제1편 라즈베리파이 기본

I. 라즈베리파이 첫 걸음

1. 라즈베리 파이란?

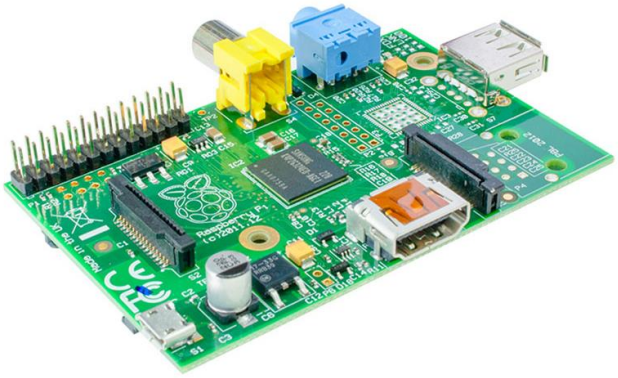
라즈베리파이는 영국의 라즈베리파이재단(Raspberry Pi Foundation)이 학교에서 기초 컴퓨터 과학 교육을 증진하기 위해 만든 싱글 보드(Single Board Computer) 컴퓨터입니다. PC는 메인 보드에 CPU와 메모리, 하드디스크를 장착하고 주변장치를 연결하여 사용합니다. 싱글보드 컴퓨터란 CPU, 메모리, 사운드카드 등이 전부 하나의 보드에 장착이 되어 있고, 여기에 키보드, 마우스, 모니터를 연결하여 사용하는 컴퓨터입니다. 아래 그림의 라즈베리파이에 USB 키보드와 마우스를 연결하고 hdmi로 모니터에 연결하면 완벽한 컴퓨터가 됩니다. PC는 하드디스크에 운영체제를 설치하지만 라즈베리 파이는 SD카드에 운영체제를 설치합니다.

라즈베리파이 모델 A는 25달러, B는 35달러로 저렴하고 그래픽 성능이 뛰어난 게 장점입니다. 2012년 2월 출시돼 2013년 11월 200만대 판매를 돌파했습니다. 100만대 판매는 1년이 걸렸지만 200만대 판매는 8개월이 걸리지 않았습니다. 라즈베리파이 모델 B는 512메가바이트(MB) 램, 2개의 USB 포트, 음성 · 영상 입출력 단자, SD카드 슬롯, 10/100MB 이더넷 포트로 구성됩니다. 라즈베리파이 외에도 비글보드, 판다보드, 아두이노 등이 등장하면서 초소형 컴퓨터(PC)에 대한 인기가 점차 높아지고 있습니다.

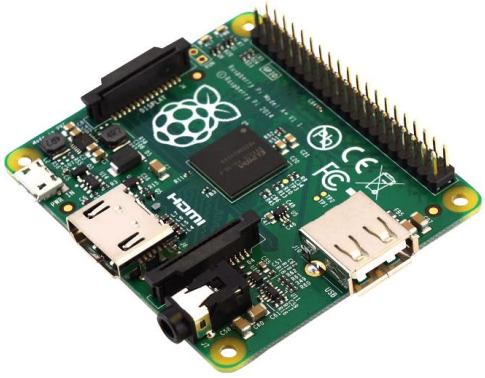
또한 GPIO 포트라는 것이 있어서, 이를 통해서 외부 전자장치(LED, 모터, 각종 센서 등)를

제어할 수 있어서 크기는 작지만 아주 강력한 효용성을 갖는 컴퓨터입니다.

【01】라즈베리 파이의 역사



라즈베리파이1 모델 A 256MB



라즈베리파이1 모델 A+

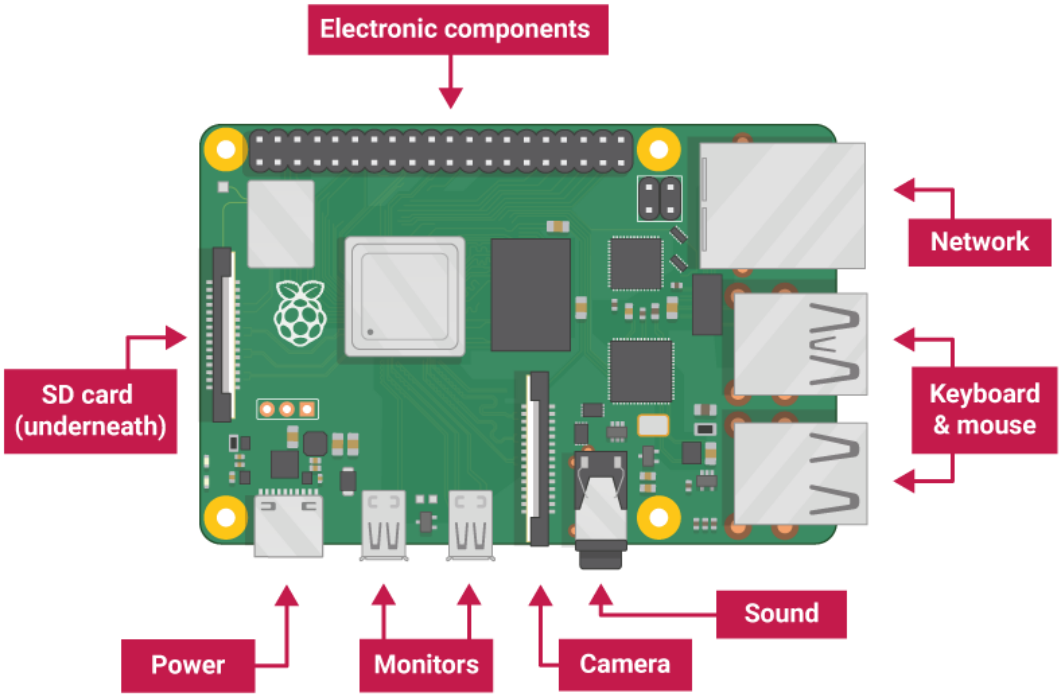
라즈베리파이1 모델 B

라즈베리파이1 모델 B+

라즈베리파이2 모델 B

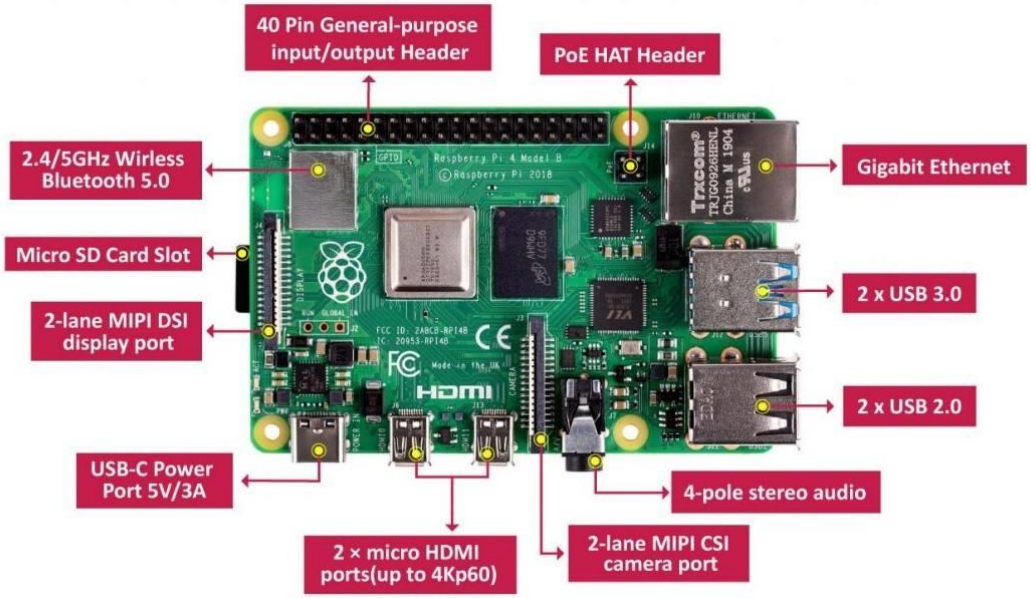
라즈베리파이2 모델 B+

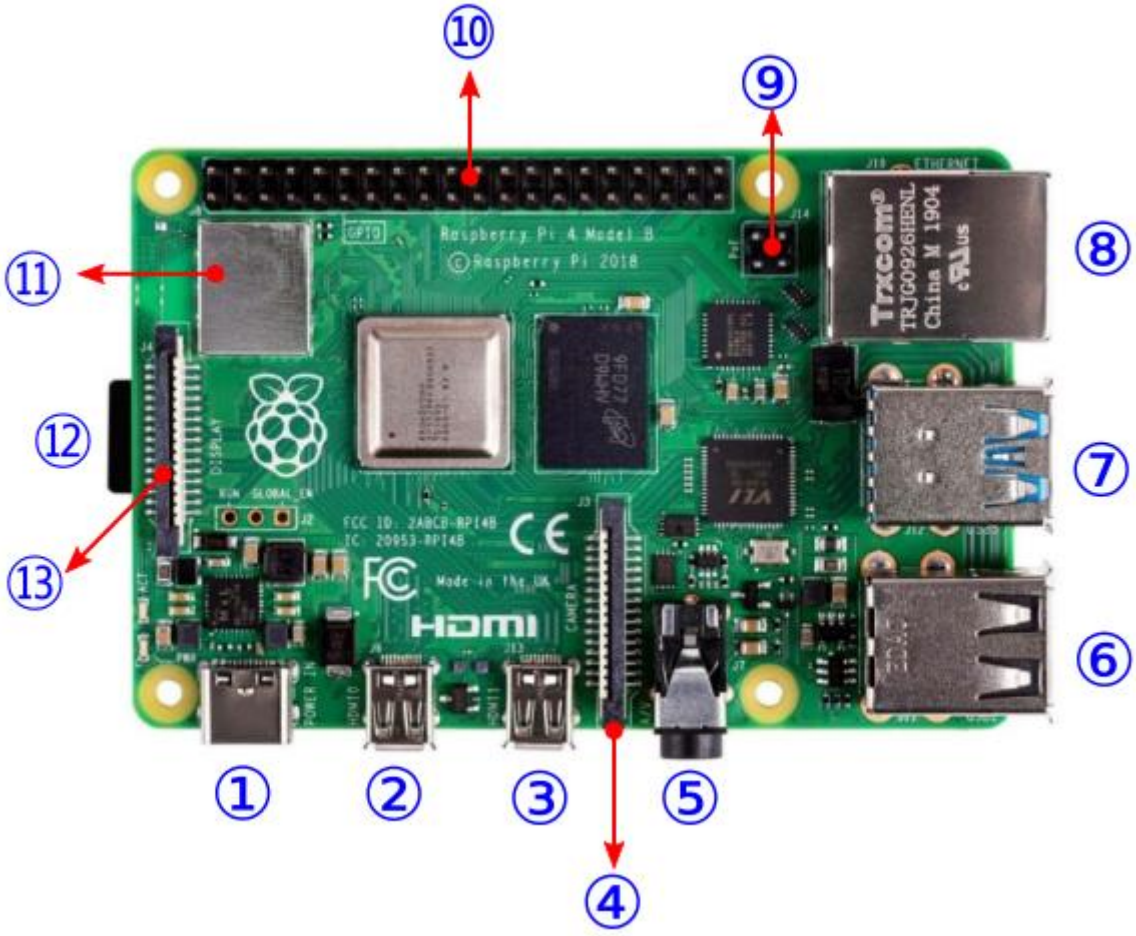
【02】라즈베리파이의 구성



【03】주변장치 연결

현재 사용하고 있는 라즈베리파이는 라즈베리파이4이므로 이를 기준으로 설명하겠습니다.





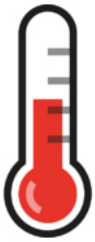
01) 전원

라즈베리파이의 전원 공급은 3가지 방법으로 할 수 있습니다.

전원 공급 방법

- USB-C 커넥터
- GPIO 헤더
- PoE(별도 PoE Hat 필요)

어느 방법이던지 5V DC, 최소 3A 의 전원을 공급하여야 합니다. 공급 전압이 4.63V(±5%)이하로 떨어지면 다음과 같은 경고 아이콘을 모니터에 표시합니다.

저전압	온도경고 1	온도경고 2
		

추가적으로 온도경고1 아이콘은 Soc(System on a Chip)의 온도가 80~85℃가 되면 표시가 되고 코어의 온도를 떨어트리기 위해 CPU 속도와 전압을 떨어뜨기기 시작하고 온도가 85℃를 넘으면 온도경고2 아이콘을 표시하며 CPU뿐만 아니라 GPU의 속도와 전압을 떨어뜨리기 시작합니다.

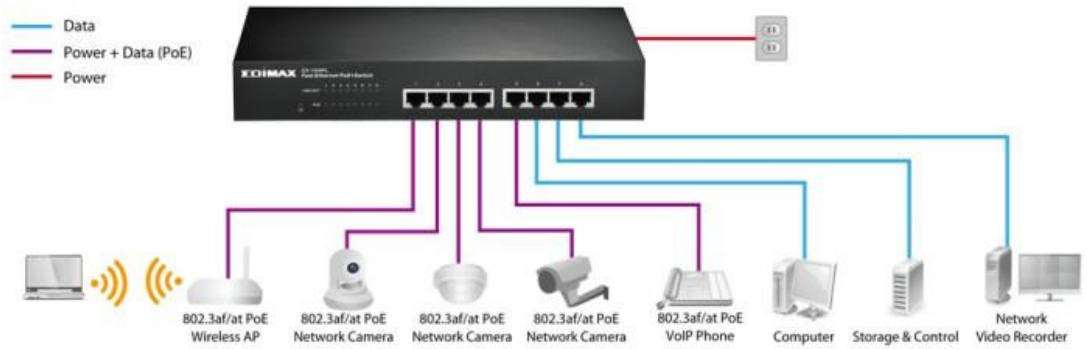
①은 USB-C 타입으로 전원을 공급합니다. 스마트폰 충전기에 USB-C타입의 케이블을 연결하여 전원을 공급하여도 되지만 스마트폰 충전기는 5V이지만 3A의 전류를 공급하지 못하는 경우가 대부분이므로, 다음과 같은 전용 아답터를 사용할 것을 권장합니다.

전용 아답터	스마트폰 충전기

⑧로도 전원 공급이 가능합니다. ⑧은 네트워크를 연결하는 이더넷포트(LAN Port)이지만 이 포트로 전원을 공급하는 방식을 PoE라고 합니다.

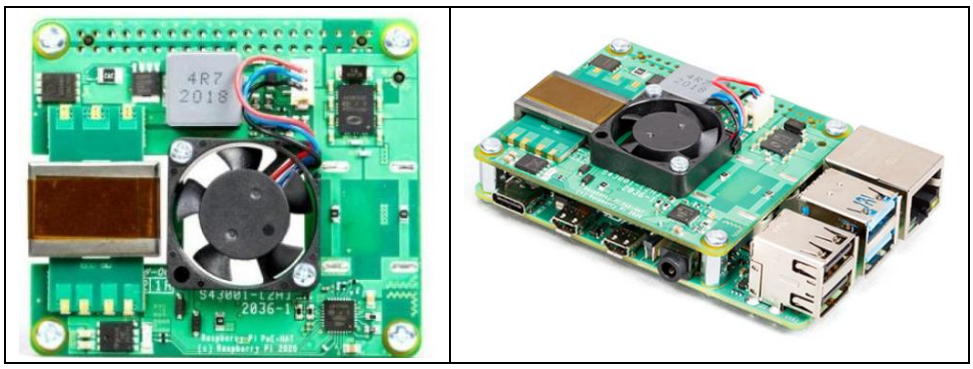
PoE는 "Power Over Ethernet" 의 줄임말로 기존의 랜 케이블 (RJ45, UTP)에 추가적인 전원을 공급하여 허브에 연결된 각종 기기를 별도의 추가 전원 입력이 필요 없이 구동가능하게 만든 일종의 규격입니다. PoE (Power Over Ethernet)는 이를 지원하는 네트워크 허브 혹은 연결되는 기기들의 전원부(어댑터등)를 따로 연결하지 않고 Ethernet Cable(UTP)을 통해 데이터와 전원을 동시에 보낼 수 있는 기술입니다. 현재 표준 전압은 직류로 약 48V 이며, 이 표준안 2003년 6월에 만들어졌습니다. PoE 의 출력은 포트당 15.4W 가 표준이나 실제 케이블 길이에 따른 전력의 저하로 인해 포트당 12.95W 정도를 표준으로 봅니다. 그리고 이 효율을 등급으로 메기는 경우도 있는데 등급이 높을 수록 전력 손실량이 적습니다. 그리고 이 보다 높은 전력 출력이 가능한 PoE+ 는 포트당 30W 이나 최소 25.5W 입니다.



[출처: https://www.brainbox.co.kr]

대부분의 무선공유기나 장치들은 PoE를 지원하므로 이더넷 포트를 연결하면 전원이 공급이 되지만, 라즈베리파이는 다음과 같은 PoE Hat이라는 추가 장치가 필요합니다.

PoE Hat	PoE Hat 장착 후



이 PoE Hat을 라즈베리파이에 부착시킨 다음에 이더넷 포트에 연결을 하면 전원이 공급이 됩니다. PoE를 사용하려면 네트워크 스위치가 PoE를 지원하여야 합니다. 이를 지원하지 않으면 PoE방식으로는 전원공급을 할 수 없습니다.
GPIO 핀을 사용하여 전원을 공급하는 방법은 뒤에서 상술합니다.

02) 모니터

②와 ③에는 다음 그림과 같은 micro-hdmi 케이블을 사용하여 모니터나 TV에 연결할 수 있는데, 둘 중 하나만 연결해도 되지만 라즈베리4는 듀얼모니터를 지원하므로 2개를 다 연결할 수 있습니다.
듀얼로 연결하는 경우에는 전원에 가까이 있는 ②의 해상도가 ③의 해상도보다 높습니다. 같은 모니터를 연결하더라도 2개의 모니터에 같은 해상도를 구현할 수는 없습니다.

<table>
<tr>
<td>  </td>
<td>  </td>
</tr>
</table>

03) 스피커

라즈베리파이에서 음성 신호를 내보내는 방법은 2가지가 있습니다. hdmi는 영상과 음성을 동시에 전송하므로 TV나 스피커가 내장되어 있거나 스피커를 연결할 수 있는 모니터라면 문제가 없지만 그렇지 않은 경우에는 ⑤에 스피커를 연결합니다.

⑤는 4극 스테레오이지만 3극 스테레오도 연결 가능합니다.

4 극 스테레오	
3 극 스테레오	

【04】운영체제 설치

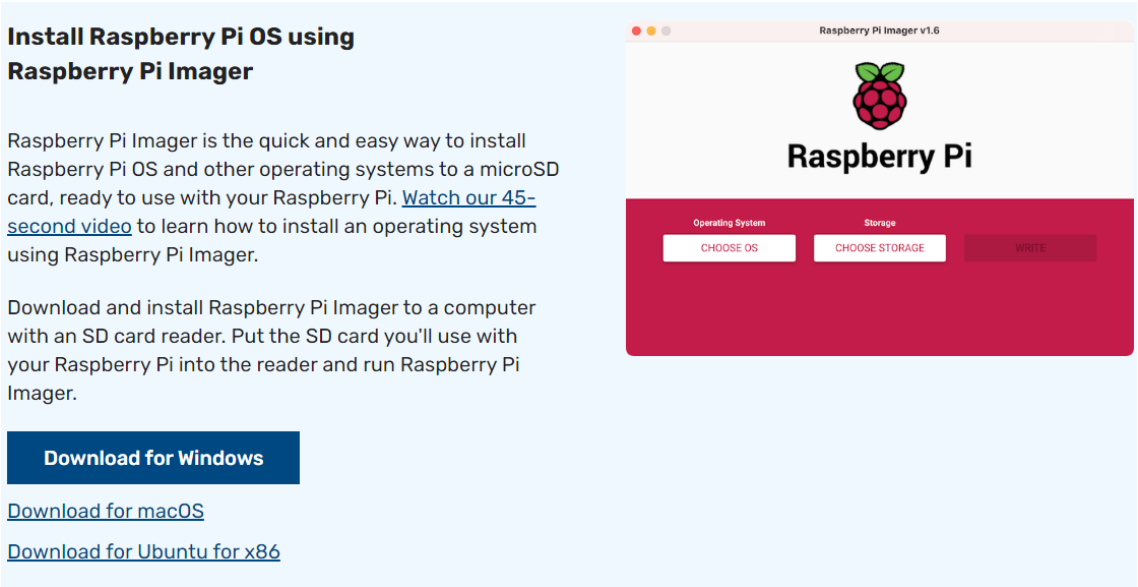
PC에서 MS-윈도를 사용하거나 리눅스를 사용하거나 맥에서 맥 운영체제를 사용하거나 데스크탑 컴퓨터는 운영체제를 하드디스크에 설치하여 사용합니다. 라즈베리파이는 하드디스크가 없으므로 운영체제를 SD에 담아서 사용합니다. 그러므로 라즈베리파이를 사용하기 위해서는 어떤 운영체제를 사용하든지 사용하고자 하는 운영체제를 SD에 설치하여야 합니다. 다른 운영체제(페도라, 데이안 등)를 사용할 수 있지만 라즈베리파이에서 공식적으로 제공하는 운영체제를 사용하는 것이 바람직합니다. 라즈베리파이의 공식 운영체제는 라즈비언(Raspbian)입니다. 이 라즈비언을 사용하는 방법은 2가지가 있습니다.

01) 운영체제 다운로드

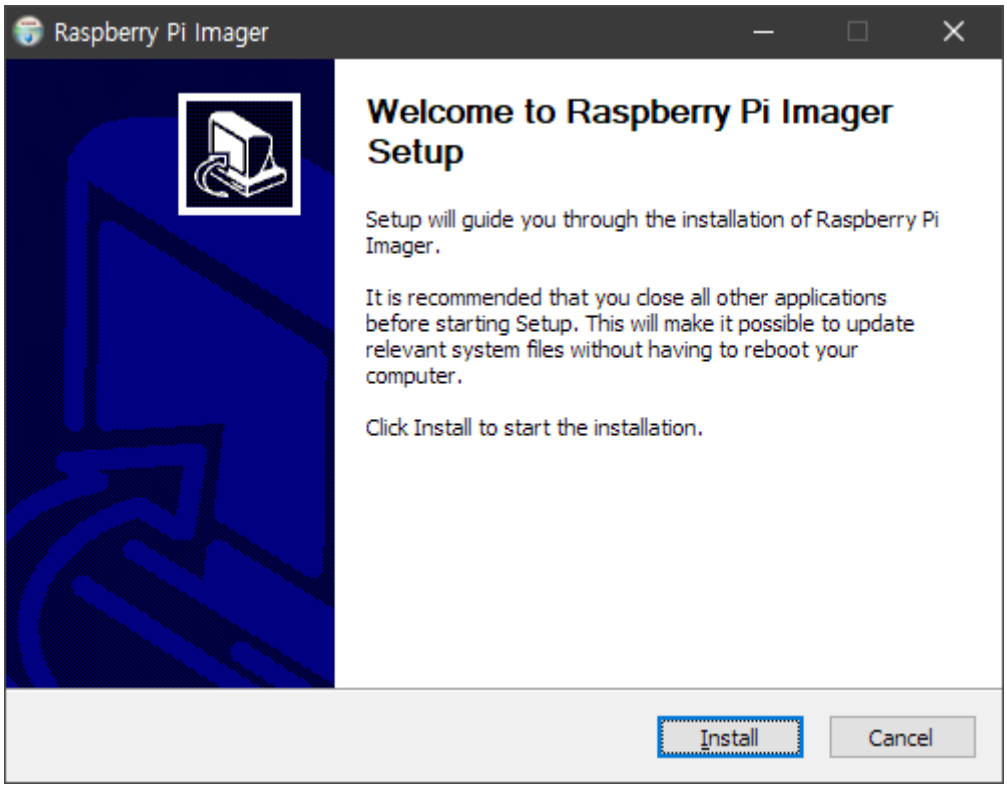
1번째는 이미 라즈비언이 설치되어 있는 SD를 구입하는 방법입니다.
2번째 방법은 Pi OS를 다운받아서 직접 SD에 설치하는 방법입니다.

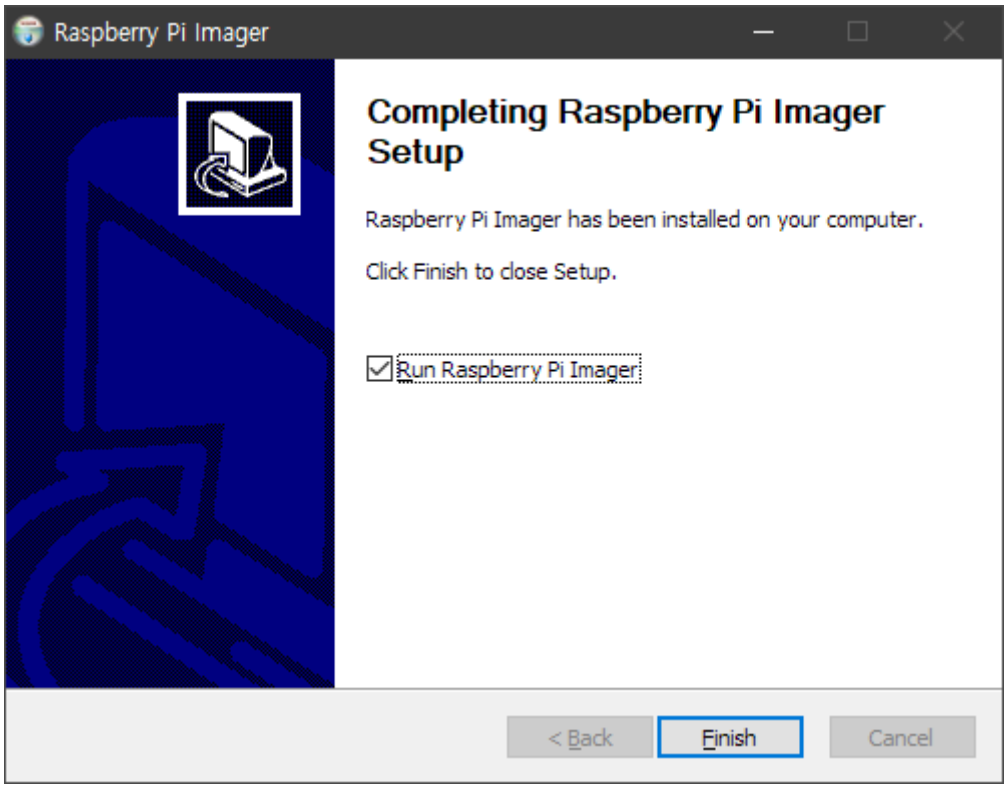
1번째 방법은 방법이라 할 수 없으므로 2번째 방법으로 Pi OS를 다운받아서 설치하면 됩니다. 그러기 위해서 라즈베리파이 공식 싸이트에 접속하여 다운받아서 설치합니다. 주소 http://www.raspberrypi.org/를 입력하면 자동으로 https://www.raspberrypi.org/ 바뀝니다. 공식 홈페이지에서는 다운로드를 찾을 수 없고, https://www.raspberrypi.com/software/에 가면 다운로드 받을 수 있습니다.

라즈베리파이의 공식 운영체는 예전에는 Raspbian이라 불렸고, 지금은 Raspberry Pi OS라 불립니다.



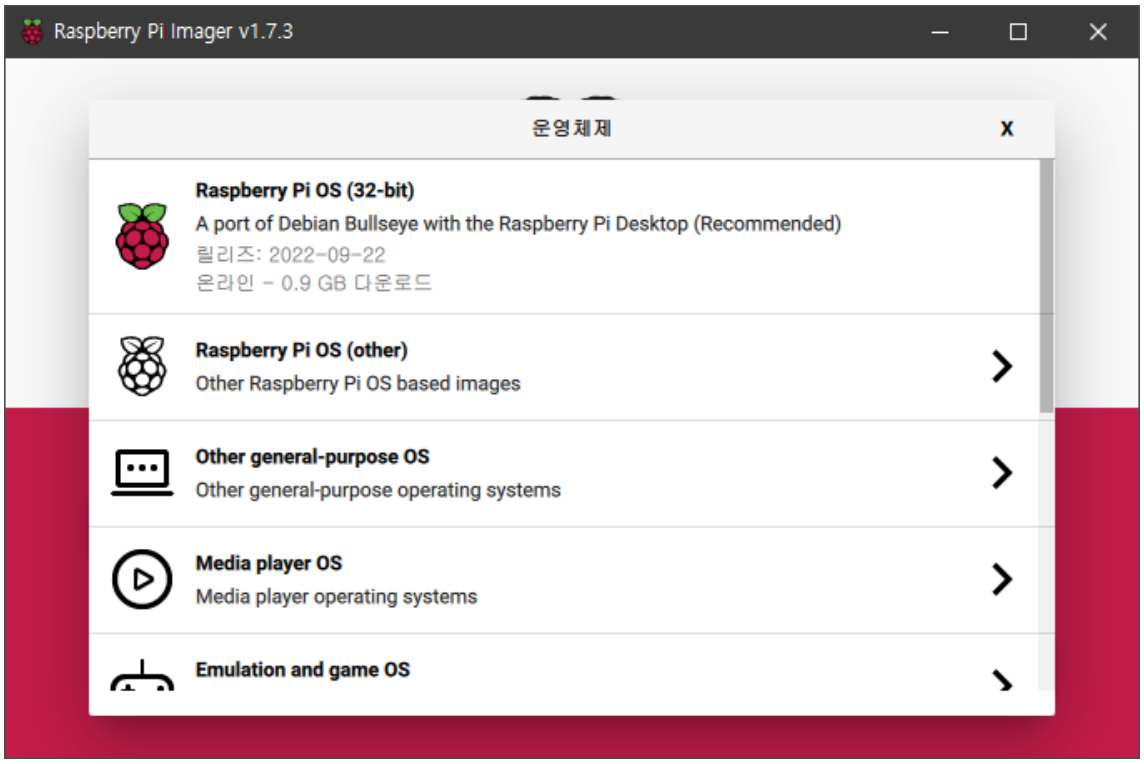
라즈비언을 설치하는 방법은 먼저 pc에 'Raspberry Pi Imager' 라는 프로그램을 설치하고 Imager 프로그램을 사용하여 다운 받은 이미지 파일을 SD 에 넣은 방법입니다.
2023년 1월 25일 현재 Imager 프로그램 버전은 1.7.3입니다. 이 프로그램을 다운 받습니다.





[운영체제 선택] 을 선택하고

Raspberry Pi OS(32-bit)를 선택합니다.

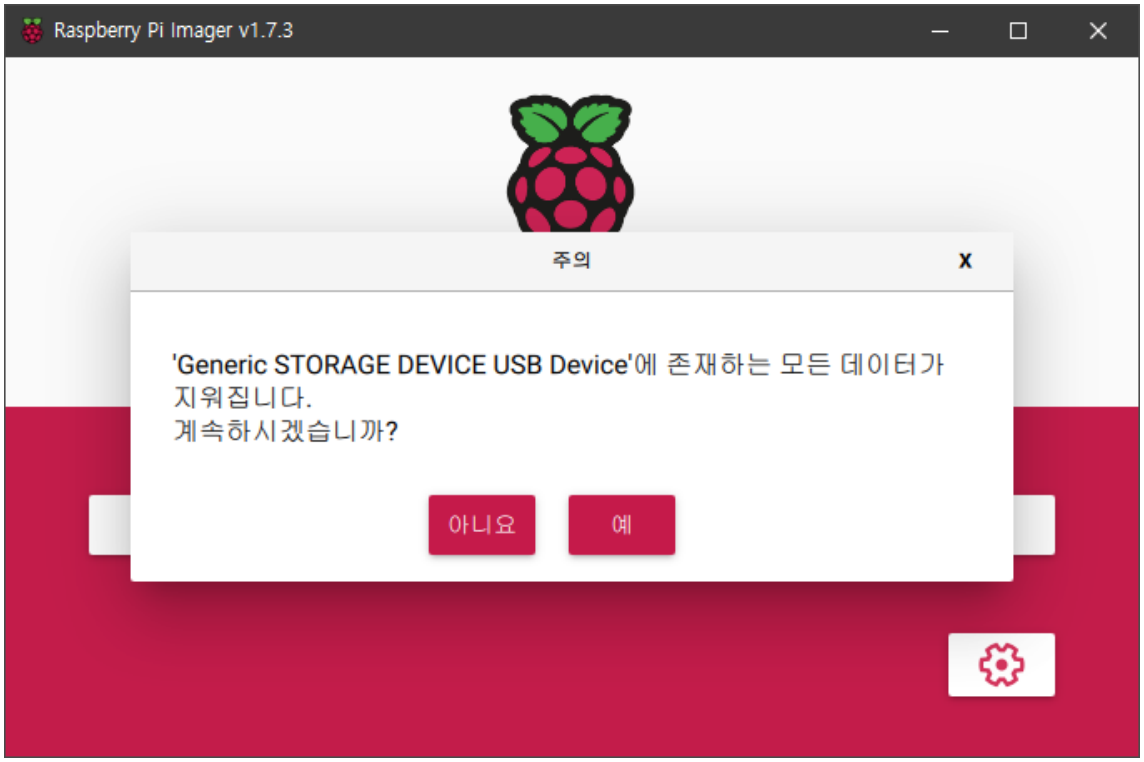


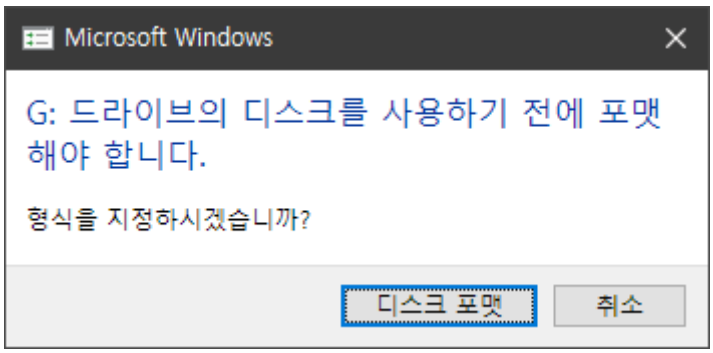
[저장소 선택]을 선택합니다.

'F:\로 마운트'는 PC에 연결되어 있는 외장하드이고 'E:\로 마운트'가 SD카드입니다. 'E:\로 마운트'를 선택합니다.

이제 [쓰기]를 클릭하면 Raspberry Pi OS가 SD카드에 설치됩니다. 오른쪽 하단에 설정으로 보이는 아이콘은 설치 옵션입니다. 이 것은 설정하지 않고 설치할 때 설정하는 것이 합리적입니다.


Raspberry Pi Imager v1.7.3
Raspberry Pi
운영체제
저장소
RASPBERRY PI OS (32-BIT)
GENERIC STORAG...
쓰기


Raspberry Pi Imager v1.7.3
주의
'Generic STORAGE DEVICE USB Device'에 존재하는 모든 데이터가 지워집니다.
계속하시겠습니까?
아니요
예



위의 것은 Pi Imager이 출력하는 것이고 아래의 것은 운영체제가 출력하는 것입니다. 운영체제가 출력하는 것은 리눅스 파일시스템을 윈도에서 인식하지 못하기 때문에 나오는 것이므로 반

드시 [취소]를 하여야 합니다. [디스크 포맷]을 하면 작성된 리눅스 파일 시스템이 삭제되므로 정상적으로 부팅되지 않습니다.

【Trouble Shooting】

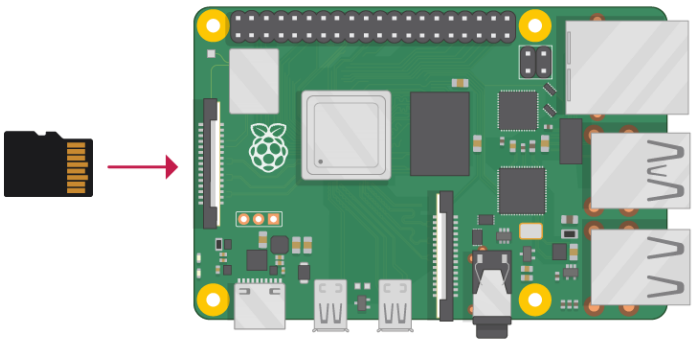
간혹 부팅이 되다가 화면이 정지되는 경우가 있는데 물론 당연히 원인은 여러가지입니다.

1. 시중에 판매되는 전용아탑터는 불량이 많으므로 윈도우를 띄우면서 전원이 부족하면 윈도우를 띄우지 못하고 정지합니다. 필자도 이런 경험이 무척 많아서 몇 번이나 SD 카드를 다시 만든 적이 있는데 전원을 바꾸면 바로 해결이 됩니다.
2. SD 카드 불량입니다.

02) 첫 부팅과 종료

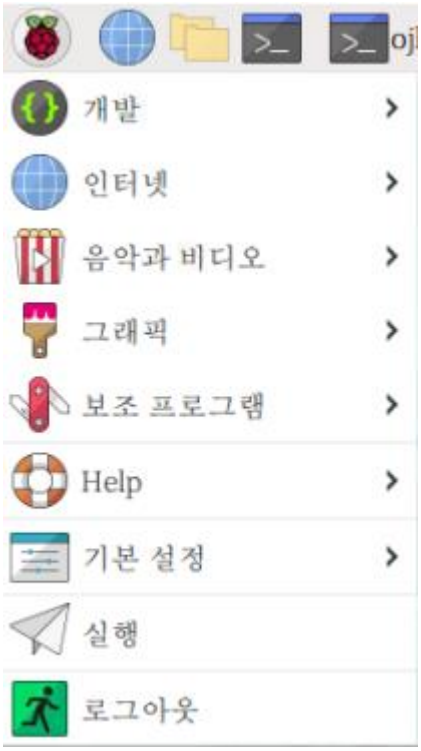
첫부팅시 여러가지 환경설정을 하는데 언어을 영어로 설정하여야 합니다. 한국어로 설정하면 한국어 폰트를 설치하지 않은 상태에서 한국어가 나오기 때문에 문자가 깨져서 나옵니다. 우선 영어로 설정한 다음에 한글 폰트를 설치하고 그 다음에 한국어로 바꾸어야 합니다.

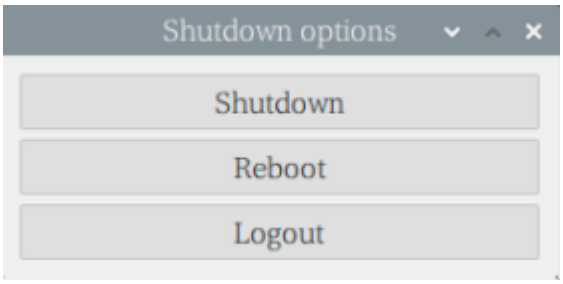
SD를 다음 그림처럼 넣고 키보드, 마우스, HDMI를 연결하고 전원을 연결합니다.



라즈베리파이 끄기

파이메뉴 – 로그아웃 – Shutdown





【05】운영체제 설정

설치할 때 만든 계정으로 자동 로그인 되고 윈도우 화면을 보여줍니다.

예전에는 부팅후에 파일시스템의 용량을 늘려 주었는데 이제는 자동으로 파일시스템의 용량을 늘려 주므로 용량을 확인하 후에 늘어나지 않았을 경우에 다음 과정에 따라서 늘려줍니다.

명령어 창에서 할 수 있지만 윈도우 화면으로 부팅이 되었으므로 윈도우 메뉴를 사용하여 파일시스템을 확장하여 줍니다.

01)해상도 설정

윈도우에서 해상도를 설정하려고 하면 해상도가 640 하나만 나오기 때문에 설정할 수가 없습니다.

/boot/config.txt

```
# uncomment to force a specific HDMI mode (this will force VGA)
# hdmi_group=1
# hdmi_mode=1
```

3. 이것을 일반적으로 22인치 이상 모니터에 많이 사용하는 **1920*1080(16:9) 1080p 해상도**로 할 경우에는 아래 숫자로 바꾸어 주면 됩니다.

```
# uncomment to force a specific HDMI mode (this will force VGA)
hdmi_group=2
hdmi_mode=82
```

처음에는 어려울 수도 있습니다. 이런 경우에는 파이를 끄고 SD카드를 꺼내서 윈도우에 가지고 가서 /boot 드라이브에 있는 config.txt 파일을 수정하고 저장하면 됩니다.

【한걸음 더】

HDMI 모드 옵션

Raspberry Pi 4B 에는 2 개의 HDMI 포트가 있으므로 일부 HDMI 명령을 어느 포트에나 적용할 수 있습니다. <command>:<port>, 여기서 포트는 0 또는 1 이며 설정을 적용할 포트를 지정합니다. 포트를 지정하지 않으면 기본값은 0 입니다. 포트 번호가 필요없는 명령에서 포트 번호를 지정하면 포트가 무시됩니다.

hdmi_group

hdmi_group 명령은 HDMI 출력 그룹을 CEA (TV 에서 일반적으로 사용하는 표준인 소비자 전자 협회) 또는 DMT(디스플레이 모니터 타이밍, 일반적으로 모니터에서 사용하는 표준)로 정의합니다. 이 설정은 hdmi_mode 와 함께 사용해야합니다.

hdmi_group	결과
0	EDID 에서 자동 감지
1	CEA
2	DMT

hdmi_mode

hdmi_group 과 함께, hdmi_mode 는 HDMI 출력 형식을 정의합니다.

이 값들은 hdmi_group=2 (DMT) 인 경우에 유효합니다.

hdmi_mode	해상도	주파수	노트
1	640x350	85Hz	
2	640x400	85Hz	
3	720x400	85Hz	
4	640x480	60Hz	
5	640x480	72Hz	
6	640x480	75Hz	
7	640x480	85Hz	
8	800x600	56Hz	

hdmi_mode	해상도	주파수	노트
9	800x600	60Hz	
10	800x600	72Hz	
11	800x600	75Hz	
12	800x600	85Hz	
13	800x600	120Hz	
14	848x480	60Hz	
15	1024x768	43Hz	라즈베리 파이와 호환되지 않음
16	1024x768	60Hz	
17	1024x768	70Hz	
18	1024x768	75Hz	
19	1024x768	85Hz	
20	1024x768	120Hz	
21	1152x864	75Hz	
22	1280x768		블랭킹 감소
23	1280x768	60Hz	
24	1280x768	75Hz	
25	1280x768	85Hz	
26	1280x768	120Hz	블랭킹 감소
27	1280x800		블랭킹 감소
28	1280x800	60Hz	
29	1280x800	75Hz	
30	1280x800	85Hz	
31	1280x800	120Hz	블랭킹 감소
32	1280x960	60Hz	

hdmi_mode	해상도	주파수	노트
33	1280x960	85Hz	
34	1280x960	120Hz	블랭킹 감소
35	1280x1024	60Hz	
36	1280x1024	75Hz	
37	1280x1024	85Hz	
38	1280x1024	120Hz	블랭킹 감소
39	1360x768	60Hz	
40	1360x768	120Hz	블랭킹 감소
41	1400x1050		블랭킹 감소
42	1400x1050	60Hz	
43	1400x1050	75Hz	
44	1400x1050	85Hz	
45	1400x1050	120Hz	블랭킹 감소
46	1440x900		블랭킹 감소
47	1440x900	60Hz	
48	1440x900	75Hz	
49	1440x900	85Hz	
50	1440x900	120Hz	블랭킹 감소
51	1600x1200	60Hz	
52	1600x1200	65Hz	
53	1600x1200	70Hz	
54	1600x1200	75Hz	
55	1600x1200	85Hz	
56	1600x1200	120Hz	블랭킹 감소

hdmi_mode	해상도	주파수	노트
57	1680x1050		블랭킹 감소
58	1680x1050	60Hz	
59	1680x1050	75Hz	
60	1680x1050	85Hz	
61	1680x1050	120Hz	블랭킹 감소
62	1792x1344	60Hz	
63	1792x1344	75Hz	
64	1792x1344	120Hz	블랭킹 감소
65	1856x1392	60Hz	
66	1856x1392	75Hz	
67	1856x1392	120Hz	블랭킹 감소
68	1920x1200		블랭킹 감소
69	1920x1200	60Hz	
70	1920x1200	75Hz	
71	1920x1200	85Hz	
72	1920x1200	120Hz	블랭킹 감소
73	1920x1440	60Hz	
74	1920x1440	75Hz	
75	1920x1440	120Hz	블랭킹 감소
76	2560x1600		블랭킹 감소
77	2560x1600	60Hz	
78	2560x1600	75Hz	
79	2560x1600	85Hz	
80	2560x1600	120Hz	블랭킹 감소

hdmi_mode	해상도	주파수	노트
81	1366x768	60Hz	
82	1920x1080	60Hz	1080p
83	1600x900		블랭킹 감소
84	2048x1152		블랭킹 감소
85	1280x720	60Hz	720p
86	1366x768		블랭킹 감소

픽셀 클럭 제한이 있습니다. 가장 높은 지원 모드는 블랭킹 감소가 있는 60Hz 에서 1920x1200 입니다.

03) 시스템 설정

시스템 설정은 복잡하지만 여기에서는 기본적인 설정만 한다.

[메뉴-Preferences-Raspberry PI Configuration]를 클릭하여 [System Tab]에서 Filesystem: 의 Expand Filesystem을 클릭하고 재부팅을 합니다.

라즈베리파이는 기본적으로 국가가 영국으로 설정되어 있으므로 우선적으로 Localisation을 설정한다.

[메뉴-Preferences-Raspberry PI Configuration]를 클릭하여 [Localization Tab]에서 keyboard를 Korea, Republic of 로 바꾸어 줍니다.

WiFi country : US
WiFi country 를 Kr로 설정하면 무선 공유기에 따라서 접속이 되지 않는 경우가 있으므로 US 로 설정한다.

04) 사용자 설정

예전에는 라즈베리파이의 기본 유저가 pi로 되어 있었습니다. 그러나 지금 OS는 설치시 사용자를 설정하도록 되어 있기 때문에 설치 시 생성한 사용자로 로그인 하면 됩니다.

사용자는 최고 권한자인 root와 임의로 생성한 사용자(예를 들면 ojk)로 나눌 수 있는데 시스템을 사용하기 위해서는 먼저 최고 권한자인 root 암호를 지정 또는 변경하여야 합니다.

sudo /bin/bash 를 사용하여 최고사용자 권한으로 로그인합니다. 물론 이것은 처음 설치하고 한 번만 사용할 수 있다. root암호을 입력하고 나면 그 다음 부터는
$ su – 명령으로 암호를 입력하여 root 권한으로 작업할 수 있습니다.

```
ojk@raspberrypi:~ $ sudo /bin/bash
root@raspberrypi:/home/ojk# passwd root

새 암호:


새 암호 재입력:


passwd: 암호를 성공적으로 업데이트했습니다

root@raspberrypi:/home/ojk# exit
exit
ojk@raspberrypi:~ $
```

05) 네트워크 설정

네트워크 설정은 뒤에서 상세히 다루지만 네트워크가 설정되어 있지 않으면 요즘은 컴퓨터가 아니니까 무작정 따라 하기로 설정합니다.

/etc/dhcpcd.conf 파일에서 # Example static IP configuration: 부분을 다음과 같이 설정합니다.

```
$ nano /etc/dhcpcd.conf

interface eth0
static ip_address=192.168.45.5/24
static routers=192.168.45.1
static domain_name_server=210.220.163.82
static domain_name_servers=
static domain_search=
```

Ctrl+o 키를 눌러 저장하고, Ctrl+x 키를 눌러 빠져 나옵니다.
재부팅합니다.

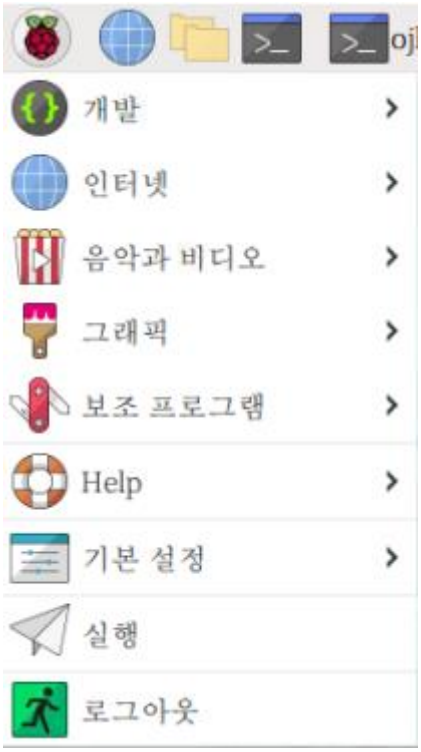
06) 한글 설치

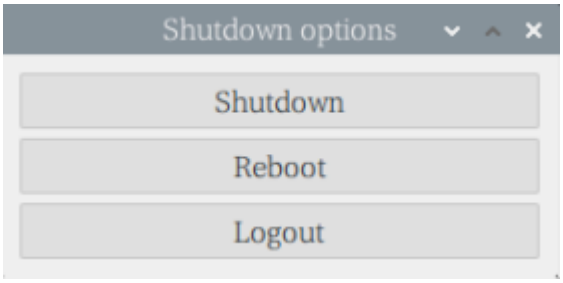
한글을 설치하기 위해서 무작정 다음 명령어를 입력합니다. 뒤에서 상세히 다루므로 여기에서는 관리자 권한으로 운영체제에 한글을 설치한다. 그 정도로만 알아두면 됩니다.

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

라즈베리파이 끄기

파이메뉴 – 로그아웃 – Reboot

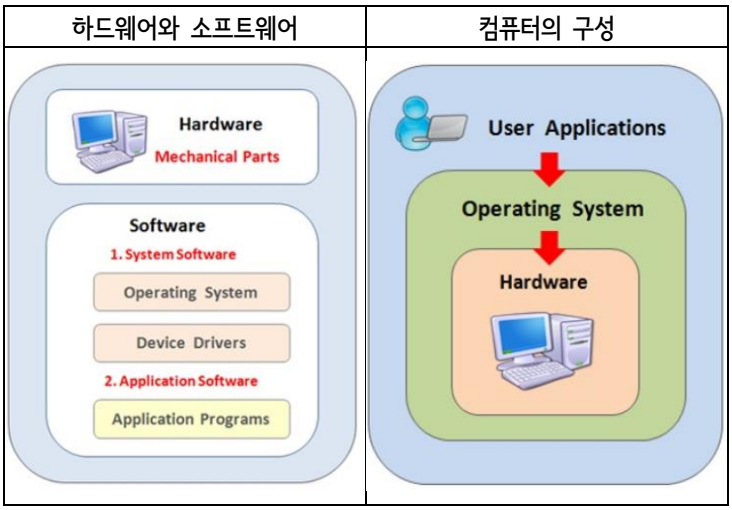




2. 리눅스 명령어

【01】쉘

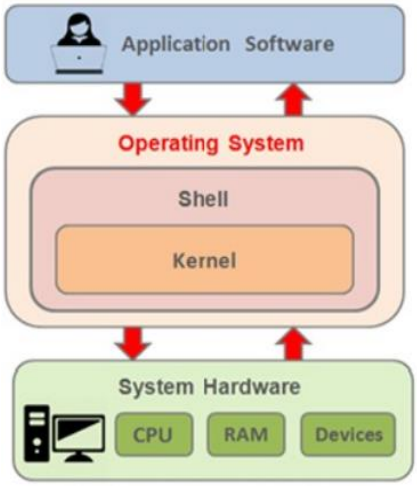
컴퓨터를 사용하기 위해서 절대적으로 운영체제가 필요합니다. 이러한 운영체제는 커널과 쉘로 구성이 됩니다. 구성하는 하드웨어와 소프트웨어를 살펴보겠습니다.



운영체제는 쉘과 커널(운영체제의 심장부)로 구성이 됩니다. 쉘은 그림과 같이 사용자와 커널 사이에서 인터페이스 역할을 하는 특별한 프로그램입니다.

쉘은 사용자와 커널 사이에서 인터페이스 역할을 하는 특별한 프로그램

커널은 시스템이 부팅될 때 메모리에 적재되어 시스템이 종료될 때까지 시스템을 관리합니다. 또한 커널은 프로세스의 생성과 제어, 메모리, 파일시스템 , 통신 및 다방면의 관리를 담당합니다.
쉘은 운영체제 명령어 해석기로서 다음 그림과 같이 하드웨어와 직접적으로 연결하는 커널과의 긴밀한 소통을 통하여 명령어를 해석하고 전달합니다.



소프트웨어		하드웨어
응용 프로그램 운영체제 쉘 커널	⟷	

쉘을 비롯한 다양한 프로그램은 디스크에 저장되어 있는데, 커널은 디스크에 저장되어 있는 이 프로그램들을 메모리로 읽어 들여서 실행하고 , 프로그램이 종료되면 시스템에서 제거하는 역할을 합니다. 사용자가 로그인할 때 실행되는 유틸리티 프로그램이고, 사용자가 명령창에 입력한 명령이나 스크립트 파일안의 명령들을 해석하여 사용자가 커널과 대화할 수 있도록 중계 역할을 합니다.

사용자가 로그온이면 대화형 쉘이 구동되어 사용자의 입력을 기다립니다. 사용자가 명령을 입력하면 쉘은

- 명령을 분석하고
- 와일드카드, 리다이렉션, 파이프, 작업 제어등의 특별한 처리를 하고
- 명령을 찾아봐서 존재하면 실행시키는 역할을 합니다.

유닉스와 리눅스는 하나의 쉘만 사용하는 것이 아니라 여러가지 다양한 쉘을 사용합니다. 주요 쉘로는 본쉘(bourne shell), C쉘, 콘쉘(Korn shell) 등이 있는데 이 세가지 모두 대화형으로 거의 동일하게 동작하지만, 스크립트 언어로 사용될 때는 문법과 효율성에서 일부 차이가 있습니다.
리눅스 쉘은 GNU 배시쉘(GNU Bourne Again Shell, GNU bash)이 기본 쉘로, 본 쉘에서 기능이 강화되어 프로그래밍뿐만 아니라 대화형으로 사용될 때도 사용자가 효율성 향상을 위해 자신의 작업 환경 구성과 단축키 생성이 가능합니다.
자신의 리눅스 버젼에서 어떤 쉘을 사용할 수 있는 지를 알고 싶으면 /etc/shell을 살펴보면 됩니다.

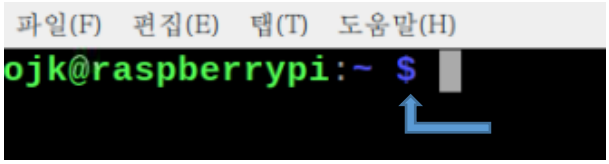
/etc/shells

```
ojk@raspberrypi:~ $ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/usr/bin/bash
/bin/rbash
/usr/bin/rbash
/bin/dash
/usr/bin/dash
/bin/mksh
/usr/bin/mksh
/bin/mksh-static
/usr/lib/klibc/bin/mksh-static
ojk@raspberrypi:~ $
```

cat 명령은 텍스트 파일의 내용을 보여주는 명령입니다.

cat /etc/shells 은 /etc/ 폴더에 있는 shells 라는 텍스트 파일의 내용을 보여주라는 명령입니다. 이러한 것들이 전부 쉘입니다.

【02】기초명령어



$ 모양을 프롬프트(Prompt)라고 하고 뒷 부분의 깜박이는 것을 커서(Cursor)라고 합니다.

일반사용자의 프롬프트는 쉘의 종류에 따라 다릅니다. 일반적으로 $ (bash 사용), % (csh 사용) 입니다.
쉘의 종류에 관계 없이 root 의 프롬프트는 전부 # 입니다.

01) 시스템 정보

uname 옵션

시스템 정보를 출력합니다.

uname 은 현재 작동 중인 머신과 운영체제에 대한 정보를 출력합니다. 아무런 옵션도 주어지지 않으면, uname -s의 결과가 출력됩니다. 여러 개의 옵션이 주어 지거나 -a 옵션이 주어지면 각 항목마다 스페이스 문자로 구분하여 'snrvm' 순서대로선택된 정보를 출력합니다.

옵션

s, --sysname
운영체제의 이름을 출력

-m, --machine
머신(하드웨어) 타입을 출력

-n, --nodename
머신의 네트워 노드 호스트명을 출력

-r, --release
운영체제 릴리즈 넘버를 출력.

-v
운영체제의 버전을 출력한다.

-p, --processor
CPU종류를 출력

-i, --hardware-platform

hardware platform을 출력

-o, --operating-system

operating system 을 출력

-a, --all

위 모든 정보를 출력한다.

--help

도움말을 출력한다.

--version

표준출력으로 버전정보를 출력하고 정상적으로 종료

위에서 -a, --all 처럼 하이픈이 하나인 옵션과 하이픈이 두개인 옵션이 있는데 둘을 다 사용할 수 있습니다.

【예제】

```
ojk@raspberrypi:~ $ uname
Linux
ojk@raspberrypi:~ $ uname -s
Linux
ojk@raspberrypi:~ $ uname -n
raspberrypi
ojk@raspberrypi:~ $ uname -r
5.15.61-v7l+
ojk@raspberrypi:~ $ uname -v
#1579 SMP Fri Aug 26 11:13:03 BST 2022
ojk@raspberrypi:~ $ uname -m
armv7l
ojk@raspberrypi:~ $ uname -a
Linux raspberrypi 5.15.61-v7l+ #1579 SMP Fri Aug 26 11:13:03 BST
2022 armv7l GNU/Linux
ojk@raspberrypi:~ $ uname --version
uname (GNU coreutils) 8.32
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<https://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute
it.
```

```
There is NO WARRANTY, to the extent permitted by law.


David MacKenzie 이(가) 만들었습니다.

ojk@raspberrypi:~ $ uname --all
Linux raspberrypi 5.15.61-v7l+ #1579 SMP Fri Aug 26 11:13:03 BST
2022 armv7l GNU/Linux
ojk@raspberrypi:~ $
```

07) 사용자

a) Login 되어 있는 사용자, 터미널, Login 시간 출력

【형식】

```
$who
```

```
ojk@raspberrypi:~ $ who
ojk      tty1         2023-02-21 22:48
ojk      tty7         2023-02-21 22:48 (:0)
ojk@raspberrypi:~ $
```

b) 사용자 계정 이름 보여줌

사용자에 대한 간단한 정보를 보여줍니다.

【형식】

```
$whoami
```

```
ojk@raspberrypi:~ $ whoami
ojk
ojk@raspberrypi:~ $
```

c) 사용자 id 보여줌

【형식】

```
$ id
```

사용자의 uid, gid를 보여준다.

```
ojk@raspberrypi:~ $ id
uid=1000(ojk) gid=1000(ojk)
groups=1000(ojk),4(adm),20(dialout),24(cdrom),27(sudo),29(audio)
,44(video),46(plugdev),60(games),100(users),104(input),106(rende
r),108(netdev),117(lpadmin),997(gpio),998(i2c),999(spi)
ojk@raspberrypi:~ $
```

d) 모든 로그인 사용자

현재 로그인되어 있는 사용자 보여줌

【형식】

```
$ users
```

```
ojk@raspberrypi:~ $ users
ojk ojk
ojk@raspberrypi:~ $
```

앞의 ojk는 콘솔에서 로그인 사용자 ojk이고, 뒤의 ojk는 원격으로 로그인 사용자입니다.

콘솔(console)이란 쉽게 말하면 원격으로 접속하는 경우가 아니고, 운영체제가 설치되어 있는 컴퓨터에 장착되어 있는 모니터와 키보드 등을 말합니다.

e) password 변경

【형식】

```
$ passwd 사용자
```

현재 로그인한 사용자는 ojk입니다. 암호를 바꾸기 위해서는

$passwd

또는

$passwd ojk

를 입력합니다.

```
ojk@raspberrypi:~ $ passwd ojk

ojk 에 대한 암호 변경 중

Current password:

새 암호:

새 암호 재입력:

passwd: 암호를 성공적으로 업데이트했습니다

ojk@raspberrypi:~ $
```

```
ojk@raspberrypi:~ $ passwd

ojk 에 대한 암호 변경 중

Current password:

새 암호:

새 암호 재입력:

passwd: 암호를 성공적으로 업데이트했습니다

ojk@raspberrypi:~ $
```

f) root 로 전환

【형식】

```
$ su -
```

root 사용자로 로그인 변경

```
ojk@raspberrypi:~ $ su -

암호:

root@raspberrypi:~#
```

ojk는 일반사용자이므로 프롬프트가 $ 이었지만, root 사용자는 최고권한자이므로 프롬프트가 #입니다.

root는 최고권한자이므로 일반사용자의 암호를 강제로 바꿀 수 있습니다.

```
root@raspberrypi:~# passwd ojk

새 암호:

새 암호 재입력:

passwd: 암호를 성공적으로 업데이트했습니다

root@raspberrypi:~#
```

08) 날짜 , 시간 확인과 달력보기

a) 날짜, 시간 확인

【형식】

```
$ date
```

```
ojk@raspberrypi:~ $ date

2023. 02. 23. (목) 22:53:18 KST

ojk@raspberrypi:~ $
```

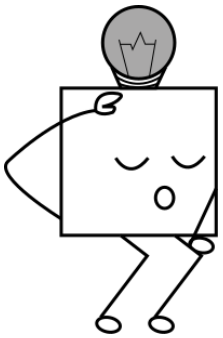
b) 달력 보기

【형식】

```
$ cal
```

```
ojk@raspberrypi:~ $ cal

bash: cal: 명령어를 찾을 수 없음
```

cal 명령은 리눅스(라즈베리파이도 리눅스의 일종) 기본 명령입니다. 그런데 명령어가 없다니????



Pi OS에는 기본적으로 설치되어 있지 않습니다. cal 명령어를 실행할 수 있도록 프로그램을 설치하여야 합니다.
cal 명령은 ncal 패키지의 일부입니다. 따라서 ncal을 설치하여야 합니다. 프로그램 설치는 뒤에서 다루므로 여기에서는 "음!! 이렇게 설치하는구나"하고 넘어가도록 합니다.

```
ojk@raspberrypi:~ $ sudo apt install ncal

패키지 목록을 읽는 중입니다... 완료

의존성 트리를 만드는 중입니다... 완료

상태 정보를 읽는 중입니다... 완료

다음 패키지가 자동으로 설치되었지만 더 이상 필요하지 않습니다:
```

```
  libfuse2
Use 'sudo apt autoremove' to remove it.

다음 새 패키지를 설치할 것입니다:

  ncal

0 개 업그레이드, 1 개 새로 설치, 0 개 제거 및 141 개 업그레이드 안 함.

28.2 k 바이트 아카이브를 받아야 합니다.

이 작업 후 56.3 k 바이트의 디스크 공간을 더 사용하게 됩니다.

받기:1 http://raspbian.raspberrypi.org/raspbian bullseye/main

armhf ncal armhf 12.1.7+nmu3 [28.2 kB]

내려받기 28.2 k 바이트, 소요시간 4 초 (6,478 바이트/초)

Selecting previously unselected package ncal.

(데이터베이스 읽는중 ...현재 110568 개의 파일과 디렉터리가 설치되어

있습니다.)

Preparing to unpack .../ncal_12.1.7+nmu3_armhf.deb ...
Unpacking ncal (12.1.7+nmu3) ...

ncal (12.1.7+nmu3) 설정하는 중입니다 ...

Processing triggers for man-db (2.9.4-2) ...
ojk@raspberrypi:~ $
```

이제 cal 명령을 실행하여 보겠습니다.

```
ojk@raspberrypi:~ $ cal

     2 월 2023

일 월 화 수 목 금 토

         1  2  3  4
 5  6  7  8  9 10 11
```

```
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28

ojk@raspberrypi:~ $
```

```
ojk@raspberrypi:~ $ ncal

    2 월 2023

일      5 12 19 26

월      6 13 20 27

화      7 14 21 28

수    1  8 15 22

목    2  9 16 23

금    3 10 17 24

토    4 11 18 25

ojk@raspberrypi:~ $
```

09) 디렉토리와 파일 작업

디렉토리란 폴더를 말합니다. 폴더는 마이크로소프트 운영체제에서 이름을 붙인 것이고 이 것이 일반화되면서 폴더라는 용어를 사용하게 되었습니다. 그러나 원래 Unix나 Linux에서는 디렉토리(directory)가 정식 명칭입니다. 물론 리눅스에서도 폴더라는 용어를 일반적으로 사용하므로 폴더를 사용하도 되지만 명령어는 여전히 디렉토리로 되어 있습니다.

a) 파일과 디렉토리 목록 보기

파일과 디렉토리 목록을 보여 줍니다. 목록이 영어로 list 이므로 list의 약자인 ls를 사용합니다.

【형식】

```
ls 옵션
```

기 능 DOS의 'dir'과 유사한 명령으로 파일명, 디렉토리명 등을 출력시키며 옵션에 따라 다양한 정보와 함께 출력됩니다.

옵 션

-a 디렉토리 내의 모든 파일 출력한다.
-i 파일의 inode와 함께 출력한다.
-l 파일 허용 여부, 소유자, 그룹, 크기, 날짜 등을 출력한다.
-m 파일을 쉼표로 구분하여 가로로 출력한다.
-r 정렬 옵션이 선택되었을 때, 그 역순으로 출력한다.
-s KB 단위의 파일 크기를 출력한다.
-t 최근에 생성된 파일순으로 출력한다.
-x 파일 순서를 세로로 출력한다.
-F 파일의 형태와 함께 출력한다.
출력되는 파일의 형태는 '*', '@', '|', '=' 등이며, 이것은 각각 실행 파일, 심볼릭 링크, FIFO 소켓을 나타낸다.
-R 서브 디렉토리의 내용을 포함하여 출력한다.
-S 파일 크기가 큰 순서로 출력한다.
-U 정렬하여 출력한다.
-1 라인당 한 파일씩 출력한다.
--help 도움말을 출력한다.
--version 'ls'의 파일 버전과 함께 출력한다.

```
ojk@raspberrypi:~ $ ls
Bookshelf  Desktop  Documents  Downloads  Music  Pictures  Public
Templates  Videos  hello.py
ojk@raspberrypi:~ $ ls -l
합계 40

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:14 Bookshelf

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Desktop

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Documents

drwxr-xr-x 2 ojk ojk 4096  2 월 18 03:10 Downloads

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Music

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Pictures

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Public

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Templates

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Videos

-rw-r--r-- 1 ojk ojk   20  2 월 20 06:21 hello.py
ojk@raspberrypi:~ $
```

drwxr-xr-x 2 ojk ojk 4096 9월 22 09:37 Music

맨 앞의 d 는 Music가 디렉토리임을 나타냅니다.

-rw-r--r-- 1 ojk ojk 20 2월 20 06:21 hello.py

맨 앞의 – 는 hello.py 가 파일임을 나타냅니다.

b) 디렉토리 변경

기 능 작업 디렉토리를 변경합니다.
change dirctory 이므로 명령어는 cd 입니다.

【형식】

```
$ cd  directory 이름
```

Directory는 바꾸려고 하는 디렉토리입니다. 디렉토리를 따로 입력하지 않으면, login한 디렉토리($HOME 또는 '~'로 표시됨)로 이동합니다

```
ojk@raspberrypi:~ $ cd Music
ojk@raspberrypi:~/Music $ ls -l

합계 0

ojk@raspberrypi:~/Music $
```

Music 디렉토리로 이동하였고 Music 디렉토리 내에는 아무 것도 없음을 나타냅니다.

c) 현재 디렉토리

현재 디렉토리는 현재 작업 디렉토리이므로 present working directory의 약자인 pwd를 사용합니다.

【형식】

```
$ pwd
```

```
ojk@raspberrypi:~/Music $ pwd
/home/ojk/Music
ojk@raspberrypi:~/Music $
```

그러면 현재 Music 위치에 있는데 상위 디렉토리로 이동하려면 cd ojk 하면 되겠지....

```
ojk@raspberrypi:~/Music $ cd ojk

bash: cd: ojk: 그런 파일이나 디렉터리가 없습니다
```

```
ojk@raspberrypi:~/Music $
```

"."은 현재 디렉토리이며, ".."은 상위 디렉토리입니다. 상위 디렉토리로 이동하려면

cd ..

를 사용하여야 합니다.

```
ojk@raspberrypi:~/Music $ cd ..
ojk@raspberrypi:~ $ pwd
/home/ojk
ojk@raspberrypi:~ $
```

d) 디렉토리 생성

디렉토리를 생성하려면 make directory 의 약자인 mkdir을 사용합니다.

【형식】

```
$mkdir 디렉토리이름
```

옵 션

-p 지정된 모든 서브 디렉토리까지 함께 생성한다.

-m mode

mode에 해당하는 사용 허가로 디렉토리를 생성한다.

```
ojk@raspberrypi:~ $ mkdir temp
ojk@raspberrypi:~ $ ls -l
합계 44

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:14 Bookshelf

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Desktop

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Documents

drwxr-xr-x 2 ojk ojk 4096  2 월 18 03:10 Downloads
```

```
drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Music

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Pictures

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Public

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Templates

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Videos

-rw-r--r-- 1 ojk ojk   20  2 월 20 06:21 hello.py

drwxr-xr-x 2 ojk ojk 4096  2 월 24 00:15 temp
ojk@raspberrypi:~ $
```

e) 디렉토리 삭제

디렉토리를 삭제하려면 remove directory의 약자인 rmdir 을 사용합니다.

【형식】

```
$rmdir 디렉토리이름
```

옵 션

-p 지우고자 하는 디렉토리의 상위 디렉토리까지 포함하여 지웁니다.

```
ojk@raspberrypi:~ $ rmdir temp
ojk@raspberrypi:~ $ ls -l
합계 40

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:14 Bookshelf

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Desktop

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Documents
```

```
drwxr-xr-x 2 ojk ojk 4096  2 월 18 03:10 Downloads

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Music

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Pictures

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Public

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Templates

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Videos

-rw-r--r-- 1 ojk ojk   20  2 월 20 06:21 hello.py

ojk@raspberrypi:~ $
```

f) 파일과 디렉토리의 복사

복사가 영어로 copy이므로 cp를 사용합니다.

【형식】

```
cp 대상 파일    대상디렉토리
```

기 능 파일(들)을 다른 파일 이름으로 복사하거나 다른 디렉토리로 복사합니다.

문 법 cp <file1> <file2> ...<fileN> <destination>

<file1>부터 <fileN>까지를 <destination>으로 복사한다. <destination>은 파일이나 디렉토리이다.

옵 션

-a -dpR의 조합과 같습니다.

-b 덮어쓰거나 지울 때 백업 파일을 만듭니다.

-d 심볼릭 링크 파일 그대로 복사합니다.
(디폴트는 연결된 원래 파일을 복사함).

-f 같은 파일명을 갖는 파일이 있을 경우, 지운 후 복사합니다.
-i 같은 파일명을 갖는 파일이 있을 경우, 사용자 확인후 복사합니다.
-l 하드 링크를 만듭니다.
-p 원시 파일의 소유자, 그룹, 허용 여부, 시간 등을 그대로 복사합니다.
-r 서브 디렉토리를 포함한 모든 파일 복사합니다.
-s 심볼릭 링크를 만듭니다.
-u 복사할 파일이 구 버전일 경우만 복사합니다.
-v 복사하기 전에 각각의 파일명을 출력합니다.
-x 파일 시스템이 같을 경우만 복사합니다.
-P 원시 파일이 존재하는 디렉토리까지 포함하여 복사합니다.
-R 디렉토리를 포함하여 복사합니다.
-S 환경 변수 SIMPLE_BACKUP_SUFFIX에 의해 지정된 백업 꼬리말로 백업 파일 생성합니다.

일반적인 사용법은 파일을 특정의 디렉토리에 같은 이름으로 복사하는 경우입니다.

현재 디렉토리의 hello.py 파일을 Music 디렉토리에 hello.py 라는 이름으로 복사합니다.

```
ojk@raspberrypi:~ $ cp hello.py Music
ojk@raspberrypi:~ $ cd Music
ojk@raspberrypi:~/Music $ ls -l

합계 4


-rw-r--r-- 1 ojk ojk 20  2 월 24 00:28 hello.py

ojk@raspberrypi:~/Music $
```

현재 디렉토리의 hello.py 파일을 Public 디렉토리에 h.py 라는 이름으로 복사합니다.

```
ojk@raspberrypi:~ $ cp hello.py Public/h.py
ojk@raspberrypi:~ $ cd Public
ojk@raspberrypi:~/Public $ ls -l

합계 4


-rw-r--r-- 1 ojk ojk 20  2 월 24 00:29 h.py

ojk@raspberrypi:~/Public $
```

g) 파일과 디렉토리의 이름 변경과 이동

【형식】

```
mv 디렉토리명 또는 파일명
```

파일이나 디렉토리를 다른 파일이나 디렉토리로 이동시킵니다. 이 명령은 복사와 같으나 원본이 지워집니다. 따라서 파일이나 디렉토리의 이름을 바꿀 때도 사용할 수 있습니다.

옵 션

-b 지워지기 전에 백업본을 만듭니다.

-f 옮겨질 디렉토리에 존재하는 파일이 있으면 덮어씁니다.

-i 옮겨질 디렉토리에 존재하는 파일이 있으면 확인합니다.

-u 옮겨질 디렉토리에 구 버전의 파일이 있을 경우만 옮김니다.

-v 옮기기 전에 파일명을 출력합니다.

```
ojk@raspberrypi:~ $ mkdir ojk
ojk@raspberrypi:~ $ ls -l

합계 44

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:14 Bookshelf

.............................................

drwxr-xr-x 2 ojk ojk 4096  2 월 24 02:16 ojk

ojk@raspberrypi:~ $ mv ojk ttt
ojk@raspberrypi:~ $ ls -l

합계 44

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:14 Bookshelf

.............................................

drwxr-xr-x 2 ojk ojk 4096  2 월 24 02:16 ttt

ojk@raspberrypi:~ $
```

h) 파일 삭제

【형식】

```
rm 파일명
```

파일들을 지우는 명령입니다. 주의할 것은 유닉스나 리눅스 하에서 파일들이 지워지면, 복구가 불가능하므로 주의하여야 합니다.

옵 션

-d 'unlink' 디렉토리를 지운다.
수퍼 유저만 가능하며, 사용 후 fsck가 필요하다.

-f 존재하지 않은 파일일 경우에도 에러 메시지 출력을 하지 않는다.

-i 지우기 전에 확인한다.

-r 서브 디렉토리를 포함하여 지운다.

-v 지워지는 파일명을 출력한다.

-R -r과 같다.

10) 명령어 매뉴얼

【형식】

```
man 명령어
```

옵 션

-a 발견되는 모든 manual page를 출력

-c 사용 할 man.conf 파일을 지정; 기본 값은 /etc/man.config이다.

-d 디버깅 정보 출력

-f whatis와 같음

-h 도움말 출력

-k apropos와 같음

-t groff(troff)를 사용하여 프린트를 위한 형식으로 출력

```
$man ls


LS(1)                                          User Commands
```

```
LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information about the FILEs (the current directory
by default).  Sort entries alphabetically if none of
       -cftuvSUX nor --sort is specified.

       Mandatory arguments to long options are mandatory for
short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

 Manual page ls(1) line 1 (press h for help or q to quit)
AUTHOR
       Written by Richard M. Stallman and David MacKenzie.

REPORTING BUGS
       GNU coreutils online help:
<https://www.gnu.org/software/coreutils/>
       Report any translation bugs to
<https://translationproject.org/team/>

COPYRIGHT
       Copyright  ©  2020  Free  Software  Foundation,  Inc.
```

```
License GPLv3+: GNU GPL version 3 or later
      <https://gnu.org/licenses/gpl.html>.
      This is free software: you are free to change and
redistribute it. There is NO WARRANTY, to the extent per-
      mitted by law.

SEE ALSO
      Full documentation
<https://www.gnu.org/software/coreutils/ls>
      or available locally via: info '(coreutils) ls
invocation'

GNU coreutils 8.32
```

다음 페이지를 보려면 space를 누르면 되고 종료하려면 q 를 누르면 됩니다.

man 명령어의 결과를 텍스트 파일로 저장하기 위해서는 #man 명령어 | col -b > 텍스트파일 을 사용합니다. 위 명령을 사용하면 텍스트는 저장이 되나 화면에 출력되는 것처럼 highlighting이나 밑줄(underlines)등의 효과는 없어집니다. 화면에 출력되는 것과 같은 내용을 출력하려면 -t 옵션(한글 지원하지 않음)을 사용하여 저장한 다음 프린터로 출력하여야 합니다.

man 명령의 결과를 파일에 저장하려면 > 파일명을 사용합니다.

【형식】

```
$man 명령어 > 파일명
```

```
ojk@raspberrypi:~ $ man ls > ls.txt
ojk@raspberrypi:~ $ ls -l
합계 52

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:14 Bookshelf

.............................................

-rw-r--r-- 1 ojk ojk   20  2 월 20 06:21 hello.py
```

```
-rw-r--r-- 1 ojk ojk 8070  2 월 24 02:25 ls.txt

drwxr-xr-x 2 ojk ojk 4096  2 월 24 02:16 ttt

ojk@raspberrypi:~ $ cat ls.txt
```

11) 파일 내용 보기

a) 파일 내용 출력

【형식】

```
cat 파일명
```

```
ojk@raspberrypi:~ $ cat ls.txt
LS(1)                                                User Commands
LS(1)
NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...
       -a, --all
  ..............................................

SEE ALSO
       Full documentation
<https://www.gnu.org/software/coreutils/ls>
       or available locally via: info '(coreutils) ls
invocation'

GNU coreutils 8.32                                   September 2020
LS(1)
ojk@raspberrypi:~ $
```

b) 파일의 내용 한 화면씩 보기

【형식】

```
$more   파일명
```

기 능 주어진 파일의 내용을 한 화면씩 출력시킨다.

문 법 more <file1> <file2> ... <fileN>

<file1>부터 <fileN>까지의 파일을 출력시킨다.

유닉스상의 more는 DOS와는 달리 실행중에 몇가지 명령어가 제공되어 그 명령어를 이용하므로써 파일의 내용을 편리하게 볼 수 있습니다. 명령어는 아래와 같습니다.

h	도움말
SPACE, z	다음 페이지
RETURN	1 라인 스크롤
d, ^D	반 페이지 스크롤
q, Q	종료
f	다음 페이지
b, ^B	이전 페이지
/pattern	검색
=	현재 라인 출력
!<command>	
:!<command>	<command> 명령어 실행
^L	화면 다시 출력
:f	현재 파일명과 라인 출력

사용예

```
more  temp/readme
```

temp/readme 파일을 페이지 단위로 출력시킨다.

12) 디스크 및 파일

a) 디스크 사용량 검사

【형식】

```
$ du
```

화일 시스템에서 현재의 위치와 그 이하 디렉토리의 크기를 블럭 단위로 출력

```
ojk@raspberrypi:~ $ du
8   ./.config/dconf
8   ./.config/chromium/FileTypePolicies/58/_metadata
28  ./.config/chromium/FileTypePolicies/58
32  ./.config/chromium/FileTypePolicies
300 ./.config/chromium/ShaderCache/GPUCache
  ..............................................
44  ./.cache/menus
25192       ./.cache
16  ./Downloads
8   ./Music
102944      .
ojk@raspberrypi:~ $
```

b) 사용 가능한 디스크 용량

【형식】

```
$ df  또는 df -h
```

df는 사용가능한 디스크 용량을 블럭 단위로 보여줍니다.

```
ojk@raspberrypi:~ $ df
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/root       30343244 4109112  24944268  15% /
devtmpfs         1709312       0   1709312   0% /dev
tmpfs            1874176       0   1874176   0% /dev/shm
tmpfs             749672    1248    748424   1% /run
tmpfs               5120       4      5116   1% /run/lock
```

```
/dev/mmcblk0p1    261108   50934    210174  20% /boot
tmpfs             374832      24    374808   1% /run/user/1000
```

df -h (h는 human) 는 인간이 알아보기 쉬운 형태의 M, G 단위로 남은 용량을 출력합니다.

```
ojk@raspberrypi:~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        29G  4.0G   24G  15% /
devtmpfs        1.7G     0  1.7G   0% /dev
tmpfs           1.8G     0  1.8G   0% /dev/shm
tmpfs           733M  1.3M  731M   1% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
/dev/mmcblk0p1  255M   50M  206M  20% /boot
tmpfs           367M   24K  367M   1% /run/user/1000
ojk@raspberrypi:~ $
```

c) 파일의 모음

윈도에서 사용하는 것처럼 파일 압축이라고 생각하면 안 됩니다. 유닉스나 리눅스 운영체제는 서버로 많이 사용하기 때문에 백업이 중요하고 백업 대상을 파일이나 백업장치에 직접 백업하는 경우가 있습니다. 가장 안전한 장치는 테이프 장치로의 백업이 대표적입니다. 물론 파일에 백업할 수도 있습니다.

【형식】

```
$tar command  저장파일명 또는 장치이름  압축 대상  -option
```

Options:

-f : 대상 tar 아카이브 지정. (기본 옵션)

-c : tar 아카이브 생성. 기존 아카이브 덮어 쓰기. (파일 묶을 때 사용)

-x : tar 아카이브에서 파일 추출. (파일 풀 때 사용)

-v : 처리되는 과정(파일 정보)을 자세하게 나열.

-z : gzip 압축 적용 옵션.

-j : bzip2 압축 적용 옵션.

-t : tar 아카이브에 포함된 내용 확인.

-C : 대상 디렉토리 경로 지정.

-A : 지정된 파일을 tar 아카이브에 추가.
-d : tar 아카이브와 파일 시스템 간 차이점 검색.
-r : tar 아카이브의 마지막에 파일들 추가.
-u : tar 아카이브의 마지막에 파일들 추가.
-k : tar 아카이브 추출 시, 기존 파일 유지.
-U : tar 아카이브 추출 전, 기존 파일 삭제.
-w : 모든 진행 과정에 대해 확인 요청. (interactive)
-e : 첫 번째 에러 발생 시 중지.

tape device 로 backup 하는 경우에는 file_name 대신에 tape_device_name

【주의!】

- **압축 대상에 절대경로로 지정하는 것과 상대경로로 지정하는 것은 해제할 때 다름.**
- **option 을 앞에 기술하면 안됨**

```
# tar cvf  user1.tar  /home/user1/
```

d) 파일의 압축

【형식】

```
$gzip file-name
$gzip -9 file-name
```

9는 압축률을 높이는 옵션

【주의!】

윈도와는 다르게 리눅스에서는 파일을 압축하면 원본은 없어집니다. 따라서 원본을 보존하려면 미리 복사하여 두어야 합니다.

```
ojk@raspberrypi:~ $ ls -l

합계 52

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:14 Bookshelf

  ..............................................
```

```
-rw-r--r-- 1 ojk ojk   20  2 월 20 06:21 hello.py

-rw-r--r-- 1 ojk ojk 8070  2 월 24 02:25 ls.txt

drwxr-xr-x 2 ojk ojk 4096  2 월 24 02:16 ttt

ojk@raspberrypi:~ $ gzip ls.txt
ojk@raspberrypi:~ $ ls -l

합계 48

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:14 Bookshelf

drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Desktop

  ..............................................

-rw-r--r-- 1 ojk ojk   20  2 월 20 06:21 hello.py

-rw-r--r-- 1 ojk ojk 3134  2 월 24 02:25 ls.txt.gz

drwxr-xr-x 2 ojk ojk 4096  2 월 24 02:16 ttt

ojk@raspberrypi:~ $
```

e) 파일 압축 풀기

【형식】

```
gunzip fine-name.gz 또는 gunzip -d file-name.gz
```

```
ojk@raspberrypi:~ $ gunzip ls.txt.gz
ojk@raspberrypi:~ $ ls -l
합계 52
drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:14 Bookshelf
drwxr-xr-x 2 ojk ojk 4096  9 월 22 09:37 Desktop
  ..............................................
-rw-r--r-- 1 ojk ojk   20  2 월 20 06:21 hello.py

-rw-r--r-- 1 ojk ojk   20  2 월 20 06:21 hello.py
-rw-r--r-- 1 ojk ojk 8070  2 월 24 02:25 ls.txt
drwxr-xr-x 2 ojk ojk 4096  2 월 24 02:16 ttt
ojk@raspberrypi:~ $
```

tar로 파일을 모으고 gzip로 압축

```
# tar cvfz user1.tgz /home/user1
```

f) file 의 유형의 판독

일반적으로 파일의 이름을 보면 파일의 유형을 짐작할 수 있습니다. 하지만 파일의 이름이 변경되거나 파일의 특성을 알 수 있도록 파일명이 지정되지 않은 경우에는 그 파일의 유형을 알 수 없는데 이를 알 수 있게 하는 명령이 file 명령입니다. 파일은 header 에 magic number이라는 것을 가지고 있는데 /usr/share/magic 에 구축된 정보를 이용하여 file header의 magic number를참조하여 파일의 유형을 출력합니다.

【형식】

```
$ file file_name
```

```
ojk@raspberrypi:~ $ file hello.py
hello.py: ASCII text
ojk@raspberrypi:~ $
```

13) 파일시스템의 사용

a) 디바이스 이름

Sata Interface

Sata Interface 첫번째 하드 드라이브: /dev/sda

첫 번째 파티션 /dev/sda1, 두 번째 파티션 /dev/sda2

Sata In terface 두번째 하드 드라이브: /dev/sdb

첫 번째 파티션 /dev/sdb1, 두 번째 파티션 /dev/sdb2

CD-ROM /dev/sr0

라즈베리파이에서는 USB를 /dev/sda, /dev/sdb 로 인식합니다. 일반적으로 USB의 파티션은 하나로 사용하기 때문에 대부분의 USB는 /dev/sda1 으로 인식합니다.

CD-ROM은 파티션이 1개이므로 /dev/sr0

b) mount

mount의 사용

mount란 file system을 OS가 사용할 수 있도록 연결해 주는 과정을 말합니다. 즉, file을 저장할 수 있는 device를 OS가 사용할 수 있도록 준비하는 절차입니다. Linux에서는 file system device는 반드시 mount과정을 거쳐 연결하고 umount과정을 통해서 연결을 해제합니다.

mount 명령은 다음과 같이 실행합니다.

【형식】

```
# mount -t <type > <device name> <mount 될 directory>
```

type ext2, ext3, ext4 : linux file system

iso9660 : cd-rom file-system

vfat : fat32 file system

ntfs : ntfs file system

hfs : Mac file system

마운트하지 전에 반드시 마운트 될 디렉토리가 있어야 합니다.

usb 장치를 /root/usb 에 마운트

```
# mount -t vfat /dev/sda1 /root/usb
```

c) mount 해제

```
# umount <device-name> or <mount-point>
```

주의 : shell 입력창에서 mount 된 directory가 현 directory 인 경우에는 umount 가 되지 않음

예) windowsXX가 설치되어 있는 partition의 mount

```
#  mount -t vfat /dev/hda1  /mnt/c
```

예) ide cd-Rom 의 mount

```
# mount -t iso9660 /dev/hdc /mnt/cdrom
```

예) scsi cd-rom의 mount

```
# mount -t iso9660 /dev/scd1 /mnt/cdrom
```

예) floppy disk 의 mount

```
# mount -t vfat /dev/fd0 /mnt/floppy
```

예) 아이오메가 사의 scsi 형 jazz 의 mount

```
# mount -t vfat /dev/sda4 /mnt/jazz
```

예) Mac용 HDD를 mount

```
# mount -t hfs /dev/sdc1 /mnt/mac
```

d) 부팅시 mount

부팅시 mount가 되도록 지정하려면 /dev/fstab 파일에 mount 정보를 지정하면 됩니다. 이 파일을 잘못 지정하는 경우에는 booting 이 되지 않으므로 주의하여야 합니다.

```
/etc/fstab 파일의 예

#㉠        ㉡           ㉢           ㉣           ㉤     ㉥
/dev/hdb1  /            ext2         defaults          1
    1
/dev/hda1  /mnt/dos     msdos        defaults          0
    0
/dev/hda5  swap         swap         defaults          0
    0
/dev/fd0   /mnt/floppy  ext2         noauto       0    0
/dev/cdrom /mnt/cdrom   iso9660             noauto,ro   0
    0
none              /proc        proc         defaults
    0      0
```

㉠ : 장치파일 이름

㉡ : mount point

㉢ : file system

㉣ : mount option

defaults : rw suid : Set User ID 비트가 유효한 상태

dev : 장치 파일을 사용할 수 있게 해줌

exec : 실행파일을 실행할 수 있게 mount

auto : booting 과정에서 자동으로 mount

nouser : 일반 사용자가 mount 할 수 없게 한다.

async(Asynchronous) :파일관련 작업을 동시에 수행하지 않음

㉤ : file system backup 여부

dump command 로 file system을 백업할 지의 여부

1이면 backup 0이면 backup 하지 않음

㉥ : file system 점검 순서

fsck command 로 파일시스템의 점검여부설정

0이거나 아무 것도 없으면 점검을 하지 않음

14) 퍼미션과 링크

a) ls 파일 종류 표시 문자

문자	나타내는 파일의 종류
d	directory
b	block 형 특수 화일
c	문자형 특수 화일
l	symbolic link
p	pipe
s	소켓
-	일반 화일

b) 화일 허가의 형태

- read : 화일의 내용조사, 변경 삭제 불가, 화일 복사 가능, 복사본 편집
- write: 화일에 정보 첨가, 변경
- execute : 화일을 program 처럼 실행하도록 함.

read(4) write(2) excute(1)

$$\text{excute} : 2^0, \text{write} : 2^1, \text{read} : 2^2$$

c) permission 의 지정과 변경

다중 사용자 환경인 Unix는 사용자를 크게 3가지 그룹으로 분류하는데 이에는 소유자(owner), 그룹(group), 다른 사용자(other)가 있습니다. owner은 사용자 자신을 말하는 것이고 group는 사용자가 속하여 있는 그룹은 의미하며 other은 사용자가 속하여 있는 그룹이 아닌 다른 사용자를 말한다.

Unix에서는 여러 사용자가 시스템을 사용하여 공동작업을 하기 때문에 각각 사용자에 따른 권한이 다른데 이를 permission 이라고 합니다. 예를 들면 ojkojk라는 사용자가 작성한 파일을 widget라는 사용자가 변경하거나 삭제한다면 여러 사용자가 시스템을 안전하고 효율적으로 사용할 수 없기 때문에 이를 막기 위해서 사용자와 그 사용자가 속하여 있는 그룹이 작성한 파일과 디렉토리의 권한을 제한하는데 이 때 사용하는 명령이 chmod("취모드" 라도 발음

함)입니다.

```
$ ls -l sample.txt
-rwxrwxr-x 1 ojkojk  teacher  30 June 10:20
```

[설명]
sample.txt 는 파일이고 소유자는 ojkojk 이며 소유자가 속하여 있는 그룹은 teacher이고 6월 30일 10시 20분에 이 파일을 생성하였는데 소유자는 읽고(r), 쓰고(w), 실행(x)할수 있으며 이 소유자가 속하여 있는 teacher 그룹에 해당하는 다른 사용자는 읽고 쓰고 실행할 수 있으며 다른 사용자는 읽고 실행할 수만 있다.

이러한 파일과 디렉토리의 속성의 변경을 다음 명령을 사용합니다.

【형식】

```
$ chmod  속성  대상 옵션
```

read (4) , write (2) , excute(1)

예) sample.txt 라는 파일에 owner에는 모든 권한 group에는 읽고 쓰는 권한 other에는 읽는 권한만 부여

```
$ chmod 764 sample.txt
```

예) sample.txt 라는 파일에 owner에는 모든 권한 group에는 읽고 실행하는 권한 other에는 접근 금지

```
$ chmod 750 sample.txt
```

예) sample.txt 라는 파일에 owner에는 모든 권한 나머지는 읽고 실행하는 권한 부여

```
$ chmod 755 sample.txt
```

d) 파일 link

하드 링크 : root 만이 할 수 있음

inode와 용량이 같은 파일 생성

같은 파일시스템에서만 가능

원본 파일이 삭제되더라도 나머지link된 파일을 통해 참조할 수 있음

【형식】

```
# ln file-name-1 file-name-2
```

symbolic link :

root 뿐만 아니라 누구나 가능

inod 가 다른 파일 생성, 링크되는 파일의 정보만 가지고 있어 용량이 적다.

파일 시스템이 다르더라도 가능

원본파일 삭제되면 내용을 참조할 수 없으나 링크된 파일은 남아 있어 상황에 따라 삭제되고 생성되는 파일을 동적으로(Dynamic) 사용할 수 있다.

Link 된 파일은 뒤에 @ 기호가 붙음

directory 는 symbokic link 만 가능

【형식】

```
# ln -s 원본파일명 심볼릭링크파일명
```

e) 접근한계 기본 설정

Unix 환경하에서는 파일과 디렉토리는 반드시 퍼미션을 갖게 되며, 처음으로 생성되는 파일과 디렉토리도 어떤 퍼미션을 가져야 한다. 처음으로 생성되는 파일과 디렉토리의 퍼미션 지정은 umask를 사용한다. 즉, umask는 directory 와 file을 새로 생성할 때 적용되는 permission 을 지정하면 이의 default는 일반 users는 002이고 root 022 이다.

umask 값 display

【형식】

```
$ umask
```

umask 값 새로 지정 (umask 값은 chmod 값과 반대)

【형식】

```
$ umask umask_value(숫자)
```

umask 로 지정한 excute 속성은 directory 에만 적용되고 file에는 적용되지 않음
즉, umask 077 지정후 새로 생성된 file 의 permission 은 600 mode

1. vi editor

vi 편집기는 입력 모드(insert mode)와 명령어 모드(command mode) 그리고 라인 모드(line mode)로 나누어진다. 입력 모드는 일반적으로 원하는 문자를 파일에 입력할 수 있는 상태를 말한다. 명령어 모드는 입력한 내용을 편집하기 위해서 다양한 편집 용어를 사용할 수 있게 해준다.
처음 vi 편집기를 들어가면, 명령 모드에 위치하게 되는데, i/I(nsert)나 a/A(pend) 또는 o/O(pen)을 입력하거나 □를 누르면 입력 모드로 전환된다. 그렇게 해서 입력하다가 다시 명령 모드로 전환하고 싶다면, □를 누르면 된다.

작업한 내용을 저장하기 위해서는 명령 모드에서 :(콜론)을 누르면, 화면 아래의 왼쪽에 :이 찍히게되며, 저장하고 종료할 것인지(wq), 아니면 저장하지 않고 종료할 것인지(q)를 정해서 명령을 내리고 나오면 된다. 저장하지 않고 강제로 빠져 나오려면, !(느낌표)를 추가하여 :q! 처럼 해주면 된다.

이제, 명령 모드에서 입력 모드로의 전환에 사용되는 명령어와 이동 명령어 및 vi 편집기에서 사용하는 기본 명령들을 나열하겠다.

[1] 사용

■ 입력 모드
. a : 커서 다음부터 입력
. A : 커서가 있는 행의 끝부터 추가
. i : 커서 앞에서부터 입력
. I : 커서가 있는 행의 처음부터 추가
. o : 커서가 있는 행의 다음 줄에 행 삽입하며 입력
. O : 커서가 있는 행의 이전 줄에 행 삽입하며 입력
. s : 커서가 있는 단어를 지우고 입력
. S : 커서가 있는 행을 지우고 입력
. □ : I와 같은 수행

■ 이동 명령어
- 문자 단위 이동-
. h : 한 문자 왼쪽으로 이동
. j : 한 문자 아래로 이동
. k : 한 문자 위로 이동

. l : 한 문자 오른쪽으로 이동

- 단어 단위 이동 명령어 -
. w : 다음 단어의 처음으로 이동
. W : 다음 단어의 처음으로 공백 단위로 이동
. b : 이전 단어의 처음으로 이동
. B : 이전 단어의 처음으로 공백 단위로 이동
. e : 다음 단어의 끝으로 이동
. E : 다음 단어의 끝으로 공백 단위로 이동한다.

- 행 단위 이동 -
. 0(zero) : 현재 행의 처음으로 이동
. ^ : 현재 행의 처음 문자로 이동
. $: 현재 행의 끝으로 이동
. □ : 다음 행의 처음으로 이동

■ 편집 명령어
. x : 현재 커서의 위치 문자를 삭제
. X : 이전의 문자를 삭제
. nX : 이전의 n개를 삭제
. dw : 커서의 위치부터 단어를 오른쪽으로 삭제
. db : 커서의 위치부터 단어를 왼쪽으로 삭제
. dW : 커서의 위치부터 단어를 오른쪽으로 공백 단위 삭제
. dB : 커서의 위치부터 단어를 왼쪽으로 공백 단위 삭제
. dd : 커서가 있는 행을 삭제
. D : 커서의 위치부터 행의 오른쪽 위치문자들을 삭제하고, 이동한다.
. yy : 현재의 행을 버퍼로 복사
. p : 버퍼의 내용을 현재 커서의 아래 줄에 붙여 넣기

■ 저장 및 종료
. :q : 편집이 이루어지지 않았을 경우에 한해서 종료
. :q! : 변경된 내용을 저장하지 않고 종료
. :x : 저장하고 종료

[2] 환경 설정

명령어 모드에서 :set을 실행하면 현재 설정된 옵션들이 화면에 출력된다.

몇 가지 유용한 옵션은 다음 표와 같다.

옵션	옵션 약어	기능	기본
tabstop=n	ts=n	**탭 공백을 n 수 만큼 지정한다.**	8
number		**줄 번호가 나오게 설정한다.**	off
nonumber		**줄 번호가 나오지 않게 설정한다.**	on

이러한 옵션을 vi를 실행할 때마다 자동으로 지정되게 하려면 사용자의 홈디렉토리의 .exrc 파일을 편집하면 된다.
줄 번호가 나오게 하고 탭 간격을 3으로 지정하려면 .exrc 파일에 다음 내용을 삽입하면 된다.

```
set number
set tabstop=3
```

3. local 에서 Login 하기

새롭게 리눅스를 설치한 후, 처음으로 새롭게 부팅할 때, 화면에 복잡한 메시지가 빠르게 지나가는 것을 볼 수 있을 것이다. 이 내용들은 시스템이 부팅되는 과정을 화면에 출력해 주는 것이다. 따라서, 이 부분을 잘 이해함으로써 시스템이 어떻게 작동되는지 약간이나마 이해 할 수 있다. 시스템이 부팅될 때에 나오는 메시지는 /var/log/dmesg에 저장되어 있다. 부팅 메시지를 확인하기 위해서는 이 파일을 에디터로 본다

vi /var/log/dmesg 또는 more /var/log/dmesg

로그인할 때는 콘솔에서의 방법과 X 윈도우에서의 방법이 있다.

【01】콘솔에서 로그인 하기

콘솔에서 로그인 할 때는

Linux release 6.0

Kernel 2.2 on an i586

Login :

Password :

위와 같은 화면을 보게 된다.

처음에는 어떻게 해야 할지 당황할 수 있지만 그럴 필요는 없다. 리눅스를 설치할 때 마지막 부분에서 root 암호를 입력하는 부분이 있었을 것이다. 설치 시에 입력했던 암호를 입력해야 한다.

Linux release 6.1

Kernel 2.2 on an i586

Login : root

Password : ******** <-- root 암호 입력

▶ root(루트)
새로 설치한 리눅스의 모든 계정과 자원을 관리하는 책임자 계정을 말한다. root는 시스템 내에서 모든 일을 할 수 있기 때문에, 항상 주의해야 한다. 따라서, root도 자신의 사용자 계정을 가지고 있어야 한다. root로서 해야 할 일이 아니면, root로 로그인하는 것은 결코 좋은 생각이 아니다.

그런데, 암호를 입력하는 부분이 화면에 나오지 않는다고 입력이 되지 않고 생각할 필요는 없다. 보안을 위해서 이러한 방식을 사용하는 것이다.

▶ 암호
이전까지는 최소한 여섯 글자이며, 여덟 글자까지 인식하지 못했다. 하지만, 리눅스 6.0부터는 256자까지 사용할 수 있게 되었다.

리눅스에서는 대소문자를 구분하기 때문에 Root와 root는 서로 다른 계정으로 인식한다. 그리고, 암호를 입력할 때, 공백이나 다른 기호를 입력하는 것은 다른 암호를 나타내므로 입력할 때 주의를 기울여야 한다.
올바르게 계정과 암호를 입력했다면, 엔터를 눌러 로그인 하도록 한다. 로그인이 올바르게 되었다면 다음과 같은 명령 명령행을 얻게 될 것이다.

[root@localhost /root]#

위에서도 언급했지만, root로 로그인하는 것은 좋은 습관이 아니다. 일반 사용자 계정으로 로그인한 후 root가 아니면 할 수 없는 일이 발생했을 경우에만 su로 root가 되어서 작업을 수행하는 것이 좋다.

【02】X 윈도우에서 로그인하기

X 윈도우로 로그인 할 때는

위와 같이 보일 것이다. (실제 사용자가 보는 화면은 설정에 따라 다를 수 있다.)
콘솔에서와 마찬가지로 Login 부분에 마우스를 가져다 놓고, root라고 입력한다. 그리고, □를 누르거나 마우스를 이동해서 root 암호를 입력한다. 입력했다면 □나 Login을 눌러 로그인 한다. 올바로 입력되었다면, 다음과 같은 화면이 나타날 것이다.

5.시스템의 종료

시스템 종료는 root만이 할 수 있으므로, 일반 사용자 계정 상태에 있다면 su를 이용해 root로 바꾸어야 한다.

만약 일반 사용자가 종료시키려고 한다면 root의 암호를 입력하라는 메시지가 나올 것이다.

가. 콘솔모드에서 시스템 종료하기

시스템을 종료시키는 방법은 간단히 shutdown 명령으로 할 수 있다.

명령행에서 다음과 같이 입력한다.

```
[root@localhost /root]# shutdown -h now
```

시스템 종료가 아니라 시스템을 재부팅하고 싶다면,

```
[root@localhost /root]# shutdown -r now
```

라고 입력하면 된다.

-h 옵션이 halt의 뜻을 나타내는 반면에 -r 옵션은 reboot를 뜻한다. now는 이 명령을 지금 바로 실행하라는 뜻이다

now 대신에 +5도 바꾸어서 실행하면 시스템을 5분 후에 종료시킬 수도 있다. 이 시간적인 여유를 둠으로써 현재하고 있던 작업을 안전하게 종료시킬 수 있게 되는 것이다.

만약 시스템을 종료 시켰다면, 모든 프로세스가 종료되었다는 메시지와 함께 'The system is halted' 라는 메시지를 보게 될 것이다.

이제 모든 작업이 종료된 것이다. 컴퓨터의 전원을 내려도 안전하다.

나.GNOME에서 시스템 종료하기

GNOME에서 시스템을 종료하거나 재부팅하기 위해서는 현재의 X 화면을 종료하고 로그인 화면으로 나와야 한다. 패널에서 로그아웃(Log out)을 선택하여 X를 종료할 수 있다. 로그인 화면으로 나왔다면, 옵션(Options)을 선택하고 System에서 Halt나 Reboot를 선택한다.

그러면, 당신은 컴퓨터를 정말 종료하고 싶은지 아니면 다시 시작하고 싶은지에 대한 질문을 받을 것이다. 정말로 종료하려면 Yes를 선택한다. 다시 시작하고 싶다면 No를 선택해라.

6.디렉토리 구조

대부분의 리눅스는 FHS(Filesystem Hierarchy Standard) 표준 파일시스템 계층을 사용하기 때문에 시스템 자원이나 프로그램들을 쉽게 찾을 수 있다. 디렉토리 구조의 시작은 ‘ / ’(루트 디렉토리)이다.

‘ / ’(루트 디렉토리) 바로 하위단계에는 시스템에 매우 중요한 몇 개의 디렉토리들이 존재한다. /bin, /etc, /dev, /usr, /sbin, /var, /lib, /tmp, /proc, /lost+found, /home 등을 있다.

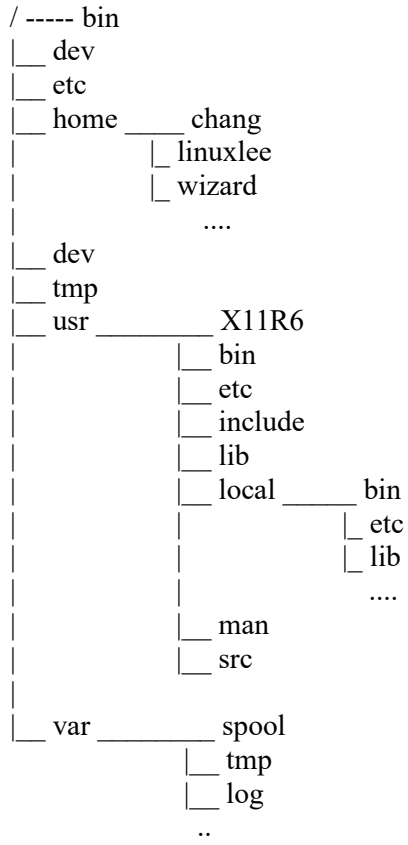
각각의 디렉토리를 간단하게 살펴보면 다음과 같다.

■ /bin : binaries의 약어. 이진파일들이며, 기본 실행 명령들이 존재한다.
■ /etc : 시스템 설정파일을 모아 놓은 디렉토리이다. 리눅스에서 없어서는 안될 디렉토리이다.
■ /dev : 리눅스에서 사용하는 장치를 모아 놓은 디렉토리이다. 대표적으로 하드 드라이브, 플로피, CD-ROM 그리고 루프백 장치 등이 존재하며, 그밖에 다양한 장치들이 존재한다.
■ /usr : 역시 가장 중요한 디렉토리 중 하나로써, 사용하면서 공유할 수 있는 파일과 디렉토리들을 가지
고 있다. 그 내용은 다음과 같다.
·/bin : 루트 디렉토리의 /bin 디렉토리에서 찾을 수 없는 유용한 프로그램들이 들어 있다.
·/etc : 응용 프로그램들의 설정 파일이 위치한다.
·/include : C 프로그램의 헤더 파일이 저장된다.
·/lib : 루트의 /lib의 한 부분이라고 생각할 수 있다.
·/src : 대부분 커널 소스들이 위치한다.
·/tmp : 루트의 /tmp와 같은 역할
·/man : 맨 페이지를 출력하는데 필요한 파일들이 있다.
■ /home : 시스템 사용자들의 홈 디렉토리가 존재하는 디렉토리이다.
■ /var : 시스템 로그 파일들이 저장되어 있는 디렉토리이다.
■ /lib : 프로그램들의 라이브러리들이 존재한다. 대부분 공유 라이브러리로써 사용 할 때 더 편리하게 사
용할 수 있으며, 파일의 크기를 줄여서 실행할 때 불러 사용하게 된다.
■ /proc : 이 디렉토리는 실제로 존재하는 것은 아니다. 시스템이 관리를 목적으로 메모리 상

에 만들어

놓은 가상의 디렉토리이다. 그 디렉토리 안의 파일들은 현재의 시스템 설정을 보여준다.

트리로 나타내면 다음과 같다. 이러한 기본 구조에 자신이 필요하면 만들 수 있다.



다른 리눅스 배포본일지라도, 리눅스 시스템은 서로 호환이 쉽게 될 수 있다. 이유는 파일시스템이 표준이기 때문이다.

II. 시스템 설정

1. Network

Raspberry Pi OS (formerly know as Raspbian) uses DHCP (Dynamic Host Configuration Protocol) to assign an IP address to the Raspberry Pi automatically whenever it is rebooted. If you have any WIFI router around your Raspberry Pi, your Raspberry Pi will get automatically any ip address.

If you have physical access to your Raspberry Pi and have a display, then the process of finding your IP address is straightforward. All you need to find out the IP address of your Raspberry Pi is to run the following command in the terminal.

```
hostname -I
```

Make sure that the I is capitalized to retrieve the IP addresses for all hostnames. If you use a lowercase i you will be grabbing the hostname instead.

【Example】

```
ojk@raspberrypi:~ $ hostname -I
192.168.45.169
```

Or you will use the following command in the terminal.

```
ip r
```

【Example】

```
default via 192.168.45.1 dev eth0 proto dhcp src 192.168.45.169
metric 202
192.168.45.0/24 dev eth0 proto dhcp scope link src 192.168.45.169
metric 202
```

Or you can still use the following command in the terminal.

```
ifconfig
```

【Example】

```
ojk@raspberrypi:~ $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.45.169  netmask 255.255.255.0  broadcast
192.168.45.255
        inet6 fe80::66f0:12d:3938:e7d  prefixlen 64  scopeid
0x20<link>
```

```
        ether dc:a6:32:90:df:b7  txqueuelen 1000  (Ethernet)
        RX packets 27150  bytes 3067853 (2.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1993  bytes 1214569 (1.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 52  bytes 5887 (5.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 52  bytes 5887 (5.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

【01】DHCP IP Settings

If you want to get dhcp ip address on your raspberry Pi, you need not to work. Just do boot your PI.

【02】Static IP Settings

To change Raspberry Pi OS's behavior so that it uses the same static IP address each time, you will need to modify the configuration file for the DHCP client daemon, **dhcpcd.conf**.

Before that, you will need some information on your current network setup so you can add the required details to the configuration file. You will require the following info:

The Raspberry Pi's currently assigned IP address – it's safest to reuse this for its static IP so that you can be sure the latter hasn't already been to another device on the network. If not, make sure another device isn't already using it.

Your router's DNS (Domain Name System) IP address. This is typically the same as its gateway address, but may be set to another value to use an alternative DNS – such as 162.168.63.1 for KT, or 210.220.163.82 for SK Broadband.

To find the current DNS IP address, enter the command:

cat /etc/resolv.conf

【Example】

```
root@raspberrypi:~# cat /etc/resolv.conf
# Generated by resolvconf
nameserver 210.220.163.82
nameserver 219.250.36.130
```

Now you have found all your network connection information, it's time to edit the **/etc/dhcpcd.conf** configuration file to add the settings you need to set up a static IP address for your Raspberry Pi:

If you haven't edited the file previously, it will mainly contain various comment lines preceded by a hash (#) symbol. At the bottom, add the following lines, replacing the emboldened names with your own network details:

interface NETWORK
static ip_address=STATIC_IP/24
static routers=ROUTER_IP
static domain_name_servers=DNS_IP

attention: not /etc/dhcpd.conf but /etc/dhcpcd.conf

Replace the emboldened names as follows:

- NETWORK – your network connection type: eth0 (Ethernet) or wlan0 (wireless).
- STATIC_IP – the static IP address you want to set for the Raspberry Pi.
- ROUTER_IP – the gateway IP address for your router on the local network.
- DNS_IP – the DNS IP address (typically the same as your router's gateway address).

Here is an example configuration to set the static IP to 192.168.45.5 with a Ethernet connection to a router at 192.168.45.1:

【Example】

```
interface eth0
static ip_address=192.168.45.5/24
static routers=192.168.45.1
static domain_name_server=210.220.163.82
```

Reboot the Raspberry Pi
With the dhcpcd.conf configuration file modified, restart your Raspberry Pi to effect the changes and set the static IP address for it:

```
sudo reboot
```

Or you can use the following command to restart networking service.

```
sudo /etc/init.d/networking restart
```

Now rather than using an address assigned automatically by DHCP, the Raspberry Pi will now

attempt to connect to the router using the new static IP address that you set in the dhcpcd.conf file.

To check that it is working correctly, enter the following command:

```
hostname -I
```

You should now see the static IP address that you set in the dhcpcd.conf configuration file.

【Example】

```
ojk@raspberrypi:~ $ hostname -I
192.168.45.5
ojk@raspberrypi:~ $
ojk@raspberrypi:~ $ ip r
default via 192.168.45.1 dev eth0 src 192.168.45.5 metric 202
192.168.45.0/24 dev eth0 proto dhcp scope link src 192.168.45.5
metric 202
ojk@raspberrypi:~ $
```

You can see the difference between dhcp and static.

```
default via 192.168.45.1 dev eth0 proto dhcp src 192.168.45.169
metric 202
192.168.45.0/24 dev eth0 proto dhcp scope link src 192.168.45.169
metric 202
```

```
default via 192.168.45.1 dev eth0 src 192.168.45.5 metric 202
192.168.45.0/24 dev eth0 proto dhcp scope link src 192.168.45.5
metric 202
```

라즈베리파이4은 기본적으로 WIFI가 장착이 되어 있으므로 무선공유기가 있으면 자동으로 네트워크 설정이 이루어집니다.

X-Window에서 설정은 간편하지만 설정 값이 적용되지 않는 경우가 많으므로 사용하지 않는 것이 좋습니다. 편리하기는 하지만 네트워크 설정이 정확하게 이루어지지 않으면 의미가 없으므로 반드시 명령어 라인에서 설정을 합니다.

사용자가 필요에 의해 수동으로 설정하는 경우에는 윈도우 오른쪽 상단의 아이콘을 클릭하여 수동으로 설정값을 입력한다.

네트워크 설정은 예전과는 다르게 변경이 되었습니다. 현재는 /etc/dhcpcd.conf 에 설정값을 저장합니다. 예전 방식으로 설정이 되어 있더라도 이 설정 파일이 우선적으로 적용이 됩니다.

【01】유선랜 eth0 설정

Ethernet 사용 시 eth0 으로 설정

【형식】

```
/etc/dhcpcd.conf
interface eth0
static ip_address=<IP address>/24
static routers=<Gateway address>
static domain_name_servers=<DNS>
```

```
$ sudo vi /etc/dhcpcd.conf
interface eth0
static ip_address=192.168.45.5/24
static routers=192.168.45.1
static domain_name_server=210.220.163.82
static domain_name_servers=
static domain_search=
```

라즈베리파이 3에서 설정

버젼이 올라가면서 설정 방식이 변하는 것은 당연하고 막을 수 없는 일입니다. 그런데 실제현장에서는 예전의 시스템을 사용하는 경우가 있으므로 예전 방식의 설정도 간단하게 나열하겠습니다.

라즈베리파이3에서 네트워크를 설정하는 파일은 /etc/network/interfaces입니다. 유동아이피로 설정되어 있다면 파일의 내용이 다음처럼 되어 있습니다.

This file describes the network interfaces available on your system
and how to activate them. For more information, see interfaces(5).

```
# File Name: /etc/network/interfaces
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp
```

고정 아이피로 바꾸려면

iface eth0 inet dhcp 을 iface eth0 inet static 로 변경하고

xxx.xxx.xxx.xxx에는 네트워크 환경에 맞는 IP를 적으면 됩니다.

```
# File Name: /etc/network/interfaces
# The loopback network interface
auto eth0
iface eth0 inet static
address xxx.xxx.xxx.xxx
netmask xxx.xxx.xxx.xxx
gateway xxx.xxx.xxx.xxx
dns-nameservers xxx.xxx.xxx.xxx
```

설정을 마치고 나면 다음과 같이 명령하여 네트워크를 재시작하거나 재부팅합니다.

```
/etc/init.d/networking restart
```

【02】WiFi 설정

WiFi 사용 시
WiFi 사용 시 인터페이스는 wlan0 이다.

$ sudo vi /etc/dhcpcd.conf
// 추가
...
interface wlan0
static ip_address=<IP address>/24
static routers=<Gateway address>
static domain_name_servers=<DNS>

국가설정을 US, GB로 한다.
첫 번째로 raspi-config를 실행해서 Localisation Options 메뉴에서 Wi-fi Country 설정을 하지 않도록 한다. 이것을 설정하지 않아도 아무 문제가 없는데 설정을 하면 무선 네트워크 검색이 되지 않는다. 하나도 검색이 안된다면 이 문제일 가능성이 있다.

wifi 상태 확인
$ sudo iwlist wlan0 scan

주변 ssid가 검색되면 하드웨어는 정상적으로 동작하고 있는 것이므로, 네트워크 설정만 해 주면 됩니다.

/etc/wpa_supplicant/wpa_supplicant.conf

network={

ssid="wifi ap ssid"

psk="wifi ap password"

}

ex)

/etc/wpa_supplicant/wpa_supplicant.conf

network={

ssid="ap ssid"

psk="wifi ap password"

}

설정을 마치고 나면 다음과 같이 명령하여 네트워크를 재시작합니다.

/etc/init.d/networking restart

DNS 서버는 1차와 2차가 있는데 라즈베리파이는 1차 DNS Server가 원활하지 않으면 2차 DNS 서버를 검색하지 않는 경우도 있으므로 1차 DNS Server를 신뢰할 수 있는 서버를 지정한다.

【03】절전 설정 해제

무선랜 설정 후 ssh 접속을 해보면 잠시만 자리를 비워도 연결이 끊어지는 경우가 생기는데 이 경우는 절전 설정이 되어 있어서 그렇습니다. 아래 방식대로 패치를 해주어야 합니다.

필자가 사용하는 ipTIME N100mini 모델은 8188cu 모델인데 설정은 8192cu 모델과 공유하는 것 같습니다. 아래 명령으로 설정 확인.

cat /sys/module/8192cu/parameters/rtw_power_mgnt

결과값이 1 또는 2가 나오면 절전 설정 상태이므로 수정

sudo vi /etc/modprobe.d/8192cu.conf

파일에 아래 내용을 입력하고 저장

options 8192cu rtw_power_mgnt=0 rtw_enusbss=0

설정이 끝나면 리부팅합니다.

그래도 여전히 같은 문제가 발생하곤 하는데... putty 같은 터미널 설정에서 null 패킷을 30초 단위로 보내도록 설정해 두면 끊김 방지를 할 수 있습니다.

【04】원격 접속 설정

라즈베리파이 3부터는 기본적으로 원격접속 설정 프로그램이 설치되어 있습니다.

만일 프로그램이 지우지거난 이상이 생긴 경우에는 다음과 같이 설치하고 환경설정을 합니다.

- VNC server: x11vnc
1) install vnc
$ sudo apt-get install x11vnc

quick test
/usr/bin/x11vnc -auth /var/run/lightdm/root/:0 -display :0

2) configuring x11vnc to start on boot

/etc/init/에 x11vnc.conf 파일을 생성한다.
$ sudo vi /etc/init/x11vnc.conf

다음과 같은 스크립트를 작성한다.
start on login-session-start
script
/usr/bin/x11vnc -xkb -noxrecord -noxfixes -noxdamage -auth /var/run/lightdm/root/:0 -passwd "MyPasswd" -forever -bg
end script

3) 작성한 스크립트를 root만 접근할 수 있게 한다.

$ sudo chown root:root /etc/init/x11vnc.conf
$ sudo chmod 600 /etc/init/x11vnc.conf

2. 프로그램 설치와 관리

【01】apt

apt-get(Advanced Packaging Tool)은 우분투(Ubuntu)를 포함안 데비안(Debian)계열의 리눅스에서 쓰이는 팩키지 관리 명령어 도구입니다. 우분투에는 GUI로 되어 있는 시냅틱 꾸러미 관리자도 있기는 하지만 이런 저런 개발관련 패키지를 설치할 때는 커맨드기반인 apt-get이 더 편하기도 합니다. sudo는 superuser권한으로 실행하기 위함입니다.

01)apt 환경 설정

```
ojk@raspberrypi:/etc/apt $ ls -l
total 36
drwxr-xr-x 2 root root 4096 Sep 21 19:33 apt.conf.d
drwxr-xr-x 2 root root 4096 Jun 10  2021 auth.conf.d
-rw-r--r-- 1 root root  150 Sep 21 19:07 listchanges.conf
drwxr-xr-x 2 root root 4096 Mar 28  2021 listchanges.conf.d
drwxr-xr-x 2 root root 4096 Jun 10  2021 preferences.d
-rw-r--r-- 1 root root  239 Sep 21 19:03 sources.list
drwxr-xr-x 2 root root 4096 Sep 21 19:03 sources.list.d
-rw-r--r-- 1 root root 1225 Sep 21 19:03 trusted.gpg
drwxr-xr-x 2 root root 4096 Sep 21 19:04 trusted.gpg.d
```

```
ojk@raspberrypi:/etc/apt $ cat sources.list
deb http://raspbian.raspberrypi.org/raspbian/ bullseye main
contrib non-free rpi
# Uncomment line below then 'apt-get update' to enable 'apt-get
source'
#deb-src http://raspbian.raspberrypi.org/raspbian/ bullseye main
contrib non-free rpi
ojk@raspberrypi:/etc/apt $
```

```
ojk@raspberrypi:/etc/apt $ sudo apt-get update
Hit:1 http://archive.raspberrypi.org/debian bullseye InRelease
Hit:2 http://raspbian.raspberrypi.org/raspbian bullseye
InRelease
Reading package lists... Done
```

```
ojk@raspberrypi:/etc/apt $ sudo vi sources.list
ojk@raspberrypi:/etc/apt $ cat sources.list
deb http://raspbian.raspberrypi.org/raspbian/ bullseye main
```

```
contrib non-free rpi
# Uncomment line below then 'apt-get update' to enable 'apt-get
source'
deb-src http://raspbian.raspberrypi.org/raspbian/ bullseye main
contrib non-free rpi
```

```
ojk@raspberrypi:/etc/apt $ sudo apt-get update
Hit:1 http://archive.raspberrypi.org/debian bullseye InRelease
Hit:2 http://raspbian.raspberrypi.org/raspbian bullseye
InRelease
Get:3 http://raspbian.raspberrypi.org/raspbian bullseye/contrib
Sources [82.2 kB]
Get:4 http://raspbian.raspberrypi.org/raspbian bullseye/main
Sources [12.2 MB]
Get:5 http://raspbian.raspberrypi.org/raspbian bullseye/non-free
Sources [141 kB]
Get:6 http://raspbian.raspberrypi.org/raspbian bullseye/rpi
Sources [1,132 B]
Fetched 12.4 MB in 12s (1,040 kB/s)
Reading package lists... Done
ojk@raspberrypi:/etc/apt $
```

15) 패키지 설치

```
sudo apt-get install 패키지이름
```

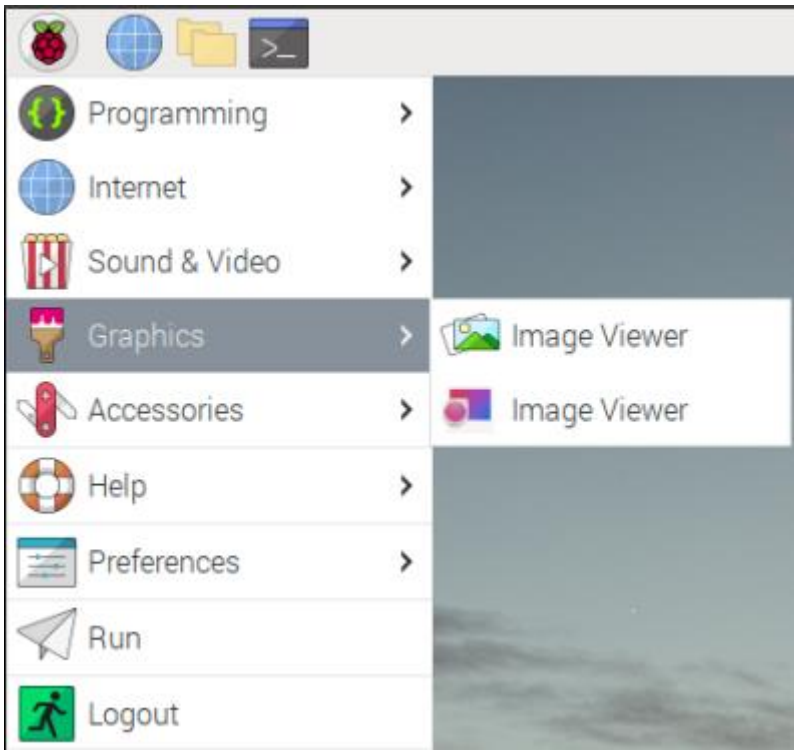
apt를 이용해서 설치된 deb패키지는 /var/cache/apt/archive/ 에 설치가 됩니다.

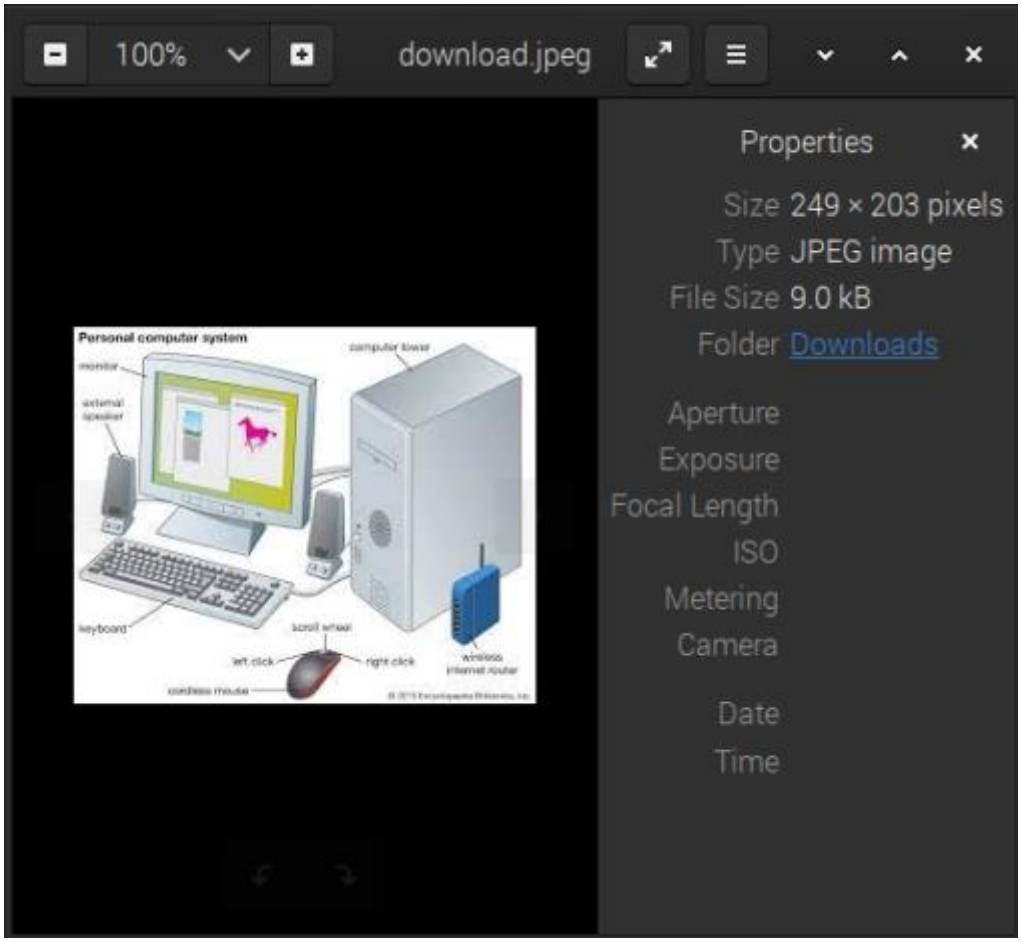
그림파일을 보여주는 eog라는 프로그램을 설치하여 봅시다.

```
apt $ sudo apt-get install eog
```

```
ojk@raspberrypi:/etc/apt $ sudo apt-get install eog
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no
longer required:
  libfuse2
.....................................................................
..............
.....................................................................
..............
Processing triggers for desktop-file-utils (0.26-1) ...
```

```
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for gnome-menus (3.36.0-1) ...
Processing triggers for libglib2.0-0:armhf (2.66.8-1) ...
Setting up docbook-xml (4.5-9) ...
Processing triggers for sgml-base (1.30) ...
ojk@raspberrypi:/etc/apt $
```






```
ojk@raspberrypi:/var/cache/apt/archives $ ls -l
total 10916
-rw-r--r-- 1 root root   84380 Jul 14  2019 docbook-xml_4.5-
9_all.deb
-rw-r--r-- 1 root root 3085704 Mar  1  2021 eog_3.38.2-
1_armhf.deb
-rw-r--r-- 1 root root   32604 May 22  2020 fonts-dejavu_2.37-
2_all.deb
-rw-r--r-- 1 root root 2070396 May 22  2020 fonts-dejavu-
extra_2.37-2_all.deb
-rw-r--r-- 1 root root   12424 Sep 24  2020 gir1.2-peas-
1.0_1.28.0-2_armhf.deb
-rw-r--r-- 1 root root  451100 Jul  1  2020 libexempi8_2.5.2-
1_armhf.deb
-rw-r--r-- 1 root root   56620 Sep 24  2020 libpeas-1.0-
0_1.28.0-2_armhf.deb
-rw-r--r-- 1 root root   53208 Sep 18  2020 libpeas-
common_1.28.0-2_all.deb
-rw-r--r-- 1 root root 1343956 Dec 25  2020 libpython3.8_3.8.7-
1_armhf.deb
-rw-r--r-- 1 root root  752848 Dec 25  2020 libpython3.8-
minimal_3.8.7-1_armhf.deb
-rw-r--r-- 1 root root 1647132 Dec 25  2020 libpython3.8-
```

```
stdlib_3.8.7-1_armhf.deb
-rw-r--r-- 1 root root  154808 Feb 25  2021 libyelp0_3.38.3-
1_armhf.deb
-rw-r----- 1 root root       0 Sep 21 19:33 lock
drwx------ 2 _apt root    4096 Feb 17 12:05 partial
-rw-r--r-- 1 root root  178976 Dec 29  2020 sgml-
data_2.0.11+nmu1_all.deb
-rw-r--r-- 1 root root  778196 Feb 25  2021 yelp_3.38.3-
1_armhf.deb
-rw-r--r-- 1 root root  444612 Feb 15  2021 yelp-xsl_3.38.3-
1_all.deb
ojk@raspberrypi:/var/cache/apt/archives $
```

16) 패키지 정보 보기

```
sudo apt-cache show 패키지이름
```

```
ojk@raspberrypi:/etc/apt $ sudo apt-cache show eog
Package: eog
Version: 3.38.2-1
Architecture: armhf
Maintainer: Debian GNOME Maintainers <pkg-gnome-
maintainers@lists.alioth.debian.org>
Installed-Size: 10074
Depends: libc6 (>= 2.29), libcairo2 (>= 1.14.0), libexempi8 (>=
2.5.0), libexif12 (>= 0.6.21-1~), libgdk-pixbuf-2.0-0 (>=
2.36.5), libgirepository-1.0-1 (>= 0.9.12), libglib2.0-0 (>=
2.42.0), libgnome-desktop-3-19 (>= 3.17.92), libgtk-3-0 (>=
3.22.0), libjpeg62-turbo (>= 1.3.1), liblcms2-2 (>=
2.2+git20110628), libpeas-1.0-0 (>= 1.0.0), librsvg2-2 (>=
2.44), libx11-6, zlib1g (>= 1:1.1.4), dconf-gsettings-backend |
gsettings-backend, gir1.2-gtk-3.0 (>= 3.22.0), shared-mime-info
(>= 0.20), gsettings-desktop-schemas (>= 2.91.92), gir1.2-peas-
1.0
Recommends: librsvg2-common, yelp
Suggests: eog-plugins
Breaks: eog-plugins (<< 3.16.4-2~)
Homepage: https://wiki.gnome.org/Apps/EyeOfGnome
Priority: optional
Section: gnome
Filename: pool/main/e/eog/eog_3.38.2-1_armhf.deb
Size: 3085704
SHA256:
3ec5fd75488b6c4fbca71eafbeec6717f489ab7a06427d649074a5198118cde5
SHA1: 1bf4c465418421855274c415e2a6d608e8a94cec
MD5sum: 79a132ad6b6cd71ddfa60567391f8a42
Description: Eye of GNOME graphics viewer program
 eog or the Eye of GNOME is a simple graphics viewer for the
GNOME
```

```
 desktop which uses the gdk-pixbuf library.  It can deal with
large
 images, and zoom and scroll with constant memory usage.  Its
goals are
 simplicity and standards compliance.
Description-md5: a17b1b698fda7b280b8e85d7b08c5d27
```

17) 패키지 검색

```
sudo apt-cache  search 패키지이름
```

```
ojk@raspberrypi:/var/cache/apt/archives $ sudo apt-cache search
eog
aa3d - ASCII art stereogram generator
adwaita-icon-theme - default icon theme of GNOME
.............................................................
.............
.............................................................
.............

eog - Eye of GNOME graphics viewer program
eog-dev - Development files for the Eye of GNOME
eog-plugin-disable-dark-theme - Disable Dark Theme plugin for
GNOME Image Viewer
eog-plugin-exif-display - Exif Display plugin for GNOME Image
Viewer
eog-plugin-export-to-folder - Export to Folderplugin for GNOME
Image Viewer
eog-plugin-fit-to-width - Fit to Width plugin for GNOME Image
Viewer
eog-plugin-fullscreen-background - Fullscreen Background plugin
for GNOME Image Viewer
eog-plugin-hide-titlebar - Hide Titlebarplugin for GNOME Image
Viewer
eog-plugin-map - Map plugin for GNOME Image Viewer
.............................................................
.............
.............................................................
.............

w3-recs - Recommendations of the World Wide Web Consortium (W3C)
ojk@raspberrypi:/var/cache/apt/archives $
```

18) 설치된 패키지 업그레이드

설치되어 있는 패키지를 모두 새버전으로 업그래이드 합니다.
sudo apt-get upgrade

19) 의존성검사하며 설치하기

sudo apt-get dist-upgrade

20) 패키지 재설치

apt-get --reinstall install 패키지이름

21) 패키지 삭제

설정파일은 지우지 않음
sudo apt-get remove 패키지이름
설정파일까지 모두 지움
sudo apt-get --purge remove 패키지이름

22) 패키지 소스코드 다운로드

sudo apt-get source 패키지이름

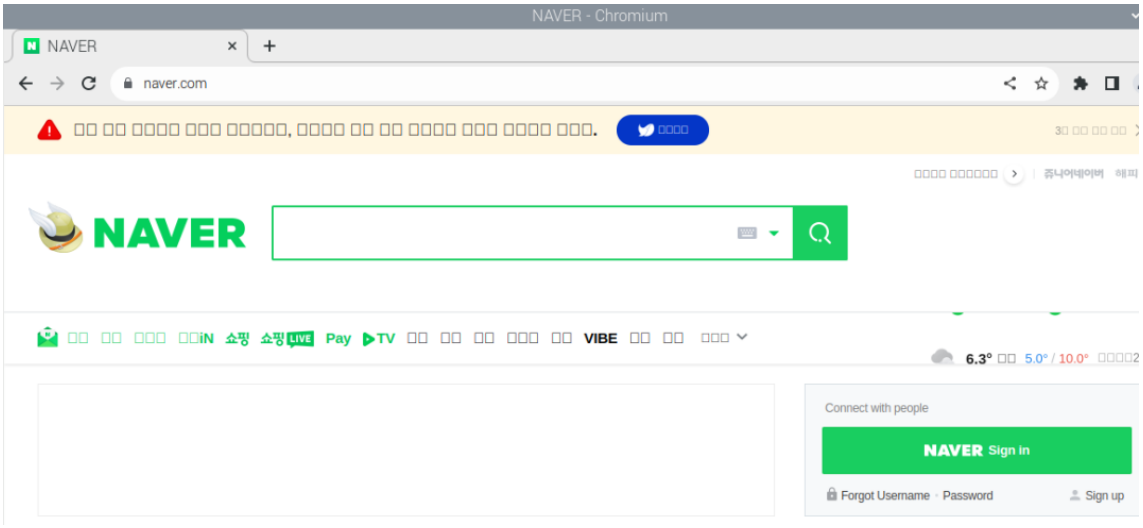
23) 위에서 받은 소스코드를 의존성있게 빌드

sudo apt-get build-dep 패키지이름

【02】한글 설정

a) 시스템 한글 설정

웹프라우저로 네이버에 접속하여 보자.



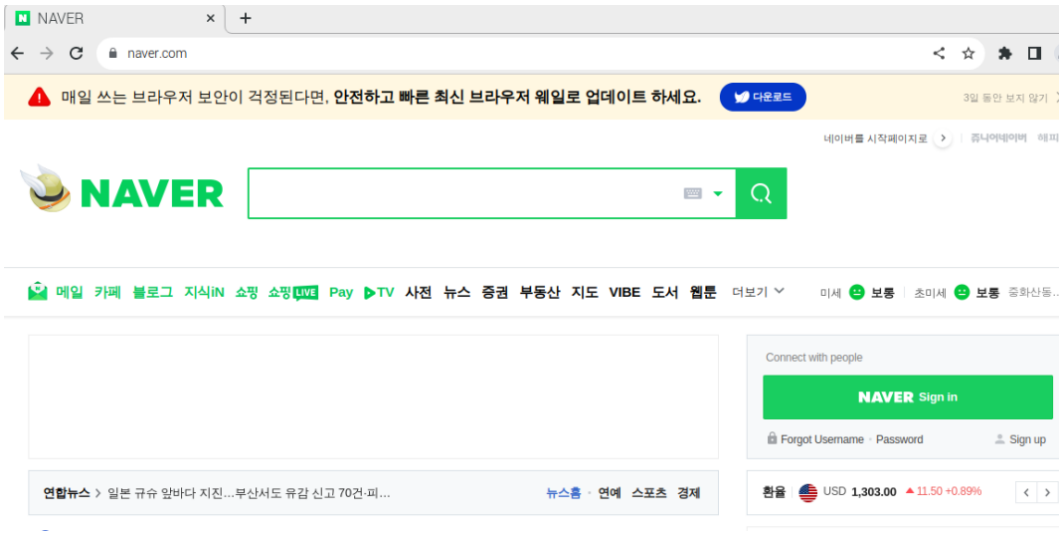
Raspberry pi는 한글 폰트가 없으므로 한글 폰트를 설치하여야 합니다. 한글 폰트를 설치하는 방법은 직접 폰트파일을 /usr/share/fonts나 사용자 홈디렉토리에 복사하는 방법과 명령어를 사용하여 다운 받아 설치하는 방법이 있습니다.

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

```
ojk@raspberrypi:~ $ sudo apt-get install fonts-unfonts-core
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no
longer required:
  libfuse2
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  fonts-unfonts-core
0 upgraded, 1 newly installed, 0 to remove and 141 not upgraded.
Need to get 14.9 MB of archives.
After this operation, 34.3 MB of additional disk space will be
used.
Get:1 http://ftp.kaist.ac.kr/raspbian/raspbian bullseye/main
armhf fonts-unfonts-core all 1:1.0.2-080608-16 [14.9 MB]
Fetched 14.9 MB in 2s (6,230 kB/s)
Selecting previously unselected package fonts-unfonts-core.
(Reading database ... 109253 files and directories currently
installed.)
Preparing to unpack .../fonts-unfonts-core_1%3a1.0.2-080608-
16_all.deb ...
Unpacking fonts-unfonts-core (1:1.0.2-080608-16) ...
Setting up fonts-unfonts-core (1:1.0.2-080608-16) ...
Processing triggers for fontconfig (2.13.1-4.2) ...
ojk@raspberrypi:~ $
```

```
$ sudo apt-get install ttf-unfonts-core
```

또는



/usr/share/fonts에 복사하는 방법

root@raspberrypi:~# cd /usr/share/fonts

root@raspberrypi:/usr/share/fonts# mkdir user

root@raspberrypi:/usr/share/fonts# cd user

root@raspberrypi:/usr/share/fonts/user# cp /home/pi/temp/* .

root@raspberrypi:/usr/share/fonts/user# ls -l

합계 27564

-rw-r--r-- 1 root root 748216 4월 29 09:35 HYCYSM.TTF

-rw-r--r-- 1 root root 1353496 4월 29 09:35 HYDNKB.TTF

-rw-r--r-- 1 root root 3485364 4월 29 09:35 HY견고딕.TTF

-rw-r--r-- 1 root root 6367340 4월 29 09:35 HY견명조.TTF

-rw-r--r-- 1 root root 16264732 4월 29 09:35 batang.ttc

root@raspberrypi:/usr/share/fonts/user#

가장 먼저 사용자 홈디렉토리 (/home/pi)에 .fonts 파일을 만들고 한글 폰트를 복사한다.

pi@raspberrypi:~ $ mkdir .fonts

pi@raspberrypi:~ $ cd .fonts

pi@raspberrypi:~/.fonts $ ls -l

합계 87412

-rw-r--r-- 1 pi pi 73312 7월 30 13:00 ARESSENCE.ttf

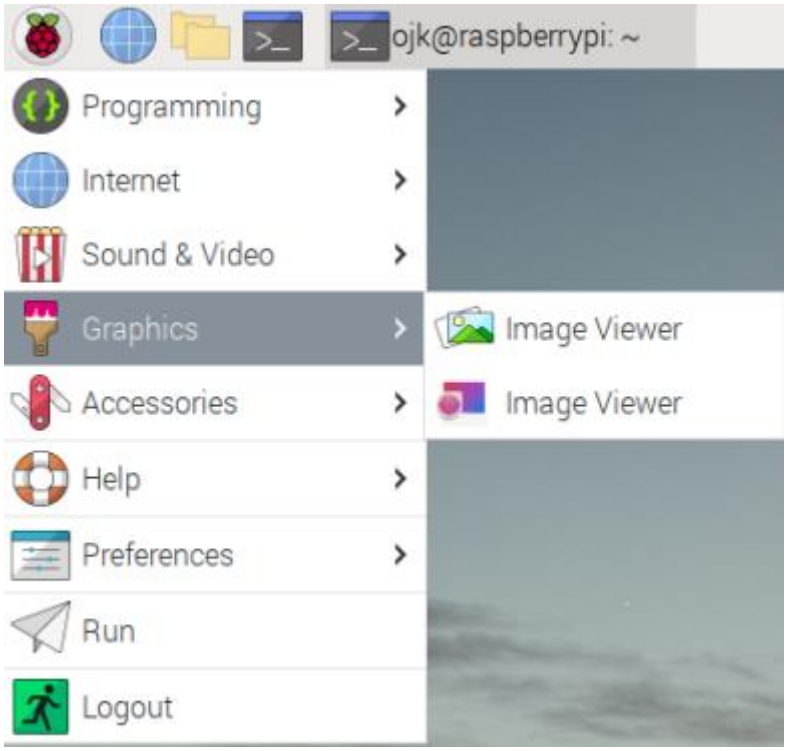
-rw-r--r-- 1 pi pi 2971120 7월 30 13:00 HANYGO240.ttf

-rw-r--r-- 1 pi pi 3485364 7월 30 13:00 HY견고딕.TTF

-rw-r--r-- 1 pi pi 6367340 7월 30 13:00 HY견명조.TTF

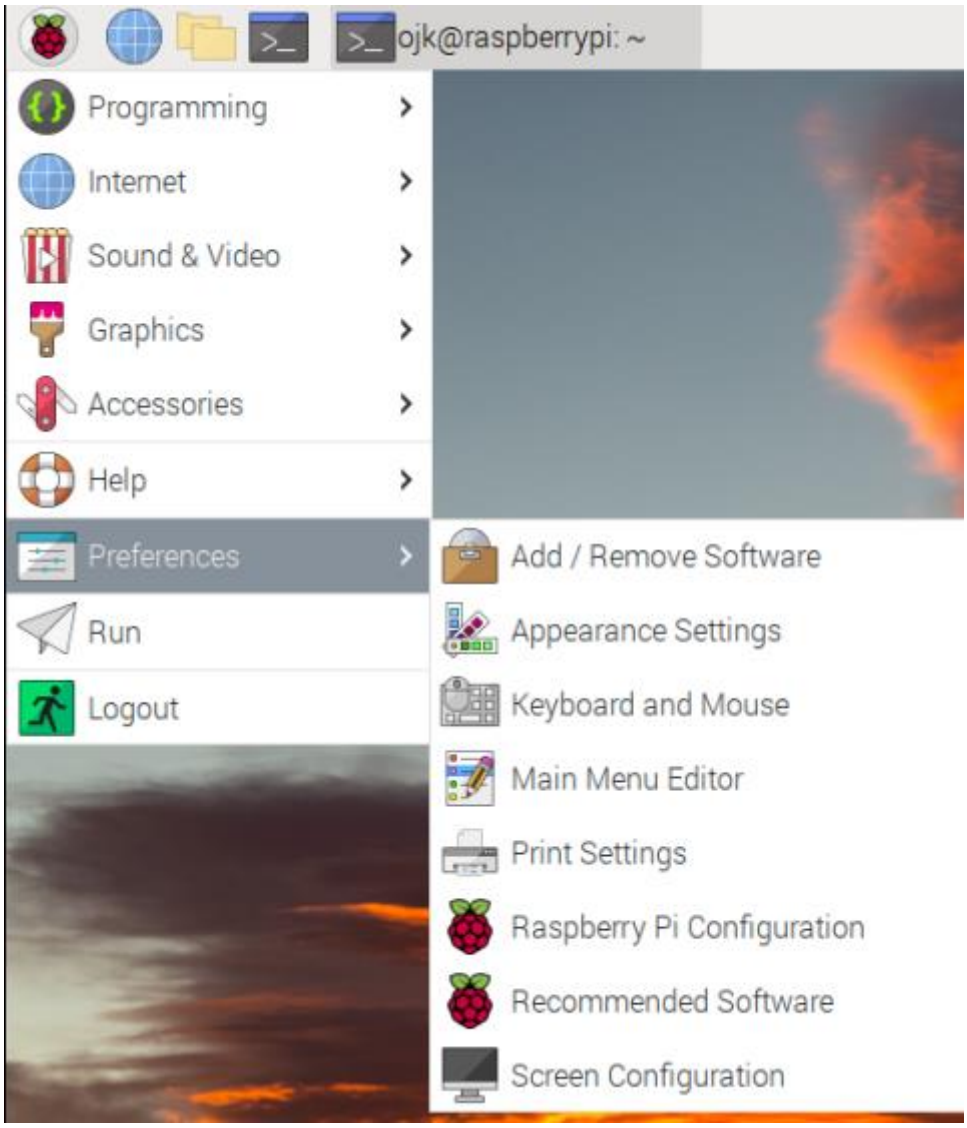
-rw-r--r-- 1 pi pi 1353496 7월 30 13:00 HY동녘B.TTF

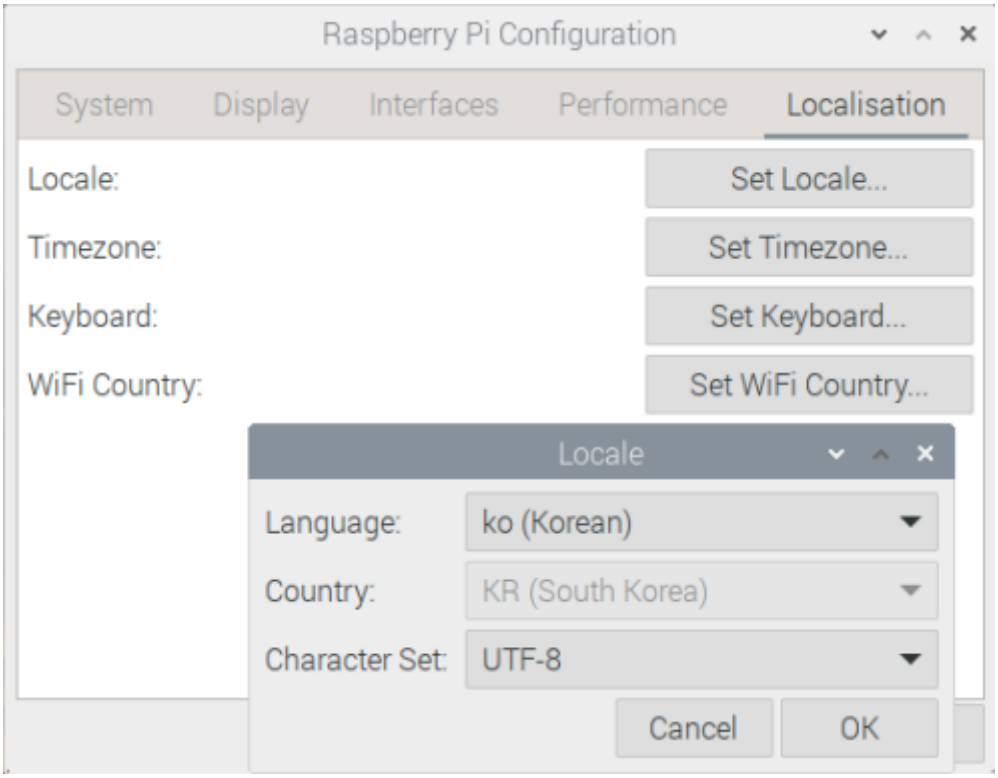
-rw-r--r-- 1 pi pi 415128 7월 30 13:00 HY수평선B.TTF



[Menu-기본설정-Raspberry Pi Configuration] Localisation Tab 에서
Locale :
Timezone:
Keyborad:
WiF Counter :

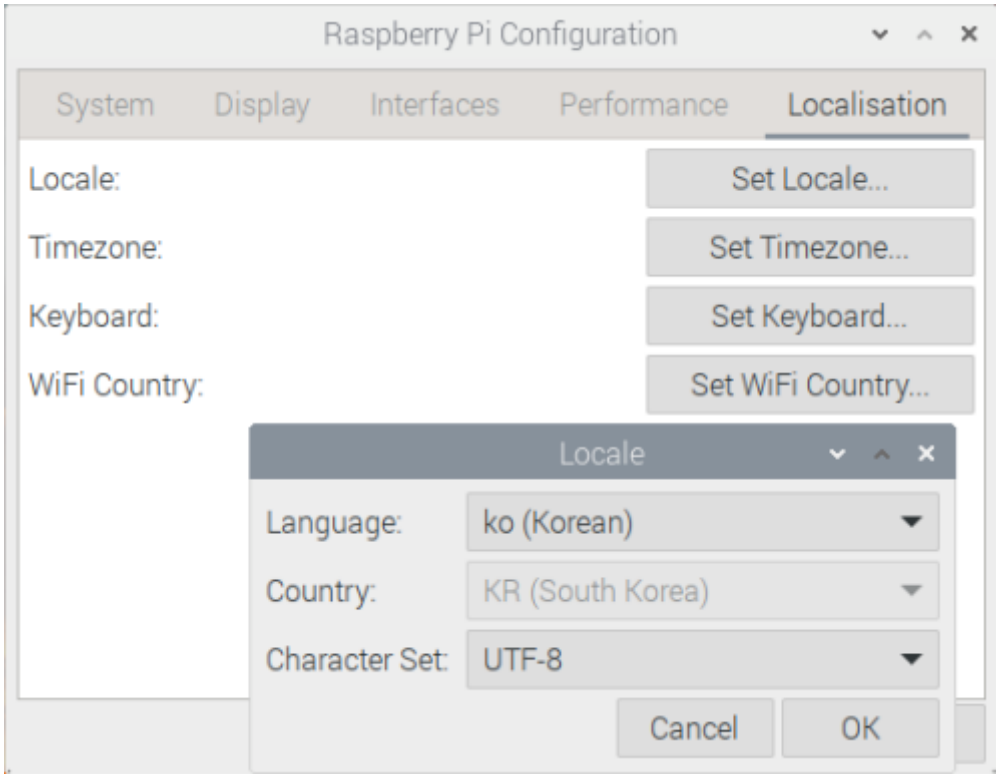
설정한다.

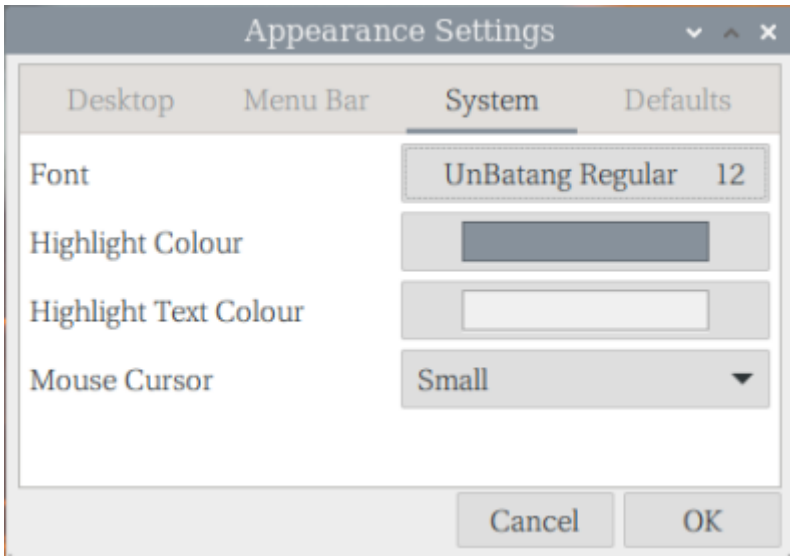




나머지 Timezone, Keyboard 등을 한국으로 설정합니다.

[Menu-기본설정-Appearance Setting] 에서 System Font 를 한글 폰트로 지정한다.
재부팅하면 한글 X-Windows를 사용할 수 있습니다.

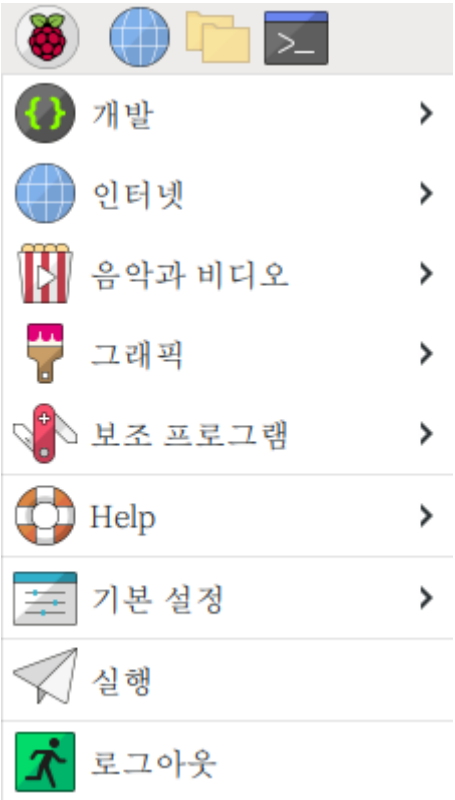




Font를 한글 폰트로 지정합니다.

재부팅합니다.

시스템 메뉴가 한글로 설정이 되어 있는 것을 확인할 수 있습니다.



b) 한글 입력기 설정

시스템에 한글을 설치하는 것과 한글을 입력하는 것은 다름니다. 따로 한글 입력기를 설치하여야 합니다.

한글 입력기로 ibus 사용

$ sudo apt-get install ibus

$ sudo apt-get install ibus-hangul

따라서 X-Window를 한글로 사용하고 한글 입력기로 ibus를 사용한다면 .bashrc 파일에

```
export LC_ALL
export LANG=ko_KR.UTF-8
export GTK_IM_MODULE=ibus
export XMODIFIERS=@im=ibus
```

```
export QT_IM_MODULE=ibus
```

를 지정하면 됩니다. .bashrc 파일은 숨김 속성 파일이라서 보이지 않습니다.

[파일관리자]-[메뉴]-[보기]-[숨김파일 표시]를 클릭하면 파일관리자에서 보인다.

재 부팅하면 한글 입력이 가능합니다.

시스템 설정
한국으로 설정

3. 화면설정

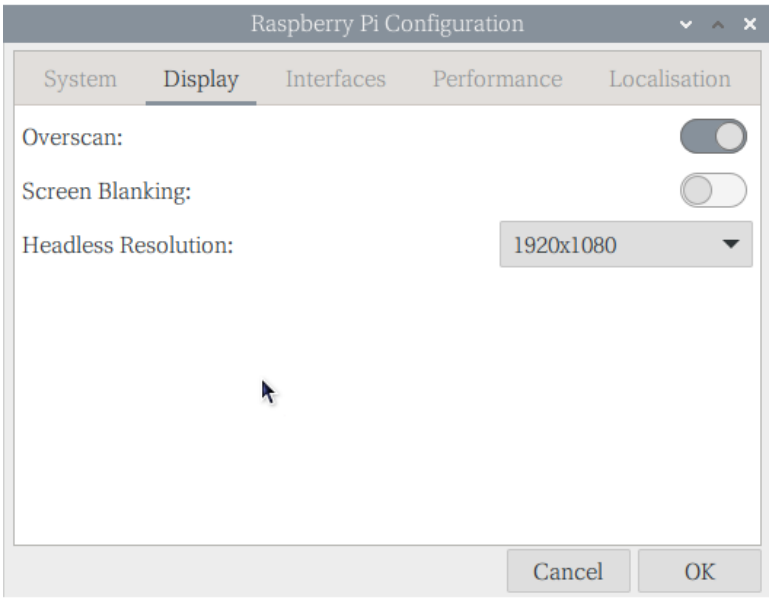
오버스캔
기본값은 16으로 설정되어 있다.
overscan

라즈베리파이는 기본적으로 스크린 세이버가 설치되어 있고, 일정 시간이 지나면 화면이 꺼짐니다. 이 후에는 키보드나 마우스를 움직여도 화면이 다시 나오지 않으므로 매우 불편합니다. 따라서 스크린 세이버를 사용할 수 없게 하거나 스크린 세이버 그림을 넣어서 동작하게 하여야 합니다.

시스템 파일을 직접 설정하여 스크린 세이버를 사용 안 함으로 설정하려면

/boot/cmdline.txt 파일의 끝에
consoleblank=0을 삽입합니다.

화면 꺼짐 기능 해제



스크린 세이버를 사용하려면
xscreensaver 을 설치합니다.

apt-get install xscreensaver

설치 후에 메뉴-preference-screensaver 메뉴에서 설정하면 됩니다.

hdmi_enable_4kp60=1

4. 응용 프로그램 사용

【01】그림보기

eog 설치
apt-get install eog

플러그인 설치
apt-get install eog-plugins

【02】화면 캡처

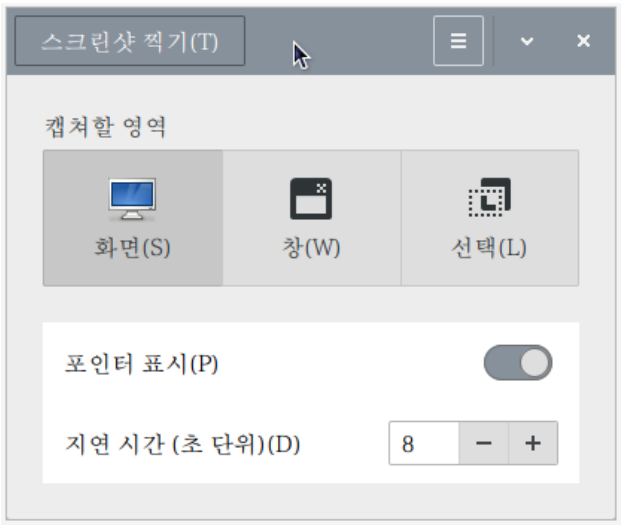
화면을 캡쳐하기 위해서는 gnome-screenshot을 설치합니다.

```
sudo apt install gnome-screenshot
```

```
sudo apt install gnome-screenshot
ojk@raspberrypi:~ $ sudo apt install gnome-screenshot
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다... 완료
상태 정보를 읽는 중입니다... 완료
다음 패키지가 자동으로 설치되었지만 더 이상 필요하지 않습니다:
  libfuse2
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  libhandy-1-0
다음 새 패키지를 설치할 것입니다:
  gnome-screenshot libhandy-1-0
0 개 업그레이드, 2 개 새로 설치, 0 개 제거 및 152 개 업그레이드 안 함.
326 k 바이트 아카이브를 받아야 합니다.
이 작업 후 1,793 k 바이트의 디스크 공간을 더 사용하게 됩니다.
계속 하시겠습니까? [Y/n] y
받기:1 http://ftp.kaist.ac.kr/raspbian/raspbian bullseye/main
armhf libhandy-1-0 armhf 1.0.3-2 [153 kB]
받기:2 http://ftp.kaist.ac.kr/raspbian/raspbian bullseye/main
armhf gnome-screenshot armhf 3.38.0-1 [173 kB]
```

```
내려받기 326 k 바이트, 소요시간 6 초 (50.2 k 바이트/초)
Selecting previously unselected package libhandy-1-0:armhf.
(데이터베이스 읽는중 ...현재 112085 개의 파일과 디렉터리가 설치되어
있습니다.)
Preparing to unpack .../libhandy-1-0_1.0.3-2_armhf.deb ...
Unpacking libhandy-1-0:armhf (1.0.3-2) ...
Selecting previously unselected package gnome-screenshot.
Preparing to unpack .../gnome-screenshot_3.38.0-1_armhf.deb ...
Unpacking gnome-screenshot (3.38.0-1) ...
libhandy-1-0:armhf (1.0.3-2) 설정하는 중입니다 ...
gnome-screenshot (3.38.0-1) 설정하는 중입니다 ...
Processing triggers for libglib2.0-0:armhf (2.66.8-1) ...
Processing triggers for libc-bin (2.31-13+rpt2+rpi1+deb11u4) ...
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for mailcap (3.69) ...
Processing triggers for desktop-file-utils (0.26-1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for gnome-menus (3.36.0-1) ...
ojk@raspberrypi:~ $
```

메뉴/보조프로그램/스크린샷 프로그램을 실행시킵니다.



5. 주변 기기 사용

【03】 카메라

1.1. raspistill

라즈베리파이에는 카메라 모듈을 사용할 수 있도록 기본적으로 4개의 프로그램이 내장되어 있습니다. raspistill, raspivid, raspiyuv 그리고 raspividyuv 입니다. raspistill과 raspiyuv는 매우유사하며 이미지를 캡처하기 위한 것이고 raspivid 및 raspvidyuv는 비디오 캡처용입니다.

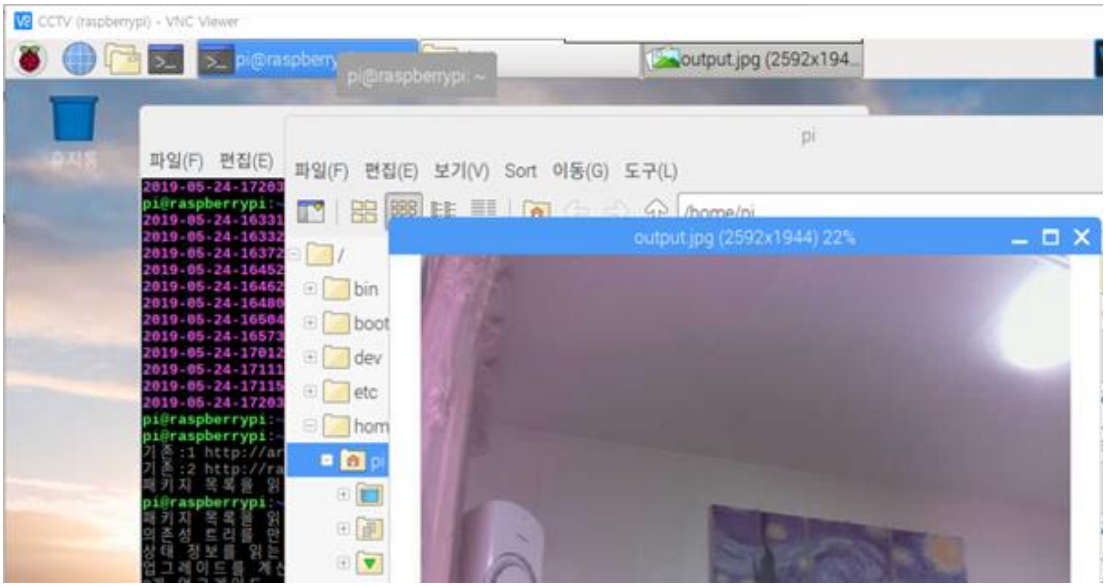
인터페이스 설정에서 Camera를 Enable 해 두었기 때문에 자동으로 테스트 환경이 설정되어 있다.

모든 카메라 응용 프로그램은 터미널 명령으로 구동되며 OpenMAX를 통해 실행되는 MMAL API를 활용하도록 작성되어 있습니다. MMAL API는 OpenMAX가 제공하는 것보다 사용하기 쉬운 시스템을 제공한다.

라즈베리파이의 터미널로 들어가서 다음 명령을 입력합니다.

$ raspistill -o output.jpg

파일 매니저로 /home/pi 폴더 안에 가면 output.jpg 파일이 만들어져 있다. 이 파일을 더블클릭 하면 찍힌 사진을 확인할 수 있다.



raspistill -br 50 -o image.jpg 실행할 때 중간에 명령어를 입력하면 촬영 조건을 넣어주거나 사진에 효과를 넣을 수 있다.

주로 사용하는 명령어 옵션은 다음과 같다.

-o 파일명 -> 파일명으로 저장

-t 시간 -> 시간 후에 촬영(단위 ms, 기본 5초)

-w. -h -> 사진 크기 세팅(w 넓이, h 높이)

-q -> 사진 파일 품질 조정(0~100)

-d -> demo 모드

-k -> keypress 모드(엔터 누르면 촬영)

-s -> signal 모드(신호가 들어오면 촬영)

-br -> 밝기 조정(0~100)

1.2. raspivid

동영상을 촬영하기 위해서는 raspivid를 사용합니다. raspivid는 기본적으로 설치되어 있으므로 곧바로 사용할 수 있습니다.

$ raspivid -o video.h264

주로 사용하는 raspivid 명령어 옵션은 다음과 같다.
-?, --help : This help information
-w, --width : Set image width <size>. Default 1920
-h, --height : Set image height <size>. Default 1080
-b, --bitrate : Set bitrate. Use bits per second (e.g. 10MBits/s would be -b 10000000)
-o, --output : Output filename <filename> (to write to stdout, use '-o -')
-v, --verbose : Output verbose information during run
-t, --timeout : Time (in ms) before takes picture and shuts down. If not specified, set to 5s
-d, --demo : Run a demo mode (cycle through range of camera options, no capture)
-fps, --framerate : Specify the frames per second to record
-e, --penc : Display preview image *after* encoding (shows compression artifacts)

촬영할 비디오의 시간을 지정하려면, -t 플래그를 밀리초 숫자와 함께 지정합니다. 예를 들어

raspivid -o video.h264 -t 10000

위의 명령은 10초 길이의 비디오를 녹화한다.
t를 0으로 지정하면 시간 간격 없이 촬영합니다.
출력을 파일로 지정하여도 화면에 영상은 나옵니다.

raspivid -o - -t 10000

Pi는 비디오를 raw H264 비디오 스트림으로 캡처합니다. 많은 미디어 플레이어는 raw H264 비디오 스트림은 플레이하지 못하므로 mp4로 변환하여야 합니다. 그래서 raw H264 비디오 스트림으로 캡처한 다음에 mp4로 변환하면 됩니다. 리눅스에서 많이 사용하는 프로그램은 MP4Box이므로 MP4Box를 사용하여 mp4로 변환하는 것이 일반적인 방식입니다.

다음 명령으로 MP4Box를 설치합니다.
sudo apt-get install -y gpac

설치가 완료되면 다음 명령으로 테스트 해보자.

raspivid -t 30000 -w 640 -h 480 -fps 25 -b 1200000 -p 0,0,640,480 -o video.h264
MP4Box -add pivideo.h264 video.mp4

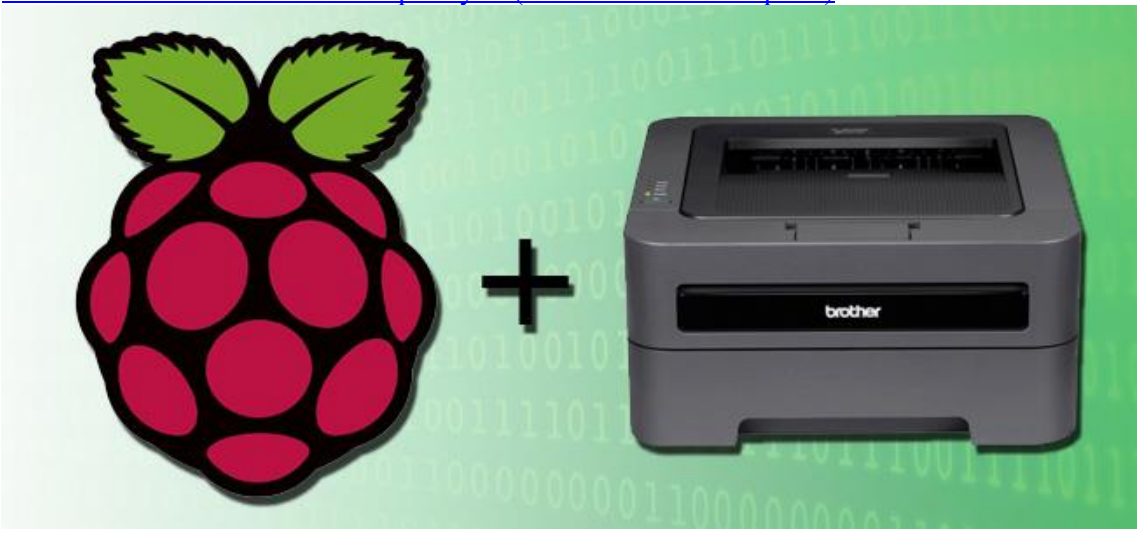
30초 동안 가로 640 세로 480 크기 25fps 설정으로 촬영하여 video.h264로 저장한 다음 video.mp4로 변환하는 명령어 입니다. 물론 파일의 저장 위치는 위 명령을 실행한 디렉토리입니다.

파일 매니저로 /home/pi 폴더 안에 가면 video.mp4 파일이 만들어져 있다. 이 파일을 더블클릭 하면 찍힌 동영상을 확인할 수 있다.

【04】printer

01) 프린터 설치

How to Add a Printer to Your Raspberry Pi (or Other Linux Computer)



Unlike a typical Windows machine, the little Raspberry Pi running Rasbian doesn't exactly come with plug-'n-play printer support. Read on as we show you how to add full-fledged print capabilities to your Pi unit.

Why Do I Want to Do This?
If you're experimenting with the Pi as a desktop replacement, want to enable a physical print output for a program or application suite you're using, or otherwise want to enable traditional printing on your Pi, this tutorial is a straight shot from printerless to happy printing; there's no previous printer-wrangling under Linux experience required.

What Do I Need?
For this tutorial you'll need the following things:
1 Raspberry Pi unit with Rasbian installed
1 USB-based or network-accessible printer
If you haven't yet configured your Raspberry Pi with a Rasbian image, we highly suggest starting with our introduction to the Raspberry Pi.

02) Installing CUPS on the Pi and Enabling Remote Access

In order to link a printer with the Raspberry Pi, we first need to install Common Unix Printing System (CUPS). It's time to fire up your Pi and navigate to the terminal (either on the Pi itself of via SSH).
At the terminal, enter the following command to begin installing CUPS:

```
sudo apt-get install cups
```

When prompted to continue, type Y and press enter. CUPS is a fairly beefy install, so feel free to go grab a cup of coffee. Once the base installation is complete, we need to make a few small administrative changes. The first order of business is to add ourselves to the usergroup that has access to the printers/printer queue. The usergroup created by CUPS is "lpadmin". The default Rasbian user (and the user we're logged into) is "pi" (adjust the following command accordingly if you want a different user to have access to the printer).

At the terminal enter the following command:

```
sudo usermod -a -G lpadmin pi
```

For the curious, the "-a" switch allows us to add an existing user (pi) to an existing group (lpadmin), specified by the "-G" switch.
Our final bit of pre-configuration work is to enable remote editing of the CUPS configuration. The rest of the configuration can be completed via the web browser on the Pi, but if you're not actually sitting right at the Pi and would prefer to use, say, the browser on your Windows desktop to complete the configuration, you'll need to toggle a small value in `/etc/cups/cupsd.conf`. At the terminal, enter the following command:

```
sudo nano /etc/cups/cupsd.conf
```

Inside the file, look for this section:

```
# Only listen for connections from the local machine
Listen localhost:631
```

Comment out the "Listen localhost:631" line and replace it with the following:

```
# Only listen for connections from the local machine
# Listen localhost:631
Port 631
```

This instructs CUPS to listen for any contact on any networking interface as long as it is directed at port 631.
Scroll further down in the config file until you see the "location" sections. In the block below, we've bolded the lines you need to add to the config:

< Location / >
Restrict access to the server...
Order allow,deny
Allow @local
< /Location >

< Location /admin >
Restrict access to the admin pages...
Order allow,deny
Allow @local
< /Location >

< Location /admin/conf >
AuthType Default
Require user @SYSTEM

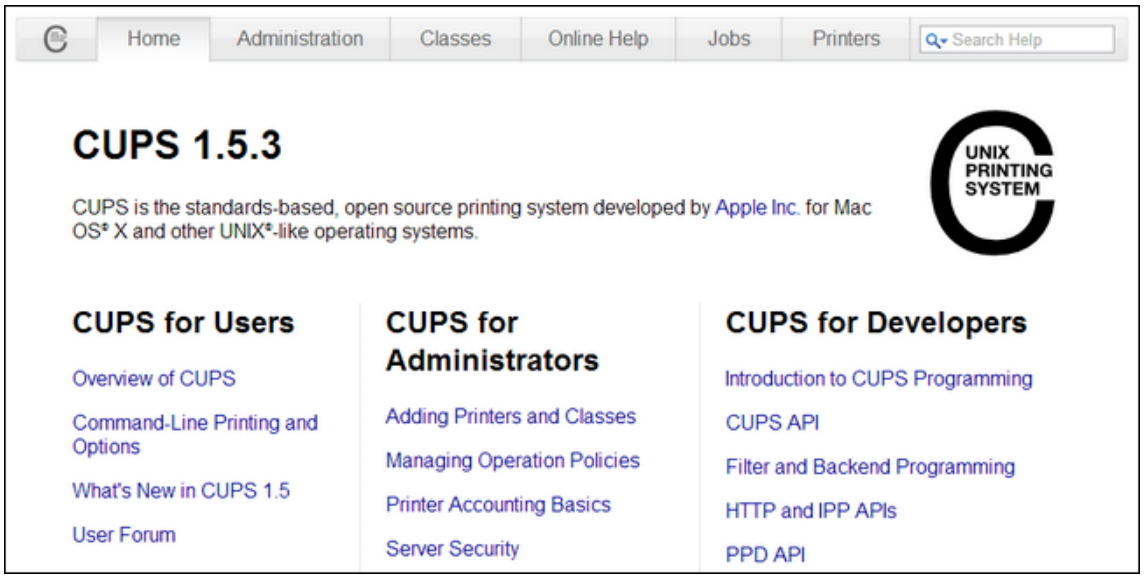
Restrict access to the configuration files...
Order allow,deny
Allow @local
< /Location >

The addition of the "allow @local" line allows access to CUPS from any computer on your local network. Anytime you make changes to the CUPS configuration file, you'll need to restart the CUPS server. Do so with the following command:

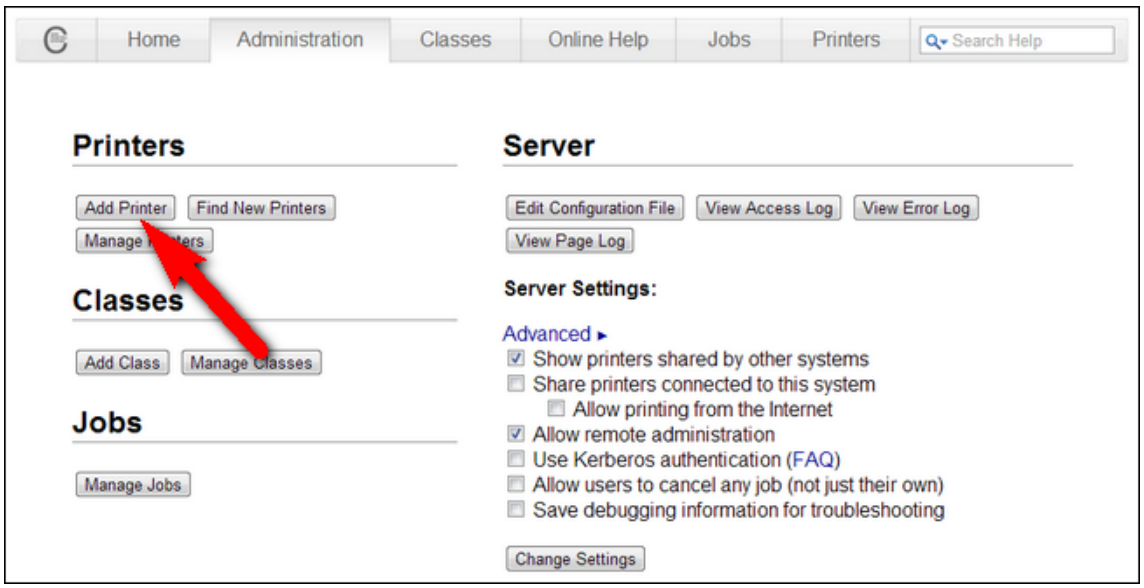
```
sudo /etc/init.d/cups restart
```

After restarting CUPS, you should be able to access the administration panel via any computer on your local network by pointing its web browser at http://[the Pi's IP or hostname]:631.

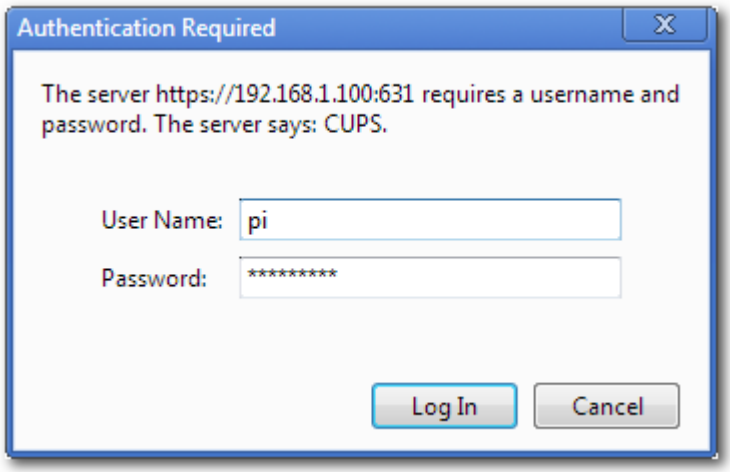
Adding a Printer to CUPS



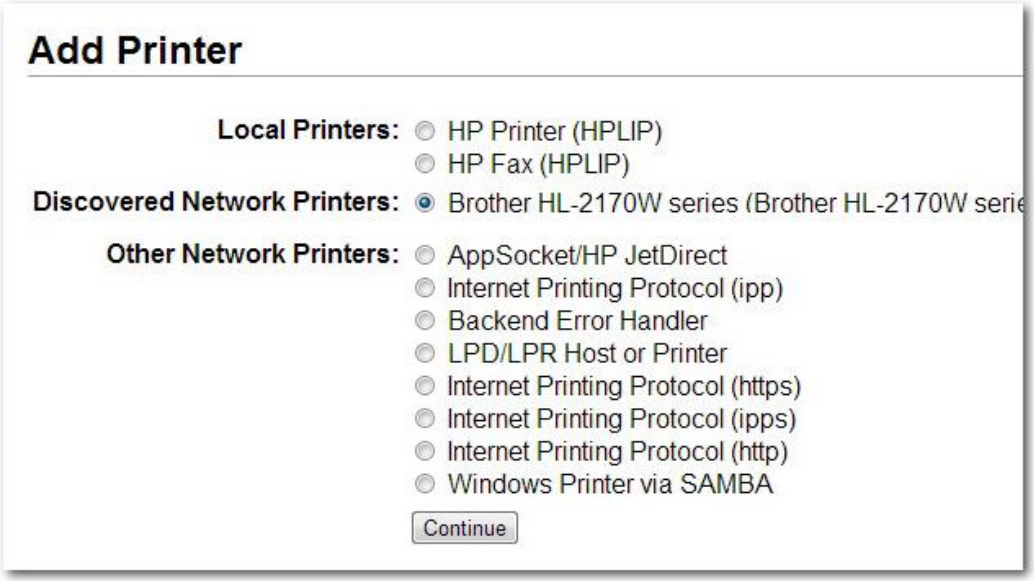
When you first navigate to http://[the Pi's IP or hostname]:631, you'll see the default CUPS homepage, as seen in the screenshot above. The section we're interested in is the "Administration" tab. Click on it now.



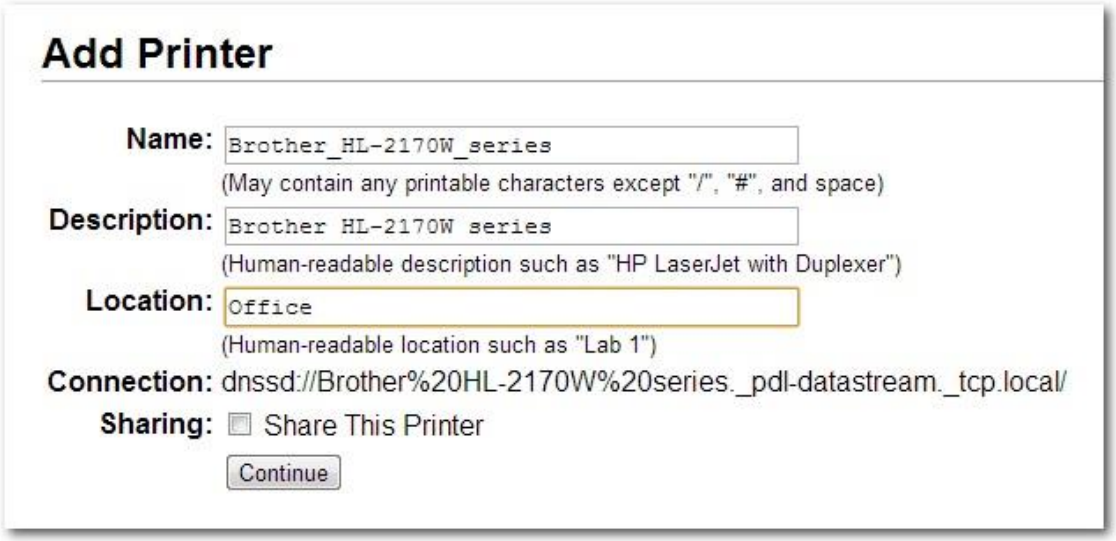
Within the Administration panel, click add printer. If you receive a warning about the site's security certificate, go ahead and click proceed anyway to ignore it. You'll be prompted to enter a username and password.



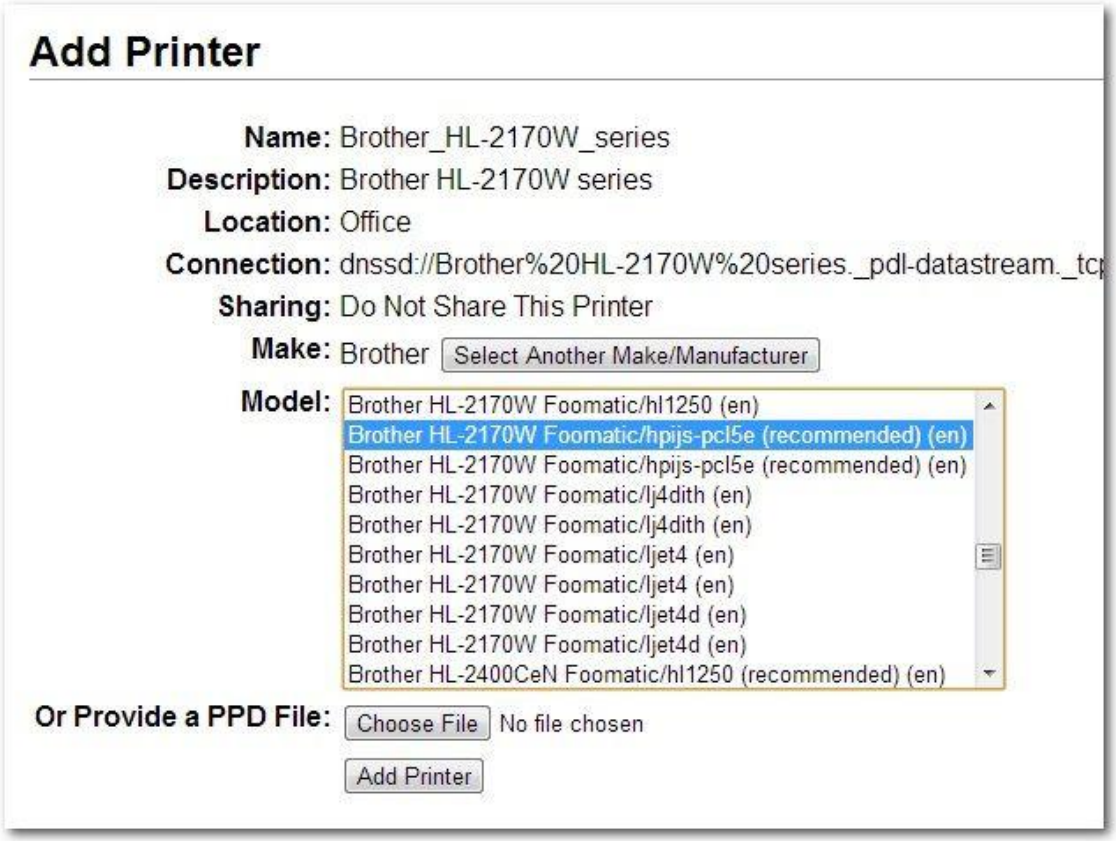
Go ahead and enter the username and password of the account you added to the "lpadmin" group earlier in the tutorial (e.g. if you're using a default Raspbian install, the login/password is "pi"/"raspberry"). Click "Log In".
After logging in, you'll be presented with a list of discovered printers (both local and networked). Select the printer you wish to add to the system:



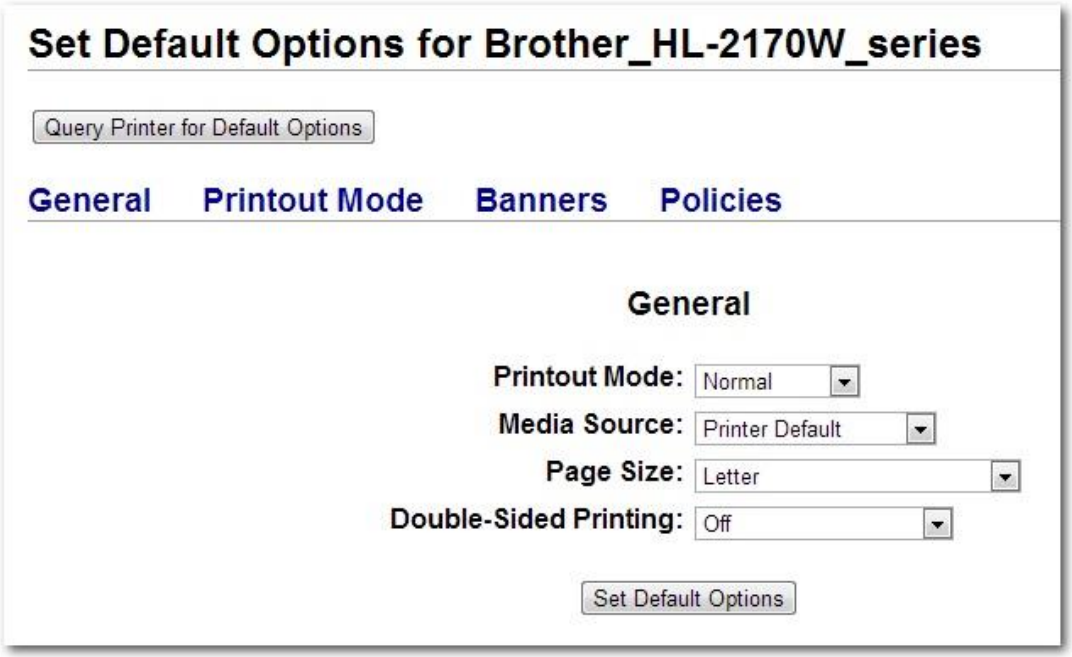
After selecting the printer, you'll be offered an opportunity to edit the name, description, and location of the printer, as well as enable network sharing. Since our printer is already a network printer, we left "Share This Printer" unchecked:



After editing the printer name and adding a location, you'll be prompted to select the specific driver you want to use for your printer. Despite the fact that it automatically discovered the printer and the printer name, CUPS makes no attempt to pick the right driver for you. Scroll until you see a model number that matches yours. Alternatively, if you have a PPD file for the printer that you have downloaded from the manufacturer, you can load that with the "Choose File" button:

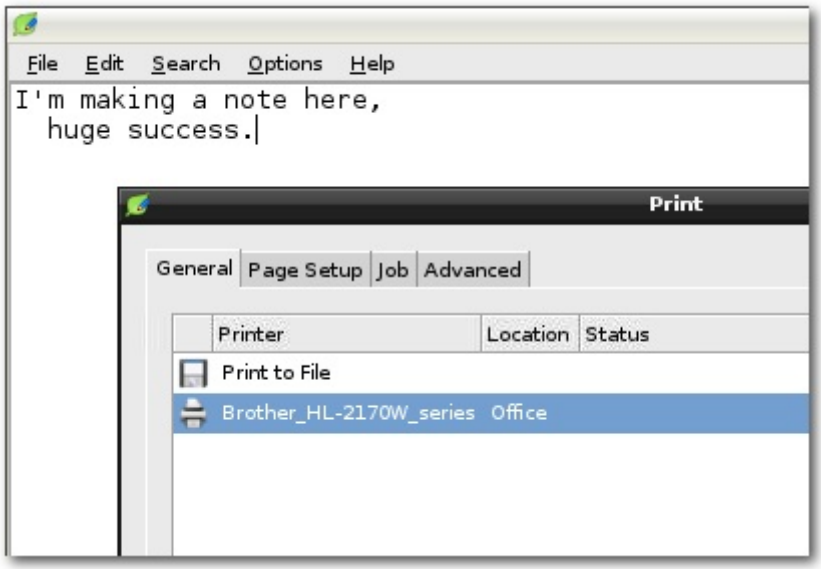


The last configuration step is to look over some general print settings like what you want the default printer mode to be, the default paper source/size, etc. It should default to the correct presets, but it never hurts to check:



After you click "Set Default Options", you'll be presented with the default administration page for the printer you just added to the CUPS system:

Everything looks good. The real test, however, is actually printing something. Let's fire up Leafpad, Rasbian's default text editor, and send a message:



While we realize it's a bit premature to write "huge success" on our test print before, you know, it's actually printed, we were that confident. Fifteen seconds or so later, the document came rolling out of the printer and dropped into the tray. Success!

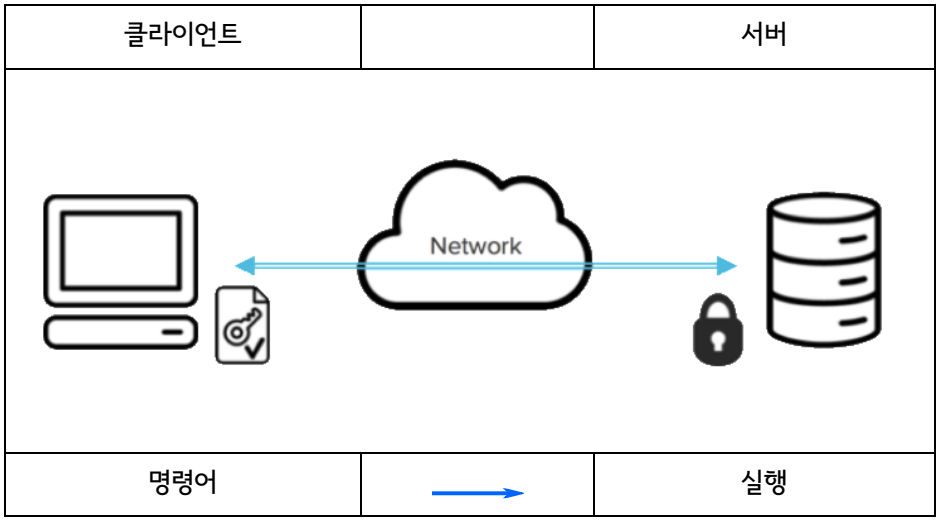
At this point, if you've added the only user that needs access to the printer to the "lpadmin" group and you've added the only printer you want to access to the CUPS system, you're done. If you have any other users you would like to add or additional printers, simply run through the respective steps in the tutorial again to do so.

III. 서버

1. SSH 서버

【01】ssh 서버란?

서버는 PC 와는 다르게 여러 사용자가 사용하는 컴퓨터입니다. 그래서 원격지에서 서버에 접속하여 명령을 실행하여야 하는 경우가 많습니다. 다름 그림처럼 내 컴퓨터(클라이언트)에서 원격지 컴퓨터(서버)에 명령을 내려야 하는 경우가 있습니다. 당연히 서버는 이런 역할을 하는 통신프로토콜과 그 프로그램을 제공합니다.



이 때 사용하는 프로그램이 **telnet** 프로그램입니다.
그런데 여기에는 약간의 문제점이 있습니다.

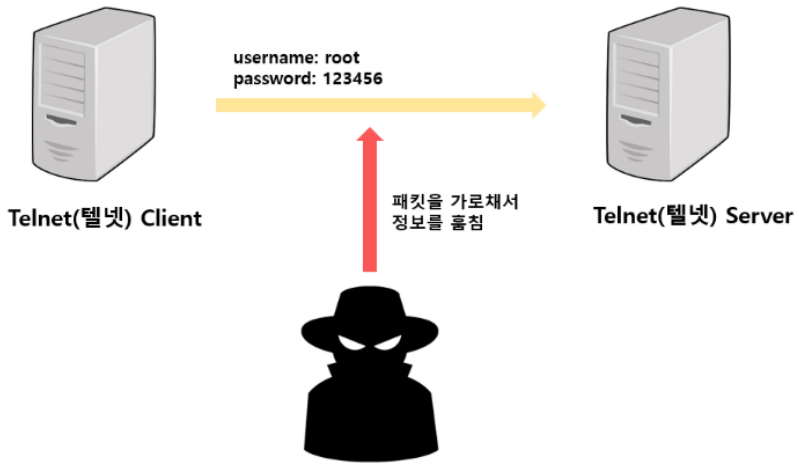


그림 1 그림제공: https://dololak.tistory.com/

보안에 취약점이 있습니다. 이를 해결하기 위해서는

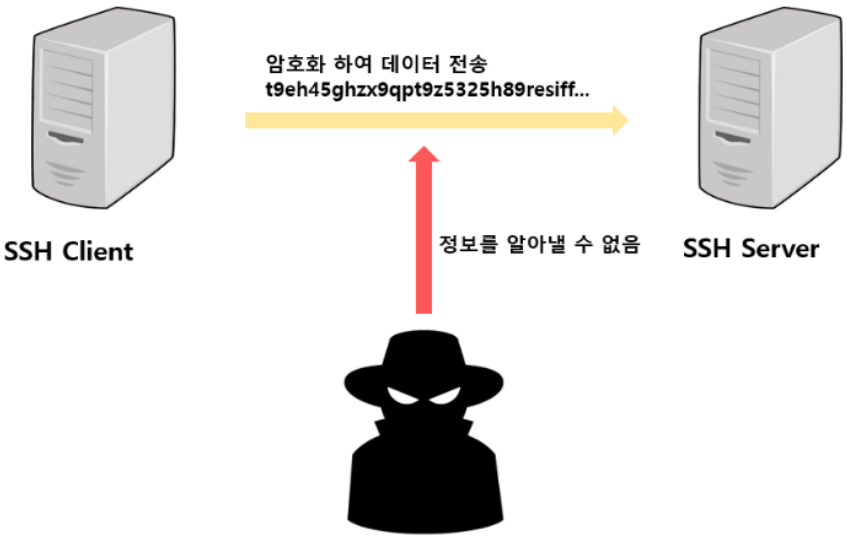


그림 2 그림제공 https://dololak.tistory.com/

SSH 는 Secure SHell 의 줄임말로 보안이 강화된 서버 원격 접속 및 제어 도구입니다.
ssh 란 클라이언트에서 원격지 서버에 접속하여 명령어를 실행할 수 있는 프로코콜을 말합니다.
명령어를 입력하는 컴퓨터가 ssh 클라이언트이고 실행을 하는 컴퓨터가 ssh 서버입니다.
이를 사용하기 위해서는 클라이언트에서는 ssh 클라이언트 프로그램을 설치하여야 하고, 서버에서는 ssh 서버 프로그램을 설치하여야 합니다. ssh 클라이언트와 서버는 시작에서부터 끝날 때까지 안전하게 암호화된 세션을 서로 제공합니다. ssh 는 22 번 포트를 사용합니다.

【02】동작 원리

기존 Telnet, Rlogin, RSH 를 대체하기 위해 만들어졌으며 네트워크 상의 다른 PC 나 서버에 로그인, 원격 명령 실행, 파일 전송을 수행할 수 있는 프로토콜이다.
기존의 원격 접속 프로토콜인 Telnet, Rlogin, RSH 는 데이터 전송 시 평문으로 전송되기 때문에 스니핑을 통해 데이터가 노출되기 쉬운 문제점을 갖고 있었다.
SSH 프로토콜은 안전한 원격 접속과 보호되지 않은 네트워크에서 안전한 네트워크 서비스를 제공하기 위해 암호화를 사용한다.
암호화를 통해 호스트(클라이언트)와 원격지(서버) 간의 연결(사용자 인증, 명령, 파일 전송)을 네트워크 공격으로부터 보호할 수 있다.

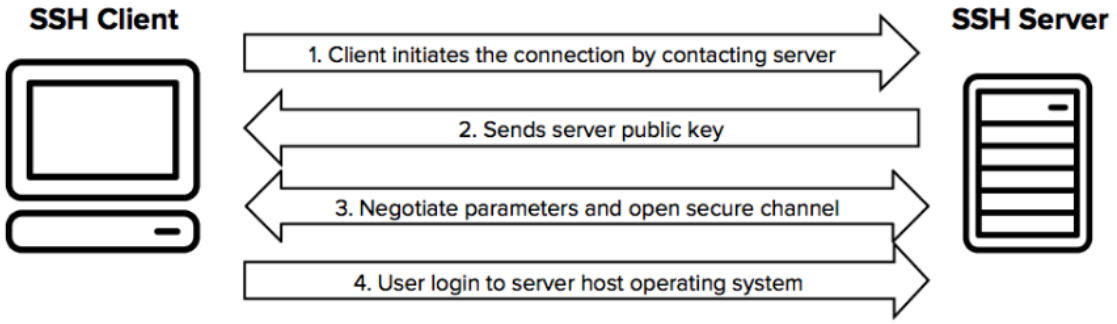
SSH 프로토콜의 일반적인 사용은 다음과 같다.

원격 접속 사용자 및 자동화 프로세스의 접근 시

자동화된 파일 전송 시
원격 명령 실행 시
네트워크 인프라와 중요 시스템 관리 시
이러한 상황에서 안전한 통신을 위해 SSH 프로토콜을 사용한다.

SSH 프로토콜은 클라이언트-서버 모델로 동작하며 대칭키 방식, 비대칭키 방식, 해시 알고리즘을 사용하여 인증 및 암호화를 수행한다.

대칭키 방식은 클라이언트-서버 간 전체 연결을 암호화에 사용되며,
비대칭키 방식은 키 교환, 클라이언트 인증, 서버 인증에 사용되고,
해시 알고리즘은 패킷의 무결성을 확인하기 위해 사용된다. (HMAC : Hash based Message Authenticated Codes)



[출처: ssh.com]

1. 클라이언트는 서버에 원격 접속하기 위해 연결을 설정하는 프로세스를 시작한다.

SSH 프로토콜은 기본적(Default)으로 TCP 22 번 포트를 사용하여 통신한다.
클라이언트가 서버에 원격 접속하기 위해 서버의 TCP 22 번 포트로 SSH 접속 요청을 보내는 것이 SSH 연결의 첫 단계이다.

서버는 클라이언트에게 서버가 지원하는 프로토콜의 버전을 응답으로 보내준다.
클라이언트는 서버가 지원하는 프로토콜의 버전 중 자신과 일치하는 것이 있다면 연결을 지속한다. (버전 교환)

2. 서버는 자신의 공개키를 클라이언트에게 전송한다.
서버는 클라이언트로부터 SSH 접속 요청을 받고 자신의 공개키를 클라이언트에게 전송하고 클라이언트는 서버로부터 받은 공개키를 로컬에 저장한다.

클라이언트는 원격 접속하는 서버들의 공개키를 로컬 사용자 홈 디렉터리의 .ssh 경로 내의 known_hosts 파일에 저장하고 있다.

※ 클라이언트 사용자의 known_hosts 경로(Default)

<Linux>
일반계정 : /home/USER/.ssh/known_hosts
root 계정 : /root/.ssh/known_hosts
<Window>
사용자 : %USERNAME%\.ssh\known_hosts

3. 클라이언트와 서버는 여러 Parameter 들을 주고 받으며 보안 채널을 확립한다.
3.1 올바른 서버인지 확인(클라이언트 관점)
클라이언트는 SSH 로 원격 접속하려고 하는 서버가 올바른 서버인지 확인할 필요가 있다.
이를 위해 클라이언트는 known_hosts 파일에 존재하는 서버의 공개키를 통해 정상적인 서버인지 확인하는 작업을 수행한다.

확인 단계는 다음과 같다.

클라이언트에서 난수 생성, 난수 해시값 생성 및 저장
난수를 서버의 공개키로 암호화 후 서버에 전송
서버에서 서버의 개인키로 데이터를 복호화하여 난수 추출
서버에서 복호화된 난수 해시값을 생성 후 클라이언트에게 전송
클라이언트에 저장된 난수 해시값과 서버에서 받은 난수 해시값을 비교
동일할 시 올바른 서버 확인

3.2 암호화된 통신을 위한 세션키 생성(대칭키 생성)

세션키는 대칭키로 전체 세션을 암호화하는데 사용되며 모든 통신을 암호화 한다.

대칭키는 비대칭키에 비해 빠르고 컴퓨팅 파워가 더 적게 든다는 장점을 가지고 있다.

그러나 대칭키가 유출되었을 경우 공격자가 암호화된 모든 통신을 복호화할 수 있는 치명적인 문제점을 가지고 있다.

이를 해결하기 위해 클라이언트와 서버는 키 교환 알고리즘을 통해 안전하게 대칭키를 공유한다.

SSH 에서 사용하는 대표적인 키교환 알고리즘인 디피-헬만(Diffie-Hellman : DH) 알고리즘은 상대방의 공개키와 나의 개인키를 통해 대칭키를 얻어는 방법이다.

이 단계에서 클라이언트와 서버는 임시 비대칭키 방식의 키 쌍을 생성하하고 공개 키를 교환한다.

DH 를 통해 클라이언트와 서버는 대칭키인 세션키를 공유하게 되고 이후 모든 통신은 세션키를 통해 암호화된다.

[주의] 대칭키 교환에 사용되는 키 쌍은 서버와 클라이언트 인증에 사용되는 SSH 키 쌍과 다름

SSH 가 DH 를 통해 얻은 대칭키는 개별 세션에 대해 생성되며 더 이상 필요하지 않은 즉시 사라진다.
따라서 클라이언트나 서버의 개인키가 유출되어도 이전 세션키를 통해 수행한 통신 내용을 복호화할 수 없다.
이는 TLS 세션에서도 사용되는 구성으로 인증서에 만료 날짜가 있는 경우 개인키는 중간자 공격으로 인해 유출되어도 인증서가 만료되었을 시 사용할 수 없으므로 만료 후에는 강한 보안 유지를 하지 않아도 된다.

※ RSA 를 통해 세션키를 암호화 했을 시에는 개인키가 유출되면 세션키를 복호화하고 전체 통신을 읽을 수 있다.

암호화 방식이 정해지고 난 후에는 전송되는 메시지에 MAC(Message Authentication Code)가 포함되어 있어야 상대방이 패킷의 무결성을 확인할 수 있다.
MAC 은 패킷의 마지막 부분에 위치하여 세션키로 암호화된 영역 바깥에서 전송되며, 일반적으로 데이터를 먼저 암호화하고 MAC 을 계산하는 방법을 권장한다.

MAC 은 공유 세션키와 메시지의 패킷 시퀀스 번호 및 실제 메시지 내용으로부터 계산되고 데이터의 무결성 및 통신 인증 확인 목적으로 사용된다.

3.3 서버에 접근할 수 있는 클라이언트인지 확인(서버 관점)
서버 또한 자신에게 접속하려는 클라이언트가 자신에게 접근할 수 있는 권한이 있는지 확인하는 단계가 필요하다.
가장 간단한 방법으로 패스워드 인증이 있다.
서버는 단순히 로그인하려는 계정의 암호를 묻고 클라이언트가 입력한 비밀번호는 세션키를 통해 암호화되고 전송되어 외부로부터 안전하게 보호된다.
패스워드가 암호화되지만 패스워드의 복잡성 설정의 한계가 있기 때문에 일반적으로 이 방법을 사용하지 않는 것이 좋다.
자동화된 스크립트를 통해 일반적인 길이의 패스워드는 공격에 의해 해제될 수 있다.
가장 많이 사용되고 권장되는 방법은 SSH 키 쌍을 사용하는 것이다.
이 방법을 사용하기 위해서는 클라이언트 측에서도 SSH 키 쌍을 생성해야 한다.

SSH 키 쌍을 통한 클라이언트 인증은 앞서 살펴본 올바를 서버인지 확인하는 과정과 비슷하다.

클라이언트는 인증할 키 쌍의 ID 를 서버에 전송
서버는 클라이언트가 접속하고자 하는 계정의 .ssh/authorized_keys 파일을 확인
ID 에 매칭되는 공개키가 있을 시, 서버는 난수를 생성하고 클라이언트의 공개키로 암호화
서버는 클라이언트에게 암호화된 메시지 전송
클라이언트의 개인키를 통해 암호화된 메시지를 복호화하여 난수 추출
클라이언트는 난수를 세션키와 결합하여 해시값 계산 후 서버 전송
서버는 저장된 난수와 세션키를 결합하여 해시값 계산 후 비교
일치할 시 클라이언트 인증
이와 같이 비대칭키를 이용하여 클라이언트가 개인키를 가지고 있는 정상적인 클라이언트라고 증명할 수 있다.

출처:
https://limvo.tistory.com/21#:~:text=SSH%20%EC%9E%91%EB%8F%99%20%EC%9B%90%EB%A6%AC,%EB%B0%8F%20%EC%95%94%ED%98%B8%ED%99%94%EB%A5%BC%20%EC%88%98%ED%96%89%ED%95%9C%EB%8B%A4.&text=1.%20%ED%81%B4%EB%9D%BC%EC%9D%B4%EC%96%B8%ED%8A%B8%EB%8A%94%20%EC%84%9C%EB%B2%84%EC%97%90,%ED%95%98%EB%8A%94%20%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4%EB%A5%BC%20%EC%8B%9C%EC%9E%91%ED%95%9C%EB%8B%A4.

4. 클라이언트가 서버에 원격 접속을 할 수 있다.

이제 세션키를 통해 클라이언트와 서버는 안전한 네트워크 통신을 수행할 수 있다.

※ 비대칭키 방식 정리

공개키로 암호화한 내용은 공개키로 복호화할 수 없고 개인키로 복호화가 가능하다.
개인키로 암호화한 내용은 개인키로 복호화할 수 없고 공개키로 복호화가 가능하다.
비대칭키 시스템에서 개인키는 자신만 가지고 있는 단일 소유권이고 공개키는 여러 사용자가 소유할 수 있는 분산 소유권이다.

두개의 노드가 통신한다고 했을 때 서로의 공개키로 암호화하여 송신한다면, 개인키를 가지고 있는 수신자를 제외하고 다른 노드는 통신 내용을 복호화할 수 없다.

그러나 개인키로 암호화하여 송신한다면, 공개키를 가지고 있는 모든 노드들이 통신 내용을 복호화할 수 있을 것이다.

그래서 개인키는 메시지에 서명하는 용도로 사용하고 통신 시 암호화에 사용하지 않는다.

메시지 서명은 작성자가 해시값을 개인키로 암호화하여 모든 사용자가 공개키로 해시값을 복호화하고 수신된 해시값과 비교하여 올바른 작성자임을 증명하는 것이다.

(= 공개키 소유자는 서명된 메시지의 출처가 개인키 소유자라는 것을 확인할 수 있다.)

※ 키교환 과정 (Elliptic Curve Diffie-Hellman 방식)

클라이언트와 서버가 서로에게 SSH_MSG_KEX_INIT 메시지를 전송 (암호화 방식 리스트 포함)
양쪽 모두 동일한 알고리즘을 사용하여 암호화 지원 목록에서 선택

클라이언트는 비대칭키 키 쌍(공개키, 개인키) 생성 → 사용 후 삭제 (키 교환에만 사용되고 나중에 폐기)
클라이언트는 SSH_MSG_KEX_ECDH_INIT 메시지를 서버에 전송 (클라이언트 공개키 전송)

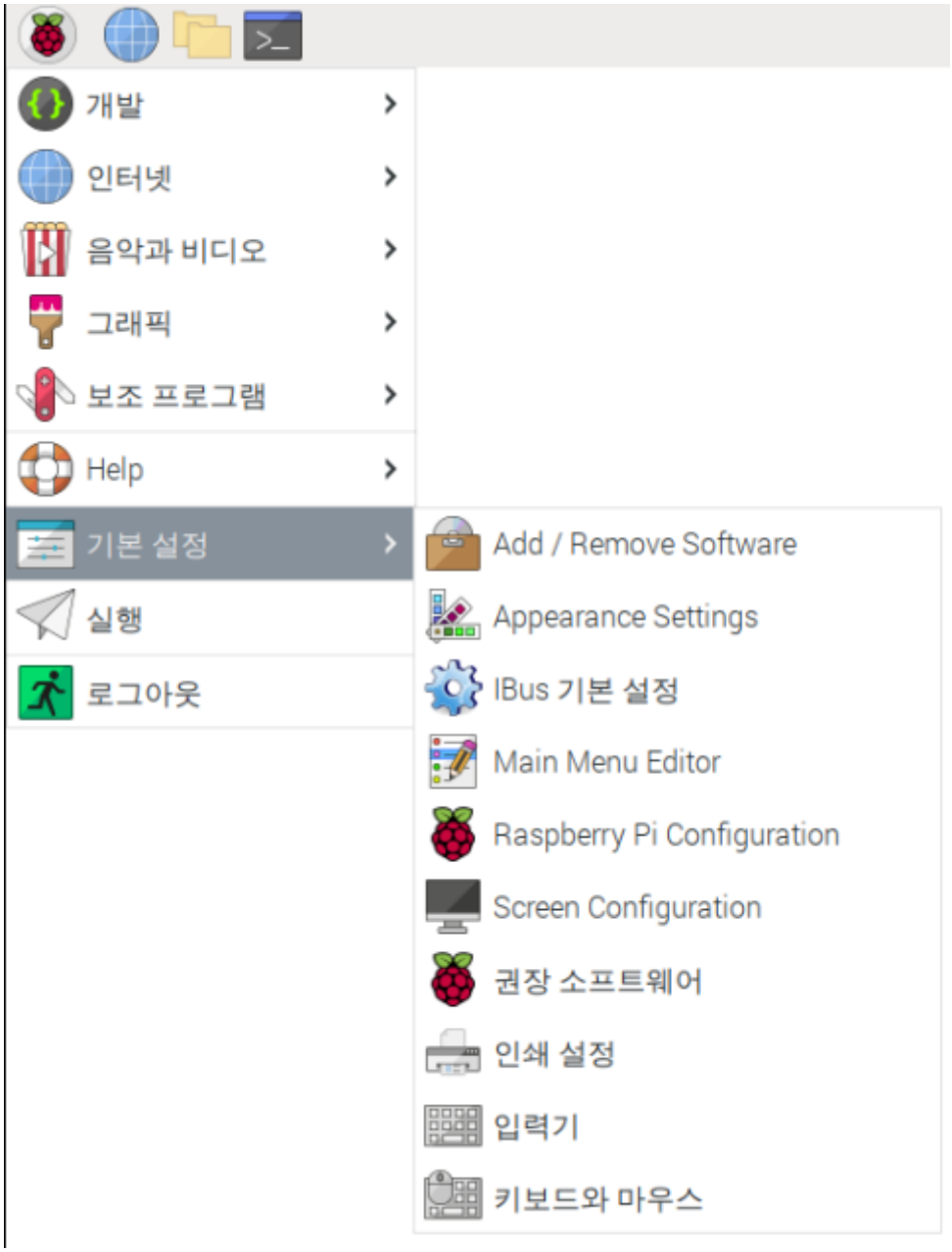
서버에서 SSH_MSG_KEX_ECDH_INIT 수신 시 비대칭키 키 쌍(임시) 생성
서버는 클라이언트의 공개키와 자신의 키 쌍으로 대칭키 K 생성
교환할 해시값 생성 (대칭키 포함, 서버의 개인키로 서명) 및 클라이언트로 전송
서명을 통해 올바른 서버임을 클라이언트가 확인 가능

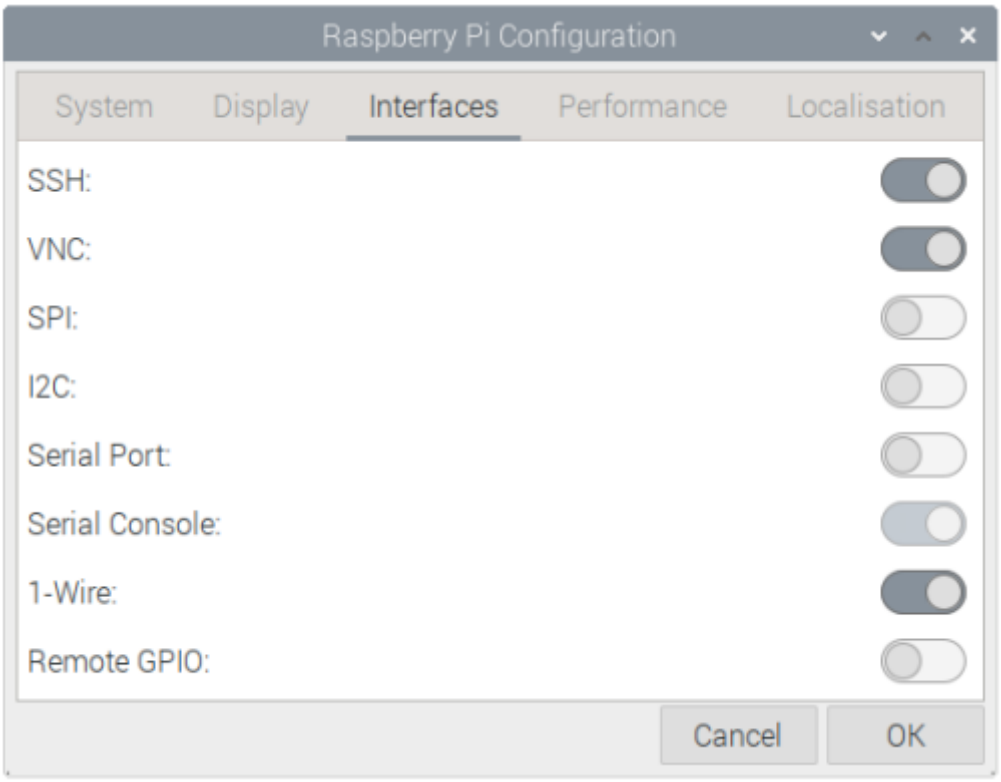
클라이언트는 서버로부터 SSH_MSG_KEX_ECDH_REPLY 수신
클라이언트가 서버의 응답으로부터 서버의 공개 키 추출, HS 의 서명 확인
로컬에 존재하는 서버 공개키와 추출한 공개키 비교, MITM 공격을 방지하기 위한 서명 확인
클라이언트에서 대칭키 공유

데이터 암호화를 시작하기 전 새로운 키 생성 (대칭키 K 로 충분하지 않음)

【03】설정

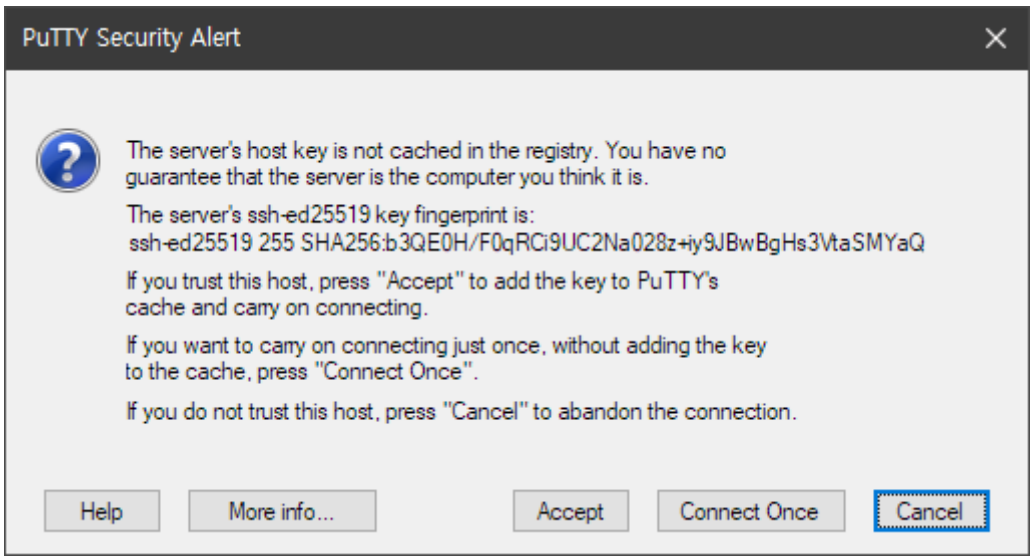
Pi OS를 설치하면 기본적으로 SSH서버를 설치가 되고, 사용할 수 있도록 설정만 하면 됩니다.


개발
인터넷
음악과 비디오
그래픽
보조 프로그램
Help
기본 설정
실행
로그아웃
Add / Remove Software
Appearance Settings
IBus 기본 설정
Main Menu Editor
Raspberry Pi Configuration
Screen Configuration
권장 소프트웨어
인쇄 설정
입력기
키보드와 마우스

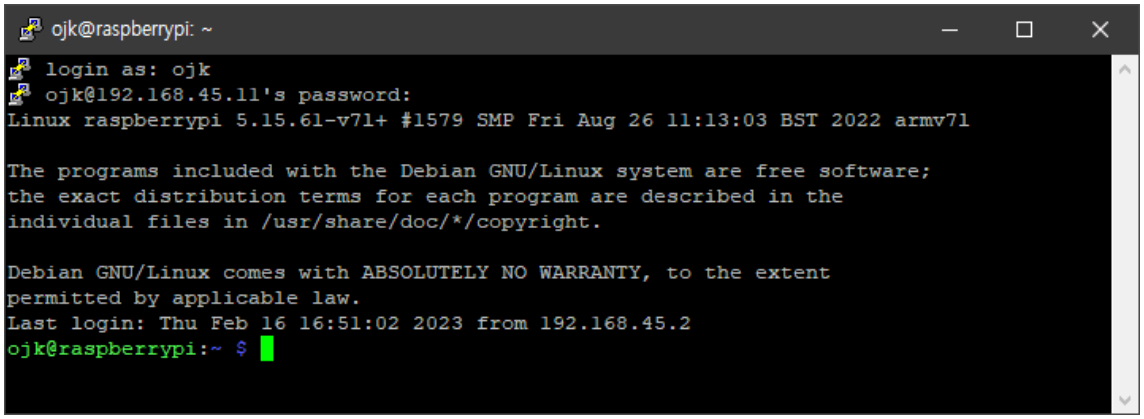


【04】 사용

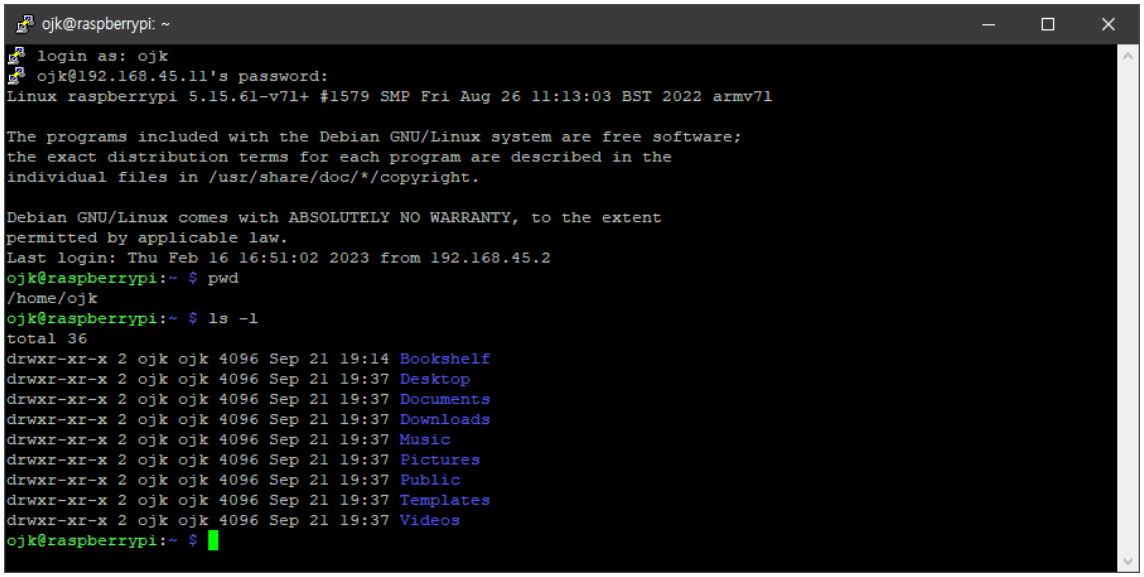
처음 접속하면 다음과 같은 창이 나옵니다. [Accept]를 클릭합니다.



사용자와 비밀번호를 입력합니다.



이제 명령어를 입력하면 실행이 됩니다.



2. LAPM 서버

데이터 베이스만 사용하려면 Maria DB만을 설치하지만 라즈베리파이를 웹서버, DB서버로 사용하기 위해서는 웹서버인 Apache, 웹프로그래밍 언어인 php를 함께 설치하여야 합니다. Apache, Php, MySQL은 늘 항상 같이 다니는 친한 친구이므로 이를 APM이라고 합니다. 현재로서는 MySQL 대신에 이와 거의 동격인 MariaDB를 설치합니다. 일반적으로 APM설치는 Apache, PHP, Maria DB를 설치를 말합니다. 리눅스에 설치하면 Linux APM이므로 LAPM이라 하고 마이크로소프트 윈도에 설치하면 Window APM이므로 WAPM이라고 합니다.
라즈베리파이는 리눅스 계열이므로 당연히 LAPM을 설치합니다.

따라서 아파치를 설치하고, MariaDB를 설치하고, php를 아파치의 모듈로서 설치하고 php가 MariaDB에 접근할 수 있도록 MariaDB 연동 모듈을 설치하면 3개의 프로그램이 연동되어 동작합니다. 물론 아파치를 설치하고, php를 아파치의 모듈로서 설치하고, MariaDB를 설치하고, php가 MariaDB에 접근할 수 있도록 MariaDB 연동 모듈을 설치하여도 상관 없습니다. 현재 설치 버젼은 어떤 것을 먼저 설치하더라고 php에서 연동 모듈만 설치하면 전부 연동 되도록 설계되어 있습니다.

【01】apt 업그레이드

다음 2개의 명령을 사용하여 apt를 최신 버젼으로 업그레이드합니다.

```
$sudo apt update
$sudo apt upgrade
```

```
root@raspberrypi:/home/ojk# apt update
Hit:1 http://archive.raspberrypi.org/debian bullseye InRelease
Hit:2 http://deb.debian.org/debian bullseye InRelease
Hit:3 http://deb.debian.org/debian bullseye-updates InRelease
Hit:4 http://security.debian.org/debian-security bullseye-
security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
110 packages can be upgraded. Run 'apt list --upgradable' to see
them.
root@raspberrypi:/home/ojk#
```

```
ojk@raspberrypi:~ $ sudo apt upgrade
```

```
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다... 완료
상태 정보를 읽는 중입니다... 완료
업그레이드를 계산하는 중입니다... 완료
다음 패키지가 자동으로 설치되었지만 더 이상 필요하지 않습니다:
  libfuse2
Use 'sudo apt autoremove' to remove it.
0 개 업그레이드, 0 개 새로 설치, 0 개 제거 및 0 개 업그레이드 안 함.
ojk@raspberrypi:~ $
```

【02】아파치 설치와 설정

01) 설치

아파치는 버젼이 2이므로 apache2 입니다.

```
$sudo apt install apache2
```

```
ojk@raspberrypi:~ $ sudo apt install apache2
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다... 완료
상태 정보를 읽는 중입니다... 완료
다음 패키지가 자동으로 설치되었지만 더 이상 필요하지 않습니다:
  libfuse2
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.3-0
제안하는 패키지:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
다음 새 패키지를 설치할 것입니다:
  apache2 apache2-bin apache2-data apache2-utils libapr1
libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.3-0
0 개 업그레이드, 9 개 새로 설치, 0 개 제거 및 0 개 업그레이드 안 함.
2,349 k 바이트 아카이브를 받아야 합니다.
이 작업 후 7,931 k 바이트의 디스크 공간을 더 사용하게 됩니다.
계속 하시겠습니까? [Y/n] y
받기:1 http://deb.debian.org/debian bullseye/main arm64 libapr1
arm64 1.7.0-6+deb11u2 [99.6 kB]
받기:2 http://deb.debian.org/debian bullseye/main arm64
libaprutil1 arm64 1.6.1-5+deb11u1 [89.6 kB]
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
:::::::::::::
Created symlink /etc/systemd/system/multi-
user.target.wants/apache2.service →
/lib/systemd/system/apache2.service.
Created symlink /etc/systemd/system/multi-
user.target.wants/apache-htcacheclean.service →
/lib/systemd/system/apache-htcacheclean.service.
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for libc-bin (2.31-13+rpt2+rpi1+deb11u5) ...
ojk@raspberrypi:~ $
```

02) 설치 확인

03) 환경 설정

홈페이지를 저장할 디렉토리를 변경하려면 DocumentRoot /var/www/html를 변경합니다. /var/www/html에서 /home/ojk/html로 변경하려면

```
DocumentRoot /home/ojk/html
```

일반적으로 홈페이지를 저장할 디렉토리는 /var/www/html 이지만, 홈페이지는 클라이언트에서 만들어서 /var/www/html 에 업로드합니다.

/var/www/html 에 업로드하려면

/var/www/html 를 홈디렉토리로 하는 사용자를 만들어서 그 사용자로 업로드하여야 하고 권한 설정도 하여야 하므로 복잡합니다. 그러나 일반 유저의 홈디렉토리를 홈페이지를 저장할 디렉토리로 지정하면 쉽게 업로드할 수 있으므로 편리합니다. 그러나 가상의 사용자의 홈페이지를 사용하는 경우에 혼돈이 되고, 보안 상 좋지 않으므로 권장하지 않습니다.

【03】PHP

01) 설치

```
$sudo apt install php
```

모듈 설치의 명령어를 한꺼번에 지정하여 한꺼번에 설치할 수도 있지만 설치 중에 에러가 발생하는 경우에는 조치를 하여야 하므로 하나씩 설치하는 것을 권장합니다.

확장 모듈 중에서 아파치와 mysql과의 연결은 필수적이므로 반드시 설치하여야 합니다. 설치 버젼에 따라서 php를 설치할 때에 libapache2-mod-php 는 같이 설치되기도 합니다. 설령 php 설치 시에 libapache2-mod-php가 설치되더라도 다시 설치하면 설치 여부를 확인할 수 있기 때문에 연결 모듈을 그대로 설치합니다.

libapache2-mod-php : apache 와 PHP 연결 모듈
php-mysql : MySQL 연동 모듈

```
ojk@raspberrypi:~ $ sudo apt install libapache2-mod-php
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다... 완료
상태 정보를 읽는 중입니다... 완료
다음 패키지가 자동으로 설치되었지만 더 이상 필요하지 않습니다:
  libfuse2
Use 'sudo apt autoremove' to remove it.
다음 새 패키지를 설치할 것입니다:
  libapache2-mod-php
0 개 업그레이드, 1 개 새로 설치, 0 개 제거 및 0 개 업그레이드 안 함.
6,460 바이트 아카이브를 받아야 합니다.
이 작업 후 17.4 k 바이트의 디스크 공간을 더 사용하게 됩니다.
받기:1 http://deb.debian.org/debian bullseye/main arm64
libapache2-mod-php all 2:7.4+76 [6,460 B]
내려받기 6,460 바이트, 소요시간 0 초 (14.6 k 바이트/초)
Selecting previously unselected package libapache2-mod-php.
(데이터베이스 읽는중 ...현재 98164 개의 파일과 디렉터리가 설치되어
있습니다.)
Preparing to unpack .../libapache2-mod-
php_2%3a7.4+76_all.deb ...
Unpacking libapache2-mod-php (2:7.4+76) ...
libapache2-mod-php (2:7.4+76) 설정하는 중입니다 ...
ojk@raspberrypi:~ $
```

```
ojk@raspberrypi:~ $ sudo apt install php-mysql
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다... 완료
상태 정보를 읽는 중입니다... 완료
다음 패키지가 자동으로 설치되었지만 더 이상 필요하지 않습니다:
  libfuse2
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  php7.4-mysql
다음 새 패키지를 설치할 것입니다:
  php-mysql php7.4-mysql
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::
php-mysql (2:7.4+76) 설정하는 중입니다 ...
Processing triggers for libapache2-mod-php7.4 (7.4.33-
1+deb11u3) ...
Processing triggers for php7.4-cli (7.4.33-1+deb11u3) ...
ojk@raspberrypi:~ $
```

나머지 모듈도 설치합니다.

php-curl : CURL 모듈
php-fpm : 서버사이드, HTML 기반 스크립트 언어(FPM-CGI)
php-gd : GD 모듈
php-intl : Internationalisation 모듈
php-mbstring : MBSTRING 모듈
php-soap : SOAP 모듈
php-xml : DOM, SimpleXML, XML, XSL 확장 모듈
php-xmlrpc : XMLRPC-EPI 모듈
php-zip : zip 모듈

02) 설치 확인

아파치의 환경설정에서 홈페이지가 저장되는 디렉토리가 /var/www/hmtl/이므로 /var/www/hmtl/ 에 phpinfo.php 파일을 생성하여 다음 내용을 작성합니다.

```
<?php
echo "오재관";
phpinfo();
?>
```

위 프로그램은 웹브라우저에 "오재관"을 출력하고, php 정보를 출력하는 프로그램입니다.

클라이언트 웹브라우저에 URL을 입력합니다.

【04】Maria DB

01) Maria DB 소개

MariaDB는 독립적인 프로그램이므로 APM 설치 순서와 관계 없이 설치할 수 있습니다. 그렇지만 본래의 목적이 웹기반에서 DB를 사용하고자 하므로 설치하고 php와 연동합니다.

02) 설치

```
$sudo apt install mariadb-server
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다... 완료
상태 정보를 읽는 중입니다... 완료
다음 패키지가 자동으로 설치되었지만 더 이상 필요하지 않습니다:
  libfuse2
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  galera-4 gawk libcgi-fast-perl libcgi-pm-perl libclone-perl
libconfig-inifiles-perl libdbd-mariadb-perl libdbi-perl
libencode-locale-perl libfcgi-bin libfcgi-perl libfcgi0ldbl
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl
libhttp-date-perl libhttp-message-perl libio-html-perl liblwp-
mediatypes-perl libmariadb3 libsigsegv2
  libterm-readkey-perl libtimedate-perl liburi-perl mariadb-
```

```
client-10.5 mariadb-client-core-10.5 mariadb-common mariadb-
server-10.5 mariadb-server-core-10.5 mysql-common socat
제안하는 패키지:
  gawk-doc libmldbm-perl libnet-daemon-perl libsql-statement-
perl libdata-dump-perl libipc-sharedcache-perl libwww-perl mailx
mariadb-test
다음 새 패키지를 설치할 것입니다:
  galera-4 gawk libcgi-fast-perl libcgi-pm-perl libclone-perl
libconfig-inifiles-perl libdbd-mariadb-perl libdbi-perl
libencode-locale-perl libfcgi-bin libfcgi-perl libfcgi0ldbl
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl
libhttp-date-perl libhttp-message-perl libio-html-perl liblwp-
mediatypes-perl libmariadb3 libsigsegv2
  libterm-readkey-perl libtimedate-perl liburi-perl mariadb-
client-10.5 mariadb-client-core-10.5 mariadb-common mariadb-
server mariadb-server-10.5 mariadb-server-core-10.5
  mysql-common socat
0 개 업그레이드, 32 개 새로 설치, 0 개 제거 및 141 개 업그레이드 안 함.
14.8 M 바이트 아카이브를 받아야 합니다.
이 작업 후 142 M 바이트의 디스크 공간을 더 사용하게 됩니다.
계속 하시겠습니까? [Y/n] y
받기:1 http://ftp.kaist.ac.kr/raspbian/raspbian bullseye/main
armhf libsigsegv2 armhf 2.13-1 [34.3 kB]

받기:32 http://raspbian.raspberrypi.org/raspbian bullseye/main
armhf mariadb-server all 1:10.5.15-0+deb11u1 [35.3 kB]
내려받기 14.8 M 바이트, 소요시간 22 초 (668 k 바이트/초)
패키지에서 템플릿을 추출하는 중: 100%
패키지를 미리 설정하는 중입니다...
Selecting previously unselected package libsigsegv2:armhf.
(데이터베이스 읽는중 ...현재 109272 개의 파일과 디렉터리가 설치되어
있습니다.)

Selecting previously unselected package libmariadb3:armhf.
Preparing to unpack .../06-libmariadb3_1%3a10.5.15-
0+deb11u1_armhf.deb ...
Unpacking libmariadb3:armhf (1:10.5.15-0+deb11u1) ...
Selecting previously unselected package mariadb-client-core-
10.5.
Preparing to unpack .../07-mariadb-client-core-10.5_1%3a10.5.15-
0+deb11u1_armhf.deb ...
Unpacking mariadb-client-core-10.5 (1:10.5.15-0+deb11u1) ...

Unpacking socat (1.7.4.1-3) ...
mysql-common (5.8+1.0.7) 설정하는 중입니다 ...
update-alternatives: using /etc/mysql/my.cnf.fallback to provide
```

```
/etc/mysql/my.cnf (my.cnf) in auto mode
mariadb-common (1:10.5.15-0+deb11u1) 설정하는 중입니다 ...
update-alternatives: using /etc/mysql/mariadb.cnf to provide
/etc/mysql/my.cnf (my.cnf) in auto mode
Selecting previously unselected package mariadb-server-10.5.
(데이터베이스 읽는중 ...현재 109889 개의 파일과 디렉터리가 설치되어
있습니다.)
Preparing to unpack .../00-mariadb-server-10.5_1%3a10.5.15-
0+deb11u1_armhf.deb ...
Unpacking mariadb-server-10.5 (1:10.5.15-0+deb11u1) ...
Selecting previously unselected package libhtml-tagset-perl.
Preparing to unpack .../01-libhtml-tagset-perl_3.20-
4_all.deb ...


...
libhttp-message-perl (6.28-1) 설정하는 중입니다 ...
libcgi-pm-perl (4.51-1) 설정하는 중입니다 ...
libhtml-template-perl (2.97-1.1) 설정하는 중입니다 ...
mariadb-server-10.5 (1:10.5.15-0+deb11u1) 설정하는 중입니다 ...
Created symlink /etc/systemd/system/multi-
user.target.wants/mariadb.service →
/lib/systemd/system/mariadb.service.
mariadb-server (1:10.5.15-0+deb11u1) 설정하는 중입니다 ...
libcgi-fast-perl (1:2.15-1) 설정하는 중입니다 ...
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for libc-bin (2.31-13+rpt2+rpi1+deb11u4) ...
ojk@raspberrypi:~ $
```

마지막 설정 중에서

```
mariadb-server-10.5 (1:10.5.15-0+deb11u1) 설정하는 중입니다 ...
Created symlink /etc/systemd/system/multi-
user.target.wants/mariadb.service →
/lib/systemd/system/mariadb.service.
```

부분을 눈여겨 볼 필요가 있습니다. 마리아DB는 운영체제입장에서는 하나의 프로그램에 불과하지만 실제로는 상당히 규모가 큰 프로그램입니다. 따라서 외부로 서비스하는 서비스 프로그램 관리도구인 systemctl로 관리를 하는데, 이 systemctl 프로그램이 관리할 수 있게 등록하는 과정입니다.

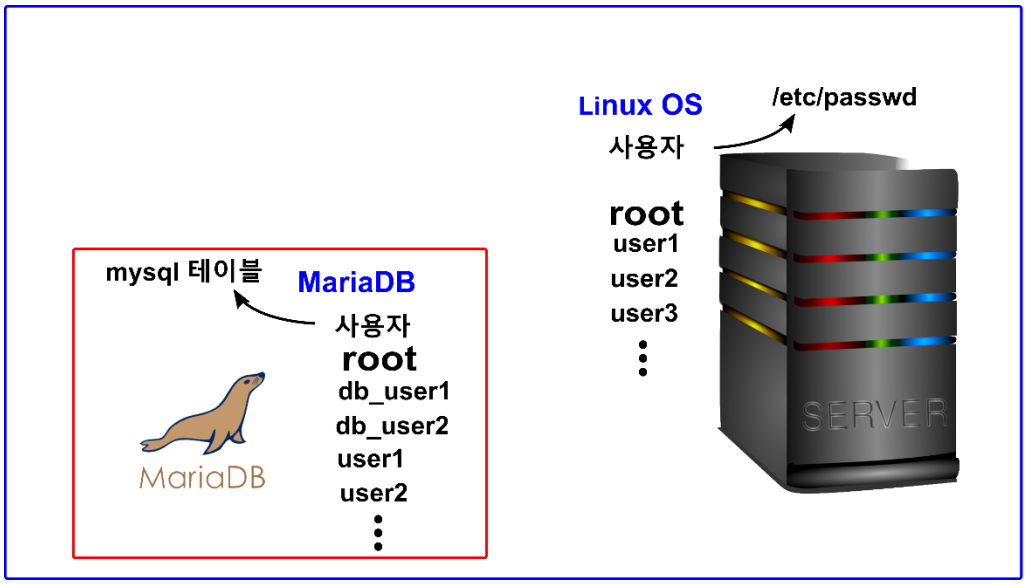
03) 설정

a) 마리아 DB 에 진입

최고 사용자인 root로 진입합니다.

```
ojk@raspberrypi:~ $ mysql -u root
ERROR 1698 (28000): Access denied for user 'root'@'localhost'
ojk@raspberrypi:~ $
```

설치할 때 암호를 입력하였으면 DB에 진입하지만 암호를 넣지 않았기 때문에 접근할 수 없습니다.



여기에서 root는 마리아DB 관리자 이름입니다. 시스템(운영체제)의 최고관리자인 root와는 다릅니다. 그림에서와 같이 사용자는 운영체제의 사용자(root, user1, user2, user3, ...) 와 마리아DB사용자(root, db_user1, db_user2, user1, user2,...) 이 있는데 운영체제의 사용자는 시스템의 자원 즉, 하드디스크, 메일, 프린터 등을 사용할 수 있고, 프로그래밍도 가능한 사용자입니다. DB사용자는 말 그대로 다른 자원은 사용하지 못하고 마리아DB에 데이터를 조회하고, 삽입하고, 수정하고, 삭제만 할 수 있는 데이터베이스 내의 사용자입니다. 운영체제의 사용자 user1, user2와 DB사용자 user1, user2는 이름은 같지만 권한은 다릅니다. 운영체제의 사용자의 계정과 비밀번호는 /etc/passwd에 저장이 되지만 DB사용자는 마리아DB내의 mysql이라는 이름의 테이블에 저장이 됩니다.

b) root 비밀 번호

마리아DB를 설치할 때에 root의 비밀번호을 입력하지만 하지 않은 경우에는 여기에서 입력하

여야 합니다. 방법은 2가지가 있습니다.

root 비밀번호 변경

- DB관리 유틸리티 사용 : mysqladmin
- mysql DB 수정

c) mysqladmin 사용

root 비밀번호를 변경합니다.

```
$sudo mysqladmin -u root password '비밀번호'
```

```
ojk@raspberrypi:~ $ mysqladmin -u root password 'sweet0330'
mysqladmin: connect to server at 'localhost' failed
error: 'Access denied for user 'root'@'localhost''
```

방금 전에 설명한 것처럼 마리아DB는 규모가 큰 프로그램이고 수많은 사용자가 사용하는 DB이므로 강력한 보안이 필요합니다. 따라서 mysqladmin을 실행할 때는 운영체제의 root 권한으로 실행하여야 합니다. 따라서 앞에 sudo를 반드시 붙여 주어야 합니다.

```
ojk@raspberrypi:~ $ sudo mysqladmin -u root password 'sweet0330'
ojk@raspberrypi:~ $
```

마리아db를 재시작하여 root비밀번호 변경을 적용합니다. 마리아DB는 운영체제 입장에서는 하나의(물론 규모가 큰) 프로그램에 불과하므로 운영체제를 재부팅할 필요는 없습니다.

설치할 때 systemctl이라는 도구를 사용하여 프로그램을 관리하도록 설정하였으므로 프로그램을 재시작하기 위해서 다음 명령을 사용합니다.

```
$ sudo systemctl restart mariadb
```

```
ojk@raspberrypi:~ $ sudo systemctl restart mariadb
```

```
ojk@raspberrypi:~ $
```

마리아DB가 실행이 되었으므로 이제 사용할 수 있습니다.

이제 마리아DB에 접속하기 위해서는 암호를 입력하여야 합니다.

```
$ mysql -u root -p
```

```
ojk@raspberrypi:~ $ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 30
Server version: 10.5.15-MariaDB-0+deb11u1 Raspbian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and
others.

Type 'help;' or '\h' for help. Type '\c' to clear the current
input statement.

MariaDB [(none)]>
```

```
ojk@raspberrypi:~ $ mysql -u root -p
```

부분은 쉘에서 내린 명령이고

```
MariaDB [(none)]>
```

은 이제 마리아DB 프로그램 안에 들어와서 사용할 수 있는 상태입니다. 이제 데이터를 조회, 입력, 수정, 삭제를 할 수 있습니다.

d) mysql DB 사용

show databases; 명령은 현재 마리아DB에서 사용하고 있는 DB 목록을 보여주는 명령입니다.

MariaDB [(none)]> show databases;

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.001 sec)
MariaDB [(none)]>
```

3개의 데이터베이스가 기본적으로 작성이 되어 있고, 이중 mysql 이라는 데이터베이스가 사용자 정보를 저장하고 있는 데이터베이스입니다.

먼저 데이터 베이스에 접속하여 mysql DB로 현재사용하고 있는 DB를 변경합니다.

```
use mysql;
```

LAMP는 설치도 복잡하고 용량도 제법 많습니다. SD카드의 남은 용량을 확인하여 봅시다.

```
ojk@raspberrypi:~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        29G  4.0G   24G  15% /
devtmpfs        1.7G     0  1.7G   0% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
tmpfs           759M  1.3M  758M   1% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
/dev/mmcblk0p1  255M   31M  225M  13% /boot
tmpfs           380M   28K  380M   1% /run/user/1000
ojk@raspberrypi:~ $
```

현재 24G byte가 남아 있음을 확인할 수 있습니다.

04) 사용

현재 생성되어 있는 데이터베이스를 확인하여 봅시다.

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.001 sec)

MariaDB [(none)]>
```

데이터 베이스 생성

【형식】

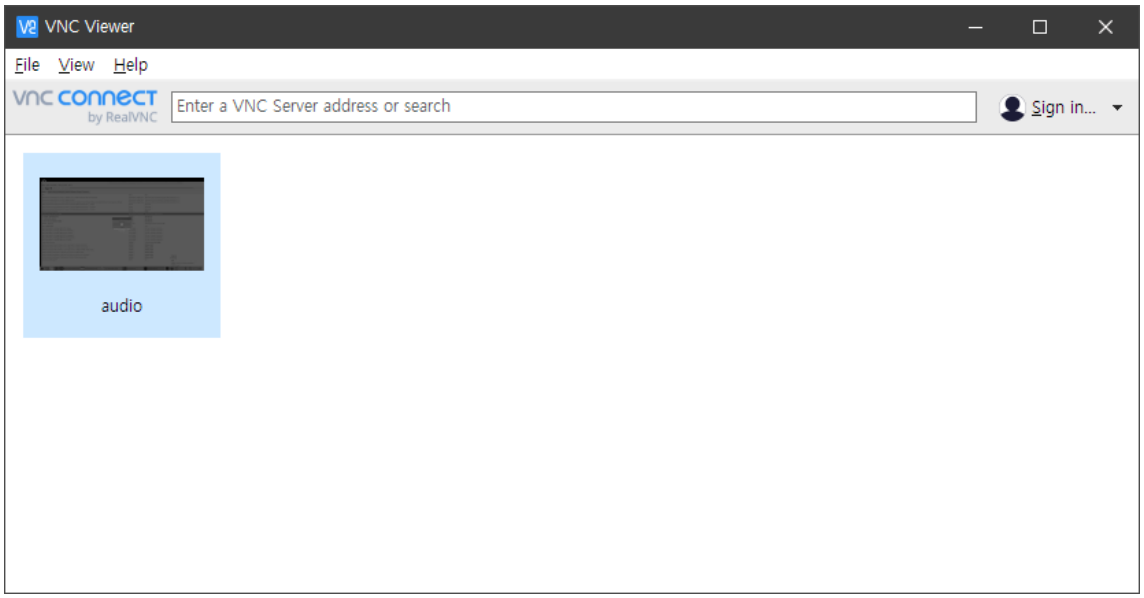
```
create database 데이터베이스명;
```

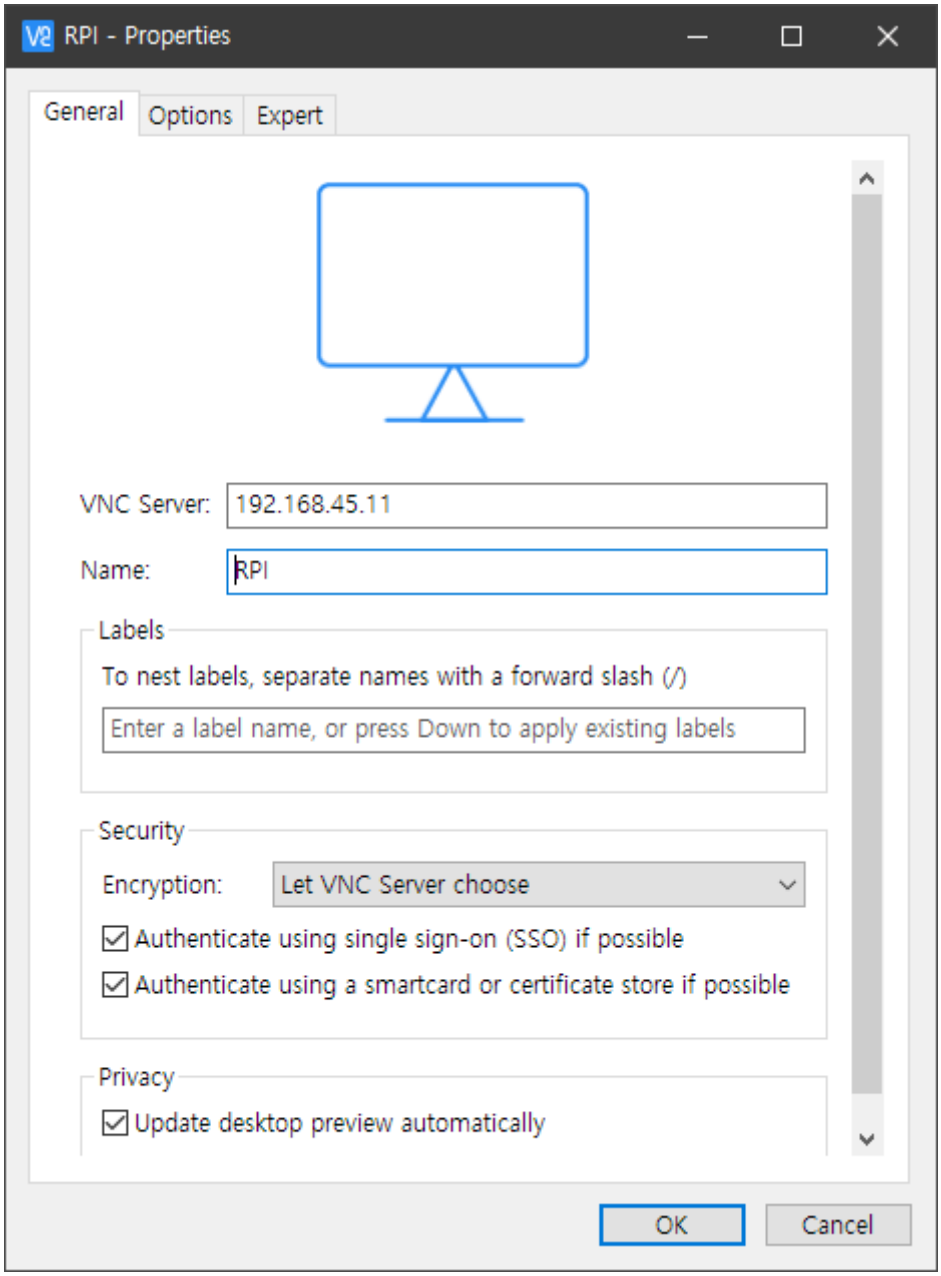
【예제】

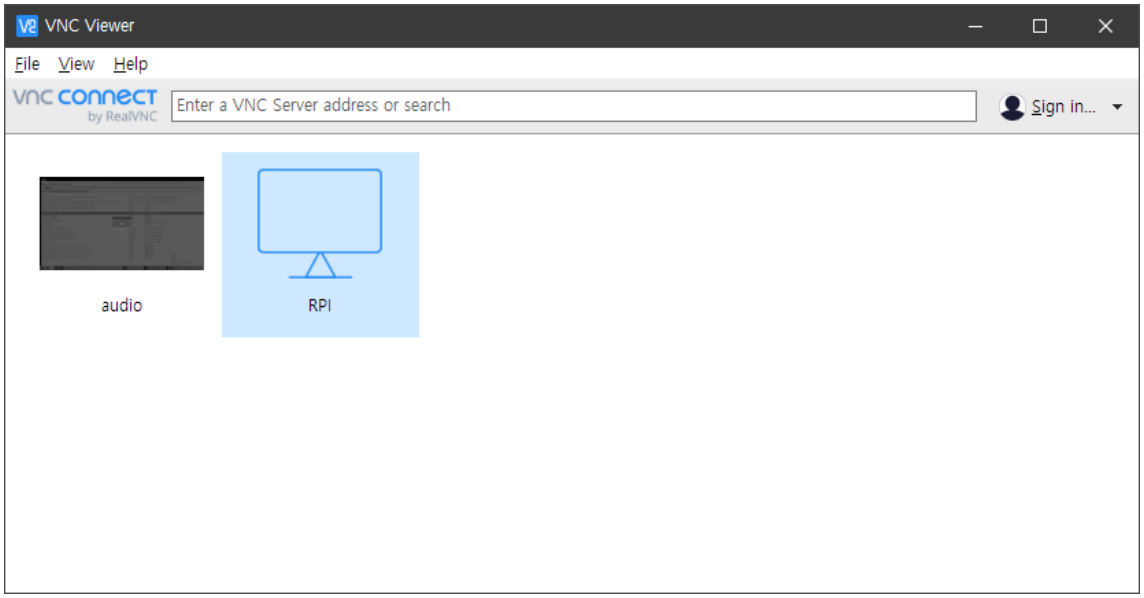
```
create database smartFarm;
```

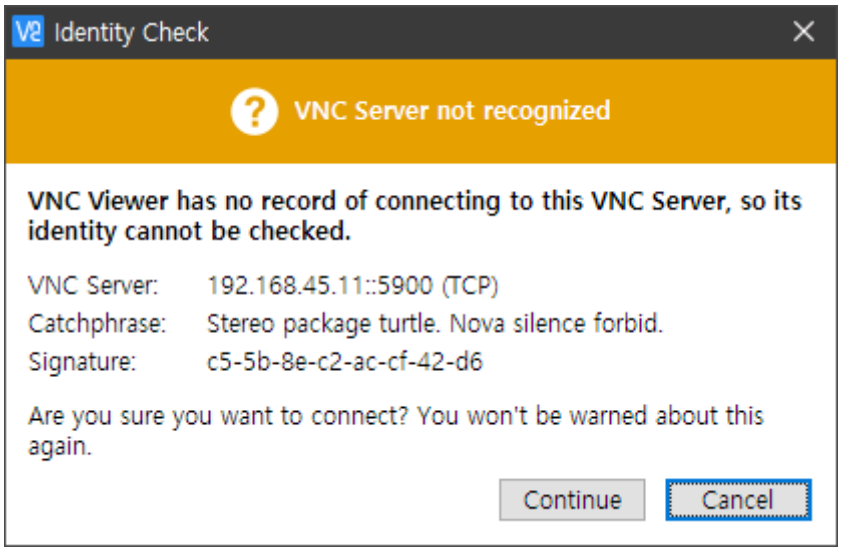
```
MariaDB [(none)]> create database smartFarm;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| smartFarm          |
+--------------------+
4 rows in set (0.001 sec)

MariaDB [(none)]>
```

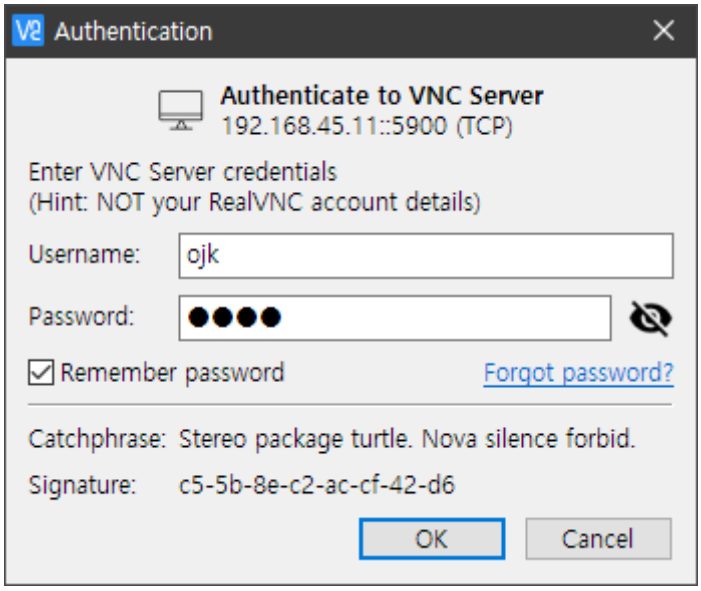
3. Vnc Server


VNC Viewer
File View Help
vnc connect by RealVNC
Enter a VNC Server address or search
Sign in...
audio


RPI - Properties
General
Options
Expert
VNC Server:
192.168.45.11
Name:
RPI
Labels
To nest labels, separate names with a forward slash (/)
Enter a label name, or press Down to apply existing labels
Security
Encryption:
Let VNC Server choose
Authenticate using single sign-on (SSO) if possible
Authenticate using a smartcard or certificate store if possible
Privacy
Update desktop preview automatically
OK
Cancel


VNC Viewer
File View Help
vnc connect by RealVNC
Enter a VNC Server address or search
Sign in...
audio
RPI


Identity Check
VNC Server not recognized
VNC Viewer has no record of connecting to this VNC Server, so its identity cannot be checked.
VNC Server: 192.168.45.11::5900 (TCP)
Catchphrase: Stereo package turtle. Nova silence forbid.
Signature: c5-5b-8e-c2-ac-cf-42-d6
Are you sure you want to connect? You won't be warned about this again.
Continue
Cancel



https://help.ubuntu.com/community/VNC/Servers#Have_x11vnc_start_automatically_via_systemd_in_any_environment_.28Vivid.2B-.29

Installing x11vnc server

cat

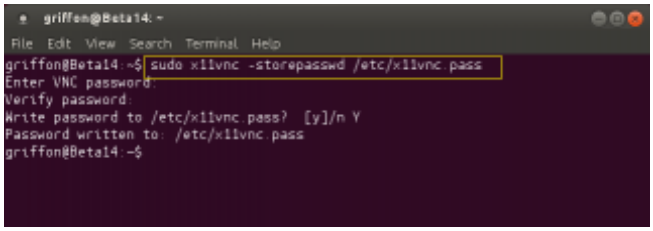
vnc server 에는 여러가지가 있으나 여기에서는 X11vnc server 를 사용하겠다. 이 설치은 꽤 쉽게 할 수 있다. 터미널을 열고서

sudo apt-get install x11vnc

최소한의 보안을 위해서 vnc연결에 암호을 사용한다. 암호는 파일에 저장되므로 암호가 저장되는 파일을 지정하면 되는데 보통 다음 명령을 사용한다.

sudo x11vnc -storepasswd /etc/x11vnc.pass

다음과 같이 암호을 물어 보고 암호는 2 번 입력하면 암호가 /etc/x11vnc.pass 파일에 저장된다.



pi user 가 패스워드 파일을 읽을 수 있도록 권한 설정

$ sudo chmod 755 /etc/x11vnc.pass

systemd를 사용하지 않고 실행

x11vnc -rfbauth /etc/x11vnc.pass -rfbport 5900 -shared

위 스크립트를

/etc/rc.local 에 삽입
x11vnc -rfbauth /etc/x11vnc.pass -rfbport 5900 -shared

/home/pi/.bashrc 파일에 넣어도 되나 콘솔에서 쉘을 실행할 때마다 x11vnc가 살행되므로 .profile에 삽입한다.

/home/pi/.profile 파일에 삽입
x11vnc -rfbauth /etc/x11vnc.pass -rfbport 5900 -forever -shared

[Menu-기본설정-Raspberry Pi Configuration] Interface Tab 에서
SSH, VNC 를 enable로 설정하고 재 부팅한다.

Create the Service Unit file
다음으로는 x11vnc server를 시스템 서비스에 등록하여야 한다.

sudo vi /lib/systemd/system/x11vnc.service
이 파일은 다음 내용이 포함되어야 한다.

[Unit]

Description=Start x11vnc at startup.

After=multi-user.target

[Service]

Type=simple

ExecStart=/usr/bin/x11vnc -auth guess -forever -loop -noxdamage -repeat -rfbauth

/etc/x11vnc.pass -rfbport 5900 -shared

[Install]

WantedBy=multi-user.target

Systemd 설정
이제 시스템 서비스를 재시작하여 x11vnc가 서비스 되도록 하여야 한다.

sudo systemctl daemon-reload

sudo systemctl enable x11vnc.service

패스워드 변경
설치할 때에 암호를 입력하였으므로 다시 실행할 필요는 없으나 때로는 암호를 변경하여야 하는 경우에 사용합니다.

x11vnc -storepasswd 파일명

ex)
$ sudo x11vnc –storepasswd /etc/x11vnc.pass

【소스 프로그램】

```
function calcAge(birth) {
    var date = new Date();
    var year = date.getFullYear();
    var month = (date.getMonth() + 1);
    var day = date.getDate();
    if (month < 10) month = '0' + month;
    if (day < 10) day = '0' + day;
    var monthDay = month + day;
    birth = birth.replace('-', '').replace('-', '');
    var birthdayy = birth.substr(0, 4);
    var birthdaymd = birth.substr(4, 4);
    var age = monthDay < birthdaymd ? year - birthdayy - 1 : year -
birthdayy;
    return age;
}
```

```
ojk@raspberrypi:~ $ sudo apt-cache search vsftpd
ccze - robust, modular log coloriser
ftpd - File Transfer Protocol (FTP) server
resource-agents - Cluster Resource Agents
vsftpd - lightweight, efficient FTP server written for security
vsftpd-dbg - lightweight, efficient FTP server written for
security (debug)
yasat - simple stupid audit tool
```

```
ojk@raspberrypi:~ $ sudo apt-get install vsftpd
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다... 완료
상태 정보를 읽는 중입니다... 완료
다음 패키지가 자동으로 설치되었지만 더 이상 필요하지 않습니다:
  libfuse2
Use 'sudo apt autoremove' to remove it.
다음 새 패키지를 설치할 것입니다:
  vsftpd
0 개 업그레이드, 1 개 새로 설치, 0 개 제거 및 152 개 업그레이드 안 함.
137 k 바이트 아카이브를 받아야 합니다.
이 작업 후 306 k 바이트의 디스크 공간을 더 사용하게 됩니다.
받기:1 http://ftp.kaist.ac.kr/raspbian/raspbian bullseye/main
armhf vsftpd armhf 3.0.3-12 [137 kB]
내려받기 137 k 바이트, 소요시간 6 초 (22.4 k 바이트/초)
패키지를 미리 설정하는 중입니다...
Selecting previously unselected package vsftpd.
(데이터베이스 읽는중 ...현재 112027 개의 파일과 디렉터리가 설치되어
있습니다.)
Preparing to unpack .../vsftpd_3.0.3-12_armhf.deb ...
Unpacking vsftpd (3.0.3-12) ...
vsftpd (3.0.3-12) 설정하는 중입니다 ...
Created symlink /etc/systemd/system/multi-
user.target.wants/vsftpd.service →
/lib/systemd/system/vsftpd.service.
Processing triggers for man-db (2.9.4-2) ...
```

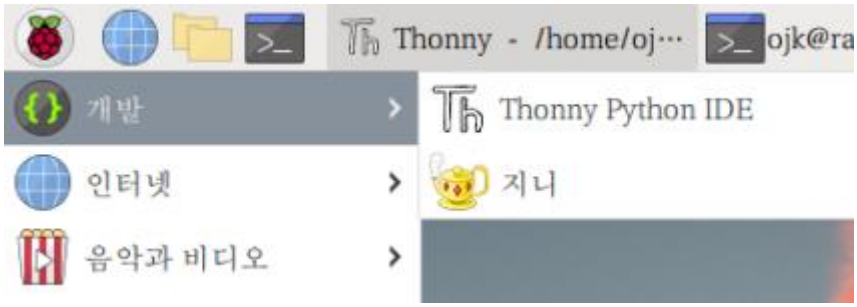
제2편 프로그래밍

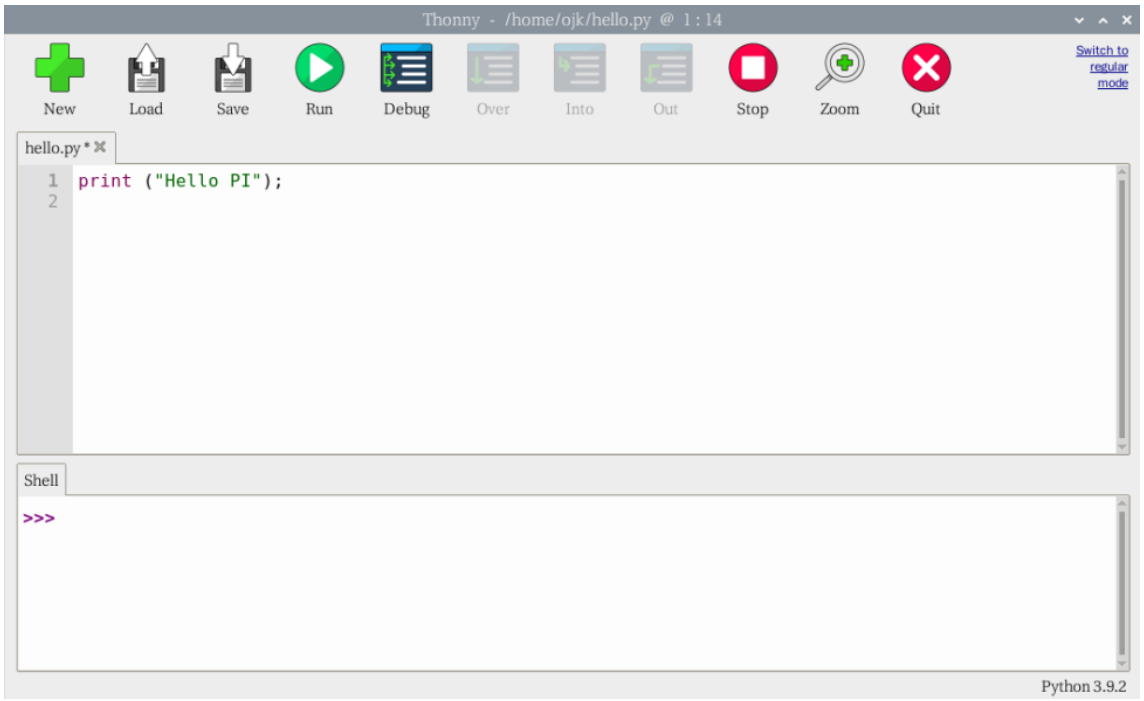
I. 파이썬

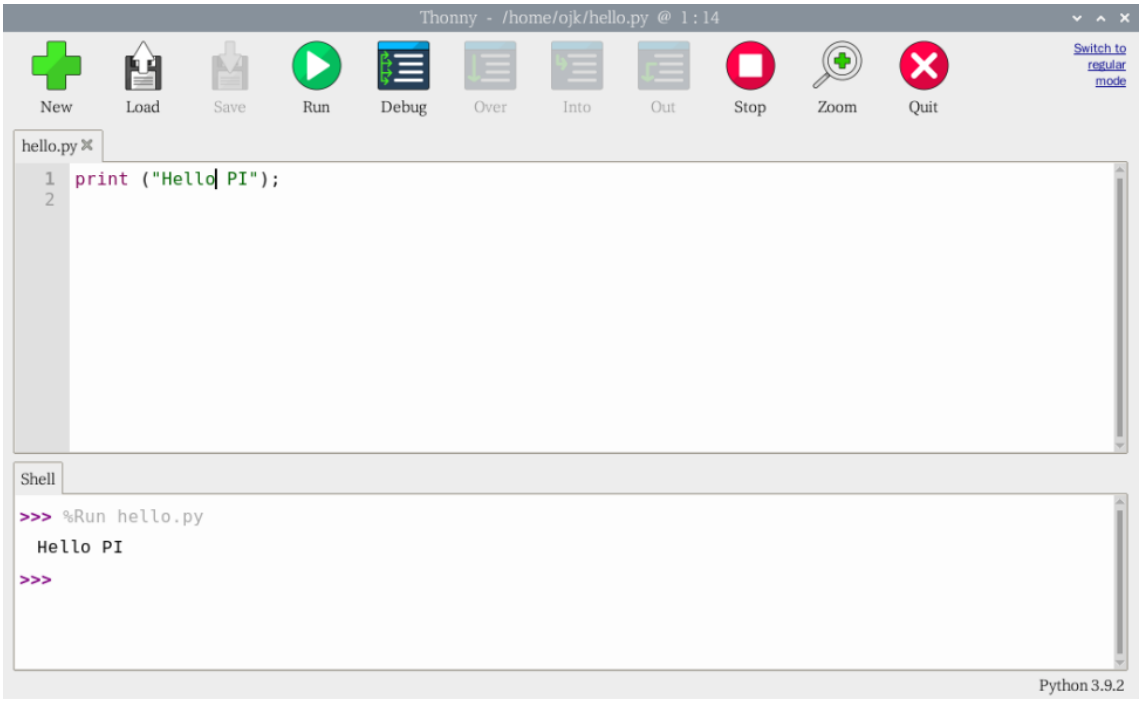
1. 설치

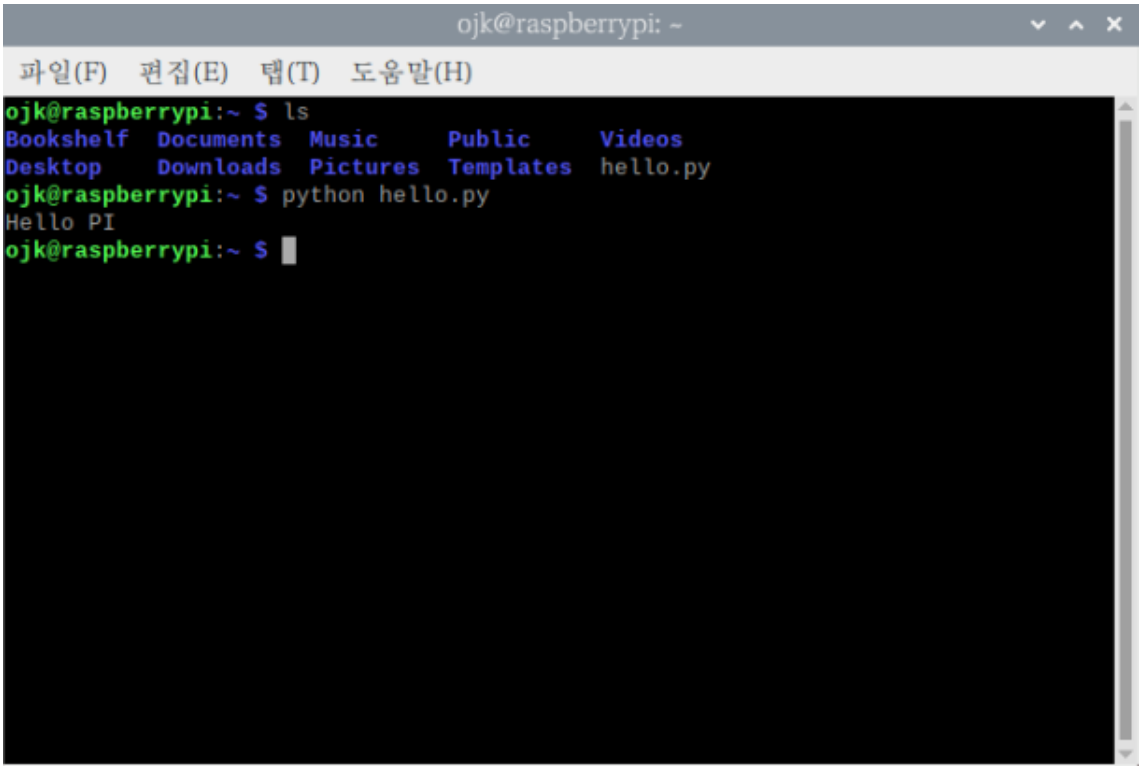
라즈베리파이는 운영체제 설치 시 기본적으로 설치되므로 따로 설치할 필요는 없습니다.

프로그램 편집






Thonny - /home/ojk/hello.py @ 1 : 14
New
Load
Save
Run
Debug
Over
Into
Out
Stop
Zoom
Quit
Switch to regular mode
hello.py
1 print ("Hello PI");
2
Shell
>>> %Run hello.py
Hello PI
>>>
Python 3.9.2


ojk@raspberrypi: ~
파일(F) 편집(E) 탭(T) 도움말(H)
ojk@raspberrypi:~ $ ls
Bookshelf Documents Music Public Videos
Desktop Downloads Pictures Templates hello.py
ojk@raspberrypi:~ $ python hello.py
Hello PI
ojk@raspberrypi:~ $

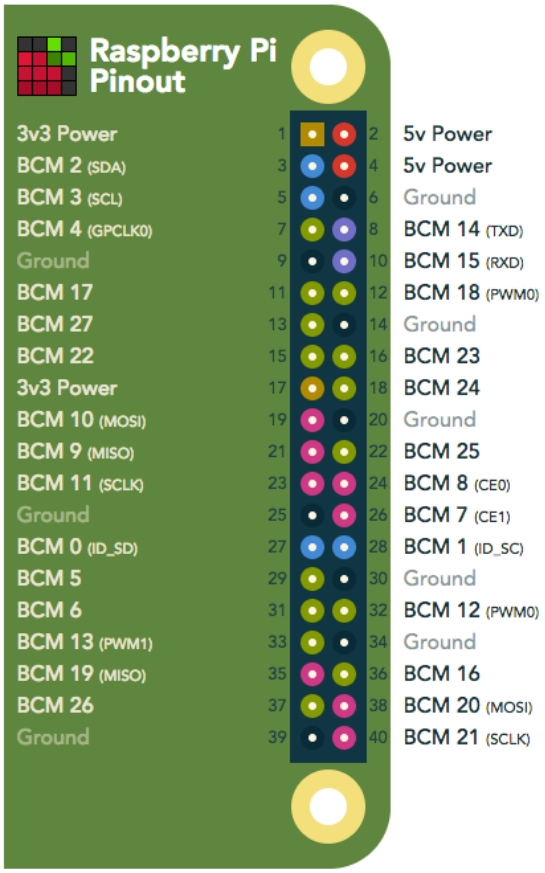
II. C 언어

III. GPIO PHYSICAL COMPUTING

1. GPIO AND PHYSICAL COMPUTING

【01】개요

라즈베리 파이는 GPIO(General Purpose Input Output), SPI, I2C, UART 통신이 가능합니다. 라즈베리파이4의 핀의 수는 40 개이고 ⊕전원(5V 2개, 3.3V 2개), ⊖전원(ground 8개), 예약핀 2개, GPIO 26개로 구성이 됩니다. GPIO 핀은 코딩을 통하여 입력과 출력으로 지정하여 제어할 수 있으며, 핀 구성은 다음과 같습니다.

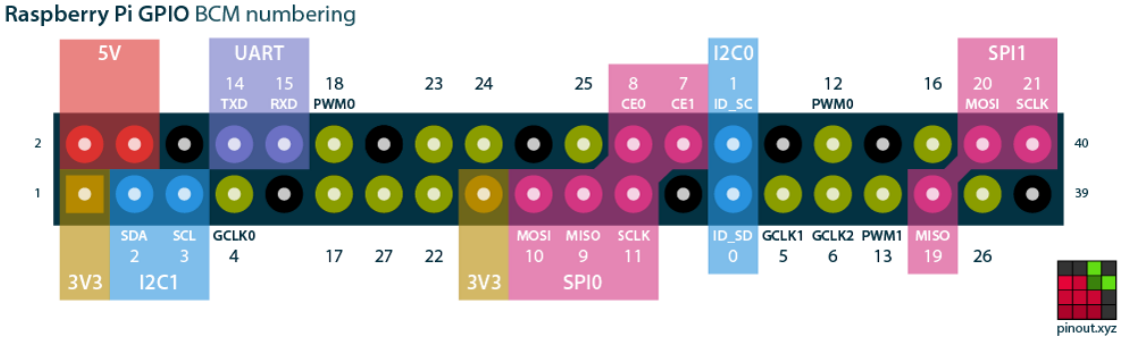


PIN	NAME	NAME	PIN
01	3.3V DC Power	5V DC Power	02
03	GPIO02 (SDA1,I²C)	5V DC Power	04
05	GPIO03 (SDL1,I²C)	Ground	06
07	GPIO04 (GPCLK0)	GPIO14 (TXD0, UART)	08
09	Ground	GPIO15 (RXD0, UART)	10
11	GPIO17	GPIO18(PWM0)	12
13	GPIO27	Ground	14
15	GPIO22	GPIO23	16
17	3.3V DC Power	GPIO24	18
19	GPIO10 (SP10_MOSI)	Ground	20
21	GPIO09 (SP10_MISO)	GPIO25	22
23	GPIO11 (SP10_CLK)	GPIO08 (SPI0_CE0_N)	24
25	Ground	GPIO07 (SPI0_CE1_N)	26
27	GPIO00 (SDA0, I²C)	GPIO01 (SCL0, I²C)	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12 (PWM0)	32
33	GPIO13 (PWM1)	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

주목하여야 할 점은 라즈베리파이는 아두이노와는 다르게 아날로그 입력을 지원하지 않습니다. 따라서 아날로그 입력 신호를 받기 위해서는 아날로그를 디지털로 변환하는 ADC(Analog to Digital Converter)을 따로 추가하여 사용하여야 합니다.

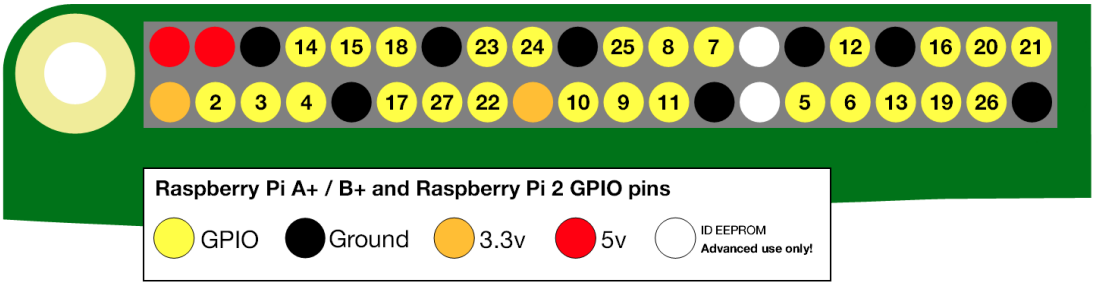
01) 전원

라즈베리파이는 3.3V와 5V 두 종류의 전압을 사용합니다. 3.3V는 2개의 핀을 가지고 있고, 1번과 17번 핀입니다. 5V는 2개의 핀을 가지고 있고, 2번과 4번 핀입니다. 그라운드는 8개의 핀을 가지고 있으며, 6번, 9번, 14번, 20번, 25번, 30번, 34번, 39번입니다. 나머지 31개의 핀은 용도가 각각 다릅니다.


Raspberry Pi GPIO BCM numbering
5V
UART
14 TXD
15 RXD
18 PWM0
23
24
25
8 CE0
7 CE1
I2C0
1 ID_SC
12 PWM0
16
SPI1
20 MOSI
21 SCLK
2
1
40
39
SDA 2
SCL 3
GCLK0 4
17
27
22
MOSI 10
MISO 9
SCLK 11
ID_SD 0
GCLK1 5
GCLK2 6
PWM1 13
MISO 19
26
3V3
I2C1
3V3
SPI0
pinout.xyz

이 핀들은 파이와 외부장치를 연결해주는 인터페이스이다. 가장 간단히 말하면 외부에서 들어오는 입력을 켜거나 끄는 스위치 또는 외부로 나가는 출력을 켜고 끄는 스위치라고 할 수 있다.

40개 중에서 26개는 GPIO 핀이고 나머지는 전원과 그라운드, 2개의 ID EEPROM 핀(pins which you should not play with unless you know your stuff!)이다.



Note that the numbering of the GPIO pins is rather weird. **Appendix 1: A note on pin numbering** below explains why.

WHAT ARE THEY FOR? WHAT CAN I DO WITH THEM?

You can program the pins to interact in amazing ways with the real world. Inputs don't have to come from a physical switch; it could be input from a sensor or a signal from another computer or device, for example. The output can also do anything, from turning on an LED to sending a signal or data to another device. If the Raspberry Pi is on a network, you can control devices that are attached to it from anywhere** and those devices can send data back. Connectivity and control of physical devices over the internet is a powerful and exciting thing, and the Raspberry Pi is ideal for this. There are lots of brilliant examples of physical computing on our blog.

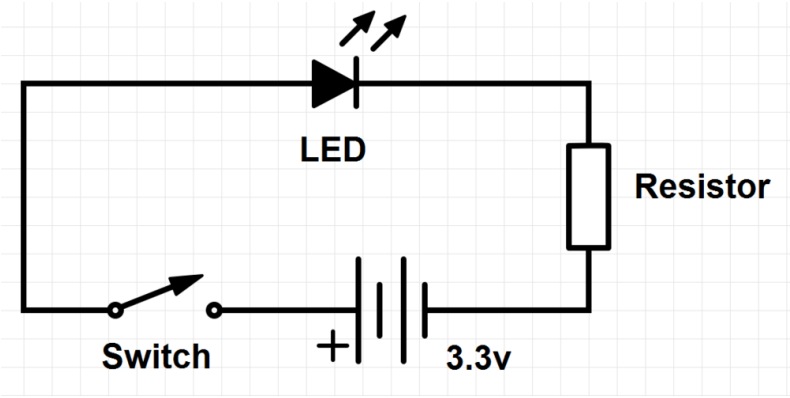
Note: Not **literally** anywhere, of course. You need things like access to the network, a network capable computing device, and electricity. Please do not write to us to point this out. :)

HOW THE GPIO PINS WORK

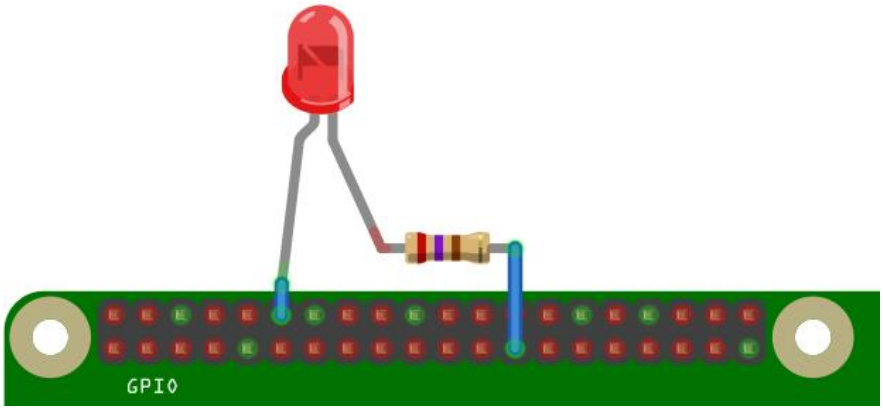
OUTPUT

WARNING: If you follow the instructions, then messing about with the GPIO is safe and fun. Randomly plugging wires and power sources into your Pi, however, may kill it. Bad things can also happen if you try to connect things to your Pi that use a lot of power; LEDs are fine, motors are not. If you are worried about this, then you might want to consider using a breakout board such as the Pibrella until you are confident enough to use the GPIO directly.

Ignoring the Pi for a moment, one of the simplest electrical circuits that you can build is a battery connected to a light source and a switch (the resistor is there to protect the LED):



When we use a GPIO pin as an output, the Raspberry Pi replaces **both the switch and the battery** in the above diagram. Each pin can turn on or off,or go HIGH or LOW in computing terms. When the pin is HIGH it outputs 3.3 volts (3v3); when the pin is LOW it is off.
Here's the same circuit using the Raspberry Pi. The LED is connected to a GPIO pin (which can output +3v3) and a ground pin (which is 0v and acts like the negative terminal of the battery):

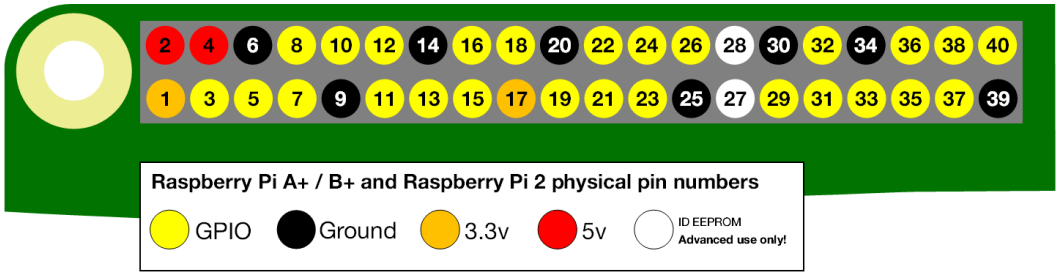


The next step is to write a program to tell the pin to go HIGH or LOW. Here's an example using Python (see Step 2), and here's how to do it in Scratch.
INPUT
GPIO **outputs** are easy; they are on or off, HIGH or LOW, 3v3 or 0v. **Inputs** are a bit trickier because of the way that digital devices work. Although it might seem reasonable just to connect a button across an input pin and a ground pin, the Pi can get confused as to whether the button is on or off. It might work properly, it might not. It's a bit like floating about in deep space; without a reference it would be hard to tell if you were going up or down, or even what up or down meant! This is why you will see phrases like "pull up" and "pull down" in Raspberry Pi GPIO tutorials. It's a way of giving the input pin a reference so it knows for certain when an input is received.
If you'd like to have a go at using the GPIO as an input then have a look at our burping jelly baby and quick reaction game tutorials for Python, or a reaction gamefor Scratch.
THE END OF THE GUIDE. THE START OF SOMETHING AMAZING
We hope that this has encouraged you to have a go at physical computing using the Pi's GPIO; it's really not as daunting as it looks. It all starts with a simple LED, but it can take you to incredible places. Do not underestimate the fun, creativity and sense of achievement you can get from a little computer and a bunch of pins. Have fun! And if you do make something cool please let us know. :)

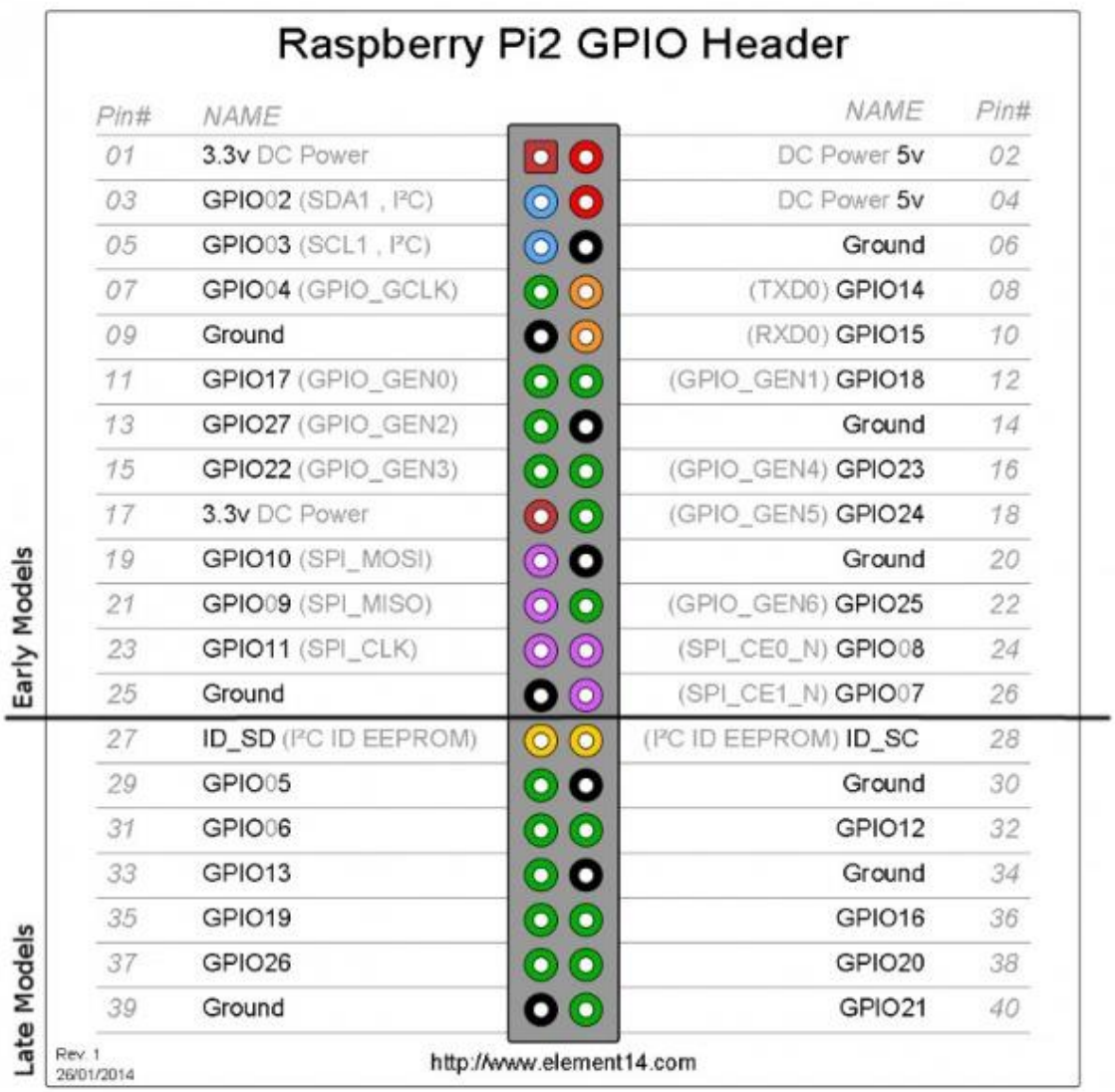
GLOSSARY
GPIO
General purpose input/output; in this specific case the pins on the Raspberry Pi and what you can do with them. So called because you can use them for all sorts of purposes; most can be used as either inputs or outputs, depending on your program.

LED
Light-emitting diode- a small, low-power light source used widely in electronics. Ideal as an introduction to physical computing on the Pi.
PHYSICAL COMPUTING
Computing that involves tangible things connected to a computer, beyond standard input and output devices like keyboards and monitors. Think buttons, lights, robots, alarms, sensors, home automation, teddy bears called Babbage in near space and so on. We love physical computing because as well as being lots of fun, it's such a powerful teaching and learning tool and encourages creativity, problem solving, and collaboration. Computing **beyond the screen** engages children of all ages, and you can make very cool stuff!

APPENDIX 1. A NOTE ON PIN NUMBERING
When programming the GPIO pins there are two different ways to refer to them: GPIO numbering and physical numbering.
GPIO NUMBERING
These are the GPIO pins as the computer sees them. The numbers don't make any sense to humans, they jump about all over the place, so there is no easy way to remember them. You will need a printed reference or a reference board that fits over the pins.
PHYSICAL NUMBERING
The other way to refer to the pins is by simply counting across and down from pin 1 at the top left (nearest to the SD card). This is 'physical numbering' and it looks like this:



WHICH SYSTEM SHOULD I USE?
Beginners and young children may find the physical numbering system simpler -- you simply count the pins. You'll still need a diagram like the one above to know which are GPIO pins, which are ground and which are power though.
Generally we recommend using the GPIO numbering. It's good practice and most resources use this system. Take your pick though -- as long as you use the same system within a program then all will be well. Note that pin numbering can also depend on what programming language you are using: Scratch GPIO, for example, uses physical pin numbers whereas in Python you can choose which to use.
For more details on the advanced capabilities of the GPIO pins see gadgetoid's interactive pinout diagram.


Raspberry Pi2 GPIO Header
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
Early Models
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
Late Models
Rev. 1
26/01/2014
http://www.element14.com

Wedge Silk	Python (BCM)	WiringPi GPIO	Name	P1 Pin Number		Name	WiringPi GPIO	Python (BCM)	Wedge Silk
			3.3v DC Power	1	2	5v DC Power			
SDA		8	GPIO02 (SDA1, I2C)	3	4	5v DC Power			
SCL		9	GPIO03 (SCL1, I2C)	5	6	Ground			
G4	4	7	GPIO04 (GPIO_GCLK)	7	8	GPIO14 (TXD0)	15		TXO
			Ground	9	10	GPIO15 (RXD0)	16		RXI
G17	17	0	GPIO17 (GPIO_GEN0)	11	12	GPIO18 (GPIO_GEN1)	1	18	G18
G27	27	2	GPIO27 (GPIO_GEN2)	13	14	Ground			
G22	22	3	GPIO22 (GPIO_GEN3)	15	16	GPIO23 (GPIO_GEN4)	4	23	G23
			3.3v DC Power	17	18	GPIO24 (GPIO_GEN5)	5	24	G24
MOSI		12	GPIO10 (SPI_MOSI)	19	20	Ground			
MISO		13	GPIO09 (SPI_MISO)	21	22	GPIO25 (GPIO_GEN6)	6	25	G25
		(no worky 14)	GPIO11 (SPI_CLK)	23	24	GPIO08 (SPI_CE0_N)	10		CD0
			Ground	25	26	GPIO07 (SPI_CE1_N)	11		CE1
IDSD		30	ID_SD (I2C ID EEPROM)	27	28	ID_SC (I2C ID EEPROM)	31		IDSC
G05	5	21	GPIO05	29	30	Ground			
G6	6	22	GPIO06	31	32	GPIO12	26	12	G12
G13	13	23	GPIO13	33	34	Ground			
G19	19	24	GPIO19	35	36	GPIO16	27	16	G16
G26	26	25	GPIO26	37	38	GPIO20	28	20	G20
			Ground	39	40	GPIO21	29	21	G21

위의 테이블은 PI 핀번호에 대한 기능, wiringPi 라이브러리 번호, 파이선에서의 번호와 보드에 프린트되어 있는 마킹정보를 정리하여 놓은 테이블입니다.

위의 테이블에서 보면 알수 있지만 라즈베리는 bi-directional I/O핀뿐 아니라, 시리얼(uart), i2c, spi, pwm 을 제공합니다.

하드웨어 셋업

GPIO 사용(digital, pwm, 버튼입력)을 실험하여 보기 위해 필요한 하드웨어를 셋업합니다. 셋업 후 C와 파이썬 예제로 하드웨어를 동작시켜 보겠습니다.

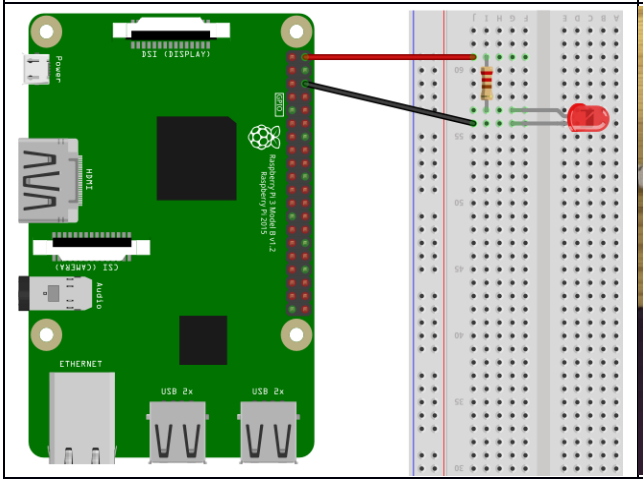
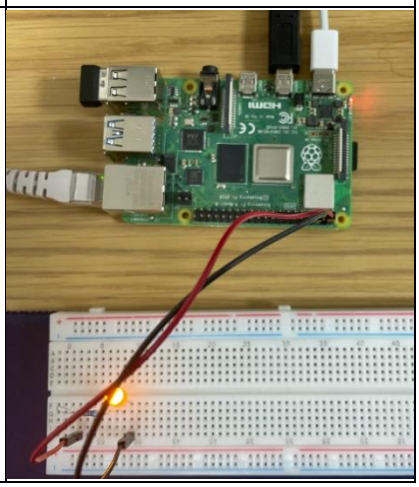
전원 구성

라즈베리파이의 공급 전원은 5V, 2.5A 이지만 CPU(BCM2837 64Bit)와 주변 장치는 3.3V를 이용하므로 GPIO를 사용할 때에는 3.3V로 회로를 구성하는 것이 바람직하다. 또 3.3V의 핀은 최대 50mA의 전류를 사용할 수 있으므로 회로를 구성할 때 이점을 반드시 참조하여야 한다.

라즈베리파이로 회로를 구성하여 봅시다. 라즈베리파이에서 전원이 나와서 저항을 거쳐서 LED를 켜고 GND로 흘러가는 가장 기본적인 회로를 만들어 봅시다. GPIO의 + 전원 포트 중 2번에서 전원이 나와서 6번 GND로 전기가 흐르는 가장 간단한 회로입니다.

아두이노와 마찬가지로 회로도를 그릴 때 Fritzing 프로그램을 사용합니다. 그러나 Fritzing의 기본에는 아두이노만 들어 있고 라즈베리파이는 들어 있지 않으므로 라즈베리파이가 들어 있는 Fritzing 자료를 다운 받아서 실행하면됩니다.

검색어로서 fritzing raspberrypi를 입력하여 검색하거나
http://fritzing.org/projects/raspberry-pi-3에 접속하여 Raspberry PI 3.fzpz를 다운 받아서 더블 클릭합니다.

회로도	실제 구성
	

회로도처럼 연결을 하면 LED가 켜집니다.

앞의 사례는 GPIO를 통하여 전기를 흐르게하여 LED를 켜는 회로입니다. 그런데 이제 LED를 켜고 끄려면 어떻게 할까요? 점퍼선을 빼면 되겠죠... 그러나 그런 방식으로는.... 그래서 우리는 이제 회로를 제어하는 프로그래밍을 해서 회로를 제어할 것입니다.

프로그래밍을 하려면 프로그래밍 언어가 필요한데 여기에서는 C언어와 파이썬을 사용하여 프로그래밍을 하겠습니다.

【02】C (WiringPi) 셋업하기

C언어로 GPIO를 제어하기 위해서는 GPIO를 제어할 수 있는 C언어 라이브러리가 필요합니다. 이 중에서 WiringPi library를 사용하겠습니다.

1) WiringPi library 설치하기

WiringPi는 라즈비안 OS에 기본으로 포함되어 있지 않습니다. 그래서 다운로드하여 설치를 하여 주어야 합니다. 라즈베리 파이를 인터넷에 연결하고 아래의 링크를 참고하여 라이브러리를 설치하여 주거나, WiringPi homepage for instructions on downloading and installing Git을 사용하여 최신 버전을 아래와 같이 다운받아 설치합니다.

```
$ git clone git://git.drogon.net/wiringPi
$ cd wiringPi
$ git pull origin
$ ./build
```

이제 다음과 같이 회로도를 작성하여 봅니다
GPIO 02 핀에서 전류가 나와서 GND로 흘러 들어가게 회로도를 그리고 구성합니다.

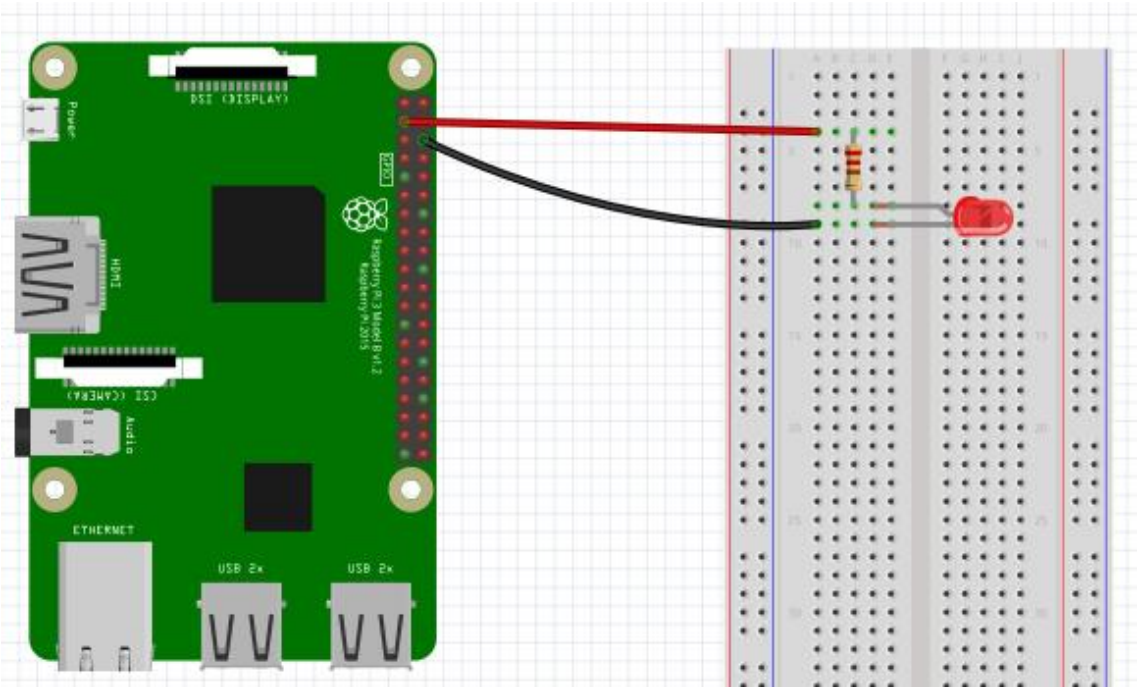
2) Wiring Pi 테스트하기

WiringPi 는 라이브러리이지만 command line 기반의 유틸리티를 포함하고 있습니다. WiringPi 가 잘 설치되었는지 확인하기 위해서 gpio 유틸리티를 사용합니다.

터미널을 열고 아래의 시스템 콜을 입력합니다.

gpio 유틸리티를 사용하여 LED를 켜고 꺼봅시다.

다음과 같이 회로를 구성합니다.



다음 명령어를 입력하면 LED를 켜지고 꺼집니다.

```
$ gpio -g mode 2 output
$ gpio -g write 2 1
$ gpio -g write 2 0
```

이렇게 명령을 주면 3번 핀(GPIO 2)에 연결된 LED가 마지막 두 명령을 받고 깜빡이게 됩니다.

버튼을 테스트하려면 아래와 같이 입력하여 보십시오.

```
pi@raspberrypi ~/code $ gpio -g mode 17 up
pi@raspberrypi ~/code $ gpio -g read 17
```

버튼이 눌렸느냐 안눌렸느냐에 따라 0이나 1이 리턴될 것입니다. 버튼을 누른 상태에서 마지막 명령을 다시 입력하여 보십시오.

gpio 유틸리티는 매우 유용한 유틸리티이므로 man 페이지를 살펴보실 것을 권장합니다.

다음으로 WiringPi 라이브러리에서 제공하는 유용한 몇몇 함수를 살펴보겠습니다.
C (WiringPi) API

라이브러리 셋업
라이브러리 사용을 위해서 아래의 헤더 파일을 include합니다.

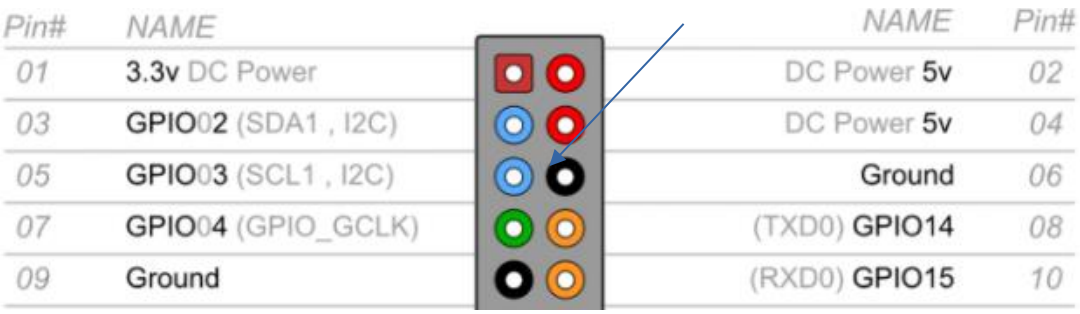
```
#include <wiringPi.h>
```

헤더파일을 include한 뒤에는 라이브러리를 초기화 하여 주어야 합니다. 이 과정에는 먼저번 예제와 같이 핀 번호 체계를 먼저 선택하여 주어야 합니다.
다음의 두 개 함수중 하나를 선택하여 라이브러리를 초기화 합니다. 여기서는 브로드컴 GPIO 번호 체계를 호출 합니다.

```
wiringPiSetup(); // Initializes wiringPi using wiringPi's simlified number system.
wiringPiSetupGpio(); // Initializes wiringPi using the Broadcom GPIO pin numbers
```

핀모드 선언
앞 그림에서 알 수 있는 것처럼 라즈베리파이 핀 번호와 라이브러리에서 사용하는 핀번호는 다르다. 예를 들면 라즈베리파일 3번 핀은 GPIO 핀 번호로는 2번이다. 따라서 GPIO PIN 2번을 사용한다고 해서 실제 라즈베리파이 오른쪽 첫 번째 핀에 연결하면 안 된다. 왼쪽 2 번째 핀에 연결해야 한다.

pinMode(2, OUTPUT)



핀을 입력이나 출력으로 설정하려면 pinMode(핀, 모드) 함수를 이용하면됩니다. 모드는 INPUT, OUTPUT, PWM_OUTPUT을 사용하여 설정할 수 있습니다.

아래의 코드는 핀 22번을 입력으로, 핀 23번을 출력으로 , 핀 18번을 PWM으로 설정합니다.

```
wiringPiSetupGpio();
 pinMode(17, INPUT);
```

```
pinMode(23, OUTPUT);
pinMode(18, PWM_OUTPUT);
```

브로드컴 GPIO번호 체계를 사용하고 있는 점을 잊지마십시오.

디지털 출력

디지털 출력은 digitalWrite(핀, [HIGH/LOW])함수를 이용하여 설정합니다.

아래는 23번 핀을 HIGH로 설정합니다.

```
digitalWrite(23, HIGH);
```

PWM (“Analog”) 출력

PWM핀은 pwmWrite(핀, [0-1023]) 함수를 이용하여 사용합니다.

아래는 18번 핀을 약 70%의 듀티 사이클로 설정하였습니다.

```
pwmWrite(18, 723);
```

디지털 입력

디지털 핀을 읽을 때는 digitalRead(핀)함수를 이용하여 읽습니다.

```
if (digitalRead(17))
        printf("Pin 17 is HIGH₩n");
else
printf("Pin 17 is LOW₩n");
```

위의 코드는 핀22번의 상태를 출력합니다. 핀이 HIGH일시 1을, LOW일시 0을 리턴합니다.

풀업/풀다운 저항

풀업 풀다운 저항 설정은 pullUpDnControl(핀, [PUD_OFF, PUD_DOWN, PUD_UP)을 이용하여 설정합니다.

예를 들어 핀22번에 버튼이 있다면 22번 핀을 풀업하기 위해서는 아래의 명령이 필요합니다.

```
pullUpDnControl(17, PUD_UP);
```

딜레이

딜레이를 주기 위해서는 delay(밀리세컨드) 함수와 delayMicroseconds(마이크로세컨드) 함수를 이용합니다.

2초간 딜레이를 주고 싶다면 아래의 함수를 사용 하십시오.

```
delay(2000);
```

이제 앞의 회로를 이용하여 5번 깜박이고 난 후에 Blink 5 times를 스크린에 출력하는 프로그램을 작성하여 보자.

[예제]

```
// File Name : ex001_ledBlink
#include<stdio.h>
#include<wiringPi.h>
int main(){

   wiringPiSetupGpio();
   pinMode(2,OUTPUT);

   int i;
   for(i=0;i<=4;i++){
     digitalWrite(2, HIGH);
     delay(1000);
     digitalWrite(2, LOW);
     delay(1000);
   }
   printf("Blink 5 times\n");
   return 0;
}
```

[결과]

```
pi@raspberrypi:~ $ ./ledBlink
Blink 5 times
pi@raspberrypi:~ $
```

【03】파이썬 pip

01) pip 란?

pip(Pip Installs Packages)는 Python 패키지를 설치하고 관리하는 패키지 매니저(Package Manneger)입니다. 운영체제를 설치하였을 때, 필요한 프로그램이 모두 설치되는 것은 아닙니다. 원하는 프로그램은 추후에 따로 설치하여야 하는 것처럼, 파이썬도 설치시에 기본만 설치되고 필요한 것은 따로 설치하여야 합니다. 이 때 사용하는 툴이 pip 입니다.

2. pip vs pip3

pip와 pip3의 차이는 파이썬 버전의 차이입니다. pip는Python2 버전 패키지 매니저이고, 참고로 파이썬2 버전은 2020년 1월 1일 자로 추가적인 업데이트 지원 종료했습니다. pip3은 Python3 버전 패키지 매니저이고, 현재 python2를 설치하여 사용하는 개발자는 없다는 가정하에서 python3용 pip3을 다루겠습니다.

라즈베리파이는 파이썬이 기본으로 설치되어 있고 GPIO등과 같은 추가 모듈이 실제로 많이 필요하므로 파이썬이 설치 될 때 pip3는 같이 설치됩니다. 따라서 설치할 필요는 없고 파이썬이나 pip를 삭제한 경우에만 재설치하면 됩니다.

02) pip 설치와 버젼 확인

Python 3.4 이후 버전을 사용하고 계시다면 pip가 내장되어 있어 따로 pip를 설치할 필요가 없습니다.

【형식】

```
pip --version
```

【주의!】

version 앞에 하이픈(-)이 2개입니다.

```
ojk@raspberrypi:~ $ pip --version
pip 20.3.4 from /usr/lib/python3/dist-packages/pip (python 3.9)
ojk@raspberrypi:~ $
```

현재 라즈베리파이에는 python 3.9와 pip 20.3.4가 설치되어 있음을 확인할 수 있습니다.
따라서 pip를 설치할 필요는 없습니다.

pip3를 실행시켜 보겠습니다. 다음에 Commands : 의 목록이 나옵니다. 이 Commands 중 하나를 선택하여 pip3를 사용합니다.

```
ojk@raspberrypi:~ $ pip3

Usage:
  pip3 <command> [options]

Commands:
  install                     Install packages.
  download                    Download packages.
  uninstall                   Uninstall packages.
  freeze                      Output installed packages in
requirements format.
  list                        List installed packages.
  show                        Show information about installed
packages.
  check                       Verify installed packages have
compatible dependencies.
  config                      Manage local and global
configuration.
  search                      Search PyPI for packages.
  cache                       Inspect and manage pip's wheel
cache.
  wheel                       Build wheels from your requirements.
  hash                        Compute hashes of package archives.
  completion                  A helper command used for command
completion.
  debug                       Show information useful for
debugging.
  help                        Show help for commands.

General Options:
  -h, --help                  Show help.
  --isolated                  Run pip in an isolated mode,
ignoring
                             environment variables and user
configuration.
  -v, --verbose               Give more output. Option is
additive, and can be
                             used up to 3 times.
  -V, --version               Show version and exit.
  -q, --quiet                 Give less output. Option is
additive, and can be
                             used up to 3 times (corresponding to
WARNING,
                             ERROR, and CRITICAL logging levels).
  --log <path>                Path to a verbose appending log.
  --no-input                  Disable prompting for input.
  --proxy <proxy>             Specify a proxy in the form
```

```
                              [user:passwd@]proxy.server:port.
  --retries <retries>         Maximum number of retries each
connection should
                              attempt (default 5 times).
  --timeout <sec>             Set the socket timeout (default 15
seconds).
  --exists-action <action>    Default action when a path already
exists:
                              (s)witch, (i)gnore, (w)ipe,
(b)ackup, (a)bort.
  --trusted-host <hostname>   Mark this host or host:port pair
as trusted,
                              even though it does not have valid
or any HTTPS.
  --cert <path>               Path to alternate CA bundle.
  --client-cert <path>        Path to SSL client certificate, a
single file
                              containing the private key and the
certificate
                              in PEM format.
  --cache-dir <dir>           Store the cache data in <dir>.
  --no-cache-dir              Disable the cache.
  --disable-pip-version-check
                              Don't periodically check PyPI to
determine
                              whether a new version of pip is
available for
                              download. Implied with --no-index.
  --no-color                  Suppress colored output.
  --no-python-version-warning
                              Silence deprecation warnings for
upcoming
                              unsupported Pythons.
  --use-feature <feature>     Enable new functionality, that may
be backward
                              incompatible.
  --use-deprecated <feature>  Enable deprecated functionality,
that will be
                              removed in the future.
ojk@raspberrypi:~ $
```

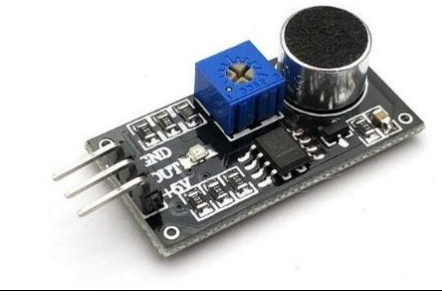
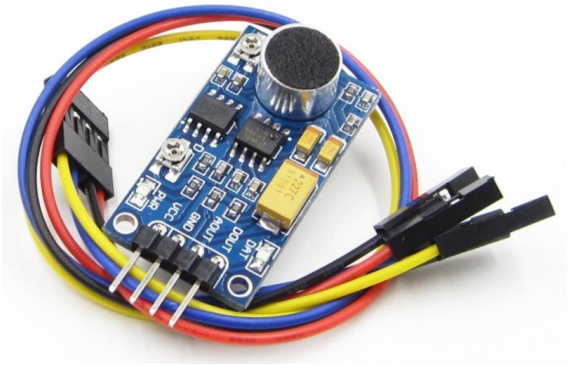
03) pip 업그레이드

다른 프로그래밍 언어도 마찬가지이지만, 특히 파이썬은 업데이트가 빈번하게 이루어 지기 때문에 자주 업그레이드 하는 것이 좋습니다.

```
pip install --upgrade pip
```

2. 센서

【01】소리 감지 센서

	
보급형 사운드 센서	고성능 사운드 센서

작동 전압 : 3.3V-5V 크기 : 32mm X 17mm X 8mm (세로 X 가로 X 세로) 주요 칩 : LM393 의 일렉트 릿 콘덴서 마이크 임계 값 조절 가능 음색이 가변 볼륨으로 설정된 임계 값보다 낮거나 높으면 모듈은 HIHG / LOW를 출력합니다

박수소리와 같은 소리가 감지되면 현재 플레이백하고 있는 음악을 다음 곡으로 옮긴다.

BCM 17 핀에 연결

```
import RPi.GPIO as GPIO
import time
import os

#GPIO Setup
channel=17
GPIO.setmode(GPIO.BCM)
GPIO.setup(channel, GPIO.IN)
print "No Sound Detected!!"
##os.system("audacious --play-pause&")

def callback(channel):
   if GPIO.input(channel):
     print "Sound Detected!!"
     print "The audio will be plying the next music"
```

```
      os.system("audacious --fwd &")

   else:
     print "Sound Detected!!"
     os.system("audacious --fwd &")
     print "The audio will be plying the next music"

GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=300)
GPIO.add_event_callback(channel, callback)

while True:
   time.sleep(1)
```

【02】인체감지 센서

SR505 라는 인체감지 센서입니다. 이 센서는 움직임이 없으면 감지하지 못합니다.

연결

표시	-	표시없음	+
연결	GND	데이터 입력 핀	5V

HC-SR505 Mini Infrared PIR Motion Sensor Precise Infrared Detector Module

Specifications:
Operating Voltage Range: DC4.5-20V
Static current: <60uA
Output level: High 3.3V / Low 0V
Trigger: repeatable trigger (default)
Delay time: Default 8S + -30% (can be customized range of a few tenths - tens of minutes)
PCB Dimensions: 10 * 23mm
Induction angle:
<100 degree cone angle
Induction distance: 3 meters
Working temperature:
-20 - +80 degrees
Sensor Lens Dimensions:
Diameter: 10mm (default)

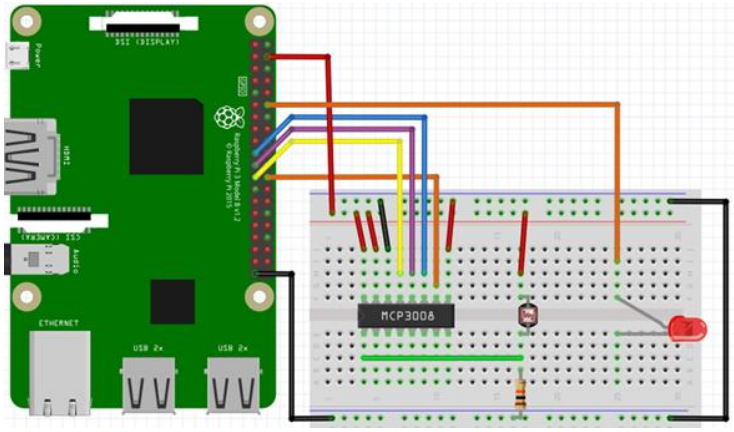
작동 전압 범위 : DC4.5-20V
정적 전류 : <60uA
출력 레벨 : High 3.3V / Low 0V
트리거 : 반복 가능한 트리거 (기본값)
지연 시간 : 기본 8S + -30 % (수십 분에서 수십 분까지 사용자 정의 가능)
PCB 크기 : 10 * 23mm
유도 각 : <100 도의 콘 각
유도 거리 : 3 미터
작동 온도 : -20 - +80도
센서 렌즈 크기 : 직경 : 10mm (기본값)선

BCM 27 핀으로 인체를 입력받습니다.

사람을 감지하면 음악을 300초 동안 플레이백하고 300초가 지나면 음악을 멈추고 1초마다 인체를 감지하여 인체를 감지하면 다시 음악을 300초 동안 플레이백하는 프로그램입니다.

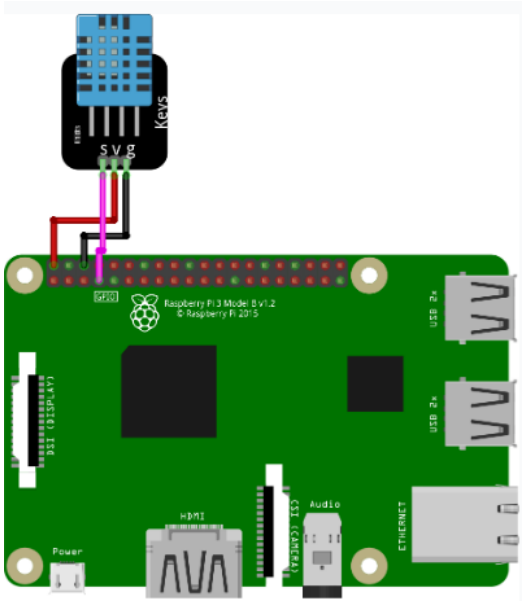
```
import RPi.GPIO as GPIO
import time
import os

#GPIO Setup
channel=27
GPIO.setmode(GPIO.BCM)
GPIO.setup(channel, GPIO.IN)


tt=0
while True:
   humanLevel=GPIO.input(channel)
   #print "Detecting Human..."

   if humanLevel==1:
     t=0
     print "Human is detected...", t, "Second"
     print "The audio will be playing..."
     os.system("audacious --play-pause &")
     os.system("audtool --mainwin-show off &")

     while t<301:
            print "Human is detected...", t, "Second"
            time.sleep(1)
            t=t+1
     tt=0
     #time.sleep(300)

     print "The audio will be pausing..."
     os.system("audacious --play-pause & ")
     time.sleep(3)

   else :
     print "Human is not detected...", tt, "Second"

   tt=tt+1

   time.sleep(1)
```

【03】조도감지 센서




```
import pymysql as ps
import spidevRead as sr
import time
conn = ps.connect(host='localhost',user='root',
    passwd='1234', db='test')
while True:
    curs = conn.cursor()
    readData = sr.analog_read(0)
    sql=f'insert into sensordb(sensing) values ({readData})'
    curs.execute(sql)
    conn.commit()
    time.sleep(5)
```

【04】온습도계



sudo python3 -m pip install --upgrade pip setuptools wheel

sudo pip3 install Adafruit_DHT

nano temp.py

temp.py----------

```
import datetime
import Adafruit_DHT as dht
import time

sensor=dht.DHT11
pin=4
```

```
h,t = dht.read_retry(sensor,pin)



if h is not None and t is not None:

while(1):

now = time.localtime()

nowtime = ("%04d/%02d/%02d %02d:%02d:%02d" % (now.tm_year, now.tm_mon, now.tm_mday, now.tm_hour, now.tm_min, now.tm_sec))

h,t = dht.read_retry(sensor,pin)

data = '{} {}% {}*C'.format(nowtime, h, t)

print (data)

time.sleep(1)

else:

print('Failed to get Data')
```

다음은 마리아 DB설정

```
sudo apt-get install rdate

date

raspi-config // 서울로 세계시간설정


sudo apt-get install mariadb-server


sudo su

mysql // root계정에서 DB접속(암호 필요없음)


create schema rpitemp; // 접속자가 사용할 DB이름 => 테이블은 HeidiSQL에서 작성

show schemas;

grant all on rpitemp.* to 사용자접속계정@'%' identified by '*******';
```

flush privileges;

show grants for 사용자접속계정; // 접속계정의 권한확인

show schemas;

exit // DB 밖으로 나오는 명령어

sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf

=> 외부 접속 허용 127.0.0.1 => 주석처리

3306포트 열기

sudo apt-get install ufw

sudo ufw status

sudo ufw allow 3306

sudo ufw allow 22

sudo ufw enable

sudo ufw status

sudo service mysql restart

import mysql.connector;

import Adafruit_DHT as dht

from datetime import datetime

import time

Maria = mysql.connector.connect(host="localhost", user="alex", passwd="*******
database="rpitemp");

Cursor = Maria.cursor();

```
sensor=dht.DHT11

pin=4


Humi, Temp = dht.read_retry(sensor,pin)


if Humi is not None and Temp is not None:

while(1):

Humi, Temp = dht.read_retry(sensor,pin)

d = datetime.today().strftime('%Y-%m-%d %H:%M:%S')

Query = "INSERT INTO temp VALUES(%s,%s,%s)";

Values = [

(d,Humi,Temp)

];

print(d,Humi,Temp)

Cursor.executemany(Query,Values);

Maria.commit();

time.sleep(10);


else:

print('Failed to get Data')
```

제3편 라즈베리파이 프로젝트

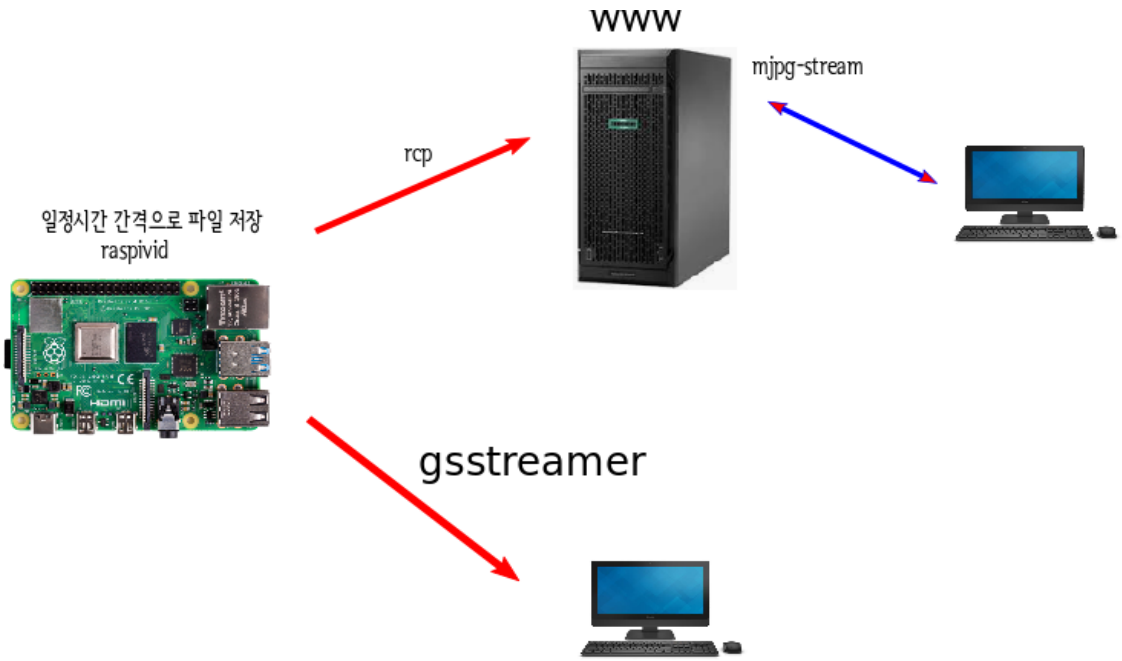
I. 스마트 팜

1. 센서 입력

파이썬 프로그래밍

C 언어 프로그래밍

II. CCTV 만들기



raspivid 프로그램을 사용하여 카메라를 통하여 촬영한 영상을 일정한 간격(10분)으로 저장하여 mp4 형식으로 변환한 다음에 웹서버에 전송한다. 전송한 후에 원래 h264, mp4 파일을 삭제한다.

녹화된 파일은 웹서버에 저장되고 웹서버를 통하여 스트림된다.

파이에서 PC로 gsstream을 사용하여 실시간으로 스트림한다.

라즈베리파이에서 카메라를 enable하고, PiCam을 연결한 상태에서 진행한다.

참고로 이 기능은 라즈비안 OS version이 Jessie 버전일 경우 동작한다. 현재(2018.4월) 라즈비안 OS는 Stretch로서 이 OS에서는 동작을 하지 않는다. 따라서 라즈비안 OS를 Jessie 버전으로 설치하여 동작을 확인할 수 있다.

1. Gstreamer 0.10 패키지 설치

$ sudo apt-get install libglib2.0-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev gstreamer-tools gstreamer0.10-plugins-base gstreamer0.10-plugins-good gstreamer0.10-plugins-bad gstreamer-plugins-ugly

plugins 가 에러가 나는 경우에는 빼고 설치

2.Gstreamer 0.10 서버 & 스크립트 다운로드

$ wget http://gstreamer.freedesktop.org/src/gst-rtsp/gst-rtsp-0.10.8.tar.bz2

$ bzip2 -d gst-rtsp-0.10.8.tar.bz2

$ tar xvf gst-rtsp-0.10.8.tar

$ cd gst-rtsp-0.10.8

3. 빌드

$ ./configure

$ make

4. 서버 실행 (스트리밍 시작)

$ cd examples

$ raspivid -t 0 -h 720 -w 1280 -fps 25 -b 2000000 -vf -hf -n -o - | gst-launch -v fdsrc ! h264parse ! gdppay ! tcpserversink host=127.0.0.1 port=5000 | ./test-launch "(tcpclientsrc host=127.0.0.1 port=5000 ! gdpdepay ! avdec_h264 ! rtph264pay name=pay0 pt=96)"

수행되는 process 확인 : $ ps -ef | more -> pi 계정으로 다음과 같이 3개의 process가 실행된다.

```
pi        2211  1928  0 18:30 pts/0    00:00:00 raspivid -t 0 -h 720 -w 1280 -fp
s 25 -b 2000000 -vf -hf -n -o -
pi        2212  1928  2 18:30 pts/0    00:00:00 /usr/bin/gst-launch-0.10 -v fdsr
c ! h264parse ! gdppay ! tcpserversink host=127.0.0.1 port=5000
pi        2213  1928  0 18:30 pts/0    00:00:00 /home/pi/gst-rtsp-0.10.8/example
s/.libs/lt-test-launch ( tcpclientsrc host=127.0.0.1 port=5000 ! gdpdepay ! avde
c_h264 ! rtph264pay name=pay0 pt=96 )
```

수행되는 네트워크 관련 상태를 확인해 본다.

$ netstat -an | more

- port 8554, 5000가 동작함을 다음과 같이 확인할 수 있다.

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:8554            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:5910          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3350          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:3389            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:5000            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:42492         127.0.0.1:5910          ESTABLISHED
tcp        0      0 127.0.0.1:5910          127.0.0.1:42492         ESTABLISHED
tcp        0      0 192.168.0.150:3389      192.168.0.83:5009       ESTABLISHED
tcp6       0      0 :::8080                 :::*                    LISTEN
tcp6       0      0 :::8000                 :::*                    LISTEN
tcp6       0      0 127.0.0.1:8005          :::*                    LISTEN
tcp6       0      0 :::8009                 :::*                    LISTEN
```

5. 확인

- PC에서 다음 팟 플레이어를 설치하고, 실행한다.

- Control-U를 누르고 주소 열기를 실행하고
rtsp://라즈베리파이IPaddress:8554/test

영상 프레임을 빠르게

화면이 처음에는 2 frame으로 매우 답답하게 느껴집니다. 아래와 같이 조정하면 조금이라도 나아지니 참고 바랍니다.

sudo vi /etc/motion/motion.conf

Maximum framerate for stream streams (default: 1)
stream_maxrate 10

Maximum number of frames to be captured per second.
Valid range: 2-100. Default: 100 (almost no limit).
framerate 10

여러가지 값으로 조정해 봤지만 10 이상은 차이가 없어 보입니다.

오늘은 gstreamer로 전송된 영상을 mp4 파일로 저장하는 방법에 대해 써보도록 하겠습니다. 아마 저번 글 (라즈베리파이 RC카 제작기 6 - gstreamer를 이용한 라즈베리파이-PC간 전송 영상을 사진으로 캡쳐하기 http://blog.naver.com/jedijaja/220868759001 참조) 의 쌍둥이 글이 되지 않을까 싶은데요. (아마 비슷한 말이 엄청 많을 겁니다. ^^)

웬지 동영상 저장이 사진 저장보다 더 있어 보일 것 같긴 하지만, gstreamer로 구현하는 부분에 있어서는 그냥 script 하나 차이 정도 밖에 되지 않습니다.

1. 송신 영상 저장의 필요성 ?

UDP나 TCP로 전송되어져 온 영상을 저장하고 싶을 때가 있습니다.(뭐 이유가 필요하겠습니까 ? ^^) 단지 제가 만든 앱 (Camera Voice Launcher http://blog.naver.com/jedijaja/220889903466 참조)을 올리면서 유사한 앱들이 있나 보았더니 유독 "Spy Camera" 라는 제목의 앱들이 많이 있더군요. 제 앱처럼 원격으로 몰래 영상을 보는 앱이겠거니 하고 봤더니, 그게 아니라, 화면을 꺼 논 상태에서 동영상 녹화를 할 수 있는 단순한 앱이더군요. (누가 이런 걸 그렇게 쓰는 지는 잘 모르겠습니다. ^^;) 아마 기존 카메라 기능이 개인 정보 보호 차원에서 화면이 꺼진 상태에서는 동작을 못하게 해 놓았는데 이를 보완하기 위해 카메라 스트리밍 정보를 캡쳐하여 동영상을 만드는 앱일 것 같습니다.

여튼, 동영상을 저장할 수 있다는 것도 큰 옵션 중에 하나라 생각되어 Camera Voice Launcher에 위와 같은 Spy Camera 녹화 기능을 추가하기 위해 gstreamer로 mp4 파일을 저장하는 방법을 찾아 보았습니다. (이는 Camera Voice Launcher에 네 번째 디폴트 기능으로 [Spy Rec]에 구현을 해놓았습니다.)

물론, 좀 뒤져보니 인터넷에 좀 있더군요 ^^.

2. gstreamer로 mp4 파일 만들기

x264enc, mp4mux를 이용한 PC에서의 카메라 영상 캡쳐하기

먼저 gstreamer가 깔려 있는 PC의 웹캠에서 오는 영상에서 mp4 파일을 만드는 방법은 아래와 같습니다.

```
gst-launch-1.0 -e ksvideosrc device-name="HD 720P Webcam"  ! video/x-raw, width=640, height=480,
framerate=30/1  ! tee name=t t. ! queue ! autovideosink sync=false t. ! queue !  videoconvert ! x264enc tune=zerolatency ! mp4mux ! filesink location=rec.mp4
```

여기서는 PC에 전술한 웹캠을 달고 그 웹캠의 영상을 한 개의 mp4 파일로 만드는 script 입니다. 위와 같이 mp4mux 명령어와 filesink 명령어를 이용하여 간단하게 영상을 mp4 파일로 저장 할 수 있습니다. mp4mux를 쓰기전에 x264enc라는 명령어를 사용하는데 mp4 파일 포멧이 영상 전송 표준인 h264를 기반으로 한 모양인지 x264enc라는 유사 h264로 엔코딩한 후에

mp4로 만드는 것 같습니다. (언제나처럼 일단 되면 더는 자세히 알려고 하지 않습니다. ^^)

omxh264enc, mp4mux를 이용한 라즈베리파이에서의 카메라 영상 캡쳐하기

gstreamer가 깔려 있는 라즈베리파이의 picam에서 오는 영상에서 mp4 파일을 만드는 방법은 아래와 같습니다.

raspivid -n -t 0 -vf -h 240 -w 320 -fps 25 -b 2000000 -o - | gst-launch-1.0 -e fdsrc ! h264parse ! avdec_h264 ! tee name=t t. ! queue ! autovideosink sync=false t. ! queue ! decodebin ! videoconvert ! omxh264enc ! h264parse ! mp4mux ! filesink location=rec.mp4

여기서는 raspivid를 이용해서 일단 picam을 돌리고 gstreamer의 fdsrc 명령어로 영상을 볼 때 tee 명령으로 분기하여 mp4로 저장을 하게 됩니다. 이 때 분기 후 decodebin 으로 h264 포멧 변경을 위해 디코딩을 하고 omxh264enc와 mp4mux를 이용해서 동영상을 저장하면 됩니다. PC와 뒤에서 설명할 안드로이드폰에서는 유사 h264 포멧을 x264enc로 만들어 줬는데 라즈베리파이에서는 이 걸 omxh264enc가 합니다. 아마도 리눅스에서의 특성이 아닌가 합니다. 언제나 처럼 일단 되니 자세히는 보지 않습니다. ^^
또 제가 라즈베리파이2를 사용하고 동영상 플레이를 vlc 를 이용하는데 위와 같이 저장한 영상이 라즈베리파이에서는 동작을 안하더군요. 하지만 PC로 옮겨 플레이하니 잘 되었습니다. 이건 나중에 라즈베리파이에 다른 플레이어를 깔아봐야 할 듯 합니다.

-e flag로 End-of-Stream 날리기

한 장의 화면만 캡쳐하는 jpg 저장과 달리 동영상 저장에서 언제까지 동영상을 저장할 지를 gstreamer의 pipeline에 꼭 알려 주어야 합니다. 그래야 동영상을 만들 때 완성을 시킬 수 있는 것 같드라구요. 이를 위해 위의 script에 보면 gst-launch-1.0 명령어 뒤에 -e flag를 적어 주었습니다. 이 플래그의 의미는 "이 스크립트를 종료시키는 시점에 강제로 end-of-stream 신호를 파이프라인에 전달해 주어라"라는 의미입니다. 이 플래그를 쓰지 않고 위와 같은 mp4mux 명령어를 실행시키면 rec.mp4 라는 파일이 만들어 지지만 "실행할 수 없는 파일입니다."라는 멘트와 함께 실행이 되지 않습니다.

3. 안드로이드에서 카메라 영상 캡쳐하기

안드로이드는 상황이 조금 다른데요. 위와 같이 gst-launch-1.0 명령어를 사용할 수 없고 바로 pipeline을 만들어야 하기 때문에 -e flag를 적용할 수가 없었습니다. 그래서 또 인터넷을 찾아보니 역시 있더군요.

http://blog.naver.com/jedijaja/220636966117 등에서 공개한 소스의 tutorial-4.c 의 gst_native_finalize() 함수의 초입에

gst_element_send_event (pipeline, gst_event_new_eos ());

```
/* wait for EOS message on the pipeline bus */
msg = gst_bus_timed_pop_filtered (GST_ELEMENT_BUS (pipeline),
GST_MESSAGE_EOS | GST_MESSAGE_ERROR, GST_CLOCK_TIME_NONE);
/* should check if we got an error message here or an eos */
gst_element_set_state (pipeline, GST_STATE_NULL);
```

위와 같이 강제로 EOS 신호를 주는 방법을 사용하더군요.

그런데..

위와 같이 적용을 했더니 저장하는 동영상의 길이나 저장하는 형식에 따라서 어쩔땐 되고 어쩔땐 안되는 식으로 너무 불안하게 작동을 해서 도저히 사용을 할 수가 없었습니다.

그래서 임시 방편으로 아래와 같이 사용하였습니다.

```
ahcsrc device=0 ! video/x-raw, width=320, heigh=240 ! videobalance contrast=1.0 brightness=0.1 saturation=1.5 hue=0.1 ! videoflip method=clockwise ! tee name=t t. ! queue ! autovideosink sync=false t. ! queue ! x264enc tune=zerolatency ! mp4mux fragment-duration=10 ! filesink location=/sdcard/test.mp4
```

여기서는 mp4mux의 인자 중 하나인 fragment-duration에 특정 숫자(ms)를 적어 동영상을 파편으로 저장하는 방식입니다. 이리하면 따로 EOS(End-of-Stream) 신호를 주지 않아도 fragment-duration만 지나면 mp4 파일이 완성이 되므로 위와 같은 문제를 해결할 수 있습니다. 단지 파편의 시간만큼 끊기는 듯한 영상이 저장되므로 연속 동작이 필요한 동영상에는 적합하지 않습니다. 하지만 뭐 아쉬운대로 사용할 수는 있더군요.
(안드로이드의 동영상 저장을 테스트해보시고 싶으신 분은 안드로이드용 Gstreamer Launcher v2.0 (http://blog.naver.com/jedijaja/220889903466 참조) 를 보시고 앱을 다운 받으셔서 [Spy Rec]이나 [Repository]에 있는 여러 예제를 실행시키시면 확인하실 수 있습니다.)

4. 전송된 영상 캡쳐하기

udp로 전송된 영상 PC에서 캡쳐하기

먼저 안드로이드폰에서 Camera Voice Launcher v2.0으로 아래와 같이 udp로 영상을 보낸다고 합시다.

```
ahcsrc device=1 ! video/x-raw, width=320, height=240 ! videobalance contrast=1.0 brightness=0.1 saturation=1.5 hue=0.1 ! videoconvert ! x264enc tune=zerolatency ! rtph264pay !
```

udpsink host=192.168.0.100 port=5000

이 때 수신하는 PC에서 아래와 같이 하면 됩니다.

gst-launch-1.0 -e udpsrc port=5000 ! application/x-rtp ! rtph264depay ! h264parse ! avdec_h264
! tee name=t t. ! queue ! autovideosink sync=false t. ! queue ! x264enc tune=zerolatency ! mp4mux
! filesink location=rec.mp4

tee 명령어로 영상을 분기해서 화면에 하나 뿌리고 x264enc와 mp4mux를 이용해서 mp4 파일로 받으면 됩니다.

udp로 전송된 영상 라즈베리파이에서 캡쳐하기

다시 안드로이드폰에서 Camera Voice Launcher v2.0으로 위에서 script를 아래와 같이 다시 udp로 영상을 보내고

ahcsrc device=1 ! video/x-raw, width=320, height=240 ! videobalance contrast=1.0 brightness=0.1 saturation=1.5 hue=0.1 ! videoconvert ! x264enc tune=zerolatency ! rtph264pay ! udpsink host=192.168.0.100 port=5000

이 때 수신하는 라즈베리파이에서는 아래와 같이 하면 됩니다.

gst-launch-1.0 -e udpsrc port=5000 ! application/x-rtp ! rtph264depay ! h264parse ! avdec_h264
! tee name=t t. ! queue ! autovideosink sync=false t. ! queue ! decodebin ! videoconvert ! omxh264enc ! h264parse ! mp4mux ! filesink location=rec.mp4

PC에서와 마찬가지로 tee 명령어로 영상을 분기해서 화면에 하나 뿌리고 omxh264enc와 mp4mux를 이용해서 mp4 파일로 받으면 됩니다

udp로 전송된 영상 안드로이드에서 캡쳐하기

다시 안드로이드폰에서 Camera Voice Launcher v2.0으로 아래와 같이 udp로 영상을 보내고

ahcsrc device=1 ! video/x-raw, width=320, height=240 ! videobalance contrast=1.0 brightness=0.1 saturation=1.5 hue=0.1 ! videoconvert ! x264enc tune=zerolatency ! rtph264pay ! udpsink host=192.168.0.100 port=5000

이 때 수신하는 안드로이드폰에서는 아래와 같이 하면 됩니다.

udpsrc port=5000 ! application/x-rtp ! rtph264depay ! h264parse ! tee name=t t. ! queue ! avdec_h264
! videoflip method=counterclockwise ! autovideosink sync=false t. ! queue ! mp4mux fragment-duration=10 ! filesink location=/sdcard/rec.mp4

여기서는 videoflip 명령어로 화면을 90도 돌려 준 후에 tee 로 분기된 영상 신호를 x264enc 를 거치지 않고 바로 mp4mux로 연결했는데 위에 script에서 보시다시피 tee 명령어 앞에 이미 h264parse를 통해서 h264 포멧으로 만들었기 때문에 x264enc는 따로 필요 없게 되는 거구요. 앞서 언급했듯이 mp4mux 명령에서 fragment-duration을 반드시 선언해 주어야 합니다.
(위의 명령어는 Camera Voice Launcher v2.0 앱의 repository에 모두 있으며 script를 받아서 ip와 port만 수정하여 사용하시면 됩니다.)

5. 안드로이드에서의 테스트를 위한 팁

위의 모든 테스트는 이미 실행해 보고 되는 것만 올린 것이니 한 번씩 해보시면 될 것 같구요. 안드로이드는 제가 앞 서 올린 여러 글들과 소스로 짜셔도 되지만 앞서 말씀드린 것처럼 안드로이드 앱 Camera Voice Launcher (https://play.google.com/store/apps/details?id=com.homecompany.gstreamer_launcher) 로 테스트 해 보실 수 있고 앱에 대한 자세한 설명은 제 블로그 글 (http://blog.naver.com/jedijaja/220889903466) 에서 확인하실 수 있습니다.

6. 글을 마치며

gstreamer 관련 일은 일단 이 정도로 대강 마무리를 지으려고 합니다. 사실 동영상 저장도 Camera Voice Launcher 업그레이드를 하면서 진행한 거라 잊어버리기 전에 한 번 정리를 하려 한 것 뿐이라 ..

전에 하다가 만 Android Segwey (안드로이드폰을 장착한 밸런싱 로봇)을 다시 시작했는데 현재 MPU-5060으로 밸런싱을 잡는 초기 단계를 진행 중입니다. 안드로이드 RC카를 해체하여 밸런싱 로봇에 맞게 좀더 변형했는데 생각보다 깔끔하게 나와서 나름 만족하고 있고 일단 MPU-5060의 통신과 간단하게 밸런스를 잡는 건 성공을 했는데 조금 더 다듬게 되면 AndroidSegwey 시리즈로 다시 글을 써보도록 하겠습니다.

III. 오디오

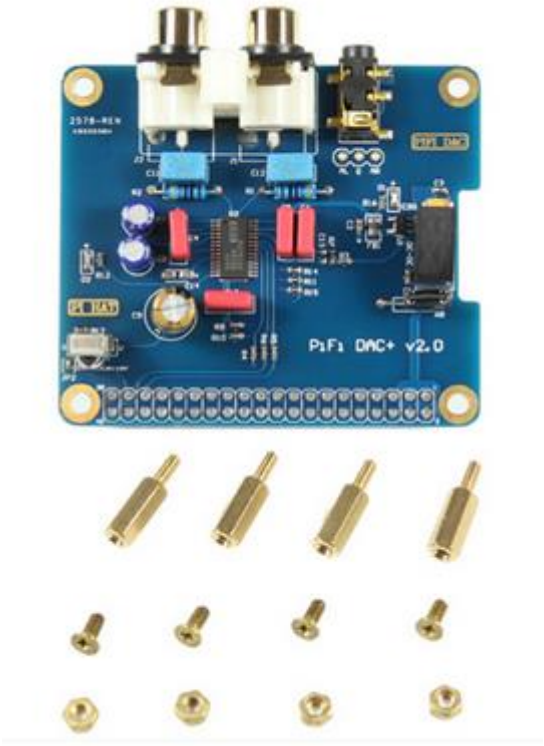
1. 음악 듣기

음악과 동영상을 플레이하는 프로그램은 vlc와 smplayer가 많이 사용하고 성능도 좋으므로 vlc나 smplayer를 사용합니다.

vlc 다운 받기
$ sudo apt-get install vlc

smplayer 다운 받기
$ sudo apt-get install smplayer

DAC 설치



$ sudo vi /boot/config.txt

아래 두 줄을 찾아서 #을 제거 후
#device_tree_param=i2s=on
#dtoverlay=hifiberry-dac

How to set up sound through DAC+?
sudo i2cdetect -y 1
0 1 2 3 4 5 6 7 8 9 a b c d e f
00: -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- ------
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- 4d -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --

pi@raspberrypi ~ $ aplay -l
aplay: device_list:252: no soundcards found...

Please provide a link to the configuration instructions.
The latest version Raspbian

/etc/modules
bcm2708_dmaengine // built-in sound card
snd_soc_bcm2708_i2 s // built-in sound card
snd_soc_pcm512x
snd_soc_hifiberry_dacplus

/etc/asound.conf
pcm.!default {
type hw card 0
}
ctl.!default {
type hw card 0
}

/boot/config.txt
dtoverlay=hifiberry-dacplus

PiFi DAC v2.0 install(나의 설정)

DAC를 모듈로 설치

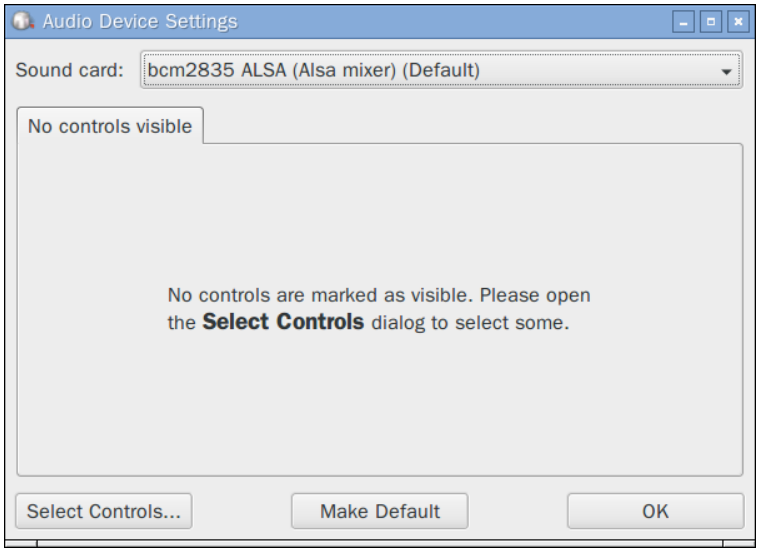
기존 /etc/modules
i2c-dev

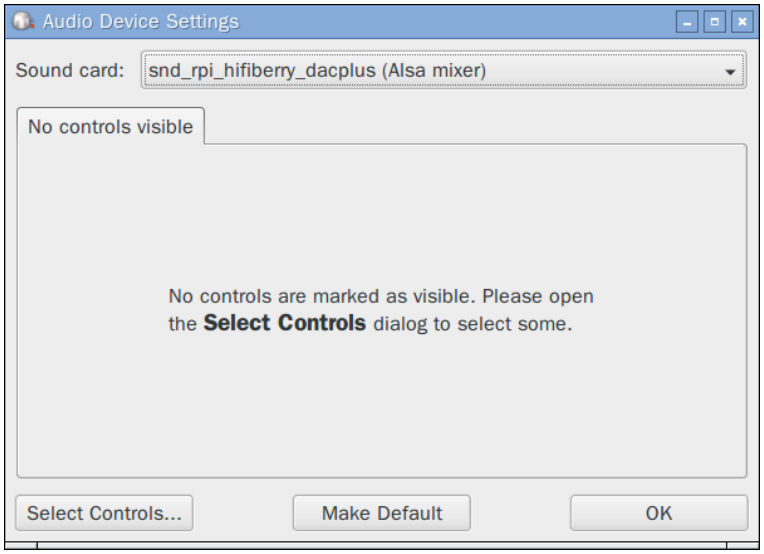
새로 생성한 /etc/modules
i2c-dev
#i2s-dev
#pcm5122_dmaengine
snd_soc_pcm512x
snd_soc_hifiberry_dacplus

처음 생성한 /etc/asound.conf
pcm.!default{
type hw card 1 //card와 1사이에 space
}
ctl.!default{
type hw card 1
}

/boot/config.txt 변경
# Uncomment some or all of these to enable the optional hardware interfaces
dtparam=i2c_arm=on
dtparam=i2s=on
dtparam=spi=on

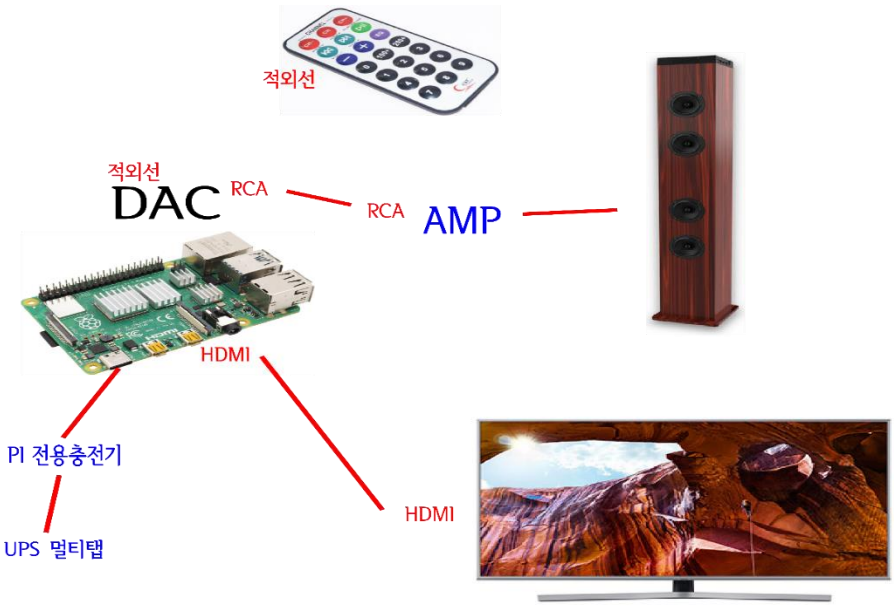
Enable pi fi Dac pcm5122 DAC chip
#device_tree_param=i2s=on
dtoverlay=hifiberry-dacplus


Audio Device Settings
Sound card:
bcm2835 ALSA (Alsa mixer) (Default)
No controls visible
No controls are marked as visible. Please open the Select Controls dialog to select some.
Select Controls...
Make Default
OK


Audio Device Settings
Sound card:
snd_rpi_hifiberry_dacplus (Alsa mixer)
No controls visible
No controls are marked as visible. Please open the Select Controls dialog to select some.
Select Controls...
Make Default
OK

vlc 환경 설정
[메뉴]-[도구]-[환경설정]-[오디오]에서
출력 모듈을 [ALSA 오디오 출력]으로 지정한다.

2. 리모콘 설정



【01】What is LIRC ?

What is LIRC ?

LIRC is a package that allows you to decode and send infra-red signals of many (but not all) commonly used remote controls.

Recent linux kernels makes it possible to use some IR remote controls as regular input devices. Sometimes this makes LIRC redundant. However, LIRC offers more flexibility and functionality and is still the right tool in a lot of scenarios.

The most important part of LIRC is the lircd daemon which decodes IR signals received by the device drivers and provides the information on a socket. It also accepts commands for IR signals to be sent if the hardware supports this.

The user space applications allows you to control your computer with your remote control. You can send X11 events to applications, start programs and much more on just one button press. The possible applications are obvious: Infra-red mouse, remote control for your TV tuner card or CD-ROM, shutdown by remote, program your VCR and/or satellite tuner with your computer, etc. Using lirc on Raspberry Pie is quite popular these days.

Supported remote controls

There are some config files for remote controls at the remotes database. This is about 2500 devices and counting. These devices should work with the general drivers or (if it lacks timing info) the driver used to create them.
If you can't find your remote control here it does not mean that your remote control is not supported. It's just that there is no config file for it yet. All remote controls that are supported by learning remote controls i.e., almost any, should also work with LIRC.

Supported capture devices

Besides a remote control you also need a capture device to read the data from the remote. Former versions focussed on home-brew capture hardware connected to the serial or parallel port. Descriptions how to build such hardware can be found here. Current versions of LIRC also support a broad range of other hardware. As a starter, you can use the kernel built-in support for many USB dongles and similar. Besides this LIRC supports basically any conceivable way to capture your data including serial devices, parallel ports, sound input etc. You can see the complete list in the left pane.

【02】Installation instructions

This chapter describes how to build and install LIRC. Note that LIRC is packaged for all major linux distributions. If you just want to use lirc, you should be able to install it like any other package. This way, you don't have to look at the dependencies, build and installation description found here.

The ./configure script is the ultimate source as to what libraries and tools LIRC requires. The list here is not complete in any way, being focused on things to install before building.

01) Dependencies

Mandatory dependencies

There are a few mandatory dependencies, all of which packaged on most (all?) Linux distributions.

- Building directly from the git source tree requires autoconf,automake and libtool.
- Compilation and linking requires the GNU toolchain including make, gcc, g++ and ld.
- python3
- Building requires modinfo, often in the package kmod. Without modinfo, all kernel drivers are excluded from the build.
- Building requires pkg-config.
- Building requires kernel headers, often in the kernel-headers package.
- xsltproc

Optional dependencies

If these are missing, ./configure can cope with it and still build LIRC in a more or less limited

way.

- Several scripts need the python3 PyYaml at http://pyyaml.org/wiki/PyYAML. Some distributions includes a python3-yaml package. There is also a pypi package at https://pypi.python.org/pypi/PyYAML. Building without this is possible but not recommended.
- Generating the HTML manpages requires man2html.
- Generating the API documentation requires Doxygen.
- The lirc-setup GUI configuration tool needs python3-gi and thus also the Gtk libs and icons. These are not required for the build, though.
- Building the X11 GUI tools like irxevent and xmode2 requires the X11 header files.
- The audio drivers needs the alsa and portaudio libs (libraries and headers).
- The ftdi and ftdix drivers require libftdi from http://www.intra2net.com/de/produkte/opensource/ftdi/

02) Kernel

As of 0.9.0+, lirc uses the kernel modules from the kernel. Some of these are formerly lirc modules which are now part of the kernel. Thus, building lirc does not involve building any kernel modules (as it used to).

Some of the former lirc modules are part of the official kernel and should be available on any reasonably updated system. However, some are in the staging area; if they are part of your kernel depends on the distro you use (unless of course if you compile your own kernel).

There are example and test kernel modules in the drivers/ directory. None of these are required for regular LIRC use, but they are supposed to be helpful while testing or writing drivers.

03) Compile and install

Since 0.9.1+ , lirc loads drivers dynamically. This means that that the build system is redesigned to always build all drivers. The former setup.sh script is dropped in favor of a standard ./configure, make, make install sequence.

When building directly from git, a first required step to create ./configure and some other files is

```
./autogen.sh
```

Whether using git sources or just using a distributed tarball, the next steps are the canonical

```
./configure
make
sudo make install
```

Notes:
When running the configure script, please pay attention at its output. Specifically, at the very end it prints a list of the enabled functionality.

Running configure without options will install lirc in /usr/local according to GNU standards. However, many examples in this manual as well as other documentation are assuming that ./configure ran with the --prefix=/usr option, installing in /etc/lirc, /usr/bin etc.

The build system supports VPATH builds which does not clutter the source tree. To use this do something like:

```
mkdir _build;
cd _build;
../configure
make
sudo make install
```

04) Checking the build

Since the dynamic drivers are not linked during the build, it's recommended to check that the expected drivers are built and can be loaded using

```
cd tools; ./lirc-lsplugins -U ../plugins/.libs
```

Is some cases, lirc-lsplugins will crash on missing libraries e. g., liblirc.so.0. If so, you need to add the path where lirc installs it's libraries (by default, /usr/local/lib) to the runtime linker path. Refer to generic ld.so(1) documentation.

With working paths lirc-lsplugins will create a list of the all drivers available, and also possible link errors not revealed during the build. *lirc-lsplugins* has a -h option providing help, and *nroff -man ../doc/man/lirc-lsplugins.1 | more* provides more complete info.

05) Configuration

Formerly, lirc was configured during build where the setup.sh script was used to select driver, configuration file, etc. Also, there was little support for starting and running the services from boot. From 0.9.1+ the configuration is instead done after the build. The configuration steps for the main lircd program involves:

- In some cases e. g., serial devices setting up kernel module options in /etc/modprobe.d or using udev rules.
- Selecting the driver and kernel device for your capture device.
- Selecting configuration file for your remote.
- Configuring and using systemd to run the services.
- Creating lircrc files for your applications.
- This is described in the configuration guide.

06) Uninstall

Remove the installed artifacts:

```
make uninstall
```

Remove the config files, if you don't need them anymore:

```
rm -rf /etc/lirc/
```

```
rm -f ~/.config/lircrc
```

You might also want to check /etc/modprobe.d for left-overs.

07) Updating from older versions

The NEWS file describes the changes since last version. Normally, upgrading from a previous version should not be too painful. However, if you have to update a really old version it's probably better to make a fresh install.

【03】Configuration guide

01) About this guide

This guide tries to describe the basic configuration steps for commonly used hardware. It's focused on the basic usage scenario to get the remote up and running, the more advanced features are not covered. This includes **irexec, lircmd, ir blasting** and the TCP/IP-based remote features.

02) Why should I use LIRC?

Recent Linux kernels have built-in support for IR remotes. Using that, pressing an up-arrow on the remote works the same way as pressing the up-arrow on a keyboard. This is a modern "just works" solution. On the other hand, LIRC is an old style linux application which can be tweaked to do almost anything, but is tricky to setup. So, why would you use LIRC?

- **You might have a remote which is supported by LIRC but not the kernel.**
- **If you have a remote which isn't supported at all, LIRC is probably your best bet to get it running.**
- **You might be on a non-Linux platform supporting lirc e. g., MacOS.**
- **You might have an application which is more or less designed to use LIRC.**
- **You might need LIRC's capabilities e. g., modes where a single remote button can be teached to deliver different keys to the application. Handling input to multiple program is also easier with lirc**
- **You might want to send IR signals to other devices (IR blasting).**
- **You might want to use lirc's applications e. g., irexec(1) which can run arbitrary commands in parallel with an application such as mythtv or kodi.**

So, while the kernel built-in handling works out of the box in many cases, there are still scenarios when LIRC is the right tool.

03) The default configuration

From 0.9.4+ LIRC is distributed with a default configuration based on the devinput driver. This should work out of the box with the following limitations:

- **There must be exactly one capture device supported by the kernel**

- **The remote(s) used must be supported by the kernel.**
- **There is no need to do IR blasting (i. e., to send IR data).**

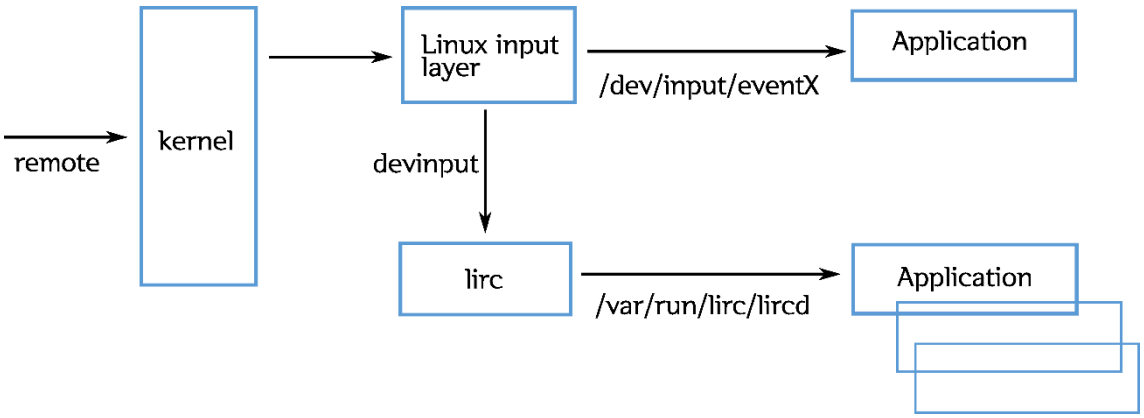
The easy way to check is to try the remotes without lircd running. If it works this way, it should also work using lircd.
To check the number of supported devices run

```
ls /sys/class/rc
```

This should list a single entry rc0.

If you want to use the default configuration you should start and enable the lircd service and possibly define lircrc files for your applications. However, you can use the lirc_options.conf file as-is. See systemd-setup

04) Overall Configuration Decisions



LIRC can be run together with the kernel in different ways. You need to decide on a general approach.

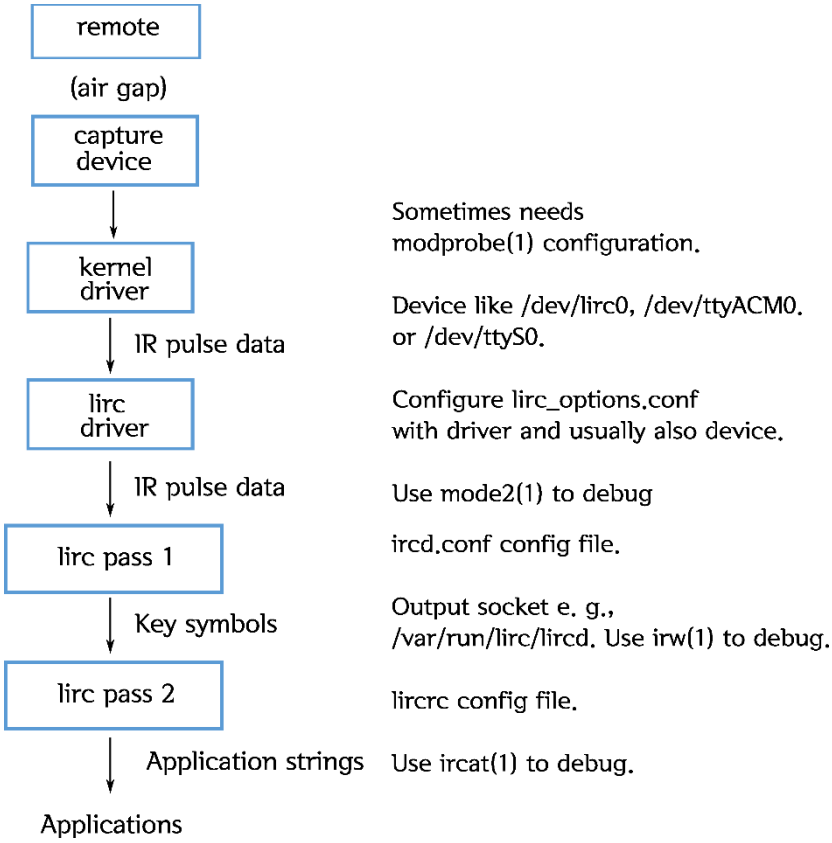
Depending on whether lirc is used or not application will get data either from the input layer (/dev/input) or from LIRC (/var/run/lirc/lircd). Using the LIRC data requires application support. Support for LIRC is common in typical linux htpc applications like mythtv, kodi and vlc; of course also LIRC applications like irexec, lircmd and irpty supports this.

The /var/run/lirc/lircd interfaces allows several applications to receive input events. On the other hand, the /dev/input interfaces only allows one application to receive each event. Despite these limitations, a common scenario is to not involve lirc at all, the upmost path in the picture (kernel → Linux input layer → application). Unless there is reason to use lirc (above)) this is probably the way to go.

If you need to use lirc, there are two cases depending on if your remote is supported by the kernel or not.

- If it's supported, the easiest path usually is using the linux input layer decoding and use that as input to lirc. This is the devinput data path in picture (kernel → Linux input layer → lirc → app).
- If the kernel built-in decoding can't be used (e. g., due to need of IR blasting or remote not being supported by the kernel) you should use a lirc driver instead. This is the bottom data path (kernel → lirc → app).

05) Basic setup flow



The overall LIRC blues:

- The remote generates an IR (or perhaps RF) signal.
- The IR data is captured by a capture device such as an IR dongle or a built-in IR port.
- Data from the capture device is caught by a linux kernel driver and made available on a kernel device such as /dev/lirc0, /dev/ttyS0 or /dev/input/eventXX.
- Data from the kernel is then caught by LIRC using a lircd driver.
- In the next step, lircd converts the kernel data from the lirc driver to key symbols using the lircd.conf file. The key symbols are presented on the output socket, by default /var/run/lirc/lircd.
- In the next step the key symbols are converted to application-specific strings using the ~/.config/lircrc file and a lirc library.

06) Determine driver and device

To determine the driver to use you might need to know the name of your capture device, what module the kernel has loaded for it and the kernel device it's connected to.

If our remote is bundled with a capture device such as a usb dongle, your first stop is the remote database. If you can find your device here, look in lircd.conf file's header for the following comment:

> this config file was automatically generated
> # using lirc-0.8.5-CVS(awlibusb) on Thu Oct 30 11:03:30 2008

Here you can learn that this file was recorded using the awlibusb driver. Take a note to the final

decision.

Next thing to do is to invoke ir-keytable:

```
$ ir-keytable
Found /sys/class/rc/rc0/ (/dev/input/event11) with:
Driver em28xx, table rc-pinnacle-pctv-hd
Supported protocols: NEC RC-5 RC-6
Enabled protocols: RC-5
Extra capabilities: <access denied>
```

If you get this kind of output you know the event device (/dev/input/event11) and the kernel module loaded (em28xx). Furthermore, since ir-keytable finds the device you know that the driver is part of the rc subsystem. Not all devices are recognized by ir-keytable, though.

Next step is to inspect dmesg, possibly after reconnecting your device. If you have a standard IR remote which is recognized by the kernel you can find how it's registered as rc0:

```
usb 3-2: Product: eHome Infrared Transceiver
Registered IR keymap rc-rc6-mce
input: Media Center Ed. eHome Infrared Remote Transceiver (0609:031d)
      as /devices/pci0000:00/0000:00:14.0/usb3/3-2/3-2:1.0/rc/rc0/input16
rc0: Media Center Ed. eHome Infrared Remote Transceiver (0609:031d)
      as /devices/pci0000:00/0000:00:14.0/usb3/3-2/3-2:1.0/rc/rc0
input: MCE IR Keyboard/Mouse (mceusb) as /devices/virtual/input/input17
rc rc0: lirc_dev: driver ir-lirc-codec (mceusb) registered at minor = 0
```

If you just find something like this you have a device which isn't an IR device (in this case an RF remote):

```
usb 2-2: Product: RF receiver
usb 2-2: Manufacturer: X10 WTI
```

Even if you have an IR device, you might see something like this if the kernel sees it as a keyboard rather than a remote. Here, an usb keyboard from JITTEL:

```
Product: JTTEL Composite Devices
hid-generic 0003:20E8:5820.0001: input,hidraw0: USB HID v10.01 Keyboard
    [JTTEL Inc. JTTEL Composite Devices] on usb-0000:00:1d.1-1/input0
```

For devices like these which not are registered as rc devices (and thus not recognized by ir-keytable) you might need to find out the corresponding event device as described in Appendix 2

Knowing the capture device name, the kernel module loaded (if any) and perhaps also a /dev/input device you have to select a driver:

- If ir-keytable located the device you can use the devinput driver. This means that the kernel decodes the ir signals and converts them to button press symbols. Use something like (with device as from ir-keytable):

  --driver devinput --device /dev/input/event11

  If ir-keytable only locates one device (the most common case) you can use

--driver devinput --device auto

This is actually the default setup, and needs no modifications.

- From 0.9.5+ you can list all usable event devices using
  mode2 --driver devinput --list-devices

- If you dont wan't to use the decoding done by the kernel, but the device is recognized by ir-keytable you can have lirc decode the raw signal from the driver. This means using the default driver which accesses the kernel on a /dev/lirc device, usually /dev/lirc0. Use something like:
  --driver default --device /dev/lirc0

  From 0.9.5+ you can list all available devices using
  mode2 --driver default --list-devices

- If you could find out the driver used to record this device in the driver database (above) you should try this driver if it makes sense. If it doesn't make sense e. g., it refers to some hardware you don't have just proceed.
- if you can find the device name in the generic driver list you might try to use this driver. Refer to Appendix 3 for details.
- If you have a device such as a keyboard which is not recognized and can't find a driver in the driver list or in the lirc-lsplugins output, the feasible option is using the devinput driver (above).

After selecting the driver and device you should check if there is any driver documentation. After this, check the driver and device using using e. g., something like

mode2 --driver default --device /dev/lirc0

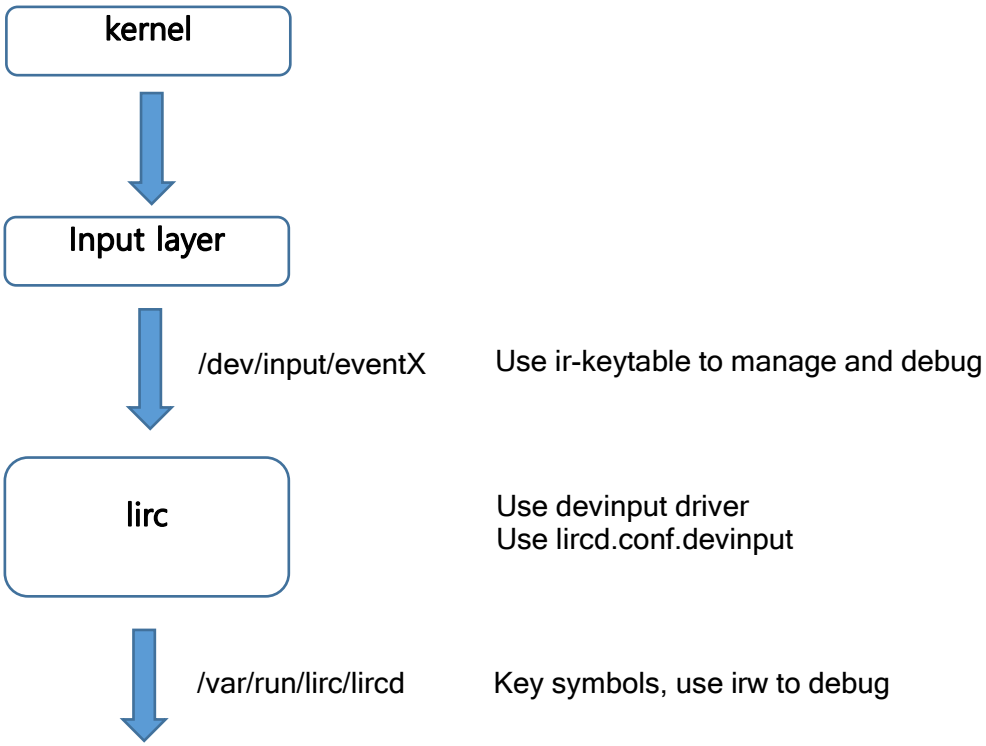
Refer to EXAMPLES in the mode2 manpage too see expected output.

The lirc-setup tool can be used to run mode2 with different drivers and devices in a GUI environment.

If you don't see anything, try to find out: (a) if you selected the correct driver b) If the driver needs settings (I/O base address, IRQ) in modprobe.d, (c) if the remote which works and (d) if your capture hardware works.

If you are to use the devinput driver, read on. Otherwise proceed to Getting the key symbols using lirc driver

07) Getting the key symbols using linux input layer



If you're lucky, your remote is already supported by the kernel. In order to find out, the first task is to locate the event device, something like /dev/input/event12 which is connected to your IR device. This is described in appendix 2.

With the device known use ir-keytable to test if your remote works:

```
# ir-keytable -t -d /dev/input/event13
```

Press buttons on the remote. If it starts to print out scan codes and key symbols everything is fine. Otherwise, try to change the protocol (see the ir-keytable manpage). If this doesn't work, it might be the end of the road and you might need to use the lirc driver option instead.

Check that all buttons generate output when testing. If there are buttons which are not mapped (no key symbol) you might not be able to fix this unless you go for the lirc driver option (to change the key symbol is perfectly possible, but probably not what you want here).

Then, activate the devinput.lircd.conf template which comes with lirc:

```
$ cd /etc/lirc/lircd.conf.d; sudo cp devinput.lirc.dist
   devinput.lircd.conf
```

If the devinput.lircd.dist file is not available or have some problem it can be re-generated using the script lirc-make-devinput(1). This might become necessary if the running kernel is different from the one used when packaging the lirc files.

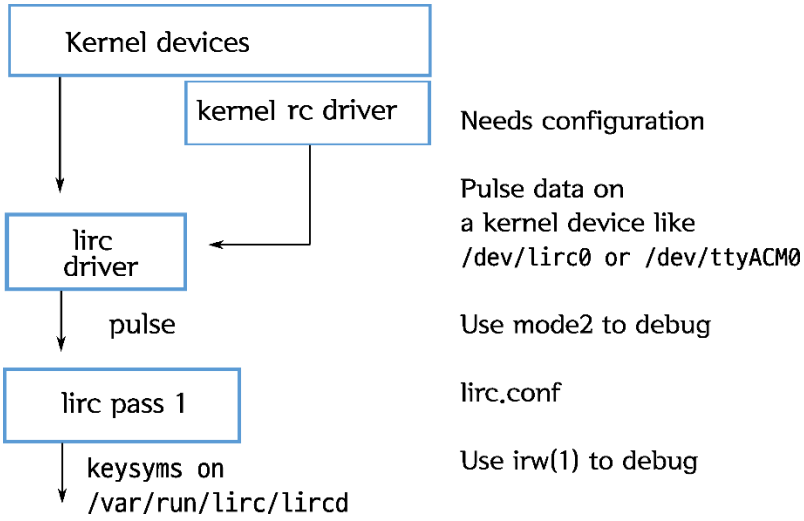
Start the lircd daemon and use irw to check:

```
$ lircd --device /dev/input/event13 --driver devinput
$ irw
```

Press remote buttons. You should see the key symbols being printed. When so you are done and can proceed to Configure systemd

Depending on your box, it might be that the event device found this way changes after a reboot. If this becomes o problem, look into appendix 6

08) Getting the key symbols using lirc drivers



You have already determined the driver and device to use which is verified using mode2. Make sure the lirc driver can read the remote, and produce pulses:

lirc pass 1: Using lircd.conf, convert pulses to key symbols like KEY_UP:

The lircd.conf is the file which lircd uses to read data from the driver and then convert (or decode) it to key symbols. It's the single most important lirc configuration file. There are some ways to find or create such a file.

- The lirc-setup GUI tool can be used to find and download both driver and configuration file which also can be tested in a GUI environment.
- If you manually selected a driver from the driver table it might need a specific lircd.conf. Such drivers are best installed using the lirc-setup tool but the corresponding configuration files can also be downloaded from the website(below)
- You can use one of the already existing configuration files on the lirc-remotes website. You can browse here or use the irdb-get tool to search and download such files.
- You can create your own configuration using the **irrecord** tool
- If you already have a config file for the libirman package you can convert it using the irman2lirc script that you can find in the contrib directory.
- It's also possible to convert CCF files and Pronto codes to a valid lircd.conf file using the pronto2lirc script.
- The irscrutinizer tool from harctoolbox.org can convert a wide range of formats to

lircd.conf files. It can also record data the same way as irrecord.

Even if you obtain the configuration file without using lirc-setup, you can still use it to verify the configuration. It runs lircd and irw as described below, making testing much simpler.

To install the file it should be copied to the lircd.conf.d directory, usually /etc/lirc/lircd.conf.d. Make sure the name ends with .lircd.conf.

If you are using several remotes you need to combine several lircd.conf files. See Appendix 8

After installation you should be able to start the the lircd daemon using something like:

```
$ lircd --nodaemon --device /dev/lirc0 --driver default
```

Verify the results using irw(1) in another window. Pressing buttons should give something like:

```
$ irw
000000037ff07bef 00 KEY_VOLUMEUP Acer_Aspire_6530G_MCE
000000037ff07bef 01 KEY_VOLUMEUP Acer_Aspire_6530G_MCE
000000037ff07bdd 00 KEY_ENTER Acer_Aspire_6530G_MCE
000000037ff07bdd 02 KEY_ENTER Acer_Aspire_6530G_MCE
```

Once irw works you are done with this step: lirc can convert the button presses to key symbols. The next step is to convert the symbols to configure the systemd service.

09) Configure systemd

By now you should know the driver and device used when running lircd. Update the configuration file /etc/lirc/lirc_options.conf with the driver and device you have determined:

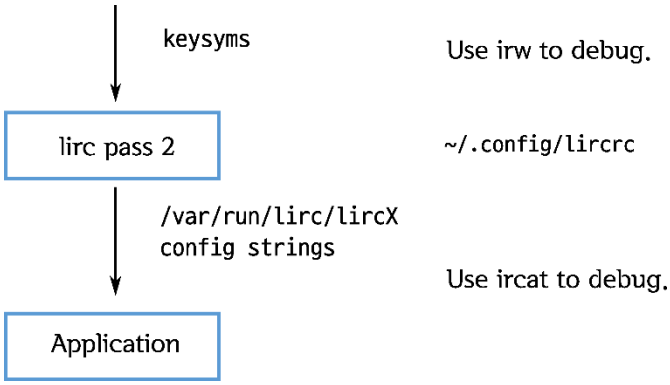
```
[lircd]
nodaemon        = False
driver          = default
device          = /dev/lirc0
....
```

You should then be able to start, stop and inspect the service using:

```
# systemctl start lircd.socket
# systemctl stop lircd.socket
# systemctl status lircd.socket lircd.service
# journalctl -b 0    /usr/sbin/lircd
```

Check that you can start/stop a working service, **irw** is your friend. If you are using another init system than systemd, you need to make similar steps.

10) Converting key symbols to application strings



Using *~/.config/lircrc*, convert the key symbols to application-specific strings.

The first step is to create a simple configuration for just one key for the irexec application, dipping a toe into the water. Create the following file and store it as ~/.config/lircrc:

```
begin
        remote = mceusb
        button = KEY_RED
        prog   = irexec
        config = echo "foo"
end
```

Notes:

- The remote (here "mceusb") is the name attribute from the lircd.conf file i. e., the line starting with 'name'.
- The button is a key symbol from the *.lircd.conf. If you don't have a KEY_RED, use another button.
- The whole idea with this step is that each application has its own translation. We will use the simple irexec app as prog for now, and check what kind of data it will receive when pressing the KEY_RED button.
- The config string is what irexec will receive when we press the KEY_RED button.

With this irexec file in place, we use ircat(1) to check what irexec receives when we push the KEY_RED button:

```
$ ./ircat irexec
echo "foo"
echo "foo"
echo "foo"
^C
```

```
$
```

So, at this point you can start irexec, and it will do actually echo some "foo" when you press the red button:

```
$ ./irexec
foo
foo
^C
$
```

With this simple example working, you now need to create complete config files for your application(s). First, you should read .lircrc chapter . Then, create your application config setup in ~/.config/lircrc and test it with ircat as above.

Depending on your application, lirc-config-tool might be able to generate a starting point. i See appendix 5.

Once the application is up, you might want to exploit LIRC's capabilities:

- **Using irexec(1) you can configure lirc to run arbitrary program when a button is pressed.**
- **Using lircmd(1) you can use lirc to let the remote emulate a mouse.**
- **You can setup lirc to transmit IR signals (IR blasting) to other devices e. g., let the remote send ir signals to a TV set. The program is irsend(1), you might want to scan the web for howto:s.**

11) Appendixes

a) A1: Configuring the kernel

When using the default LIRC IR driver, the kernel IR driver must be configured to send the data only to the /dev/lirc device and not to the general input layer. If not, each button event will delivered twice to the application, both through /var/run/lirc and /dev/input.

As of 0.9.1+ this is configured automatically by lircd, and neither the echo 'lirc' >/sys/class/rc/... nor the protocol udev rule should normally be required.

lircd can run either as root or as a regular user. In the latter case you might need to adjust device permissions.

Also, some lirc drivers conflicts with the kernel drivers. A common example is the lirc atilibusb driver which conflicts with the kernel ati_remote driver. Another example is lirc serial drivers which conflicts with the kernel default tty driver. Such conflicts shows up as dmesg output about not being able to open the involved device, plus various other symptoms.

If required, the default driver configuration can be done using /sys/class/rc interfaces or using a udev rule. Conflicting kernel drivers must be blacklisted. Conflicts on serial ports can be handled by disabling the kernel serial driver for that port.

When using serial or parallel port hardware the proper kernel module must be loaded with correct options. This requires modprobe(1) configurationi.

Using TV-cards requires some extra attention.

Kernel IR driver runtime configuration.

The builtin ir driver subsystem is aware of LIRC, and is capable to send all data through /dev/lirc0. If lircd fails to configure this automatically it can be done manually:

```
# echo -- 'lirc' > /sys/class/rc/rc0/protocols
```

Here, 'rc0' is OK if you have only one infrared device. Note that this is not persistent, you need to do this after each boot.

Using '-lirc' instead restores the normal kernel operation when stopping LIRC.

Kernel IR driver udev configuration

Likewise, if lircd fails to configure the kernel IR driver automatically you can create a file /etc/udev/rules.d/99-remote-control-lirc like:

```
SUBSYSTEM=="rc", ATTR{protocols}="lirc"
```

This is persistent and makes all ir (i. e., rc-based) devices send data only through /dev/lirc0 where it can be retrieved by the 'default' driver. The file is available in the contrib/ directory.

Kernel module conflicts

When using remotes which are not infrared, the corresponding driver is not affected by the methods above. One example is an RF remote which uses the atilibusb LIRC driver. This conflicts with the ati_remote kernel module, which thus needs to be disabled. Do this by creating the file /etc/modprobe.d/blacklist-atiremote.conf like:

```
# Conflicts with LIRC.
blacklist ati_remote
```

For known cases the lirc-setup tool generates blacklisting configuration files.

The contrib directory contains a file 61-lirc.blacklist-all.conf which blacklists all kernel drivers known to conflict with any lirc plugin. While in most cases an overkill, it can be helpful.

In general, finding out what module to blacklist is not always easy. dmesg(1) sometimes gives a hint about conflicts on a device. Another method is to boot the system without the usb device connected, and do a lsmod. After that, connect the device and make a new lsmod. Comparing the different outputs might give a clue.

Adjusting kernel device permissions

When lircd not runs as root, it needs read and write access to the kernel device it communicates with. Since devices in Linux are handled by udev this is handled by udev rules.

Many drivers including the default driver uses the /dev/lirc or USB devices. In the contrib directory is an example file for the these devices called 60-lirc.rules containing:

```
KERNEL=="lirc[0-9]*", SUBSYSTEM=="lirc", GROUP="lirc", MODE="0660"
ACTION=="add", SUBSYSTEM=="usb", \
    RUN+="/usr/bin/setfacl -m g:lirc:rw $env{DEVNAME}"
```

The file should be stored in /etc/udev/rules.d/60-lirc.rules This example gives users in the group lirc full permissions to the /dev/lirc0 devices using regular group permissions. It also gives members of the same group access to all USB devices using extended permissions (ACL). It should be simple to adopt to other users (USER=), usernames and devices. Refer to more generic udev docs.

The devinput driver uses the /dev/input/event* devices. These are often accessible for members of a specific group; the best solution is then to add this group the the lircd user's supplementary group e. g., using usermod -aG input lirc which adds the input group to user lirc. See Running as a regular user

Disabling kernel serial port reservation

Usually the default kernel serial port driver grabs all ports it auto-detects as soon as it is loaded and the LIRC modules won't be able to use any of them.

This needs to be resolved using the setserial(1) tool. An example how to load a lirc_serial module on /dev/ttyS0:

```
setserial /dev/ttyS0 uart none
modprobe lirc_serial
```

This could be added to lirc_options.conf as a modprobe section e. g.,:

```
[modprobe]
code = setserial /dev/ttyS0 uart none; modprobe lirc_serial
```

This section is parsed by the lircd-setup tool which runs as root when lircd is started.

lirc-setup can generate this section in many cases.

Debian users should adjust their /etc/serial.conf. Note that lirc_serial probably needs some modprobe setup, see below.

modprobe configuration

The kernel loads the appropriate driver for most modern, USB-based using udev hotplugging - as soon as the device is connected the corresponding kernel module is loaded. However, when using hardware connected to e. g., serial or parallel ports the proper kernel module must be manually loaded. This is done using modprobe(1) options or modprobe.d(5) files. Options to modprobe include things like the actual port, interrupt to use etc.

lirc-setup can generate proper /etc/modprobe.d files in such cases. modinfo(1) also provides useful info. In any case, the proper kernel module must be loaded with correct options before lircd is started.

TV cards

To use any remote control receivers connected directly to a bttv based TV card you will need a working bttv setup in your kernel. For most TV cards we rely on bttv autodetection. That way you don't have to give any parameters to the module as they are selected internally depending on the

information the bttv module gives us. This means that you should pay attention that your TV card is detected correctly by bttv, as can be checked using dmesg(1).

b) A2: Finding the event device

c) A3: Understanding the driver table

d) A4: Normalizing the lircd.conf

e) A5: Generating the .lircrc

f) A6: Addressing changing event devices

g) A7: Running irexec

h) A8: Using multiple remotes

i) A9: Using multiple capture devices

j) A10: LIRC configuration files

k) A11: Using home-brewn hardware

l) A14: Running as a regular user

【04】Configuration reference

01) lircd.conf file format

A description of the format is available in the lircd.conf(5) manpage. In fact you probably don't need to know anything about it except that it's maybe the most important part of the package.

02) Configuring lircmd

ircmd can be used to emulate a mouse with your remote control. Depending on the config file described in the next section it converts IR signals into mouse events. It currently supports three mouse protocols (MouseSystems, IntelliMouse and IMPS/2). For compatibility reasons the default protocol is the MouseSystems protocol but the preferred is the IntelliMouse protocol. The advantage of this protocol is its wheel-mouse support. That way you can for example configure Netscape to scroll if you press certain buttons.

IMPS/2 used to be the preferred protocol since it also has wheel-mouse support and IntelliMouse was not available. However PS/2 protocol specifies that the mouse must accept and reply to specific commands, and that can not be done through the pipe lircmd uses. For this reason IntelliMouse support was written and is currently the preferred protocol.

lircmd can basically be used with two applications: X11 and gpm Using gpm makes no sense with a modern X installation since this supports multiple mouses natively.

a) X11

Put this section in your Xorg.conf file to use the lircmd emulated mouse in addition to your normal one.

```
Section "InputDevice"
        Identifier    "LIRC-Mouse"
        Driver          "mouse"
        Option          "Protocol" "IntelliMouse"
        Option          "SendCoreEvents"
        Option          "Buttons" "5"
        Option          "ZAxisMapping" "4 5"
EndSection
```

b) uinput and effective user

If your distribution does not have uinput enabled by default (i. e., there is a /dev/uinput device) add to the file /etc/modules the entry:

```
uinput
```

To make this device writable for regular users create a new file /etc/udev/rules.d/55-uinput with one line

```
KERNEL=="uinput", MODE="0666"
```

However, this has security implications. The preferred setup is to use the example rule in the contrib directory which makes /dev/uinput accessible for the lirc user, and then run lircmd as this user

Another alternative is to run lircmd as root. In this case this no udev rule is required, but it raises other security concerns.

Finally, run lircmd with the --uinput option (which can be set in lirc_options.conf).

c) systemd

LIRC ships with systemd support files. To start lircmd, use

```
# systemd start lircmd.service
```

To make it start at boot time use:

```
# systemd enable lircmd.service
```

03) lircmd.conf file format

The config file for lircmd is quite simple. Just look at the example in the contrib directory. Some drivers even already bring their config file for lircmd with them so lircmd is ready to run.

PROTOCOL <protocol>
You can choose between MouseSystems, IntelliMouse and IMPS/2 protocol. The default is MouseSystems protocol.

ACCELERATOR <start> <max> <multiplier>
Change the values here if your mouse pointer is moving too fast/slow. Usually the mouse pointer moves 1 pixel every time it receives a signal. The values here specify how much mouse movement accelerates if you hold down the according button on your remote control for a longer timer. The start value is the threshold that starts acceleration. Then the amount of pixels is calculated with the following formula: x=repeat*multiplier, where repeat is the number of repeated signals. max specifies the maximum number of pixels the pointer can move due to a single command.

ACTIVATE <remote> <button>
TOGGLE_ACTIVATE <remote> <button>
I recommend that you use a special button to activate the mouse daemon with this command. You will see whenever the daemon is activated/deactivated directly on the screen. If you omit this command the daemon will always be active.

The difference between ACTIVATE and TOGGLE_ACTIVATE is how you leave the mouse mode. With TOGGLE_ACTIVATE you have to press the button that you use to enter the mode to leave it. With ACTIVATE you will leave mouse mode as soon as you press a button that is not used for any function in the config file.

MOVE_ [N [E | W] | E | S [E | W] | W] <remote> <button>
The obvious functionality. You can even get better granularity by combing different commands (copied from the config file for AnimaX remotes):

MOVE_N ANIMAX_MOUSE_PAD MOUSE_NNE
MOVE_NE ANIMAX_MOUSE_PAD MOUSE_NNE
This also demonstrates that all commands are executed beginning at the top.

MOVE_[IN|OUT] <remote> <button>
This will only work with IntelliMouse and IMPS/2 protocols and indicates movement of the wheel.

BUTTONx_CLICK, BUTTONx_DOWN, BUTTONx_UP, BUTTONx_TOGGLE <remote> <button>
This simulates according events for the left (x=1), middle (x=2) or right (x=3) mouse button.

IGNORE <remote> <button>
Pressing ignored buttons won't cause the mouse daemon to deactivate. This is useful, for example, if your remote sends separate press or release codes that you have mapped in your lircd.conf. This only makes sense if you use ACTIVATE instead of TOGGLE_ACTIVATE.

'*' is allowed as wild card for button and remote. Please note that every line that fits to the received signal will be executed. Parsing starts at the top of the file.

04) lircrc file format

The lircrc file is used to map the key symbols defined in lircd.conf to application-specific strings. Thus, this file cannot be configured until lircd has been configured to provide proper key symbols as displayed by **irw**.

The lircrc file should be placed in the home directory as ~/.config/lircrc. Optionally you can create a system-wide configuration file located in /etc/lirc/lircrc which will be used when no lircrcfile can be found in the user's home directory. The idea is to have configuration information of all clients in one place. That lets you keep a better overview of clients and simplifies the use of modes explained later.

The preferred setup is to have a main ~/.lircrc which includes a number of other files, typically living in ~/.config/lircrc/*. The contrib directory contains a number of example files, most of which created using lirc-config-tool for an ordinary MCE remote

First I will explain the syntax of the lircrcfile itself. The file consists of one or more of the following constructions:

```
begin
      prog       = ...
      remote    = ...
      button    = ...
      [button = ...] (optional, for key sequences)
      repeat    = ...
      delay     = ...
      ignore_first_events = ...
      config    = ...
      [config = ...] (optional, for toggle button behaviour)
      mode       = ...
      flags     = ...
end
```

Bringing it to the point the above says which program (prog) should do what (config, mode, flags) if you press a certain button (remote, button) a specified time (repeat, delay). By default for each remote signal received the lircrc config file is read from top to bottom and each matching configuration is executed in order of appearance.

prog
gives the name of the program that should receive the configstring given in config.

remote, button
specify a key of a remote control that launches an action. Key sequences can be specified by giving more then one remote/button string (not on the same line, but using separate remote/button tokens on separate lines). The character '*' can be used as a wild card for remote or button. The default for remote is '*'. The remote name must always be given before its according button. When using key sequences a given remote is valid for all following buttons until you specify another remote.

repeat
tells the program what shall happen if a key is repeated. A value of zero tells the program to ignore repeated keys. Any other positive value 'n' tells the program to pass the config string every 'n'-th time to the according application, when a key is repeated. The default for repeat is zero.

delay
tells the program to ignore the specified number of key repeats before using the "repeat" configuration directive above. This is used to prevent double triggers of events when using a fast repeat rate. A value of zero, which also is the default, will disable the delay function. If "repeat" value is zero but "delay" is set, there will be a single event generated after the delay period expires (with another one before delay period starts).

ignore_first_events
ignores the specified amount of first events. Same as "delay" but without an event before delay

period starts. This allows to define the reaction on the long key presses. Should not be set together with "delay".

config
is the string that will be passed to the according application whenever the specified key sequence is received by lircd. If you give more than one config string, the config strings will be passed to the applications by turns. With this feature you can for example implement toggle buttons.
You can pass non-printable characters to applications with all standard C escape sequences (most common are: \n = line-feed, \r = carriage return, \t = tab, \e = escape, \<n> = ASCII code in octal representation, \x<n> = ASCII code in hexadecimal representation, \\ = backslash). Additionally you can supply Ctrl-X by specifying \X where X is an upper character or @. For example \C is Ctrl-C.

mode
tells the program to enter a special mode. You can group several configurations by putting them into the following, where mode stands for the mode where these configurations should be active:

```
begin mode
    ...
end mode
```

All configurations embraced by this mode construct will stay inactive until the program enters the given mode by using the mode token. Please note that configurations outside a mode will always stay active even though you enter a specific mode. To prevent the execution of such "global" configurations you can place these at the end of the config file below all mode constructs and use the quit flag described below to stop execution of further configurations when a match happens inside a mode block. If mode is equal to the name of a client application this application will always start in this mode. Consider this situation: you want to start xawtv with irexec and enter the tv mode. Then irexec would enter the tv mode but xawtv would begin without any mode enabled. By renaming the mode from tv to xawtv you can solve this problem.
Another way to specify a startup mode is by using the startup_mode flag as described below.
Caveat: In order to avoid many identical entries all actions that modify the mode a program currently is in are independent of the prog token.

The following are valid flags:

once
This is only allowed in conjunction with the mode directive. The config string is passed to the application only the first time the mode is entered or you have explicitly left this mode. This is useful for starting an application whenever you enter a special mode.

quit
Usually all configurations are examined whether they have to be executed. You can stop this immediately with this flag. Configurations further below will not be executed if the current button press matches the current configuration. A match also happens if the current configuration defines a button sequence and only part of the sequence already was entered.

mode
This is only allowed within a mode block. It tells the program to leave this mode.

startup_mode
Tells the program to start in the mode given in the mode keyword. The following example tells the program to start in the browser mode

```
begin
        flags = startup_mode
```

```
        mode = browser
end
```

toggle_reset
This will only have an effect if you have specified several config lines to implement a toggle button. Usually the toggle state is always saved for the button regardless of other button presses. But with this flag the toggle state will be reset to the first config entry as soon as a different button not matching the specification in the current block is pressed.

If you press a button on your remote the lircrc is searched from top to bottom for matching configurations. Be aware that the search is not stopped by a match unless you have specified the quit flag in the matching configuration. You should also be aware that if a configuration changes the current mode, the change takes effect immediately, which means that the further search for matching configurations beginning at the next configuration further down will take place with the new mode setting.

It is possible to split the lirc configuration into several files by using the include command. It tells the parser to read the specified file before resuming the current one:

include ~/.config/lirc/xawtv
If the specified filename begins with "~/", "~" will be substituted with the content of the HOME environment variable. The filename also can be put inside <> and "" characters which in contrast to the C preprocessor do not have special meanings.

A simple example for a lircrc file (supposing you using an AnimaX remote and use the sample files for this remote from the remotes database. If you have another remote change remote= and button= according to your remote [t definitions are made in the lircd.conf file]):

```
begin
    remote = ANIMAX
    button = MENU_DOWN
    prog     = irexec
    repeat = 0
    config = echo "Hello world!"
end
```

If you have saved this as ~/.config/lircrc , start irexec. Press the button which is selected in the button= line and you will see a 'Hello world!' on your screen. As you can see irexec is a simple program launcher. Of course you can do a lot more than just start programs. Be aware that irexec will wait for the started program to finish, before it will resume it function. If this is not what you want, you should add a "&" at the end of the config line to start the desired program in background.

Differences in the order of configurations in lircrc can lead to completely different results, as this example shows:

```
    begin order
    begin
        button = OK
        prog = irexec
        config = echo "This is printed last"
    end
    end order
```

```
begin
        button = OK
        prog = irexec
        config = echo "This is printed first"
        mode = order
        flags = quit
end
```

Using this order on first key stroke of OK

```
"This is printed first"
```

will appear - the command is executed and the mode 'order' is entered. The second stroke (and every further one) will lead to

```
"This is printed last"
"This is printed first"
```

Both configs are executed, even though the second is outside the mode; the quit flag has no effect - no other config is following it in the lircrc file.

Changing the order within the lircrc to

```
begin
        button = OK
        prog = irexec
        config = echo "This is printed first"
        mode = order
        flags = quit
end

begin order
begin
        button = OK
        prog = irexec
        config = echo "This is printed last"
end
end order
```

will lead to

```
"This is printed first"
```

on every stroke. The second config is never executed: even though the mode is changed it can not take effect (because of the quit flag). To achieve unrestricted usage of keys within modes place all mode-configurations before all other configurations; and use quit flags within the mode if you don't want other configurations to be executed.

If you start a LIRC client program, it reads your ~/.config/lircrc and reacts only on prog= entries which point to itself. All programs should give you the possibility to use an alternative config file. If you have included more than one program in your lircrc, then start all these programs, they react only to their according entries in lircrc. This also leads to a disadvantage of the mode concept. If you don't start all client programs at a time the mode they currently are in may differ between

applications. Also key sequences might not be recognized equally because all programs then don't have the same starting point. In order to solve this problem there is the lircrcd program since version 0.8.0. lircrcd's purpose is to synchronise all clients and maintain a common mode for all applications. In order to use the lircrcd feature you have to explicitly enable it by adding the following line at the beginning of the file:

lircrc_class default

This directive can only be used in a top-level file, not in an included one. The string default could actually be any identifier; clients using a lircrc with the same string will synchronized.

In versions before 0.9.2 the same effect was acheived with a "shebang", a first line in the file #! lircrcd From 0.9.2+, the support for this deprecated and it will be removed in an upcoming release.

05) Sending infrared signals

The LIRC package contains the irsend tool for sending infrared signals to e.g. your TV or CD player. For reliable transmission a good config file is even more important than for receiving. A discussion of all the infrared protocols is way beyond the scope of this manual but when creating a config file at least read the hints at the end of this manual. You can find exact timing specifications for most common protocols in files retrieved from the remotes database located using irdb-get find generic

LIRC also provides interfaces to develop applications which sends data. Sine 0.9.2, the primary interface is the client API.

【05】Manpages

01) Programs and Drivers Overview

a) LIRC manpages and user-space drivers overview

LIRC contains a quite large number of tools. This chapter tries to provide an overview.

b) User programs

- irsend sends data using lirc from the command line.
- irpty emulates keyboard input from a tty using a remote.
- irrecord can create a new lircd.conf for a remote which isn't available in the database.

c) Daemons

- lircd is the main LIRC daemon and runs as a systemd service. It usually reads from a kernel device like /dev/lirc0 or /dev/ttyACM0 and presents the decoded data on an output socket like /var/run/lirc/lircd.
- lircrcd is a helper daemon which coordinates the LIRC state as defined in lircrc in different clients. It's managed, started and stopped by lircd.
- lircmd runs as a systemd service. It reads data from the lircd output socket and generates mouse events, so that a remote can emulate a mouse.

- irexec normally runs as a user daemon in the session. It is used to run arbitrary commands on various button presses.
- lircd-uinput connects to the lircd output socket and forwards the decoded button presses to the kernel uinput device. This makes the lirc events available to other applications in the same way as events from other input devices.
- lircd-setup is a small, one-shot program which runs commands as root before starting lircd .

d) Configuration tools

- irdb-get can list, search and download lircd.conf remote configuration files from the remotes database.
- lirc-setup is a GUI tool which can create the different configuration files after some user dialogs.
- lirc-lsplugins lists the plugins and drivers which are actually available in an installation. It also provides info on the drivers, including drivers that can't be loaded.
- lirc-lsremotes can parse lircd.conf configuration files and act as a static checker. It is also used to provide dense, parseable keyword information about remote config files.
- lirc-config-tool can generate a first shot for lircrc configuration file for some applications. It can also be used to sanitize old lircd.conf files which does not use symbols from the official namespace.
- lirc-make-devinput can generate a site-specific devinput.lircd.conf configuration file.

e) Test and debug tools

- mode2 monitors the data sent from the kernel to lircd. It displays either timing information or, if the hardware decodes the signals, the decoded values.
- irw monitors the data sent from lircd to the applications, normally in the /var/run/lirc/lircd socket.
- ircat connects to same socket as irw. However, it displays the application strings delivered to the application after mapping the lircd output using the lircrc file.
- irpipe works together with the irpipe kernel driver. It can be used to feed data to lircd which it can read on /dev/irpipe0. This device works the same way as /dev/lirc0 as seen from lircd.
- irsimsend can "send" data using a lircd.conf configuration file and store the data (pulse/space durations) in a file. This file can be used with irpipe to send the same data to lircd.
- irsimreceive can "receive" from a file e. g., created by mode2(1) or irsimrecieve(1).
- irtestcase can log the data from a remote together with the decoded symbols. It can also send the data to lircd and check that the decoded symbols matches the logged ones.

f) User-space drivers

Note that many drivers lacks documentation. Use lirc-lsplugins to generate the complete list of drivers with some basic info.

- alsa-usb receives IR data over the microphone input, using the alsa drivers.
- atilibusb receives RF data with various hardware using the X10 chip.
- atwf83 receives IR data using Aureal ATWF@83 ESKY chip.
- audio sends IR data using standard audio hardware.
- default sends and receives IR data using any device supported by the kernel.

- devinput receives IR data from any remote supported by ir-keytable(1) with a minimum of configuration.
- file driver logs data (pulse durations) otherwise sent, and can also be used to "receive" simulated data from a file.
- girs driver receives and sends IR data using a serially connected remote device compliant with the Girs standard (for example for usage with the Arduino).
- imon-24g handles data from the iMON 2.4G DT/LT remote.
- imon handles data from various iMON remotes.
- iguanair> receives and sends IR data using the IguanaIR devices.
- irtoy recieves and sends IR data using the Dangerous Prototypes IrToy device.
- srm7500atilibusb receives RF data using a Philips SRM-7500 capture device.
- tira sends and receives IR data using the Home Electronics Ira/Tira capture device.
- udp receives data over a network UDP port using a specific protocol.

There is also External plugins documentation.

Devices and data formats

- lirc(4) data formats and ioctl commands for the /dev/lirc[0-9] devices. This manpage has been upstreamed to the linux manpages project.
- lircd.conf: the lircd.conf format.

02) Programs and tools

a) ircat

NAME
ircat - print strings when pressing buttons

SYNOPSIS
ircat [options] <prog>
DESCRIPTION
This program prints config strings to standard output. It can be used to provide remote control input to scripts and to debug your .lircrc file.

The argument to the program is the program name, as it appears in the prog entries in .lircrc.

-h --help
Display usage summary.
-v --version
Display version.
-c --config=<file>
Set config file.

EXAMPLES
If .lircrc contains:

```
begin
        prog = myprog
```

```
        button = KEY_CHANNELUP
        config = next_file
end
then
```

$ ircat myprog
will print "next_file" (followed by newline) every time the button bound to KEY_CHANNELUP is pressed.

ENVIRONMENT
LIRC_SOCKET_PATH
The lircd(8) output socket used to retrieve data. Defaults to /usr/var/run/lirc/lircd

SEE ALSO
The documentation for lirc is maintained as html pages. They are located under html/ in the documentation directory.

b) irdb-get

NAME
irdb-get - list, search and download lirc configuration files.

SYNOPSIS
irdb-get update
irdb-get find <string>

irdb-get info <id>

irdb-get download <id>

irdb-get list [pattern]

irdb-get find-driver <string>

irdb-get yaml-config

irdb-get <-h|--help|-v|--version>

DESCRIPTION
irdb-get can list, search and download lirc remote configuration files. It works by downloading an index file, making all operations besides the actual download local.

Entries in the index are identified by an id. Such an id is on the form dir/remote e. g., x10/ati_remote_wonder_III

COMMANDS
update

Download the index from the remote repository.
find <string>
List all entries in the index file matching <string>. Matching is done using regular shell match syntax (i. e., not regex).
find-driver <string>
List all entries in the index with a given driver. The matching is exact, no wildcards.
info <id>
Print some more info about a given entry.
download <id>
Download the configuration file for a given entry. The filename is printed on stdout.
list <pattern>
List entries matching <pattern>. <pattern> is on the form dir/remote e. g., x10/ati* or */ati_remote_wonder* .
The first word in the list is something like -RL- where the letters represent:

R
Raw configuration.
T
Remote has timing information i. e., it can be used with generic capture devices.
L
Remote has a lircmd config file.
P
Remote has a photo.
yaml-config
Prints a YAML file on stdout which maps drivers to remote files with corresponding driver attribute. The format is unstable and primarely used by lirc-setup(1).

OPTIONS
-h , --help
Print help message.
-v , --version
Print version info.

ENVIRONMENT
LIRC_REMOTES_URL
The base URL used when downloading config files, defaults to https://sourceforge.net/p/lirc-remotes/code/ci/master/tree.
LIRC_REMOTES_LIST
URL to a file listing all remotes, as created by lirc-lsremotes(1). Defaults to http://lirc-remotes.sourceforge.net/remotes.list.
XDG_CACHE_HOME
If defined, relocates the cached, downloaded remotes listing file to $XDG_CACHE_HOME/remotes.list, see FILES.

FILES
~/.cache/remotes.list
The index file downloaded on demand or by the update command.

c) irexec

NAME
irexec - run programs with one button press

SYNOPSIS
irexec [options] [config_file]
DESCRIPTION
irexec executes commands on an IR signal decoded by lircd, the LIRC daemon. It uses an lircrc config file where the config = entries are executed. E. g., given the following config file snippet

```
begin
        prog     = irexec
        button = KEY_RED
        config = echo "KEY_RED"
end
```

irexec will echo KEY_RED on the terminal when the corresponding button is pushed on a remote. The command is an arbitrary shell command executed asynchronously - irexec does not wait for it to complete.

ARGUMENTS
config_file
lircrc configuration file. irexec only uses entries with prog = irexec. The path defaults to ~/.config/lircrc.

OPTIONS
-h, --help
Display usage summary
-v, --version
Display version
-d, --daemon
Make irexec fork to background. In this case a config file should be given on the command line as irexec won't be able to find any home directory.
-D, --loglevel [level]
Determine the amount of logging information. [level] can be a symbolic syslog level: error, warning, info, notice or debug. lirc also defines three additional levels trace, trace and trace2 which gives even more messages (trace2 bringing the most). However, in the log these messages are marked as debug. By default, no logging is done.
-n, --name <name>
Use this program name instead of the default irexec as identifier in the lircd.conf file.

ENVIRONMENT
LIRC_SOCKET_PATH
Path to the lircd socket irexec reads from, defaults to /usr/var/run/lirc/lircd.

FILES
~/.config/lircrc
Default config file
/usr/etc/lirc/lircrc
Config file used by the systemd irexec service.
~/.cache/irexec.log
Debug log. Setting the XDG_CACHE_HOME environment variable relocates this file to $XDG_CACHE_HOME/irexec.log

NOTES
For versions up to 0.9.1 irexec used to wait until the executed program terminated. Old configuration files thus often includes a '&' appended to the command string to avoid being stuck in the command. This is not required in 0.9.2+ which cannot wait for command completion.

irexec should run as a service. The contrib directory contains a .desktop file which could be dropped in ~/.local/autostart. Doing so creates a service which can be handled by regular desktop tools such as gnome-tweak-tool on all major desktops. If running restricted commands such as powering off the machine is required, sudo(8) can be used to allow regular users to run such commands.

An alternative is to use the irexec.service systemd service which runs as root (and can run in parallel with the desktop service). Running as root has severe security implications. See the Configuration Guide in the html documentation.

d) irpty

NAME
irpty - Pseudo tty driver.

SYNOPSIS
irpty [options] config_file -- program [args ...]
DESCRIPTION
irpty connects to lircd to receive infrared codes and converts them to key strokes. E.g. type irpty ~/.config/lircrc -- workbone to control the CD-player program workbone. Of course you will have to create an appropriate config file for this purpose first. The config string will be passed to the desired application. Note that you can use escape sequences to specify non printable characters. Have a look at the .lircrc file format description for details.

OPTIONS
-h --help
display usage summary
-V --version
display version
-e --no-echo
disable echo
-i --ignore-eof
ignore EOF
-n --non-interactive
force non-interactive mode
-v --verbose
verbose mode

SEE ALSO
The documentation for lirc is maintained as html pages. They are located under html/ in the documentation directory.

e) irrecord

NAME

irrecord - IR-codes recording tool for usage with LIRC

SYNOPSIS

irrecord [-f] [-n] [-H driver] [-d device] [file]

irrecord -a <file>

irrecord -l

irrecord --help | --version

DESCRIPTION

This program will record the signals from your remote control and create a config file for lircd. Although a good deal of effort is put in this program it is often not possible to automatically recognize all features of a remote control. See USAGE NOTES below.

If *file* is not specified it defaults to "irrecord.lircd.conf"

If *file* already exists and contains a valid config irrecord will use the protocol description found there and will only try to record the buttons.

OPTIONS

-a --analyse
Analyse a raw_codes config file, trying to convert it to a regular configuration.

-u --update
Add new buttons to an existing config file. No protocol information is updated.

-f --force
Force raw mode. Use this if recording fails otherwise. This creates a raw codes configuration file which can be used as-is or converted using the -a option.

-n --disable-namespace
Disable namespace checks.

-l --list-namespace
List valid button names.

-H --driver=driver
Use given driver. -H help lists available drivers.

-d --device=device
Read from given device. Use mode2(1) --list-devices to list available devices for a driver.

-U --plugindir=directory
Load drivers from directory. See DRIVER LOADING.

-k --keep-root
Don't drop root privileges after opening device. See RUNNING AS ROOT.

-A, --driver-options key:value[|key:value...]

Set one or more options for the driver. The argument is a list of key:value pairs delimited by '|'. The key can not contain spaces, but such are allowed in the value part. Certain characters including '#' and ';' are used as comment markers in the config file and are not allowed anywhere.

-D --loglevel=level
Determine the amount of logging information. [level] can be a symbolic syslog level: 'error','warning, 'info', 'notice' or 'debug'. lirc also defines three additional levels 'trace', 'trace1' and 'trace2' which gives even more messages ('trace2' bringing the most). However, in the log these messages are marked as 'debug'. The level can also be an integer in the range 3 (almost no messages) to 10.

-O, --options-file <path>
File containing default values for all options. A relative path is interpreted from current directory. See [FILES] below.

-h --help
Display this message.

-v --version
Display version.

USAGE NOTES

The primary options are --driver and usually also --device (some drivers does not need --device). These could be verified using mode2(1) if irrecord runs into trouble.

When driver and device is known it saves some work to update lirc_options.conf with the new values. irrecord uses the [lircd] section as a fallback, so with a proper lirc_options.conf irrecord could be run without command line options.

Using an existing file as a template can sometimes be required for irrecord to work. If using one of the generic templates it can also provide better timing. See PROTOCOL PARAMETERS AND TEMPLATES.

If the program fails to recognize the protocol of the remote control you could use the --force option to at least create a config file in raw mode.

REPEAT MASKS

After recording some buttons with irrecord, you should check the repeat masks. These are needed in order to handle repeated button presses, which are very common

Using the config file, start lircd and irw. Keeping a button pressed down, you should see something like:<

```
0000000000f40bf0 00 KEY_1 ANIMAX
0000000000f40bf0 01 KEY_1 ANIMAX
0000000000f40bf0 02 KEY_1 ANIMAX
0000000000f40bf0 03 KEY_1 ANIMAX
0000000000f40bf0 04 KEY_1 ANIMAX
```

0000000000f40bf0 05 KEY_1 ANIMAX

Note how the second field gets incremented. This must work for correct operation.

PROTOCOL PARAMETERS AND TEMPLATES.

irrecord actually works in two phases. In the first, it tries to identify the protocol used by the remote. Although this often works quite well, there are advantages using an existing file as a template.

One reason to use a template is when irrecord fails to recognize the protocol in the beginning. In such cases, try to find an existing configuration from the same vendor using something like:

```
$ irdb-get find motorola
# lots of remotes listed...
$ irdb-get download motorola/VIP_1200.lircd.conf  # Pick any
$ cp VIP_1200.lircd.conf my_remote.lircd.conf
$ irrecord my_remote.lircd.conf
```

Invoking irrecord this way re-uses the existing protocol which usually works (vendors seldom changes protocol parameters).

Another reason to use an existing file is to get more exact timing, important when planning to also transmit (blast). In such cases, using a generic template found using irdb-get find generic has advantages since the timing values here are hand-crafted from specifications. To use these, you need to know which protocol your remote uses, though.

If decoding of IR commands does not work reliably you can try to modify the eps and aeps values in the lircd config file to adjust the tolerance of signal lengths. aeps is an absolute value while eps is a relative value expressed in percent. See lircd.conf(5)

TROUBLESHOOTING

Multiple-personalities remotes

Some remotes actually emulates two or more remotes. irrecord runs into trouble if buttons from different logical remotes are mixed in the same run, in particular in the initial protocol identification. For such devices, irrecord must be invoked once for every logical device using only buttons from this device during the run.

Un-printable garbage when using default driver

If there is various non-printable garbage on the screen when running irrecord when using the default driver the rc protocol needs to be set. With a single device the protocol can be set and inspected using

```
sudo sh -c "echo 'lirc' > /sys/class/rc/rc0/protocol"
cat /sys/class/rc/rc0/protocol
```

Physical disturbances

As for physical disturbances, the primary source is fluorescent light. You should not have any such light around when using irrecord. It's also important to have a suitable distance between the remote and the capture device, which often is smaller than in typical, normal usage. However, making the distance too small might cause other problems. At a first try use a foot or two.

DRIVER LOAD PATH

Drivers are loaded dynamically. This is done from a traditional *ux ':'-separated path where each component in the path is searched (leading part first, trailing last).

The path used for this is determined by (falling priority):

-
The --plugindir option.
-
The 'plugindir' entry in the [lircd] section of the lirc_options.conf file.
-
The environment variable LIRC_PLUGINDIR.
-
A hardcoded default (/usr/lib/lirc/plugins).

RUNNING AS ROOT

In many cases irrecord needs to run as root to access devices not available to regular users. On the other hand, running as root creates problems such as log files owned by root, security concerns etc.

In order to cope with this, irrecord by default drops root privileges after opening the input device. This support is based on that root permissions are accquired using sudo(1) e. g., using

```
$ sudo irrecord --device /dev/lirc0 --driver default
```

If not using sudo, the same behaviour could be accomplished using the SUDO_USER environment variable e. g.,

```
# SUDO_USER=$LOGNAME irrecord --device /dev/lirc0 --driver default
```

The --keep-root option will make irrecord to keep root privileges for the complete run.

FILES

/usr/etc/lirc/lirc_options.conf

The options file holding default values for command line options in the [irrecord] section. For some values including debug, plugindir, driver and device irrecord falls back to the [lircd] section if not found in [irrecord].

-
The location of this file can be changed using the -O/--options-file command-line option or using the environment variable LIRC_OPTIONS_PATH.

~/.cache/irrecord.log

Debug output. Setting the XDG_CACHE_HOME environment variable relocates this file to $XDG_CACHE_HOME/irrecord.log

f) irsend

irsimsend
irtestcase
irxevent
irw
lirc-lsplugins
lirc-lsremotes
lirc-make-devinput
lirc-setup
lircd
lircd-setup
lircd-uinput
lircmd
lircrcd
mode2
xmode2

03) Devices and configuration files

lirc
lircd.conf
lircrc

04) Driver documentation

Technical
How everything works
Writing LIRC user-space drivers
Writing applications for LIRC
Writing LIRC configuration tools
Adding new remote controls
Lirc APIs
User space driver API.
lirc_client API.
lirc library API documentation
Getting additional information

FAQ, mailing list and mailing list archive
Reporting bugs

IV. Trouble Shooting

pi@raspberrypi:~ $ sudo apt-get install lirc
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
lirc is already the newest version (0.10.1-5.2).
0개 업그레이드, 0개 새로 설치, 0개 제거 및 51개 업그레이드 안 함.
1개를 완전히 설치하지 못했거나 지움.
이 작업 후 0 바이트의 디스크 공간을 더 사용하게 됩니다.
계속 하시겠습니까? [Y/n] y
lirc (0.10.1-5.2) 설정하는 중입니다 ...
Job for lircd.service failed because a fatal signal was delivered to the control process.
See "systemctl status lircd.service" and "journalctl -xe" for details.
invoke-rc.d: initscript lircd, action "start" failed.

- lircd.service - Flexible IR remote input/output application support

  Loaded: loaded (/lib/systemd/system/lircd.service; disabled; vendor preset: enabled)
  Active: failed (Result: signal) since Fri 2020-06-26 18:20:11 KST; 29ms ago
  Docs: man:lircd(8)
  http://lirc.org/html/configure.html
  Process: 24081 ExecStart=/usr/sbin/lircd --nodaemon (code=killed, signal=SEGV)
  Main PID: 24081 (code=killed, signal=SEGV)

6월 26 18:20:11 raspberrypi systemd[1]: Starting Flexible IR remote input/output application support...
6월 26 18:20:11 raspberrypi lircd[24081]: Warning: cannot open /etc/lirc/lirc_options.conf
6월 26 18:20:11 raspberrypi lircd-0.10.1[24081]: Info: lircd: Opening log, level: Info
6월 26 18:20:11 raspberrypi lircd-0.10.1[24081]: Notice: Version: lircd 0.10.1
6월 26 18:20:11 raspberrypi lircd-0.10.1[24081]: Notice: System info: Linux raspberrypi 4.19.118-v7l+ #1311 SMP Mon Apr 27 14:26:42 BST 2020 armv7l GNU/Linux
6월 26 18:20:11 raspberrypi systemd[1]: lircd.service: Main process exited, code=killed, status=11/SEGV
6월 26 18:20:11 raspberrypi systemd[1]: lircd.service: Failed with result 'signal'.
6월 26 18:20:11 raspberrypi systemd[1]: Failed to start Flexible IR remote input/output application support.
dpkg: error processing package lirc (--configure):
installed lirc package post-installation script subprocess returned error exit status 1
처리하는데 오류가 발생했습니다:
lirc
E: Sub-process /usr/bin/dpkg returned an error code (1)
pi@raspberrypi:~ $

【소스 프로그램】

```
function calcAge(birth) {
    var date = new Date();
    var year = date.getFullYear();
    var month = (date.getMonth() + 1);
    var day = date.getDate();
    if (month < 10) month = '0' + month;
    if (day < 10) day = '0' + day;
    var monthDay = month + day;
    birth = birth.replace('-', '').replace('-', '');
    var birthdayy = birth.substr(0, 4);
    var birthdaymd = birth.substr(4, 4);
    var age = monthDay < birthdaymd ? year - birthdayy - 1 : year -
birthdayy;
    return age;
}
```