </form>
  </body>
</html>
```

그냥 버튼 3개입니다.

이를 이용하여 실행해 보겠습니다.



```
pi@raspberrypi: ~/js
pi@raspberrypi ~/js $ node ax.js
Express server listening on port 8000
express deprecated res.sendfile: Use res.sendFile instead ax.js:32:9
<Buffer ff ff 01 05 03 20 90 01 45 ff ff 01 02 00 fc>
<Buffer ff ff 01 05 03 20 90 05 41 ff ff 01 02 00 fc>
<Buffer ff ff 01 05 03 20 00 00>
<Buffer d6 ff ff 01 02 00 fc>
<Buffer ff ff 01 05 03 20 90 01>
<Buffer 45 ff ff 01 02 00 fc>
<Buffer ff ff 01 05 03 20 90 05>
<Buffer 41 ff ff 01 02 00 fc>
<Buffer ff ff 01 05 03 20 00 00>
<Buffer d6 ff ff 01 02 00 fc>
```

7. 라즈베리파이와 카메라 영상처리

A. 디지털 영상처리

영상처리를 간단히 설명하자면, 사물에 반사되는 빛 신호를 처리하는 것이라 할 수 있습니다. 기존의 필름카메라 시대에서 디지털카메라 시대로 패러다임이 변하면서 물체에 반사하는 빛의 신호를 아날로그에서 디지털로 처리하고 있습니다. 이를 디지털 영상처리라 하며 그 범위와 분야도 점점 더 광범위해지고 있습니다. 특히 컴퓨터와 모바일 플랫폼의 성능이 좋아짐에 따라 더 좋은 성능의 애플리케이션들이 개발되고 있습니다.

디지털 영상처리를 분류하자면, 영상의 개선, 복원, 변환, 분석, 인식, 압축 등을 이야기 할 수 있으며, 그에 따라 다양한 알고리즘이 연구/개발되고 있습니다.

실제로 영상처리 알고리즘을 배운 후에, 그것을 이용해서 연구를하거나 제품을 개발할 때는 OpenGL, DirectX, OpenCV 등을 많이 사용합니다. 이들은 오픈소스 라이브러리인데, 최근에는 게임 개발 엔진들이 많이 상용화되고 투자가 이루어지면서 개발 쪽에서는 상용화된 라이브러리 및 엔진들을 사용하는 것이 일반적입니다. 하지만, 여전히 학계에서는 오픈소스 라이브러리를 사용하여 연구를 하는 노력이 많이 이루어지고 있습니다. 이미지 및 그림에 대한 처리 및 3D 렌더링을 할 때는 OpenGL과 DirectX가 사용되고 비디오 처리를 할 때는 OpenCV를 많이 사용합니다. 즉, 3D 그림을 그리고 처리할 때는 OpenGL 및 DirectX를 사용하는 것이 적합하며, 웹캠 등으로 스트리밍 및 얼굴 인식 등의 알고리즘을 수행할 때는 OpenCV를 사용하는 것이 좋습니다.

B. OpenCV와 Python

OpenCV는 Open Source Computer Vision의 약자로 인텔에서 만든 영상처리 라이브러리입니다. OpenCV를 사용하는 이유는 여러 알고리즘들이 함수로 구현되어 있고, 이를 사용하여 새로운 알고리즘의 개발할 수 있기 때문입니다. 이 라이브러리는 윈도우, 맥, 리눅스 등 여러가지 OS에서 사용될 수 있으며, 다음의 기능들을 수행할 수 있습니다.

- 1) 실시간 이미지 캡처 (Real time image capture)
- 2) 비디오 파일 송수신 (Video file import and export)
- 3) 이미지 프로세싱 (Image processing)
- 4) 사물 및 얼굴 인식 (Object / face recognition)

라즈베리파이에서 사용하는 파이썬(Python)은 프로그래밍 언어로써, 쉽고 빠르게 배울 수 있으며 자동으로 메모리 관리를 하는 기능을 가지고 있습니다. 파이썬을 이용한 프로그래밍을 배우는 어린이들이 늘고 있으며 다양한 서적들도 출판되고 있습니다. 즉, 인간의 언어에 상당히 가까우며, 이해하기 쉬운 구조로 되어 있습니다. 영상처리를 하기 위해서 C, C++, 혹은 Java등의 언어를 사용할 수 있고, 어떤 점에서는 C로 코딩한 영상처리 알고리즘이 성능면에서 더 탁월한 결과를 가지기도 합니다. 하지만, 본 매뉴얼에서는 파이썬을 사용하여 영상처리에 대해 보다 쉽게 배우는 기회를 가지도록 하겠습니다.

C. OpenCV 설치

OpenCV는 다음의 네 가지의 라이브러리를 포함하고 있습니다.

- CV: Computer vision algorithms (대부분의 비전처리 알고리즘)
- CVAUX: Experimental/Beta (베타버전 알고리즘)
- CXCORE: Linear algebra (선형대수)
- HIGHGUI: Media/window handling (비디오 읽고 쓰기, 윈도우 디스플레이 등)

라즈베리파이의 패키지들을 업데이트하고 업그레이드

```
pi@raspberrypi ~$ sudo apt-get update  
pi@raspberrypi ~$ sudo apt-get upgrade  
pi@raspberrypi ~$ sudo rpi-update
```

필요한 개발툴과 패키지를 설치

```
pi@raspberrypi ~$ sudo apt-get install build-essential cmake pkg-config
```

이미지 입출력(I/O) 패키지 설치, JPEG, PNG, TIFF를 읽고 처리할 수 있음

```
pi@raspberrypi ~$ sudo apt-get install libjpeg8-dev libtiff4-dev libjasper-dev  
libpng12-dev
```

이미지를 스크린을 통해서 볼 수 있도록 GTK를 설치 (Graphical User Interface Tool Kit)

```
pi@raspberrypi ~$ sudo apt-get install libgtk2.0-dev
```

비디오 입출력(I/O) 패키지 설치

```
pi@raspberrypi ~$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
```

선형대수를 위한 라이브러리 및 scs(splitting cone solver)를 위한 라이브러리 설치

```
pi@raspberrypi ~$ sudo apt-get install libatlas-base-dev gfortran
```

pip 설치 (파이썬 관련 프로그램 설치시 많이 사용)

```
pi@raspberrypi ~$ wget https://bootstrap.pypa.io/get-pip.py
pi@raspberrypi ~$ sudo python get-pip.py
```

virtualenv virtualenvwrapper을 pip를 사용하여 설치

```
pi@raspberrypi ~$ sudo pip install virtualenv virtualenvwrapper
```

profile에 다음의 라인을 추가 (profile은 etc라는 폴더에 있음)

```
pi@raspberrypi ~$ sudo nano /etc/profile
pi@raspberrypi ~$ export WORKON_HOME=$HOME/.virtualenvs
pi@raspberrypi ~$ source /usr/local/bin/virtualenvwrapper.sh
pi@raspberrypi ~$ source ~/.profile
```

Profile 수정 (다음의 라인을 추가하고 Ctrl+X, Y, Enter)

```
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
```

컴퓨터 비전 Virtual Environment를 설치

```
pi@raspberrypi ~$ mkvirtualenv cv
```

파이썬 2.7과 numpy 설치

```
(cv) pi@raspberrypi ~$ sudo apt-get install python2.7-dev
(cv) pi@raspberrypi ~$ pip install numpy
```

OpenCV 설치하고 압축풀기

```
(cv) pi@raspberrypi ~$ wget -O opencv-2.4.10.zip
http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.10/opencv-
2.4.10.zip/download
(cv) pi@raspberrypi ~$ unzip opencv-2.4.10.zip
(cv) pi@raspberrypi ~$ cd opencv-2.4.10
```

OpenCV 설치하기 (make 후 약 8시간 소요)

```
(cv) pi@raspberrypi ~$ mkdir build  
(cv) pi@raspberrypi ~$ cd build  
(cv) pi@raspberrypi ~$ cmake -D CMAKE_BUILD_TYPE=RELEASE -D  
CMAKE_INSTALL_PREFIX=/usr/local -D BUILD_NEW_PYTHON_SUPPORT=ON -  
D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D  
BUILD_EXAMPLES=ON ..  
(cv) pi@raspberrypi ~$ make  
(cv) pi@raspberrypi ~$ sudo make install  
(cv) pi@raspberrypi ~$ sudo ldconfig
```

OpenCV 설치하고 압축풀기

```
(cv) pi@raspberrypi ~$ cd ~/.virtualenvs/cv/lib/python2.7/site-packages/  
(cv) pi@raspberrypi ~$ ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so  
(cv) pi@raspberrypi ~$ ln -s /usr/local/lib/python2.7/site-packages/cv.py cv.py
```

OpenCV 설치하기

```
(cv) pi@raspberrypi ~$ workon cv  
(cv) pi@raspberrypi ~$ python  
>>> import cv2  
>>> cv2.__version__  
'2.4.10'
```

OpenCV 설치하기

```
(cv) pi@raspberrypi ~$ workon cv  
(cv) pi@raspberrypi ~$ python  
>>> import cv2  
>>> cv2.__version__  
'2.4.10'
```

D. 파이카메라를 활용한 실시간 스트리밍

IP카메라를 활용한 보안 시스템 구축 및 원격 모니터링을 요구할 때, 저렴한 라즈베리파이를 활용하여 실시간 스트리밍 시스템을 구축할 수 있습니다. 인터넷을 통하여 영상을 전달해야하기 때문에 간단히 관련 프로토콜에 대해서 집고 넘어갈 필요가 있겠습니다.

E. 영상처리 기본실습

영상처리의 기본실습의 일환으로 USB포트에 연결된 웹캠과 라즈베리파이용 파이카메라를 사용하여 실습을 하도록 하겠습니다. 기본 실습에 사용될 예제는 널리 알려진 얼굴 인식 (Face Detection)과 간단한 도형의 모양과 색깔을 사용하여 검출하는 달걀 인식 (Egg Detection)에 대한 실습을 구현해 보았습니다.

Viola/Jones Face Detector

가장 널리 알려진 얼굴 인식 (Face detection) 알고리즘으로 Viola와 Jones에 의해서 고안됨.

핵심 테크닉

- Rectangle features (Haar feature)
- Integral images for fast computation
- Ada-Boosting for feature selection
- Attentional cascade for fast rejection of negative windows



- Rectangle feature
간단히 설명하면,
 $F = \Sigma (\text{흰부분 픽셀}) - \Sigma (\text{어두운부분 픽셀})$
눈 부분은 어둡고 코와 입부분은 밝음
 F 값이 큼 >> "얼굴로 인식"



Training Data

- 5000 faces
- All frontal, 24x24 pixels
- Normalized, scaled, translated

<관련논문>

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. CVPR 2001.

P. Viola and M. Jones. Robust real-time face detection. IJCV 57(2), 2004.

• Integral images for fast computation
어느 영역의 면적을 구할 때, 단순히 더하기 빼기로 빨리 구할 수 있다는 컨셉. 즉, 점D의 값이 (1-2-D-5) 면적, B의 값이 (1-3-B-5)의 면적, A값이 (1-3-A-7)의 면적, 그리고 C의 값이 (1-2-C-7)의 면적이라고 할 때, 파란색 면적 (A-C-D-B)은 각각의 적분 면적을 더하고 빼서 구할 수 있음: A의 값-B의 값-C의 값+D의 값 = 파란색 면적

• Ada-Boosting for feature selection
간단히 설명하면, 보다 좋은 classifier를 찾기 위해 트레이닝하는 방법으로, 약한 classifier의 집합으로 strong한 classifier를 구성한다는 것. Ada는 adaptive의 약자.

• Attentional cascade for fast rejection of negative windows
Ada-boost classifier를 통해 얼굴이 아닌 것은 버려지고 얼굴인 것만 계속적으로 검출

Input image
True
True
True
• • •
Face found

```

graph LR
    Input[Input image] --> C1((Ada-boost Classifier1))
    C1 -- True --> C2((Ada-boost Classifier2))
    C2 -- True --> C3((Ada-boost Classifier3))
    C3 -- True --> Found[Face found]
    C1 -- False --> NonFace1[Non-face]
    C2 -- False --> NonFace2[Non-face]
    C3 -- False --> NonFace3[Non-face]
  
```

앞서, 가상공간에 OpenCV를 설치하였기 때문에 터미널을 연 후에 다음의 명령어를 통해서 OpenCV를 사용할 수 있습니다.

FaceDetection.py 준비하기

```

pi@raspberrypi ~$ source /usr/local/bin/virtualenvwrapper.sh
pi@raspberrypi ~$ mkvirtualenv cv
(cv) pi@raspberrypi ~$ sudo nano FaceDetection.py
  
```

얼굴 인식 using Webcam (FaceDetection.py)

```

import cv2.cv as cv

cv.NamedWindow('a', 1)

cap = cv.CaptureFromCAM(-1)

cv.SetCaptureProperty(cap, cv.CV_CAP_PROP_FRAME_HEIGHT, 240)

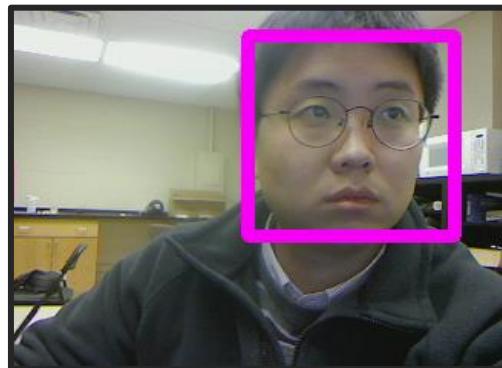
cv.SetCaptureProperty(cap, cv.CV_CAP_PROP_FRAME_WIDTH, 320)

hc = cv.Load("haarcascade_frontalface_alt.xml")

while True:

    frame = cv.QueryFrame(cap)
  
```

```
face = cv.HaarDetectObjects(frame, hc, cv.CreateMemStorage(), 1.05,2,  
cv.CV_HAAR_DO_CANNY_PRUNING, (120,120))  
  
for (x,y,w,h),n in face:  
  
    cv.Rectangle(frame,(int(x),int(y)),(int(x)+w,int(y)+h),(255,0,255),8,0)  
  
    cv.ShowImage('a', frame)  
  
    c = cv.WaitKey(1)  
  
    if c == 13:  
  
        exit(0)
```



달걀 인식 using Pi-Camera

```
import time
import serial
import picamera
import picamera.array
import cv2
import cv2.cv as cv
import numpy as np
hsv_min = np.array([20, 130, 100])
hsv_max = np.array([40, 180, 120])
width = 320
height = 240
with picamera.PiCamera() as camera:
    with picamera.array.PiRGBArray(camera) as stream:
        camera.resolution = (width,height)
        camera.vflip = True
        while True:
            camera.capture(stream, format='bgr', use_video_port=True)
            edges = cv2.Canny(stream.array, 100,300)
            hsv = cv2.cvtColor(stream.array, cv2.COLOR_BGR2HSV)
            imgray = cv2.cvtColor(stream.array, cv2.COLOR_BGR2GRAY)
            ret, thresh = cv2.threshold(imgray, 127, 255, 0)
            contours, hierarchy = cv2.findContours(thresh, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
            contours = sorted(contours, key = cv2.contourArea, reverse =
True)[:5]
            screenCnt = None
            for h, cnt in enumerate(contours):
                peri = cv2.arcLength(cnt, True)
                approx = cv2.approxPolyDP(cnt, 0.01*peri, True)
                screenCnt = approx
```

```

M = cv2.moments(cnt)
if M['m00']!=0:
    cx = int(M['m10']/M['m00'])
    cy = int(M['m01']/M['m00'])
else:
    cx = width/2
    cy = height/2

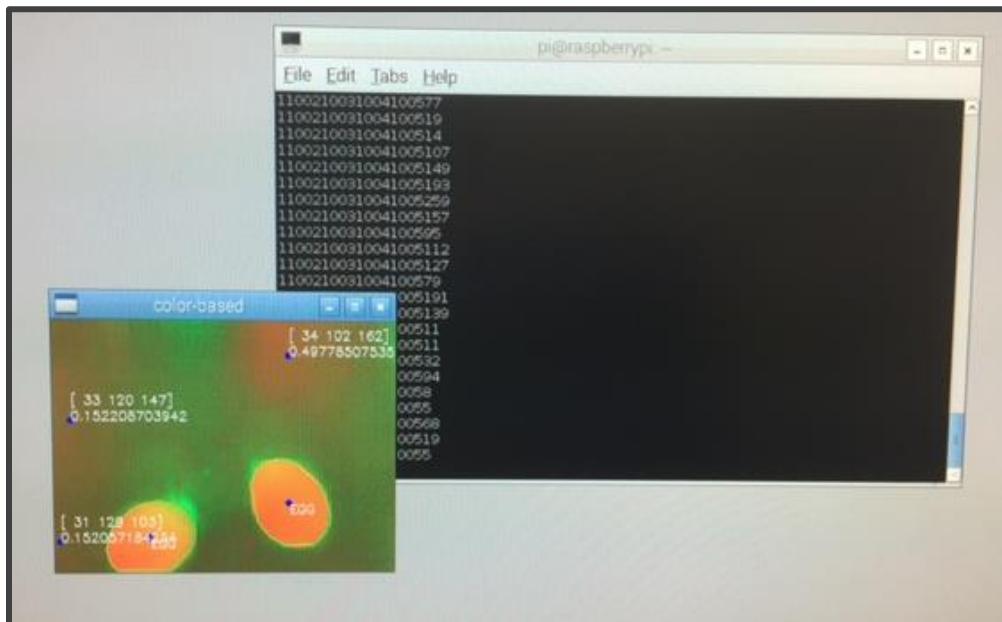
center = np.array([cx,cy])
R_arr = np.zeros((1000,5))

for idx, val in enumerate(cnt):
    R_arr[idx][h]= np.linalg.norm(val-center)
    R = R_arr[:,h]
    ratio = np.min(R[np.nonzero(R)])/max(R_arr[:,h])
    cv2.circle(hsv, (cx, cy), 2, (255,0,0),2)

    if cx < width and cy < height and ratio > 0.6 and ratio < 0.75:
        cv2.putText(hsv, 'EGG', (cx,cy+10),
cv2.FONT_HERSHEY_SIMPLEX,
            0.4, (255,255,255), 1, 8)
    else:
        cv2.putText(hsv, str(ratio), (cx,cy),
cv2.FONT_HERSHEY_SIMPLEX,
            0.4, (255,255,255), 1, 8)
        cv2.putText(hsv, str(hsv[cy-10,cx-10]), (cx,cy-15),
cv2.FONT_HERSHEY_SIMPLEX,
            0.4, (255,255,255), 1, 8)
    if ratio > 0.6 and ratio < 0.75:
        cv2.drawContours(hsv, [screenCnt], -1, (0,255,0),1) #[screenCnt
was contours]
mask = cv2.inRange(hsv, hsv_min, hsv_max)

```

```
cv2.imshow('color-based', hsv)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
stream.seek(0)
stream.truncate()
cv2.destroyAllWindows()
```



8. 참고문헌

1. <http://blog.derivatived.com/posts/OpenCV-Tutorial-on-Face-Tracking-Raspberry-Pi-Camera/>
2. <http://www.youtube.com/watch?v=JjPsW-7FUNG>
3. http://www.clien.net/cs2/bbs/board.php?bo_table=lecture&wr_id=188209&page=66&sca=%5B%EA%B8%B0%ED%83%80%5D&page=66
4. <http://www.pickupapi.com/blog/pitronics-part-7-servo-motors/>
5. <http://raspberrypicar.wordpress.com/>
6. <http://wiringpi.com/pins/>
7. <http://homepage.cem.itesm.mx/carbajal/EmbeddedSystems/SLIDES/Computer%20Vision/Computer%20Vision%20using%20SimpleCV%20and%20the%20Raspberry%20Pi.pdf>
- 8.
9. http://d4c027c89b30561298bd-484902fe60e1615dc83faa972a248000.r12.cf3.rackcdn.com/supporting_materials/Raspberry%20Pi%20Start%20Guide.pdf
10. https://www.packtpub.com/sites/default/files/9781849694322_Chapter_04.pdf
11. <http://www.ijser.org/researchpaper%5CRaspberry-Pi-and-image-processing-based-Electronic.pdf>
12. <http://blog.jozilla.net/2008/06/27/fun-with-python-opencv-and-face-detection/>
13. <http://www.intorobotics.com/9-opencv-tutorials-hand-gesture-detection-recognition/>
14. http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html
15. 스크래치 GPIO: available at <https://pihw.files.wordpress.com/2014/06/ws-setup-scratchgpio13062014.pdf>
16. <http://www.micropik.com/PDF/dht11.pdf>
17. <http://www.uugear.com/portfolio/dht11-humidity-temperature-sensor-module/>

18. <http://akizukidenshi.com/download/ds-aosong/DHT11.pdf>