지능형 자동차 개발환경 설정

HYUNDAI AUTRON

목차

HYUNDAI AUTRON

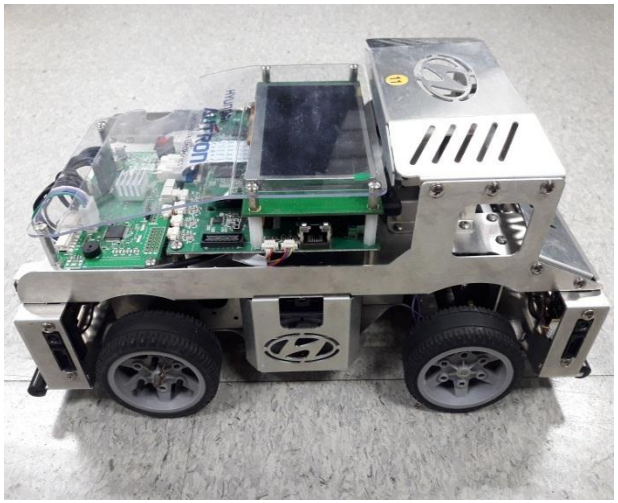
지능형 자동차 개발환경 소개

Host computer

Ethernet

Serial

Target board



개발환경

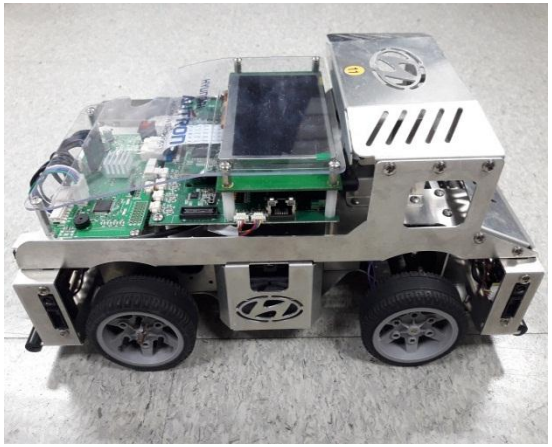
- OS : PC-Linux, target board-embedded linux
- Serial(mini USB cable): target board 터미널 확인
- Ethernet: ssh(시큐어 셸)를 통한 데이터 전송

프로세서	A15 dual core (1.5GHz)
카메라 해상도	1280x720
AD	AD port 6EA (PSD sensor용)
LCD	4.3" (해상도 480x272)
데이터 전송	Ethernet (ssh 이용)
보드간 통신	UART

HYUNDAI AUTRON

지능형 자동차의 Hardware 소개

■ 지능형 자동차 본체



■ 케이블 및 부품류

보드 전원 케이블

이더넷 케이블

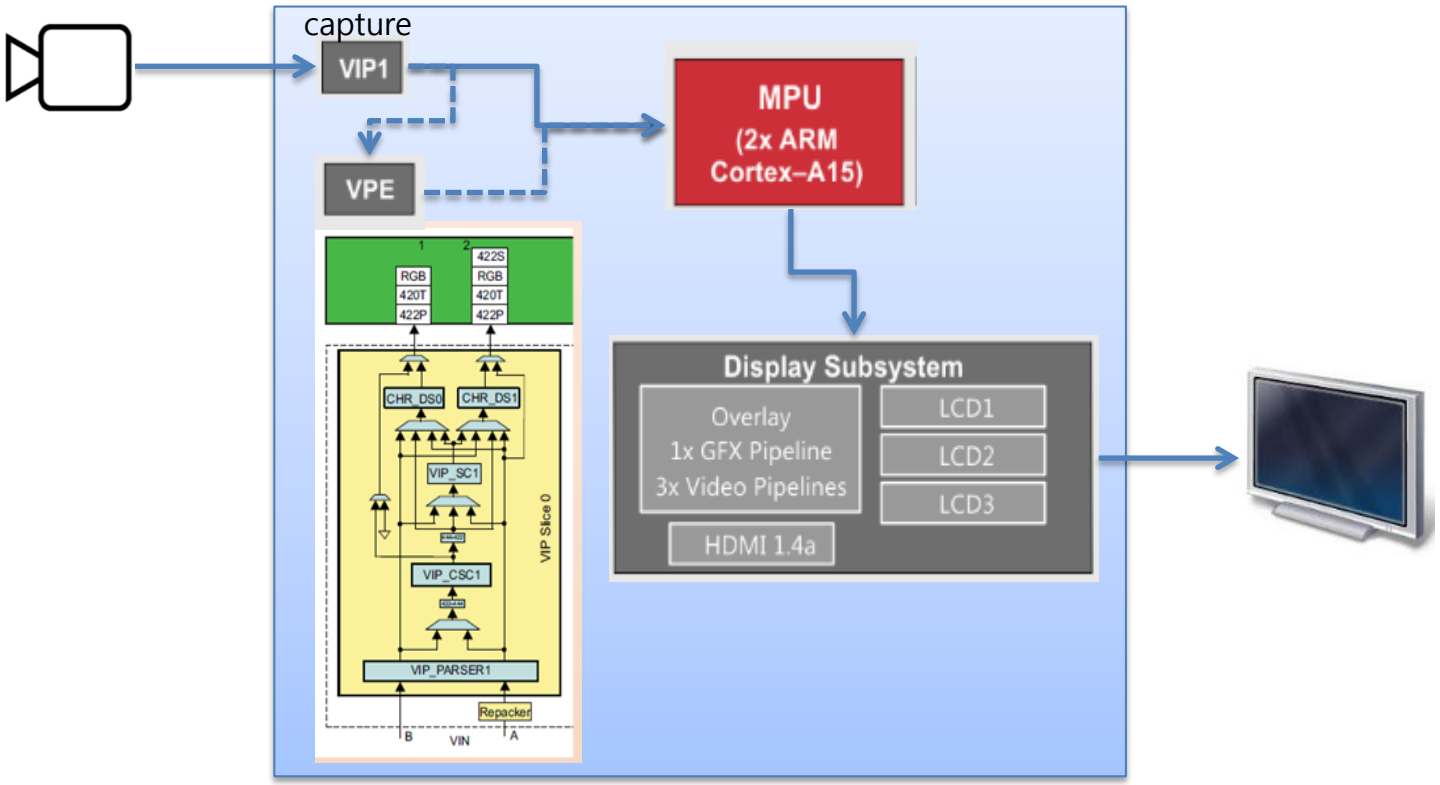
USB 케이블

배터리 충전기

HYUNDAI AUTRON

개발 SoC (System-on-Chip): 영상처리 칩소개

- **코어: A15 dual core. 1.5GHz**
- **영상처리 하드웨어**
  - VPE: Capture한 이미지의 format 변환 및 scaling 지원
  - DISPLAY: 다중 레이어 지원. Scaling 지원



HYUNDAI AUTRON

SDK 소개

■ Application 폴더

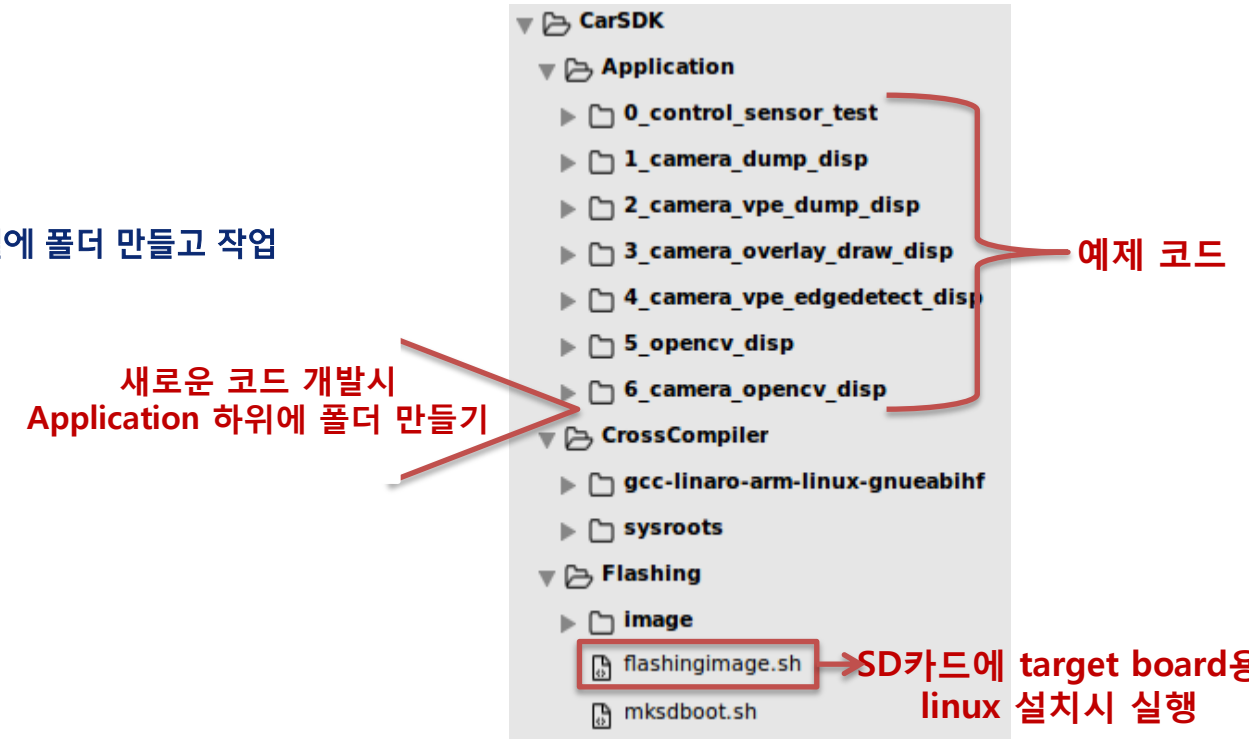
- 예제 코드 들어 있는 폴더
- 코드 개발 및 작업 공간
- 새로운 코드 개발시 Application 밑에 폴더 만들고 작업

■ CrossCompiler 폴더

- crossCompiler 폴더
- 개발 과정에서 접근할 일 없음

■ Flashing 폴더

- embedded linux 설치 폴더
- SD 카드에 embedded linux 설치 및 초기화 원할 때 flashingimage.sh 실행



HYUNDAI AUTRON

목차

HYUNDAI AUTRON

Host PC의 Ubuntu 설정 과정

■ Ubuntu 16.04 64bit 설치 후 툴 설치 리스트

```
# 패키지 인덱스 정보 업데이트
$ sudo apt-get update

# 프로그램 개발에 필요한 헤더파일과 라이브러리 설치
$ sudo apt-get install build-essential

# 크로스컴파일 관련 툴: 타겟 보드용 바이너리 컴파일에 필요
$ sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386 libz1:i386

# minicom 설치: 타겟 보드와 serial 통신하는데 사용
$ sudo apt-get install minicom

# ssh 설치: 타겟 보드와 네트워크를 통한 데이터 전송에 사용
$ sudo apt-get install ssh

# Gparted 설치: SD 카드 초기화에 사용
$ sudo apt-get install gparted

# vim 설치: 에디터(옵션)
$ sudo apt-get install vim

# Sublime text 설치: 에디터(옵션)
$ sudo add-apt-repository ppa:webupd8team/sublime-text-3
$ sudo apt-get update
$ sudo apt-get install sublime-text-installer

#기타 유용한 툴 설치
$ sudo apt-get install git python diffstat texinfo gawk chrpath dos2unix wget unzip socat
doxygen
```

Host PC의 Ubuntu 설정 과정

■ CarSDK 복사 후, 압축 풀기

```
# CarSDK를 본인 PC의 희망경로에 복사해 온 다음, 압축 풀기
$ tar -xzvf CarSDK.tar.gz
```

Host PC와 target board간의 통신 설정 목차

- [Host PC] minicom 설정 (serial 통신 용)
  - Target board와 Host PC를 USB 연결후 장치 이름 확인
  - minicom 설정 들어가서 통신 환경 설정

- [Host PC] 이더넷 설정 (scp 를 이용한 데이터 전송용)
  - 고정 IP 설정
  - 보드와 PC간의 통신 확인 방법
  - 보드 부팅 후 이더넷 통신 안될 경우 조치 방법
  - 데이터 전송 방법

HYUNDAI AUTRON

minicom 설정 (1/3)

■ Target board와 Host PC를 USB 연결후 장치 이름 확인

- USB 케이블 연결 전후 /dev/tty* 의 list 확인

USB 연결 전

```
at@ubuntu:~$ ls /dev/tty*
/dev/tty     /dev/tty23  /dev/tty39  /dev/tty54  /dev/ttyS11  /dev/ttyS27
/dev/tty0    /dev/tty24  /dev/tty4   /dev/tty55  /dev/ttyS12  /dev/ttyS28
/dev/tty1    /dev/tty25  /dev/tty40  /dev/tty56  /dev/ttyS13  /dev/ttyS29
/dev/tty10   /dev/tty26  /dev/tty41  /dev/tty57  /dev/ttyS14  /dev/ttyS3
/dev/tty11   /dev/tty27  /dev/tty42  /dev/tty58  /dev/ttyS15  /dev/ttyS30
/dev/tty12   /dev/tty28  /dev/tty43  /dev/tty59  /dev/ttyS16  /dev/ttyS31
/dev/tty13   /dev/tty29  /dev/tty44  /dev/tty6   /dev/ttyS17  /dev/ttyS4
/dev/tty14   /dev/tty3   /dev/tty45  /dev/tty60  /dev/ttyS18  /dev/ttyS5
/dev/tty15   /dev/tty30  /dev/tty46  /dev/tty61  /dev/ttyS19  /dev/ttyS6
/dev/tty16   /dev/tty31  /dev/tty47  /dev/tty62  /dev/ttyS2   /dev/ttyS7
/dev/tty17   /dev/tty32  /dev/tty48  /dev/tty63  /dev/ttyS20  /dev/ttyS8
/dev/tty18   /dev/tty33  /dev/tty49  /dev/tty7   /dev/ttyS21  /dev/ttyS9
/dev/tty19   /dev/tty34  /dev/tty5   /dev/tty8   /dev/ttyS22  /dev/ttyprintk
/dev/tty2    /dev/tty35  /dev/tty50  /dev/tty9   /dev/ttyS23
/dev/tty20   /dev/tty36  /dev/tty51  /dev/ttyS0  /dev/ttyS24
/dev/tty21   /dev/tty37  /dev/tty52  /dev/ttyS1  /dev/ttyS25
/dev/tty22   /dev/tty38  /dev/tty53  /dev/ttyS10 /dev/ttyS26
at@ubuntu:~$
```

USB 연결 후

```
at@ubuntu:~$ ls /dev/tty*
/dev/tty     /dev/tty23  /dev/tty39  /dev/tty54  /dev/ttyS11  /dev/ttyS27
/dev/tty0    /dev/tty24  /dev/tty4   /dev/tty55  /dev/ttyS12  /dev/ttyS28
/dev/tty1    /dev/tty25  /dev/tty40  /dev/tty56  /dev/ttyS13  /dev/ttyS29
/dev/tty10   /dev/tty26  /dev/tty41  /dev/tty57  /dev/ttyS14  /dev/ttyS3
/dev/tty11   /dev/tty27  /dev/tty42  /dev/tty58  /dev/ttyS15  /dev/ttyS30
/dev/tty12   /dev/tty28  /dev/tty43  /dev/tty59  /dev/ttyS16  /dev/ttyS31
/dev/tty13   /dev/tty29  /dev/tty44  /dev/tty6   /dev/ttyS17  /dev/ttyS4
/dev/tty14   /dev/tty3   /dev/tty45  /dev/tty60  /dev/ttyS18  /dev/ttyS5
/dev/tty15   /dev/tty30  /dev/tty46  /dev/tty61  /dev/ttyS19  /dev/ttyS6
/dev/tty16   /dev/tty31  /dev/tty47  /dev/tty62  /dev/ttyS2   /dev/ttyS7
/dev/tty17   /dev/tty32  /dev/tty48  /dev/tty63  /dev/ttyS20  /dev/ttyS8
/dev/tty18   /dev/tty33  /dev/tty49  /dev/tty7   /dev/ttyS21  /dev/ttyS9
/dev/tty19   /dev/tty34  /dev/tty5   /dev/tty8   /dev/ttyS22  /dev/ttyUSB0
/dev/tty2    /dev/tty35  /dev/tty50  /dev/tty9   /dev/ttyS23  /dev/ttyprintk
/dev/tty20   /dev/tty36  /dev/tty51  /dev/ttyS0  /dev/ttyS24
/dev/tty21   /dev/tty37  /dev/tty52  /dev/ttyS1  /dev/ttyS25
/dev/tty22   /dev/tty38  /dev/tty53  /dev/ttyS10 /dev/ttyS26
```

USB serial 케이블의 장치 이름 나타남

HYUNDAI AUTRON

minicom 설정 (2/3)

■ minicom 설정 들어가서 통신 환경 설정

```
$ sudo minicom -s
```

```
at@ubuntu:~$ sudo minicom -s
[sudo] password for at:
```

1. Serial port setup 선택

```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
| Exit from Minicom        |
+--------------------------+
```

2. Serial Device 이름 앞에서 확인한 이름으로 변경, 통신속도, 흐름제어 다음과 같이 설정

```
+-----------------------------------------------------------------------+
| A -    Serial Device      : /dev/tty8                                 |
| B - Lockfile Location     : /var/lock                                 |
| C -   Callin Program      :                                           |
| D -  Callout Program      :                                           |
| E -    Bps/Par/Bits       : 115200 8N1                                |
| F - Hardware Flow Control : Yes                                       |
| G - Software Flow Control : No                                        |
|                                                                       |
|    Change which setting?                                              |
+-----------------------------------------------------------------------+
        | Screen and keyboard      |
        | Save setup as dfl        |
        | Save setup as..          |
        | Exit                     |
        | Exit from Minicom        |
        +--------------------------+
```

```
+-----------------------------------------------------+
| A -    Serial Device      : /dev/ttyUSB0            |
| B - Lockfile Location     : /var/lock               |
| C -   Callin Program      :                         |
| D -  Callout Program      :                         |
| E -    Bps/Par/Bits       : 115200 8N1              |
| F - Hardware Flow Control : No                      |
| G - Software Flow Control : No                      |
|                                                     |
|    Change which setting?                            |
+-----------------------------------------------------+
```

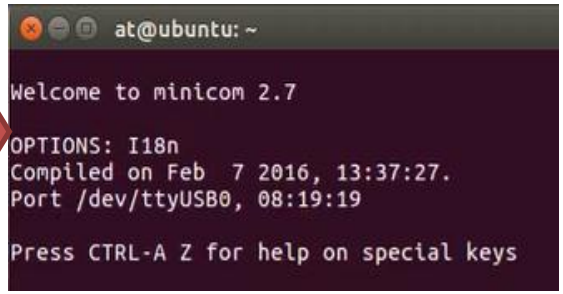

HYUNDAI AUTRON

minicom 설정 (3/3)

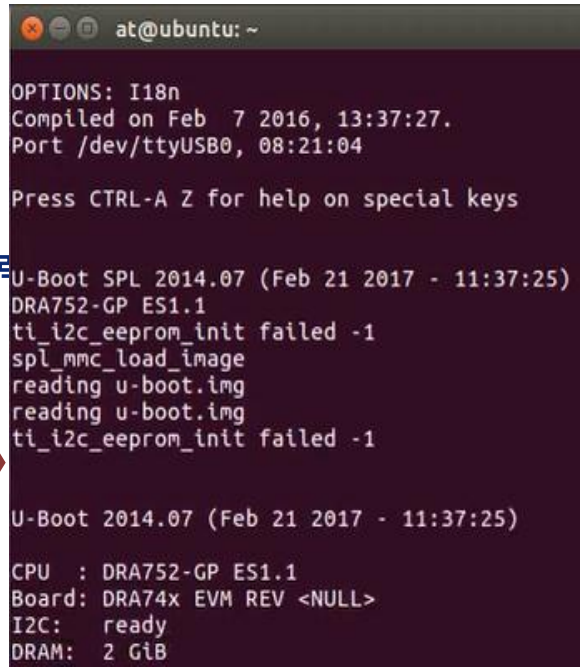
■ minicom 설정 들어가서 통신 환경 설정

3. Save setup as dfl 선택: default 세팅으로 저장

4. Exit 선택: 설정한 환경으로 minicom 실행 됨 → 보드 전원 On 시 부팅 로그 출력



보드 On



■ 향후에 다시 mincom을 실행 시킬경우 세팅하지 않고 아래 명령어로 바로 실행가능

```
$ sudo minicom
```


HYUNDAI AUTRON

이더넷 설정 (1/3) – 고정 IP 설정 방법

1. network 이름 확인: $ ifconfig

```
at@ubuntu: ~
at@ubuntu:~$ ifconfig
enp2s0    Link encap:Ethernet  HWaddr 98:83:89:27:da:87
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:109 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:18204 (18.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:5380 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5380 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:433280 (433.2 KB)  TX bytes:433280 (433.2 KB)

at@ubuntu:~$
```

2. "/etc/network/interfaces" 파일 편집: $ sudo vi /etc/network/interfaces

3. 아래와 같이 고정 IP 설정 후 저장

```
auto enp2s0
iface enp2s0 inet static
        address 10.10.70.1
        netmask
255.255.255.0
```

이름 다를 수 있음
ifconfig로 확인한 이름 입력

```
at@ubuntu:~$ sudo vi /etc/network/interfaces
[sudo] password for at:
```

```
at@ubuntu: ~
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto enp2s0
iface enp2s0 inet static
        address 10.10.70.1
        netmask 255.255.255.0
```

4. Host PC 재부팅

HYUNDAI AUTRON

이더넷 설정 (2/3) – Target board와 Host PC간의 통신 확인 방법

■ **Host PC → target board**

$ ping 10.10.70.4

```
at@ubuntu:~$ ping 10.10.70.4
PING 10.10.70.4 (10.10.70.4) 56(84) bytes of data.
64 bytes from 10.10.70.4: icmp_seq=1 ttl=64 time=0.114 ms
64 bytes from 10.10.70.4: icmp_seq=2 ttl=64 time=0.127 ms
64 bytes from 10.10.70.4: icmp_seq=3 ttl=64 time=0.085 ms
64 bytes from 10.10.70.4: icmp_seq=4 ttl=64 time=0.086 ms
^C
--- 10.10.70.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3060ms
rtt min/avg/max/mdev = 0.085/0.103/0.127/0.018 ms
at@ubuntu:~$
```

■ **Target board → Host PC (minicom에서 실행)**

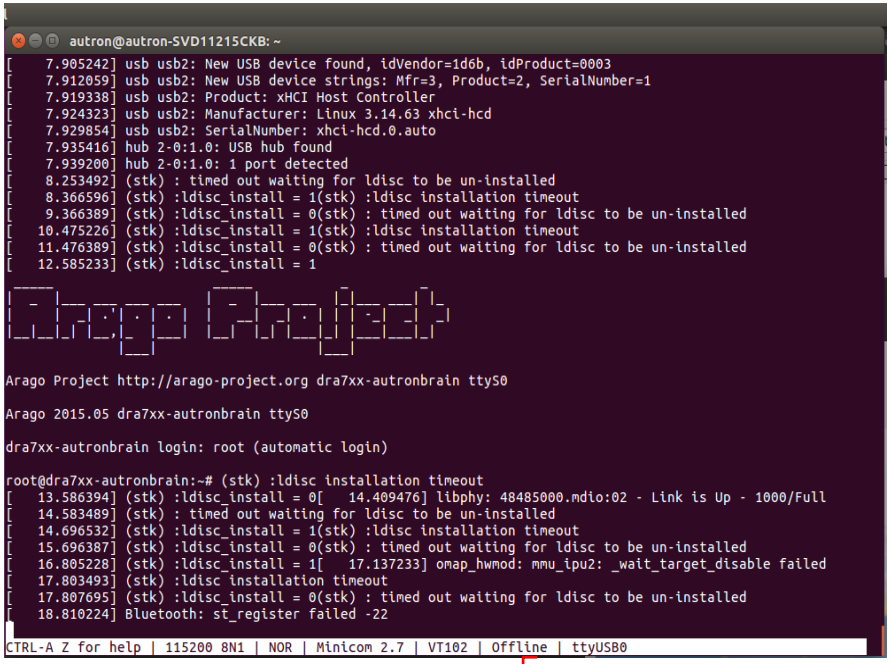
ping 10.10.70.1

```
root@dra7xx-autronbrain:~# ping 10.10.70.1
PING 10.10.70.1 (10.10.70.1): 56 data bytes
64 bytes from 10.10.70.1: seq=0 ttl=64 time=0.477 ms
64 bytes from 10.10.70.1: seq=1 ttl=64 time=0.589 ms
64 bytes from 10.10.70.1: seq=2 ttl=64 time=0.433 ms
64 bytes from 10.10.70.1: seq=3 ttl=64 time=0.155 ms
^C
--- 10.10.70.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.155/0.413/0.589 ms
root@dra7xx-autronbrain:~#
```


HYUNDAI AUTRON

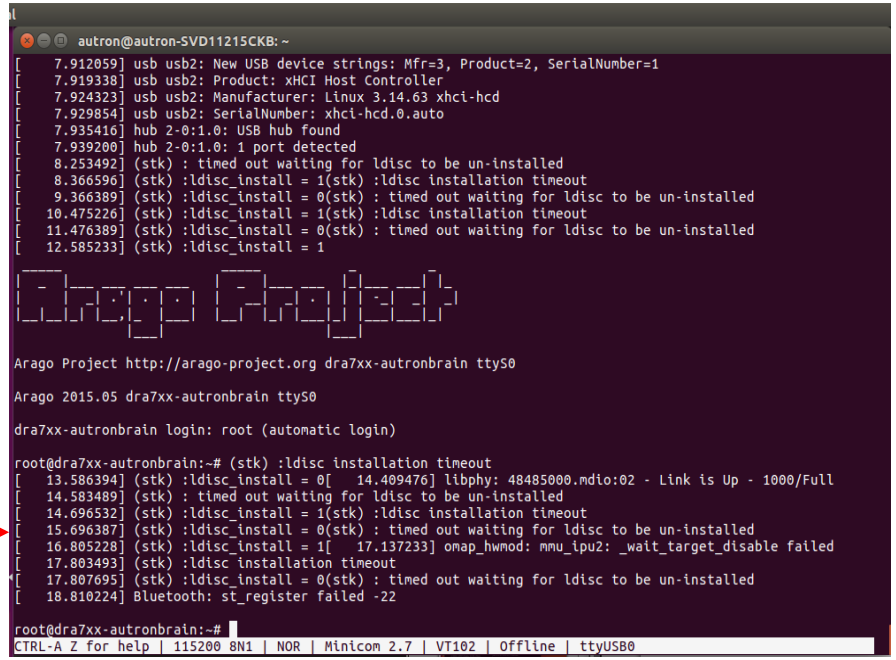
이더넷 설정 (2/3) – Target board와 Host PC간의 통신 확인 방법

■ 정상 부팅 확인



Warning 무시하고,

Enter 입력시, 커맨드라인으로 넘어감.



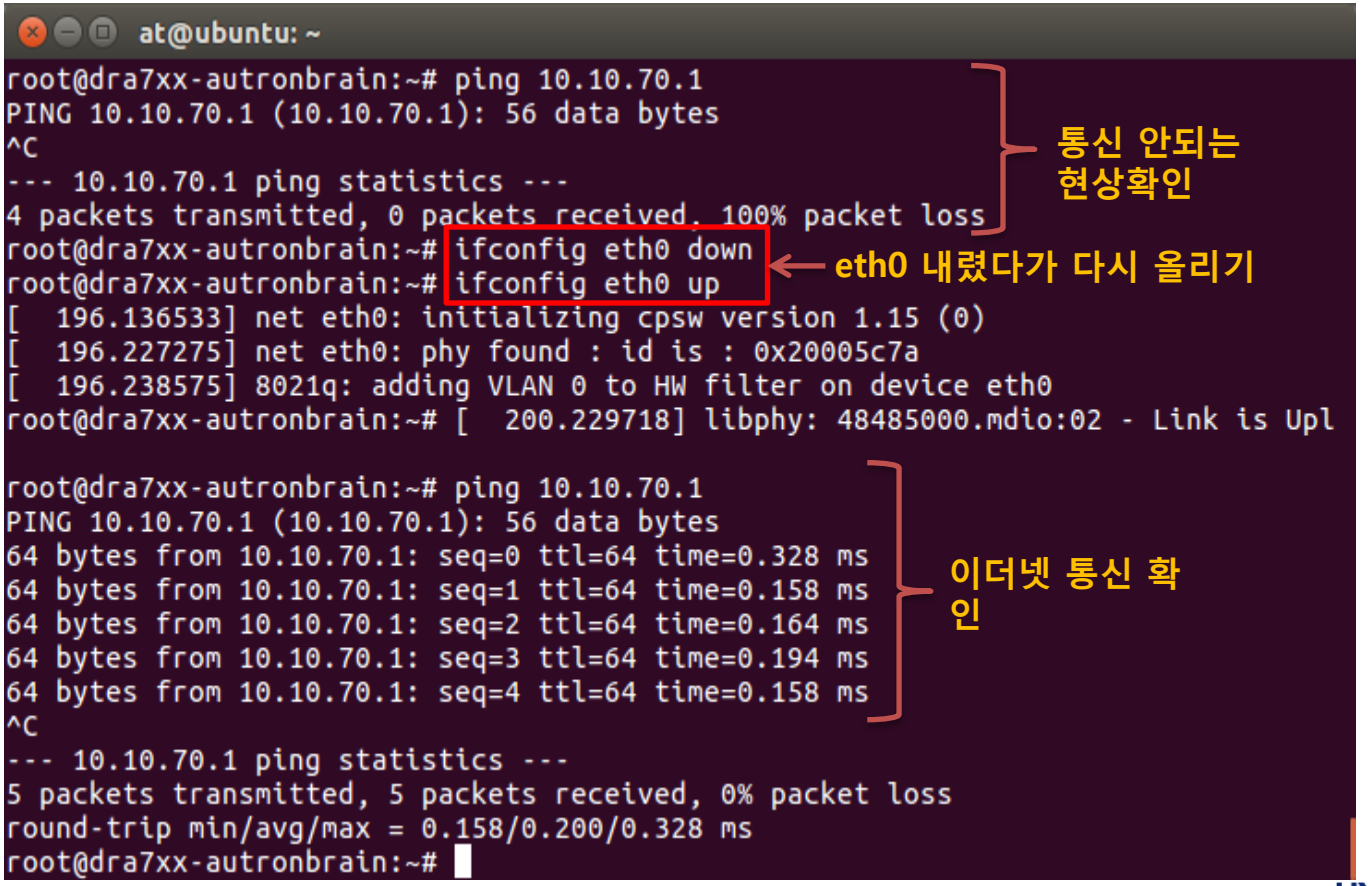
HYUNDAI AUTRON

이더넷 설정 (3/3) – 이더넷 통신 안될 때

■ 확인사항

- Host PC 고정 IP 세팅 확인
- 이더넷 케이블 연결 확인

■ 세팅에 문제 없는데도 ping 안될 때는 target board에서 eth0 내렸다가 다시 올리기 (minicom에서 실행)



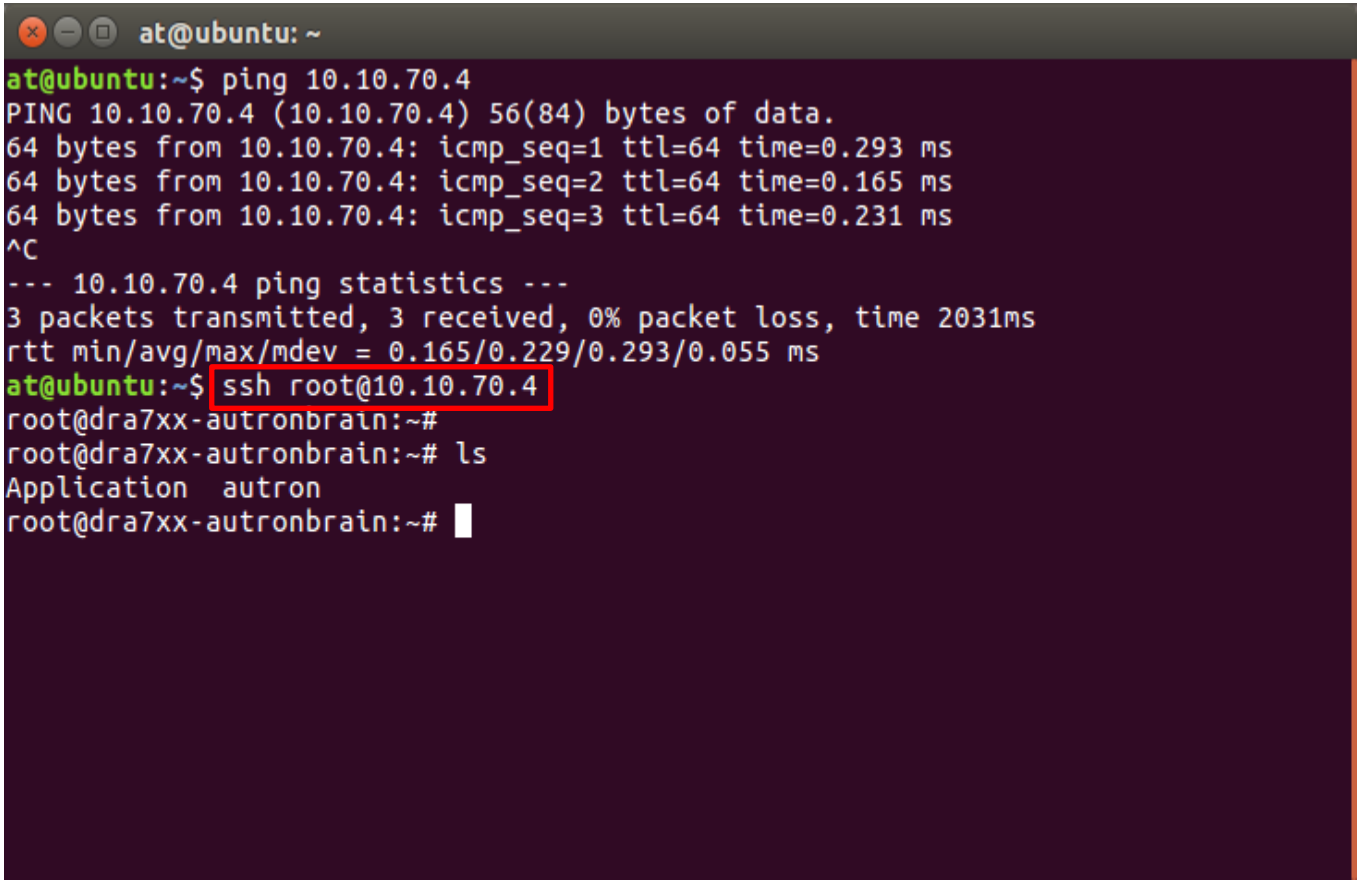
HYUNDAI AUTRON

[참고] 이더넷을 통한 target 보드 SSH 접속 방법

■ SSH 접속 방법 (serial cable 연결 및 minicom 실행 필요 없이 동일한 기능 수행)

- 단 target board 부팅 후 ping 테스트가 정상적일 때 사용가능
- 종료시에는 exit 커맨드 사용

$ ssh root@10.10.70.4

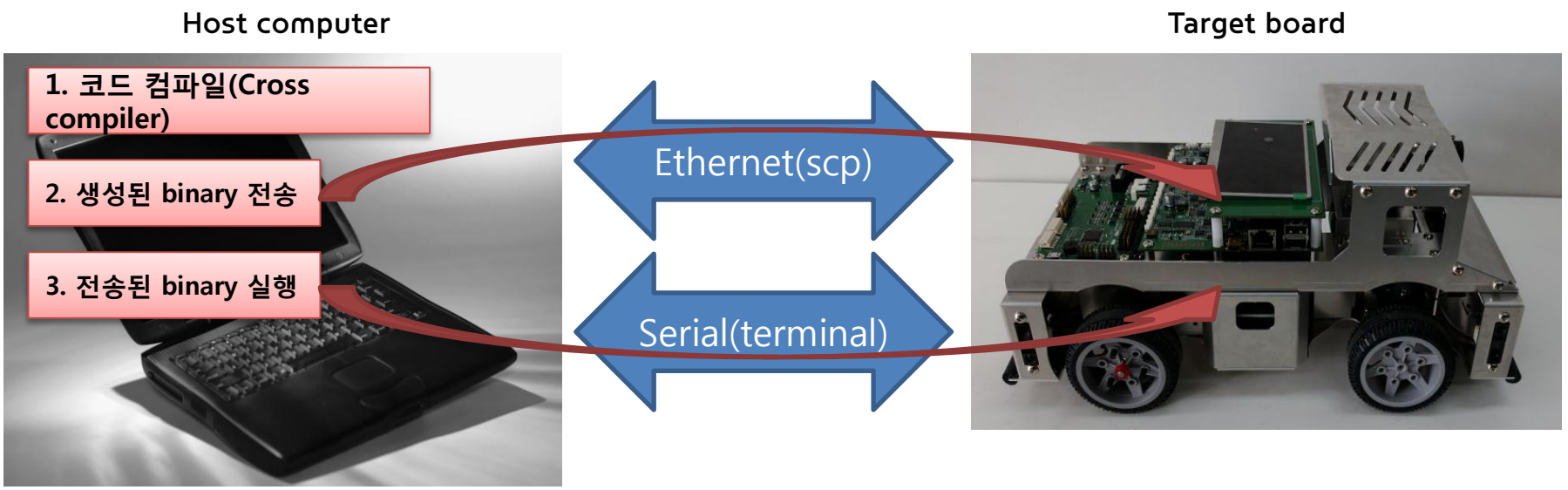


HYUNDAI AUTRON

목차

HYUNDAI AUTRON

지능형 자동차 개발순서



- **Host computer 와 Target board 부팅 완료후 (serial 및 ethernet 연결 상태 확인)**
- **Host computer**
  1. 코드 컴파일 (CarSDK/Application 폴더)
  2. 생성된 Binary를 Target board로 전송 (scp 명령어 사용. Ethernet으로 전송)
- **Target 보드**
  3. 전송된 bianry 실행 (serial로 제어)
  4. 필요에 따라 생성된 디버깅 로그나 이미지 Host computer로 전송

HYUNDAI AUTRON

코드 컴파일 및 실행 방법

■ 순서: (@PC) 컴파일 → (@PC) target board로 바이너리 전송 → (@보드)전송된 바이너리 실행

1. [Host PC] 해당 예제 폴더로 이동하여 “make” 입력

```
at@ubuntu:~$ cd CarSDK/Application/0_control_sensor_test/
at@ubuntu:~/CarSDK/Application/0_control_sensor_test$ ls
Makefile  car_lib.c  car_lib.h  controlSensor.c
at@ubuntu:~/CarSDK/Application/0_control_sensor_test$ make
/home/at/CarSDK/CrossCompiler/gcc-linaro-arm-linux-gnueabihf/bin/arm-linux-gnuea
bihf-gcc     -c -o controlSensor.o controlSensor.c
/home/at/CarSDK/CrossCompiler/gcc-linaro-arm-linux-gnueabihf/bin/arm-linux-gnuea
bihf-gcc -o controlSensor controlSensor.o car_lib.c
at@ubuntu:~/CarSDK/Application/0_control_sensor_test$ ls
Makefile  car_lib.c  car_lib.h  controlSensor  controlSensor.c  controlSensor.o
at@ubuntu:~/CarSDK/Application/0_control_sensor_test$
```

생성된 바이너리

2. [Host PC] 생성된 바이너리 scp 명령어로 target board로 전송

```
at@ubuntu:~/CarSDK/Application/0_control_sensor_test$ scp controlSensor root@10.10.70.4:/home/root/
controlSensor                                 100%   14KB  13.8KB/s   00:00
at@ubuntu:~/CarSDK/Application/0_control_sensor_test$
```

3. [Target board] 전송된 바이너리 실행

```
root@dra7xx-autronbrain:~# ls
Application  autron
root@dra7xx-autronbrain:~# ls
Application  autron  controlSensor
root@dra7xx-autronbrain:~# ./controlSensor
```

전송된 바이너리

HYUNDAI AUTRON

이더넷을 통한 데이터 전송 방법

■ Host PC → Target board

- $ scp [파일명] root@10.10.70.4:/home/root/[저장경로]
- 예시) Host PC에서 target 보드로 바이너리 전송

```
at@ubuntu:~/CarSDK/Application/0_control_sensor_test$ scp controlSensor root@10.10.70.4:/home/root
controlSensor                                 100%   14KB  13.8KB/s   00:00
at@ubuntu:~/CarSDK/Application/0_control_sensor_test$
```

```
root@dra7xx-autronbrain:~# ls
Application  autron
root@dra7xx-autronbrain:~# ls
Application  autron  controlSensor
root@dra7xx-autronbrain:~#
```

← 전송된 바이너리

■ Target board → Host PC (minicom에서 실행)

- # scp [파일명] [PC이름]@10.10.70.1:/home/[PC이름]/[저장경로]
- 예시) target board에서 Host PC로 dump한 이미지 파일 전송

```
root@dra7xx-autronbrain:~/Application/1_camera_dump_disp# scp dump_20000101_010156.uyvy at@10.10.70.1:/home/at/download
at@10.10.70.1's password:
dump_20000101_010156.uyvy                     100% 1800KB   1.8MB/s   00:00
root@dra7xx-autronbrain:~/Application/1_camera_dump_disp#
```

```
at@ubuntu:~$ cd download/
at@ubuntu:~/download$ ls
at@ubuntu:~/download$ ls
dump_20000101_010156.uyvy
at@ubuntu:~/download$
```

← 전송된 이미지 파일

HYUNDAI AUTRON

Optimization 옵션 조정

- Edge Detection을 직접 구현한 4번 예제의 경우 Optimization level에 따른 연산속도 차이를 확인할 수 있음
- Makefile 수정시 make clean 후 make 해야 binary 에 정상적으로 수정사항 반영됨.
- 예시) 4번 Edge Detection 예제의 Makefile

Makefile

```
14  INC += -I$(ROOTFS)/usr/include/gbm
15  LIBDIR := $(ROOTFS)/usr/lib
16
17
18  CFLAGS := -O1 -g -Wall -fPIC -mfloat-abi=hard -mfpu=neon -Wl,-rpath,$(ROOTF
19  CXXFLAGS = -Wall -ansi -g -fPIC -mfloat-abi=hard -mfpu=neon $(INC) -I$(ROOT
20
21  LDFLAGS = -lm -lpthread -L$(LIBDIR) -lrt -ldrm -lmtdev -ldrm_omap -lstdc++
22  TARGET = camera_vpe_edgedect_disp
```

```
18  CFLAGS := -O3
```

O1 옵션을 O3로 수정 하고, 다음과 같이 재빌드 하여 target 보드에서 실행시 연산시간 차이 확인 할 수 있음

```
$ make clean
$ make
```


HYUNDAI AUTRON

부팅 후 코드 자동실행 방법

■ Targe 보드의 부팅시 시작되는 기본 경로에서 “.profile” 파일 생성 및 편집

- vi 에디터 이용 파일 열어서 편집 (인터넷에서 vi 에디터 기본 사용법 익힌 후에 편집)

```
root@dra7xx-autronbrain:~# pwd
/home/root
root@dra7xx-autronbrain:~# ls
Application  autron
root@dra7xx-autronbrain:~# vi .profile
```

- 실행하고자 하는 파일(파일 경로 포함) 실행 명령어 입력

```
./Application/1_camera_dump_disp/camera_dump_disp
~
~
~
~
I .profile [Modified] 1/1 100%
```

- 파일 저장 후 재부팅

```
./Application/1_camera_dump_disp/camera_dump_disp
~
~
~
~
:wq
```

```
root@dra7xx-autronbrain:~# reboot
```

- 생성된 파일 확인 방법

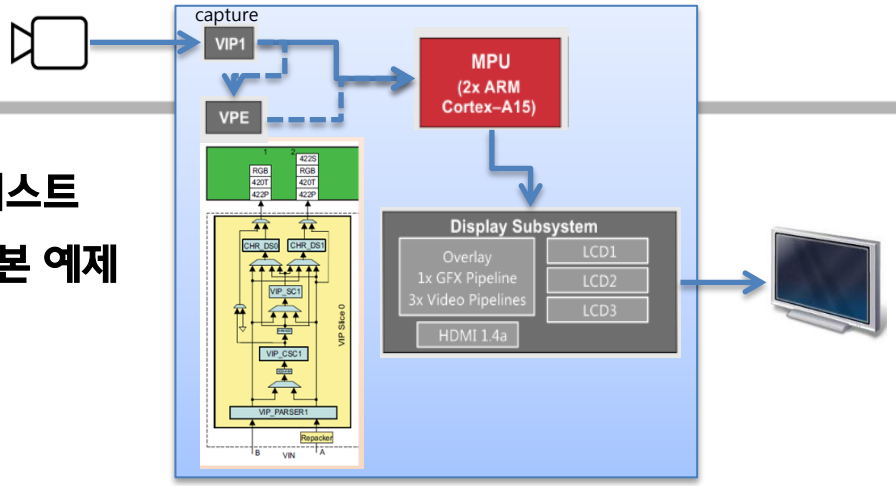
```
root@dra7xx-autronbrain:~# ls -a
.  ..  .bash_history  .profile  Application  autron
root@dra7xx-autronbrain:~#
```


HYUNDAI AUTRON

목차

HYUNDAI AUTRON

예제 코드 요약



- **0_control_sensor_test: 차량 제어 및 센서 입력 테스트**
- **1_camera_dump_disp: 카메라 입력, LCD 출력 기본 예제**
  - 카메라 입력 → LCD 출력 (키 입력시 이미지 저장)
- **2_camera_vpe_dump_disp: VPE 사용 영상 변환**
  - 카메라 입력 → VPE → LCD 출력 (키 입력시 이미지 저장)
- **3_camera_overlay_draw_disp: 다중 레이어 사용 점/선/텍스트 overlay**
  - 카메라 입력 → LCD 출력 (키 입력시 이미지 저장)
  - 점/선/텍스트 입력 → LCD 출력
- **4_camera_vpe_edgedetect_disp: 영상 처리 알고리즘 예제**
  - 카메라 입력 → VPE → Edge Detection → LCD 출력 (키 입력시 이미지 저장)
  - Edge Detection 연산 시간 → LCD 출력
- **5_opencv_disp: 그림 파일 OpenCV 처리**
  - 그림 파일 → OpenCV → LCD 출력 (키 입력시 이미지 저장)
  - OpenCV 처리 시간 → LCD 출력
- **6_camera_opencv_disp: 카메라 입력 데이터 OpenCV 처리**
  - 카메라 입력 → VPE → Hough Transform (OpenCV 사용) → LCD 출력 (키 입력시 이미지 저장)
  - OpenCV 처리 시간 → LCD 출력

HYUNDAI AUTRON

영상 입출력 관련 주요 API 설명(1/2)

■ Camera 데이터 처리 관련

- v4l2_open: 영상 capture를 위한 초기화.
- v4l2_reqbufs: 영상 저장할 큐(queue) 버퍼 메모리 할당
- v4l2_streamon: 영상 capture 시작
- v4l2_qbuf: 영상 queue 처리 권한을 driver에게 이전하여 다음 영상 프레임 요청

  즉, application에서 driver 쪽으로 소유권 이전 → 카메라에서 영상 들어오면 driver가 queue 에 데이터 저장
- v4l2_dqbuf: 영상 queue 처리 권한을 application이 가지고 와서 입력된 영상 프레임의 버퍼 인덱스를 사용하여 프레임 처리

  즉, driver에서 application 쪽으로 소유권 이전 → application이 영상 queue 버퍼에서 자료를 꺼내는 작업 실행 가능

■ Display 처리 관련

- disp_open: 영상 출력을 위한 초기화.
- disp_get_vid_buffers: 영상 출력을 위한 버퍼 할당
- disp_post_vid_buffer: 영상 출력 버퍼에 영상 데이터 입력

영상 입출력 관련 주요 API 설명(2/2)

■ VPE 처리 관련

- vpe_open: VPE 하드웨어 사용을 위한 위한 초기화.
- vpe_input_init: VPE 입력 초기화
- vpe_output_init: VPE 출력 초기화
- vpe_output_fullscreen: scaling 여부 설정
- vpe_stream_on: VPE 하드웨어 입력 또는 출력 stream on 시킴

  vpe_stream_on(vpe->fd, V4L2_BUF_TYPE_VIDEO_CAPTURE_MPLANE): VPE 하드웨어에 출력 영상 데이터 흐름 On

  vpe_stream_on(vpe->fd, V4L2_BUF_TYPE_VIDEO_OUTPUT_MPLANE): VPE 하드웨어에 입력 영상 데이터 흐름 On
- vpe_input_qbuf: VPE 입력 queue 처리 권한을 driver에게 이전 (application에서 driver 쪽으로 소유권 이전)

  → 이후 VPE driver가 입력 queue에 있는 영상 접근하여 가공 가능
- vpe_input_dqbuf: VPE 입력 queue 처리 권한을 application이 가지고 옴(driver에서 application 쪽으로 소유권 이전)

  → 이후 VPE 입력 queue 에서 가공할 이미지 저장
- vpe_output_qbuf: VPE 출력 queue 처리 권한을 driver에게 이전 (application에서 driver 쪽으로 소유권 이전)

  → 이후 VPE driver가 영상 가공 후 출력 queue 에 데이터 저장
- vpe_output_dqbuf: VPE 출력 queue 처리 권한을 application이 가지고 옴(driver에서 application 쪽으로 소유권 이전)

  → 이후 VPE 처리된 출력 queue 에서 자료를 꺼내는 작업 실행 가능

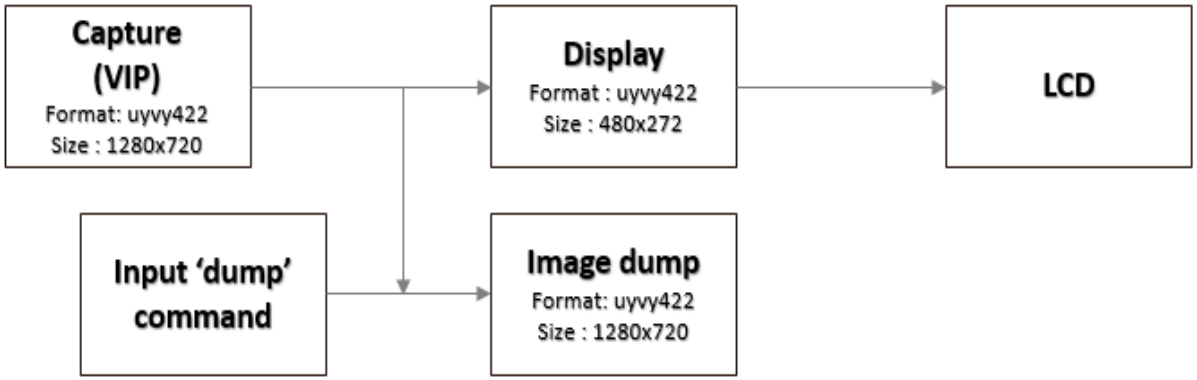
0_control_sensor_test

■ 차량 제어 및 센서 입력 테스트 예제

- 전조등, 후미등, Beep 제어
- 바퀴 위치 제어
- 바퀴 속도 제어
- 조향 제어
- 카메라 위치 제어
- Line sensor 값 읽기
- 적외선 센서 값 읽기

1_camera_dump_disp

■ 카메라 입력 영상의 LCD 출력 및 이미지 저장



쉘 명령어 (on Target board)	실행 내용
cd Application/1_camera_dump_disp	1_camera_dump_disp 폴더로 이동
./camera_dump_disp	해당 example 실행 및 camera preview 확인
dump + enter	1280x720, uyvy format 의 camera image dump. dump_yyyymmdd_hhmmss.uyvy file 로 저장

HYUNDAI AUTRON

1_camera_dump_disp – Target 보드에서 dump한 이미지 확인 방법

■ YUV/RGB Viewer: vooya 설치

- Download: http://www.offminor.de/downloads.html
- Vooya의 deb 파일 다운 받은 후 Install 방법

```
at@ubuntu:~/temp$ sudo dpkg -i *.deb
[sudo] password for at:
Selecting previously unselected package vooya.
(데이터베이스 읽는중 ...현재 183021개의 파일과 디렉터리가 설치되어 있습니다.)
Preparing to unpack vooya-1.8-amd64.deb ...
Unpacking vooya (1.8-0) ...
vooya (1.8-0) 설정하는 중입니다 ...
Processing triggers for shared-mime-info (1.5-2ubuntu0.1) ...
Processing triggers for gnome-menus (3.13.3-6ubuntu3.1) ...
Processing triggers for desktop-file-utils (0.22-1ubuntu5.1) ...
Processing triggers for bamfdaemon (0.5.3~bzr0+16.04.20160824-0ubuntu1) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for mime-support (3.59ubuntu1) ...
Processing triggers for man-db (2.7.5-1) ...
at@ubuntu:~/temp$
```

■ Target 보드에서Host PC로 dump 이미지 전송후에 vooya로 확인

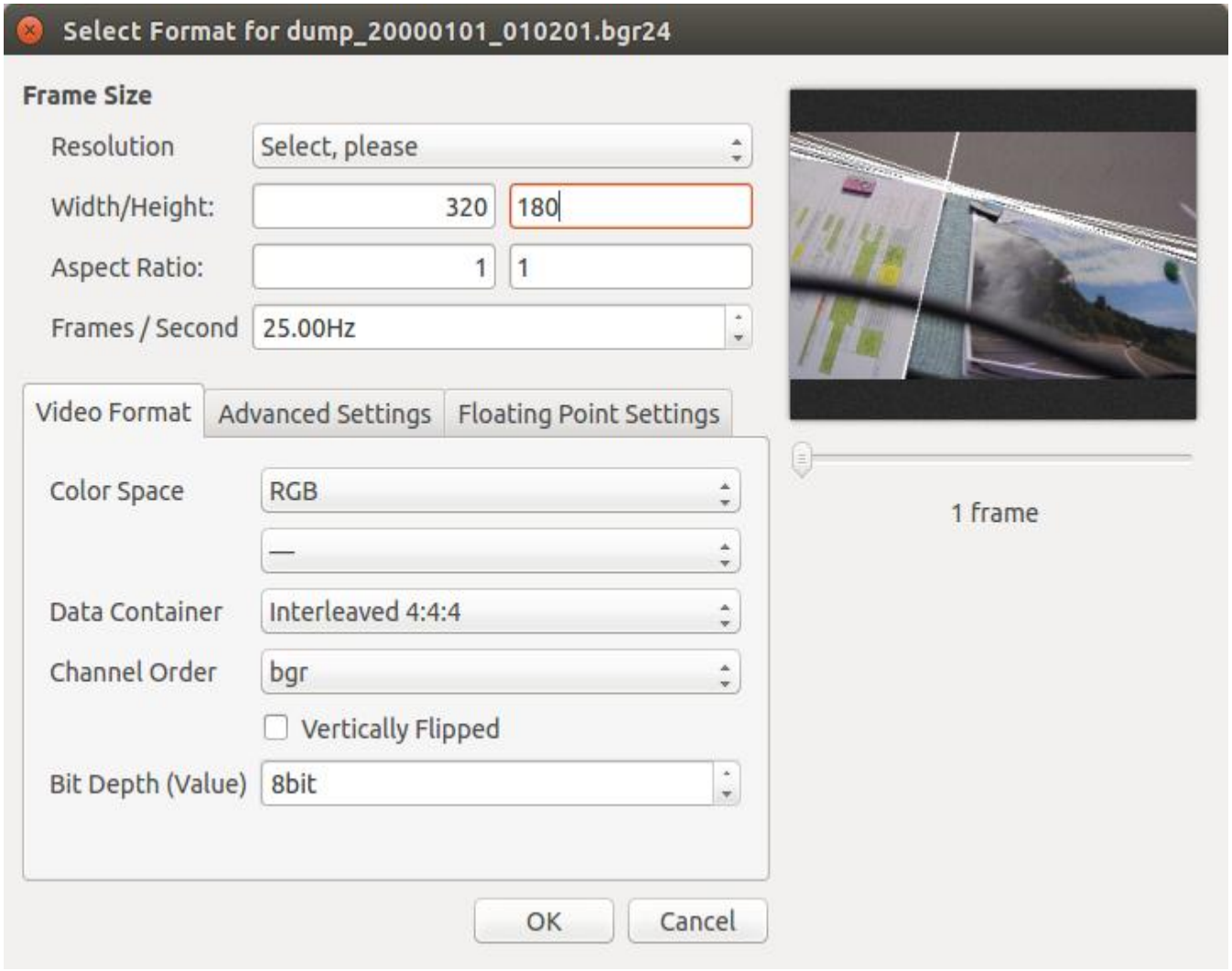
■ 설치후 vooya 실행방법

Vooya 입력 후 클릭

HYUNDAI AUTRON

1_camera_dump_disp – 이미지 로드시 vooya 설정

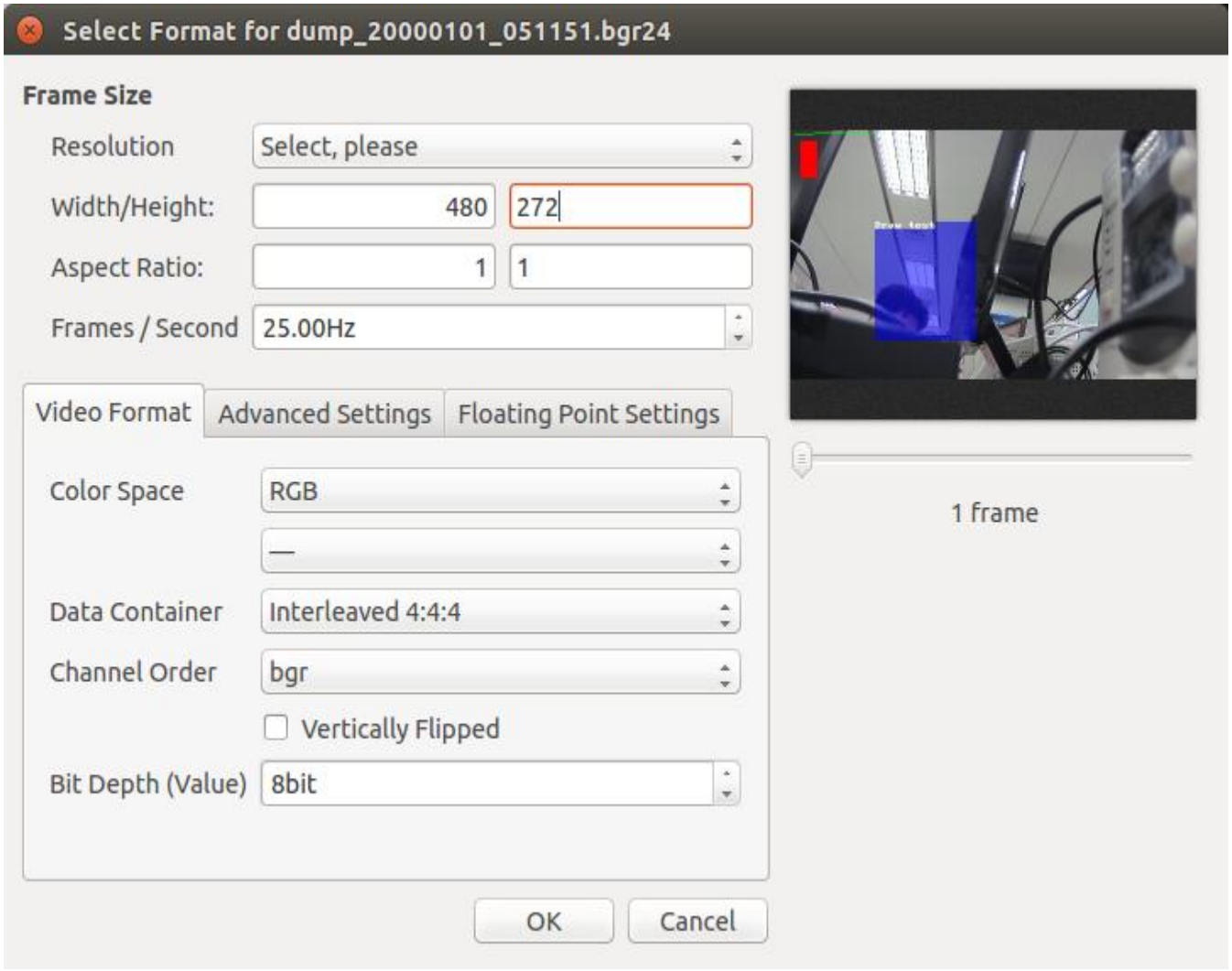
■ 영상크기: 320x180, 영상포맷: bgr24



HYUNDAI AUTRON

1_camera_dump_disp – 이미지 로드시 vooya 설정

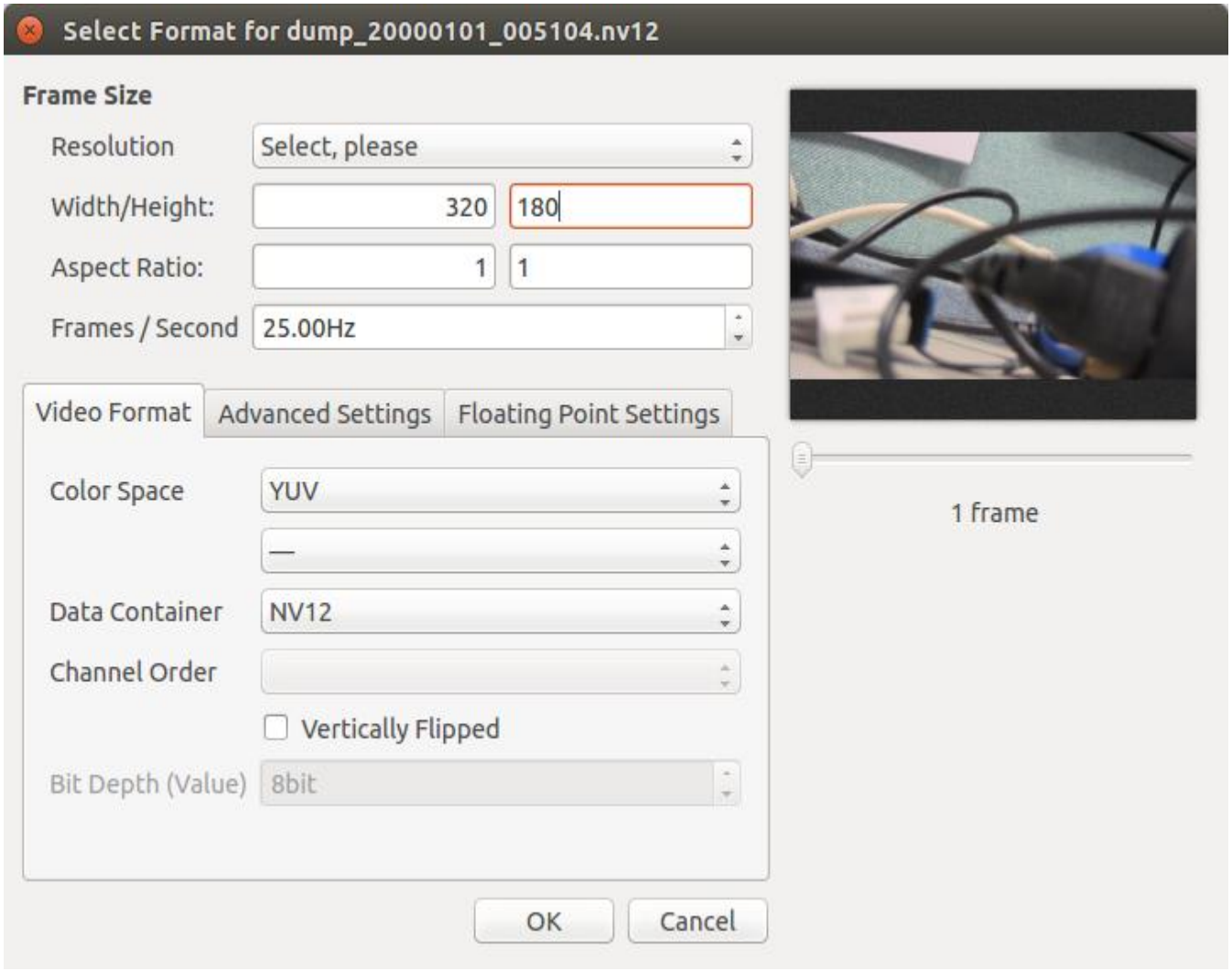
■ 영상크기: 480x272, 영상포맷: bgr24



HYUNDAI AUTRON

1_camera_dump_disp - 이미지 로드시 vooya 설정

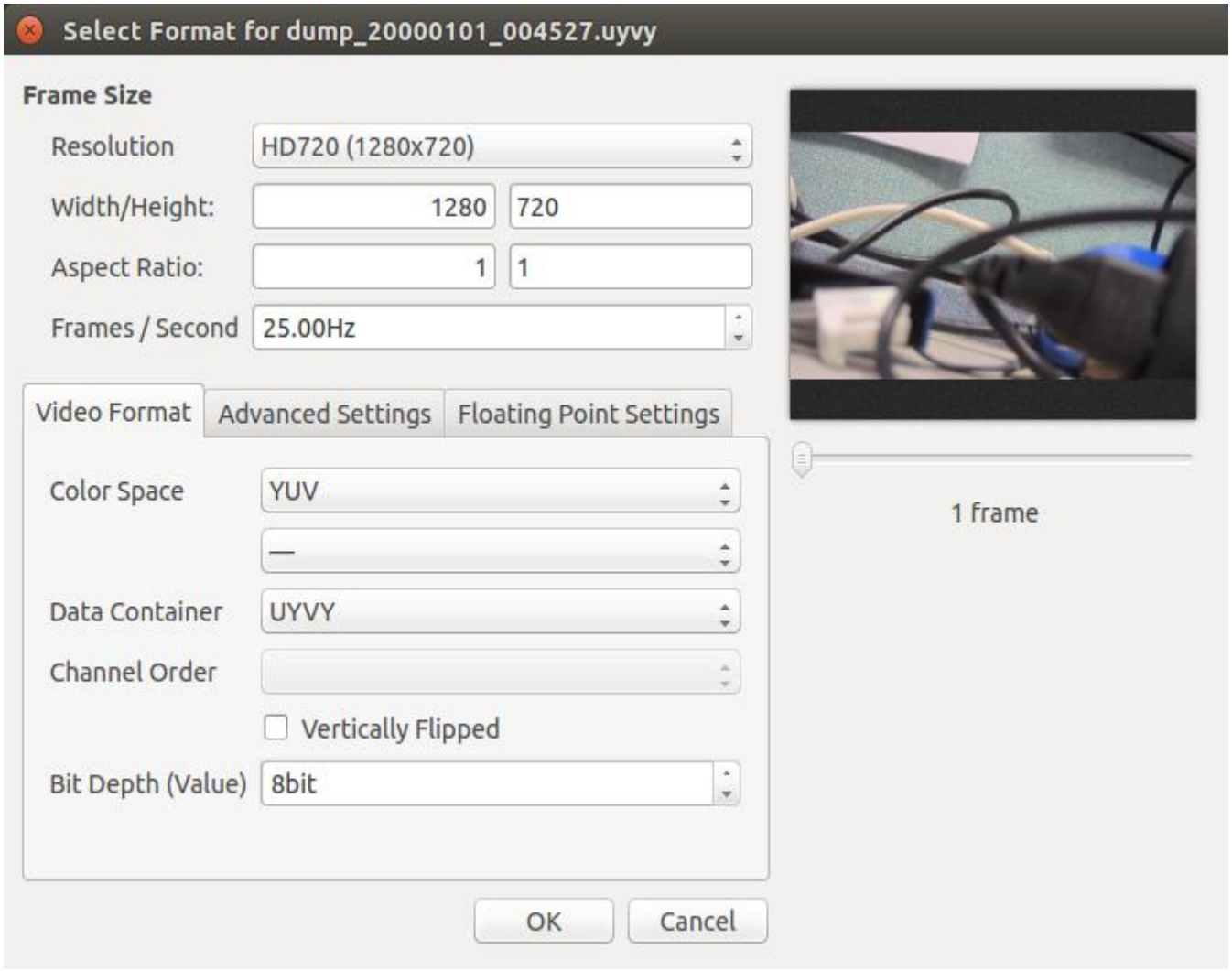
■ 영상크기: 320x180, 영상포맷: nv12



HYUNDAI AUTRON

1_camera_dump_disp – 이미지 로드시 vooya 설정

■ 영상크기: 1280x720, 영상포맷: uyvy



HYUNDAI AUTRON

1_camera_dump_disp – main.c 간략 요약

- **int main(int argc, char **argv)**
  - 영상 입출력 open
  - Thread 용 arg 초기화 (쓰레드 간에 데이터 공유)
  - Thread 생성: capture_thread, capture_dump_thread, input_thread

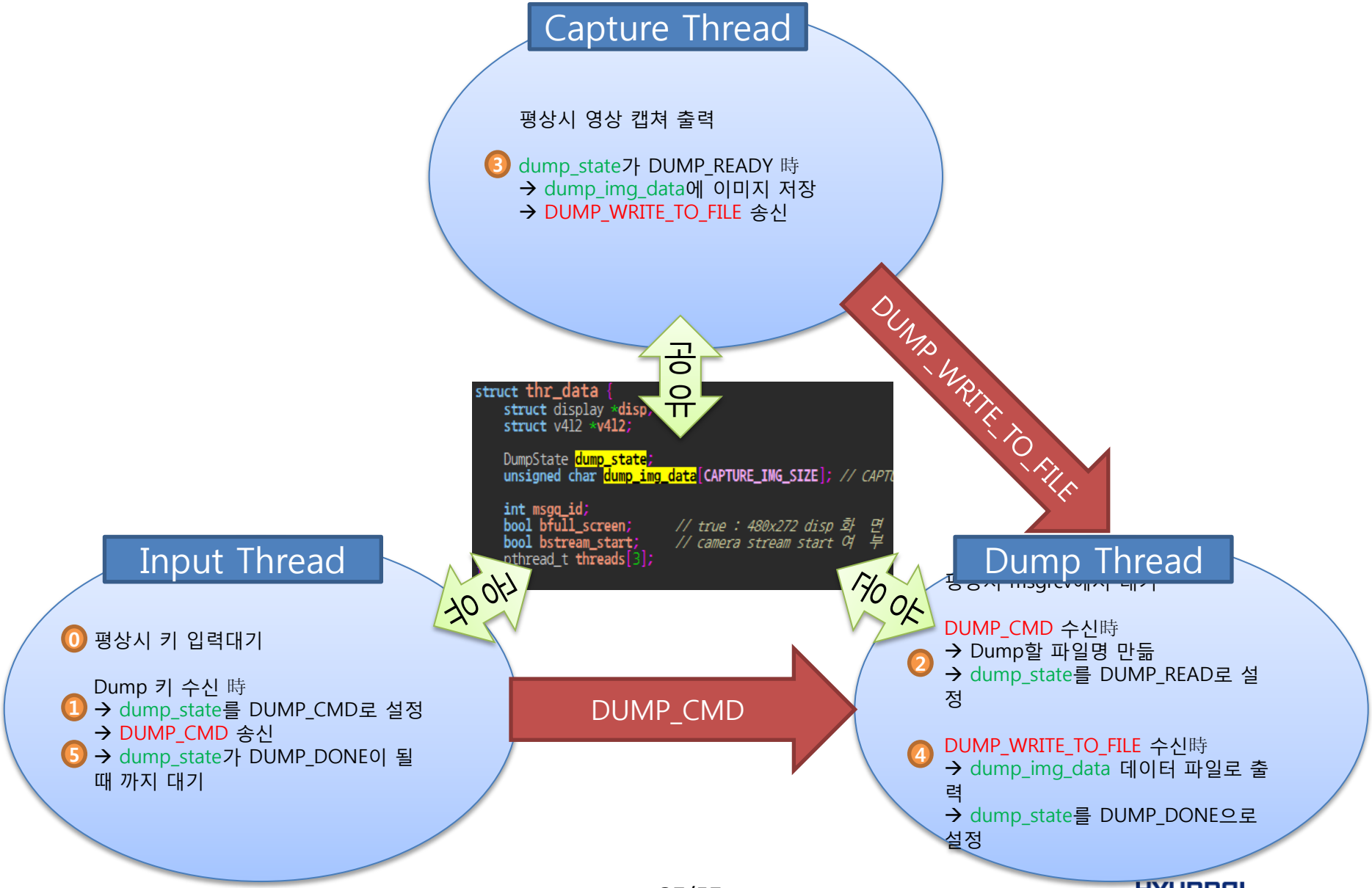
```
struct thr_data {
    struct display *disp;
    struct v4l2 *v4l2;

    DumpState dump_state;
    unsigned char dump_img_data[CAPTURE_IMG_SIZE]; // CAPT

    int msgq_id;
    bool bfull_screen;      // true : 480x272 disp 화 면
    bool bstream_start;     // camera stream start 여 부
    pthread_t threads[3];
};
```

- **void * capture_thread(void *arg): 영상 캡쳐 및 디스플레이**
  - 영상 입출력 버퍼 초기화
  - while loop:
    - v4l2_dqbuf → disp_post_vid_buffer → v4l2_qbuf: 평상시에는 카메라 캡쳐해서 display 출력
    - thread간 공유 데이터값 확인(dump_state = DUMP_READY): msgsnd로 capture_dump_thread 호출 (DUMP_WRITE_TO_FILE)

- **capture_dump_thread: 영상 데이터 파일로 저장**
  - msgrcv로 대기
    - DUMP_CMD 수신: 파일명 생성, thread간 공유 데이터 수정 (dump_state = DUMP_READY)
    - DUMP_WRITE_TO_FILE: 이미지 데이터 파일로 출력, thread간 공유 데이터 수정 (dump_state = DUMP_DONE)

- **input_thread: 키 입력 대기**
  - 키 입력시:
    1. thread간 공유 데이터 수정 (dump_state = DUMP_CMD)
    2. msgsnd로 capture_dump_thread 호출 (DUMP_CMD)
    3. Dump 완료시 까지 대기: thread간 공유 데이터값 확인(dump_state = DUMP_DONE)

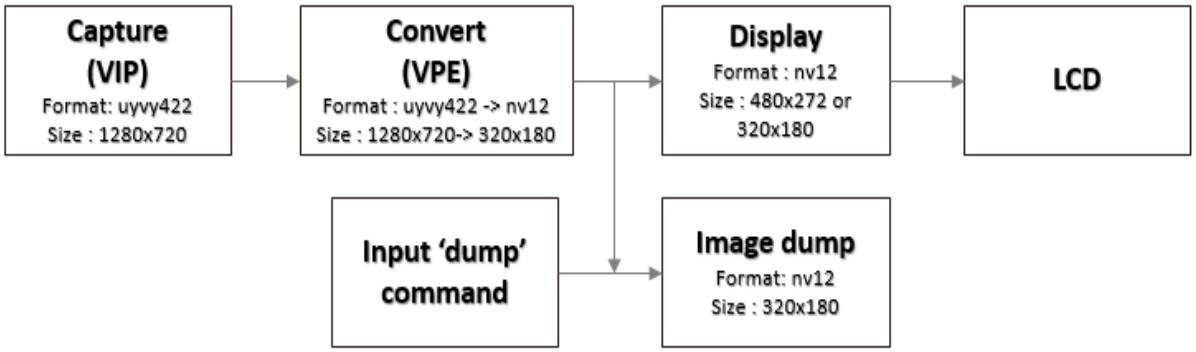
Thread 간 관계 흐름도(예제 1 기준. 다른 예제도 유사)



HYUNDAI AUTRON

2_camera_vpe_dump_disp

■ 카메라 입력 영상 VPE 이용 영상 변환후 LCD 출력 및 이미지 저장



쉘 명령어 (on Target board)	실행 내용
cd Application/2_camera_vpe_dump_disp	2_camera_vpe_dump_disp 폴더로 이동
./camera_vpe_dump_disp	해당 example 실행 및 camera preview 확인
dump + enter	320x180, nv12 format 의 camera image dump. dump_yyyymmdd_hhmmss.nv12 file 로 저장
full + enter	480x272 or 320x180 으로 화면 출력 (Default : full 화면(480x272) 출력)

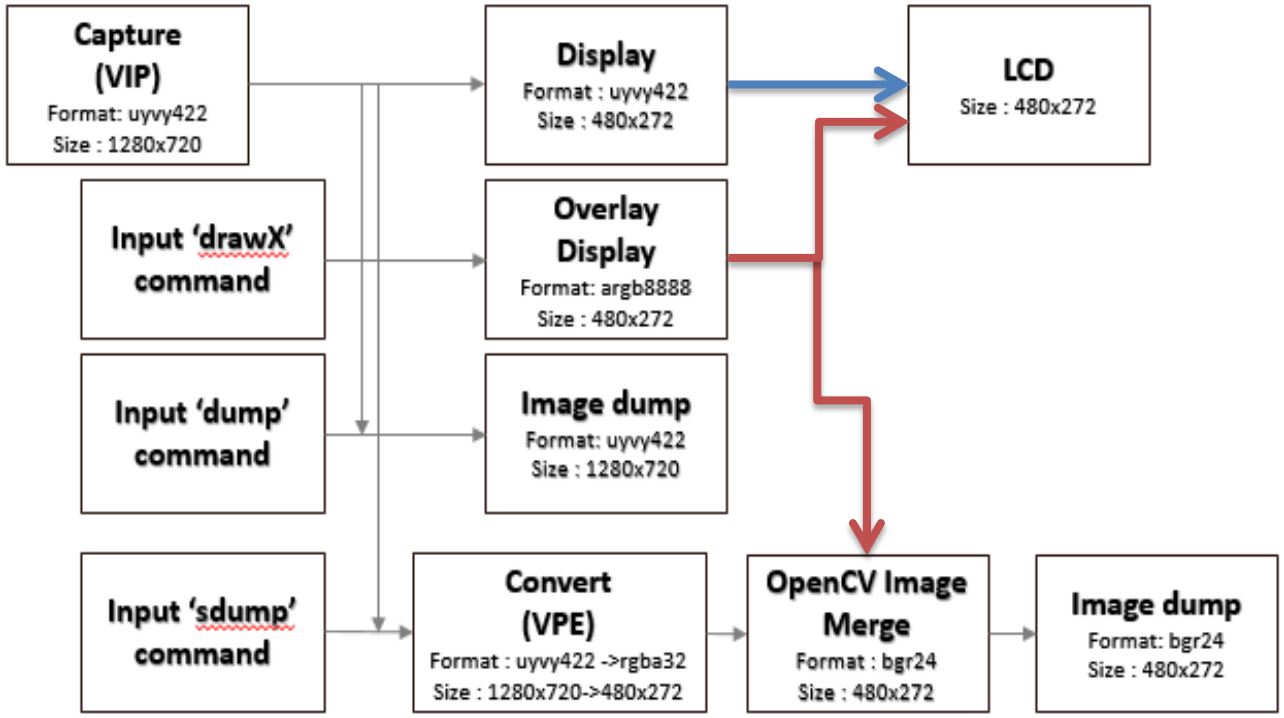
HYUNDAI AUTRON

2_camera_vpe_dump_disp – main.c 간략 요약

- **int main(int argc, char **argv)**
  - 영상 입출력 open
  - VPE 초기화: open, 입출력 parameter 저장
  - Thread 용 arg 초기화 (쓰레드 간에 데이터 공유)
  - Thread 생성: capture_thread, capture_dump_thread, input_thread
- **void * capture_thread(void *arg): 영상 캡쳐 → VPE → 디스플레이 영상 입출력 버퍼 초기화, VPE 초기화**
  - while loop:
    1. v4l2_dqbuf: 캡쳐 queue의 소유권으로 Application으로 가지고 옴
    2. vpe_input_qbuf: 캡쳐 queue를 VPE driver로 소유권 이전
       (vpe_stream_on: VPE 입력 시작 – 초기에 한번 호출)
    3. vpe_output_dqbuf: VPE 출력 queue Application으로 가지고 옴
    4. disp_post_vid_buffer: VPE 처리 결과 Display
    5. vpe_output_qbuf: VPE 출력 queue VPE driver로 소유권 이전
    6. vpe_input_dqbuf: VPE 입력 queue Application으로 소유권 이전
    7. v4l2_qbuf: VPE 입력 queue를 캡쳐 queue로 소유권 이전
- **capture_dump_thread: 영상 데이터 파일로 저장 (1번 예제와 유사)**
- **input_thread: 키 입력 대기 (1번 예제와 유사)**

3_camera_overlay_draw_disp

■ 카메라 영상 상에 점/선/텍스트 오버레이 및 LCD 출력



HYUNDAI AUTRON

3_camera_overlay_draw_disp

■ 카메라 영상 상에 점/선/텍스트 오버레이 및 LCD 출력

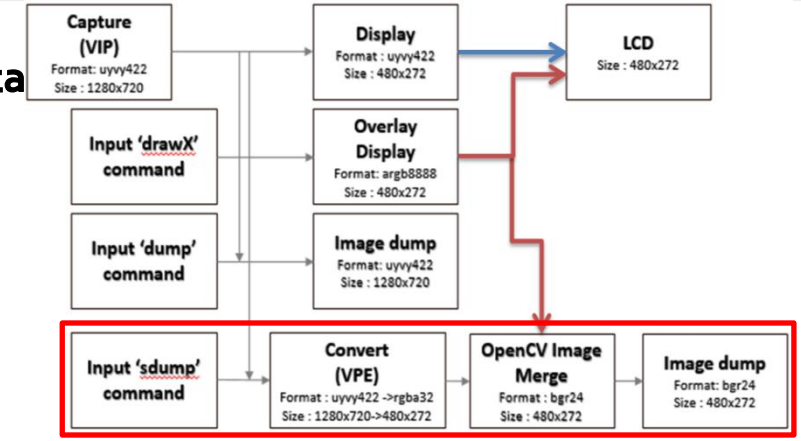
쉘 명령어 (on Target board)	실행 내용
cd Application/3_camera_overlay_draw_disp	3_camera_overlay_draw_disp 폴더로 이동
./3_camera_overlay_draw_disp	해당 example 실행 및 camera preview 확인
dump + enter	1280x720, uyvy format 의 camera image dump. dump_yyyymmdd_hhmmss.uyvy file 로 저장
sdump + endter	480x272, rgba32 format 의 screen image dump. dump_yyyymmdd_hhmmss.rgba32 file 로 저장
drawP:x,y:color + enter	좌표 x,y 에 점 출력 (ex: drawP:10,10:0xffff0000)
drawL:x1,y1:x2,y2:color + enter	좌표 x1,y1 부터 x2,y2 로 선 출력 (ex : drawL:4,4:100,2:0xff00ff00)
drawR:x,y:w,h:color + enter	좌표 x,y 에 w,h 만큼의 사각형 출력 (ex : drawR:12,12:20,40:0xff0000ff)
drawT:x,y:color:string + enter	좌표 x,y에 string 출력 (ex : drawT:100,100:0xffffffff:Draw test)
drawC + enter	Draw 모든 영역 clear

HYUNDAI AUTRON

3_camera_overlay_draw_disp – main.c 간략 요약

- **int main(int argc, char **argv)**
  - 영상 입력 open, 영상 출력 open (overlay plane 설정)
  - Thread 용 arg 초기화 (쓰레드 간에 데이터 공유)
  - Thread 생성: capture_thread, capture_dump_thread, input_thread
- **void * capture_thread(void *arg): 영상 캡쳐 및 디스플레이**
  - 영상 입출력 버퍼 초기화
  - while loop:
    - v4l2_dqbuf → disp_post_vid_buffer → v4l2_qbuf: 평상시에는 카메라 캡쳐해서 display 출력
    - thread간 공유 데이터값 확인(dump_state = DUMP_READY): msgsnd로 capture_dump_thread 호출 (DUMP_WRITE_TO_FILE)
- **capture_dump_thread: 영상 데이터 파일로 저장**
  - msgrcv로 대기
    - DUMP_CMD 수신: 파일명 생성, thread간 공유 데이터 수정 (dump_state = DUMP_READY)
    - DUMP_WRITE_TO_FILE: 이미지 데이터 파일로 출력, thread간 공유 데이터 수정 (dump_state = DUMP_DONE)
      - Capture 이미지 저장時: 1번 예제와 유사
      - Display 이미지 저장時: VPE와 OpenvCV 이용 데이터 merge 후 저장
- **input_thread: 키 입력 대기**
  - dump 또는 sdump 키 입력時: 1번 예제와 유사
  - draw 명령어 입력時: argb8888 포맷으로 overlay (사용 가능한 API: drawPixel, drawLine, drawRect, drawString)

3_camera_overlay_draw_disp – makescreendata 함수 이해



■ 함수 static int makescreendata(struct thr_data *data

- sdump 입력시 실행됨
- VPE와 OpenCV이용 출력 영상 layer 합성 목적

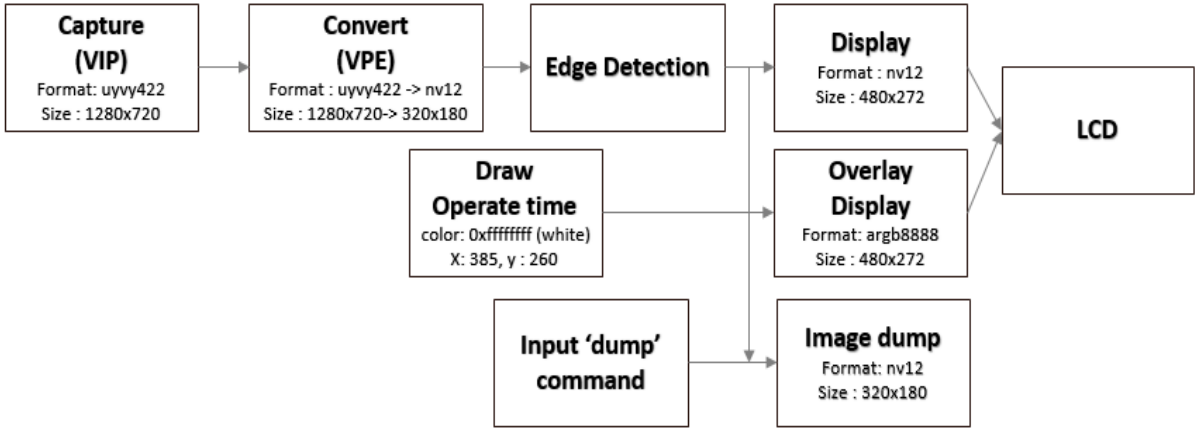
■ 흐름

1. VPE 이용 Capture 영상 크기 및 format 변경
   1) VPE 초기화: open, 입출력 parameter 저장, 입출력 buffer 할당
   2) vpe_output_qbuf: VPE 출력 queue VPE driver로 소유권 이전
   3) vpe_stream_on(vpe->fd, V4L2_BUF_TYPE_VIDEO_CAPTURE_MPLANE): VPE 하드웨어에 출력 영상 데이터 흐름 On
   4) vpe_input_qbuf: VPE 입력 queue를 VPE driver로 소유권 이전
   5) vpe_stream_on(vpe->fd, V4L2_BUF_TYPE_VIDEO_OUTPUT_MPLANE): VPE 하드웨어에 입력 영상 데이터 흐름 On
   6) vpe_output_dqbuf: VPE 출력 queue를 Application으로 소유권 이전
   7) vpe_input_dqbuf: VPE 입력 queue를 Application으로 소유권 이전
   8) vpe_stream_off(vpe->fd, V4L2_BUF_TYPE_VIDEO_OUTPUT_MPLANE): VPE 하드웨어에 입력 영상 데이터 흐름 Off
   9) vpe_stream_off(vpe->fd, V4L2_BUF_TYPE_VIDEO_CAPTURE_MPLANE): VPE 하드웨어에 출력 영상 데이터 흐름 Off
2. get_framebuf: overlay image 버퍼 가지고 오기
3. OpenCV_merge_image: OpenCV 이용 layer 합성
   1) 이미지 합성: overlay image+vpe output image
   2) 합성된 이미지 bgr24 포맷으로 변환

HYUNDAI AUTRON

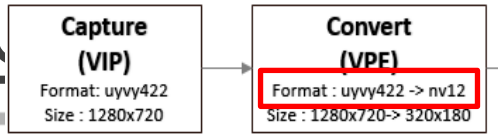
4_camera_vpe_edgedetect_disp

■ Edge Detection 예제 구현 및 처리 결과 출력: Y 데이터(gray 이미지) 사용하여 영상처리

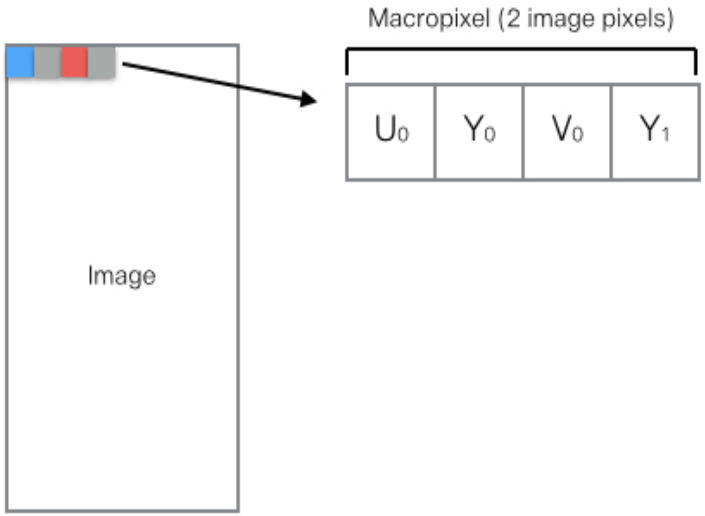


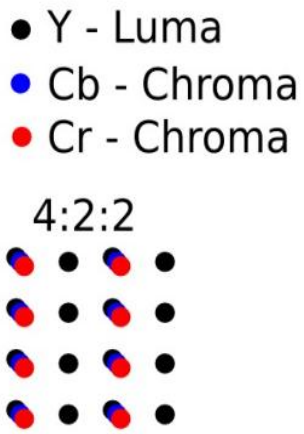
쉘 명령어 (on Target board)	실행 내용
cd Application/4_camera_vpe_edgedect_disp	4_camera_vpe_edgedect_disp 폴더로 이동
./camera_vpe_edgedect_disp	해당 example 실행 및 edge 결과 및 연산 시간 출력 확인
dump + enter	320x180, nv12 format 의 edge detect image dump. dump_yyyymmdd_hhmmss.nv12 file 로 저장

4_camera_vpe_edgedetect_disp – 영상 포맷 소

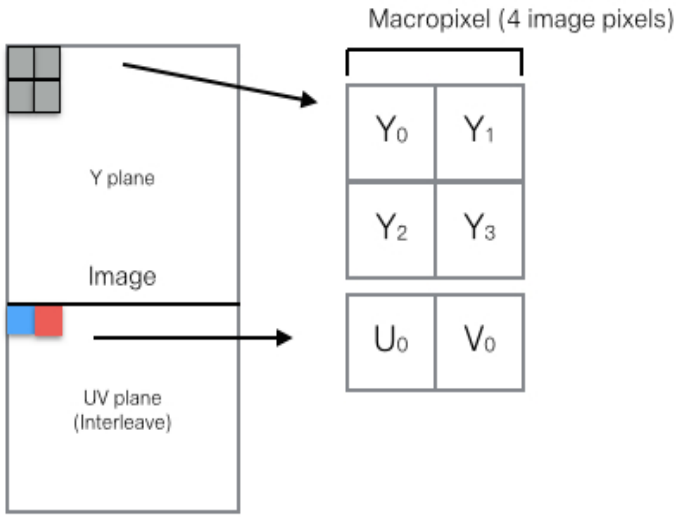


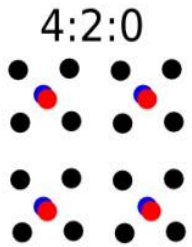
■ **uyvy422**





■ **nv12: Y plane UV plane 따로 있음** → Y 데이터만 사용하여 영상처리하기 좋은 구조





HYUNDAI AUTRON

4_camera_vpe_edgedetect_disp – main.c 간략 요약

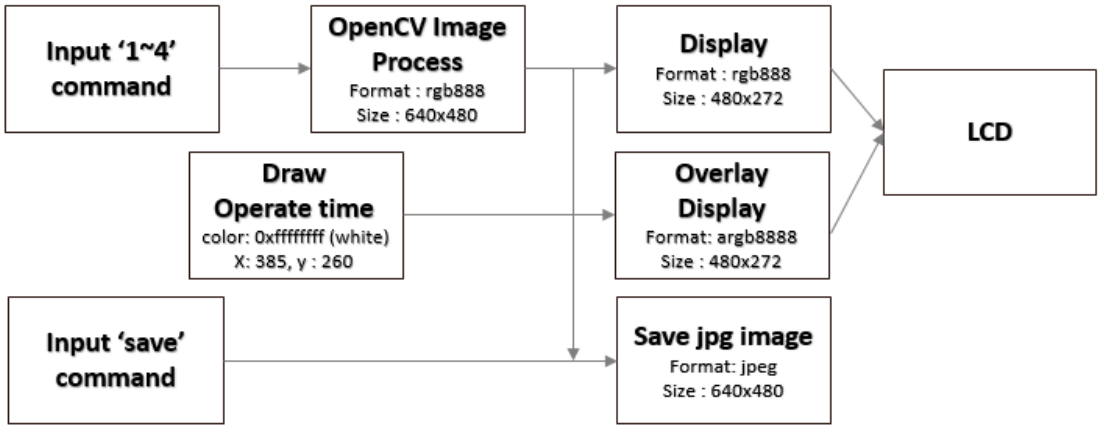
- **int main(int argc, char **argv)**
  - 영상 입력 open, 영상 출력 open (overlay plane 설정)
  - VPE 초기화: open, 입출력 parameter 저장
  - Thread 용 arg 초기화 (쓰레드 간에 데이터 공유)
  - Thread 생성: capture_thread, capture_dump_thread, input_thread
- **void * capture_thread(void *arg): 영상 캡쳐 → VPE → Edge detection → 디스플레이**
  - 영상 입출력 버퍼 초기화, VPE 초기화
  - while loop: (2번 예제와 유사)
    1. v4l2_dqbuf: 캡쳐 queue의 소유권으로 Application으로 가지고 옴
    2. vpe_input_qbuf: 캡쳐 queue를 VPE driver로 소유권 이전
       (vpe_stream_on: VPE 입력 시작 – 초기에 한번 호출)
    3. vpe_output_dqbuf: VPE 출력 queue Application으로 가지고 옴
    4. sobel_edge_detection: Edge detection 알고리즘 수행
    5. draw_operatingtime: Edge detection 수행 시간 overlay plane 에 표시
    6. disp_post_vid_buffer: 이미지 plane 출력(Edge detection 결과)
    7. update_overlay_disp: overlay plane 출력(수행시간 text)
       (DUMP_READY일때 이미지 처리결과 dump)

    } Layer가 두 개 있기 때문에 각각 출력

    8. vpe_output_qbuf: VPE 출력 queue VPE driver로 소유권 이전
    9. vpe_input_dqbuf: VPE 입력 queue Application으로 소유권 이전
    10. v4l2_qbuf: VPE 입력 queue를 캡쳐 queue로 소유권 이전
- **capture_dump_thread: 영상 데이터 파일로 저장 (1번 예제와 유사)**
- **input_thread: 키 입력 대기 (1번 예제와 유사)**

5_opencv_disp

■ OpenCV 예제 알고리즘 처리하여 LCD 출력 및 이미지 저장



쉘 명령어 (on Target board)	실행 내용
cd Application/5_opencv_disp	5_opencv_disp 폴더로 이동
./opencv_disp	OpenCV 사용 file load 후 해당 이미지 출력
1 + enter	OpenCV 사용 file load 후 해당 이미지 출력
2 + enter	OpenCV 사용 Face detection
3 + enter	OpenCV 사용 Binding image
4 + enter	OpenCV 사용 Cany edge detection
save + enter	opencv jpeg image. 640x480 크기의 yyyymmdd_hhmmss.jpg file 로 저장

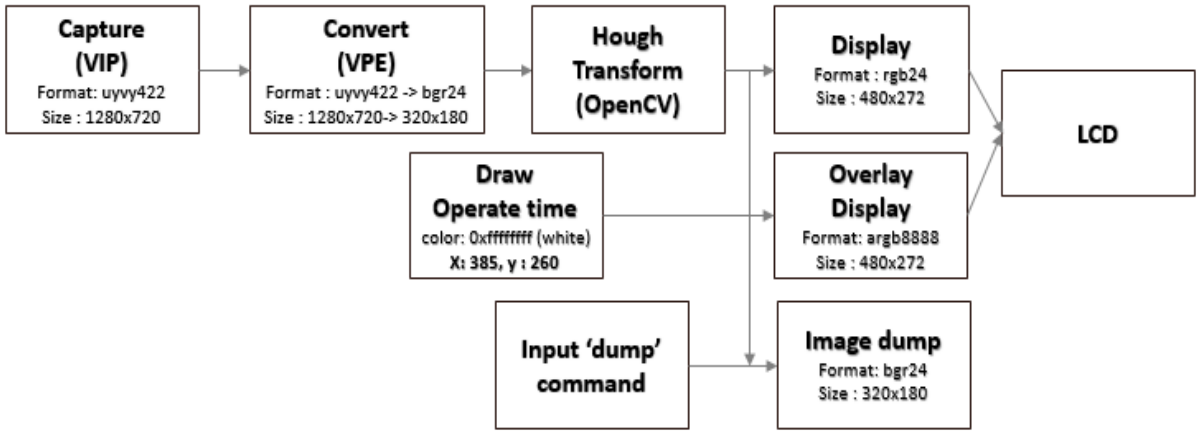
HYUNDAI AUTRON

5_opencv_disp – main.c 간략 요약

- **int main(int argc, char **argv)**
  - 영상 출력 open (overlay plane 설정)
  - Thread 용 arg 초기화 (쓰레드 간에 데이터 공유)
  - cv_exam 실행하여 이미지 파일 로드 및 출력
  - Thread 생성: capture_dump_thread, input_thread
- **capture_dump_thread: 영상 데이터 파일로 저장**
  - msgrcv로 대기
    - DUMP_CMD 수신: dump_state = DUMP_READY & cv_exam 함수 호출(→이미지 데이터 복사후, DUMP_WRITE_TO_FILE 전송)
    - DUMP_WRITE_TO_FILE: cv_savetojpeg 호출 (이미지 데이터 파일로 출력) & dump_state = DUMP_DONE
- **input_thread: 키 입력 대기**
  - save 키 입력時: 1번 예제와 유사
  - '1~4' 번호 입력時: cv_exam 함수 호출하여 영상 처리 수행 및 처리된 영상 업데이트

6_camera_opencv_disp

■ 카메라 입력 이미지에 OpenCV이용 허프변환 하여 결과 LCD 출력



쉘 명령어 (on Target board)	실행 내용
cd Application/6_camera_opencv_disp	6_camera_opencv_disp 폴더로 이동
./camera_opencv_disp	해당 example 실행 및 edge 결과 및 연산 시간 출력 확인
dump + enter	320x180, bgr24 format 의 camera image hough transform dump. dump_yyyymmdd_hhmmss.bgr24 file 로 저장

HYUNDAI AUTRON

6_camera_opencv_disp– main.c 간략 요약

- **int main(int argc, char **argv)**
  - 영상 입력 open, 영상 출력 open (overlay plane 설정)
  - VPE 초기화: open, 입출력 parameter 저장
  - Thread 용 arg 초기화 (쓰레드 간에 데이터 공유)
  - Thread 생성: capture_thread, capture_dump_thread, input_thread
- **void * capture_thread(void *arg): 영상 캡쳐 → VPE → Hough transform → 디스플레이**
  - 영상 입출력 버퍼 초기화, VPE 초기화
  - while loop: (4번 예제와 유사)
    1. v4l2_dqbuf: 캡쳐 queue의 소유권으로 Application으로 가지고 옴
    2. vpe_input_qbuf: 캡쳐 queue를 VPE driver로 소유권 이전
       (vpe_stream_on: VPE 입력 시작 – 초기에 한번 호출)
    3. vpe_output_dqbuf: VPE 출력 queue Application으로 가지고 옴
    4. hough_transform : Hough transform 알고리즘 수행 (OpenCV 이용), 수행 시간 overlay plane 에 표시
    5. disp_post_vid_buffer: 이미지 plane 출력(Edge detection 결과)
    6. update_overlay_disp: overlay plane 출력(수행시간 text)
       (DUMP_READY일때 이미지 처리결과 dump)
    7. vpe_output_qbuf: VPE 출력 queue VPE driver로 소유권 이전
    8. vpe_input_dqbuf: VPE 입력 queue Application으로 소유권 이전
    9. v4l2_qbuf: VPE 입력 queue를 캡쳐 queue로 소유권 이전

    } Layer가 두 개 있기 때문에 각각 출력
- **capture_dump_thread: 영상 데이터 파일로 저장 (1번 예제와 유사)**
- **input_thread: 키 입력 대기 (1번 예제와 유사)**

HYUNDAI AUTRON

향후 계획

■ 1차 Mission

- 부팅후 프로그램 자동 실행
- 카메라 이용 특정 물체 인식후 화면에 인식 결과, 연산시간 display
- 차량이 인식한 물체 따라가게 하기

별첨 SD 카드 관련 내용 (필요시에만 참고)

■ 지능형 자동차 개발환경 세팅 방법

- 지능형 자동차의 target board에 Embedded Linux 설치
- Gparted 이용 SD카드 초기화 방법

HYUNDAI AUTRON

Target board에 Embedded Linux 설치과정 (1/3) – 요약

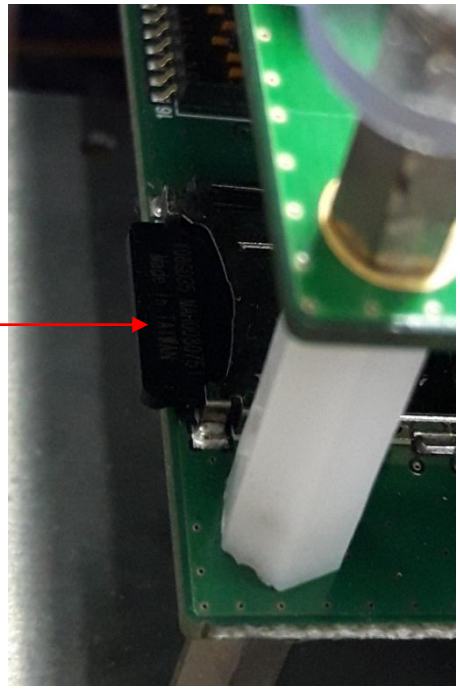
별첨

- Target board에 장착할 SD 카드 Host PC에 장착
- CarSDK/Flashing 폴더로 이동

```
$./flashingimage.sh /dev/sdc
```

→ ‘sdc’는 host PC에서 잡히는 SD 카드 드라이브명 확인 ‘sdb’, ‘sdd’과 같이 다른 이름 될 수 있음

- SD 카드 target board 장착 후 전원 On



HYUNDAI AUTRON

Target board에 Embedded Linux 설치과정 (2/3) – 예시

별첨

```
at@ubuntu: ~/CarSDK/Flashing
at@ubuntu:~$ ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sda3  /dev/sdb  /dev/sdc  /dev/sdd
at@ubuntu:~$ ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sda3  /dev/sdb  /dev/sdc  /dev/sdc1  /dev/sdd
at@ubuntu:~$ cd CarSDK/Flashing/
at@ubuntu:~/CarSDK/Flashing$ ./flashingimage.sh      /dev/sdc
----------------------------------------
Defined machine is brain
----------------------------------------
./flashingimage.sh machine device
example: ./flashingimage.sh evm
example: ./flashingimage.sh brain
example: ./flashingimage.sh evm /dev/sdc
example: ./flashingimage.sh brain /dev/sdc
----------------------------------------
----------------------------------------
Defined machine is brain
/dev/sdc
flashing!! using device /dev/sdc
[sudo] password for at:
-- Main device is: /dev/loop0
*************************************************************
*         THIS WILL DELETE ALL THE DATA ON /dev/sdc          *
*                                                           *
*         WARNING! Make sure your computer does not go      *
*                  in to idle mode while this script is     *
*                  running. The script will complete,       *
*                  but your SD card may be corrupted.       *
*                                                           *
*         Press <ENTER> to confirm....                      *
*************************************************************

unmounting device '/dev/sdc1'
1024+0 records in
1024+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.417093 s, 2.5 MB/s

Welcome to fdisk (util-linux 2.27.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x7024f393.
```

SD 카드 PC 연결시 드라이브 명 확인

Flashing 폴더로 이동후 실행 (주의: 드라이브명 /dev/sdc1이 아닌 /dev/sdc)

Password 입력

Enter 입력

SD 카드 리더기 사용하지 않고, PC에 직접 꽂으면, /dev/mmcblk0 로 인식됨

이렇게 하면 정상적으로 수행되지 않으니, 반드시 리더기를 사용하여 sdc로 인식되게 할 것

Target board에 Embedded Linux 설치과정 (3/3) – 예시

별첨

```
at@ubuntu: ~/CarSDK/Flashing
Created a new partition 1 of type 'Linux' and of size 62 MiB.

Command (m for help): Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): Partition number (2-4, default 2): First sector (129024-3862527, default 129024): Last sector, +sectors or +si
ze{K,M,G,T,P} (129024-3862527, default 3862527):
Created a new partition 2 of type 'Linux' and of size 1.8 GiB.

Command (m for help): Partition number (1,2, default 2): Partition type (type L to list all types):
Changed type of partition 'Linux' to 'W95 FAT32 (LBA)'.

Command (m for help): Partition number (1,2, default 2):
The bootable flag on partition 1 is enabled now.

Command (m for help): The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

make partitions.
Formating /dev/sdc1 ...
mkfs.fat 3.0.28 (2015-05-16)
mkfs.fat: warning - lowercase labels might not work properly with DOS or Windows
mke2fs 1.42.13 (17-May-2015)
/dev/sdc2 contains a ext4 file system labelled 'rootfs'
        last mounted on /tmp/sdk/7497/rootfs on Wed Aug 23 14:39:34 2017
Proceed anyway? (y,n) y
Creating filesystem with 466688 4k blocks and 116880 inodes
Filesystem UUID: 8f29dc3d-0668-4451-b225-8afac6a55c2e
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

Copying filesystem on /dev/sdc1,/dev/sdc2
Current Machine: brain
use uenv.brain.txt for Autron arago-brain-image-dra7xx-autronbrain.tar.gz
Extracting filesystem on /dev/sdc2 ...
unmounting /dev/sdc1 ,/dev/sdc2
completed!
at@ubuntu:~/CarSDK/Flashing$
```

y 입력

완료 → SD 카드 target board 장착 후 전원 On

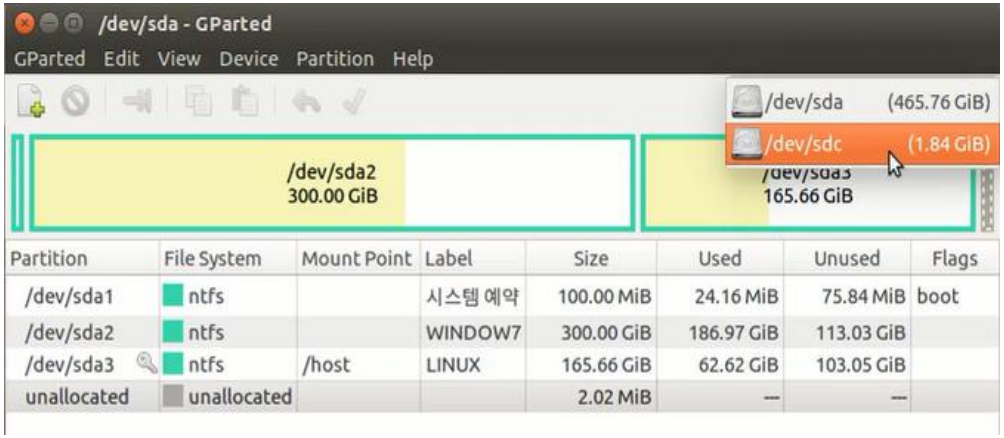
사용중인 SD 카드를 초기화 하고 다시 Flashing 하는 방법 (1/5)

별첨

■ GParted 실행 → SD 카드 드라이버 선택 → unmount → 파티션 제거(delete) → 새로운 파티션 생성(new)

1.

2.

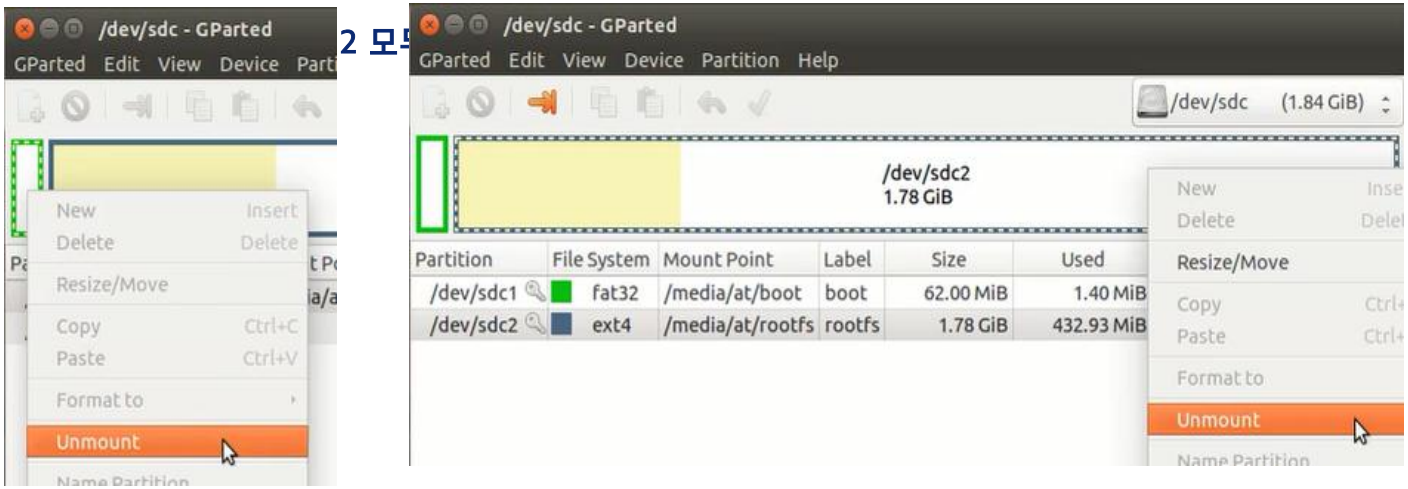


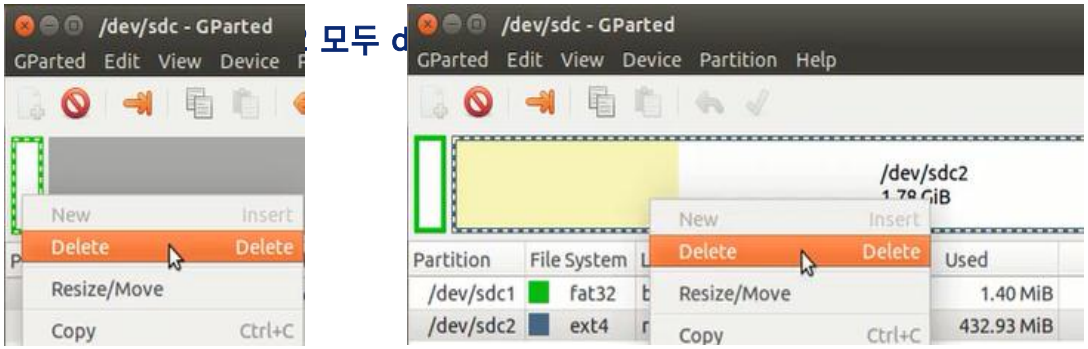
HYUNDAI AUTRON

사용중인 SD 카드를 초기화 하고 다시 Flashing 하는 방법 (2/5)

별첨

■ GParted 실행 → SD 카드 드라이버 선택 → unmount → 파티션 제거(delete) → 새로운 파티션 생성(new)

3. 

4. 

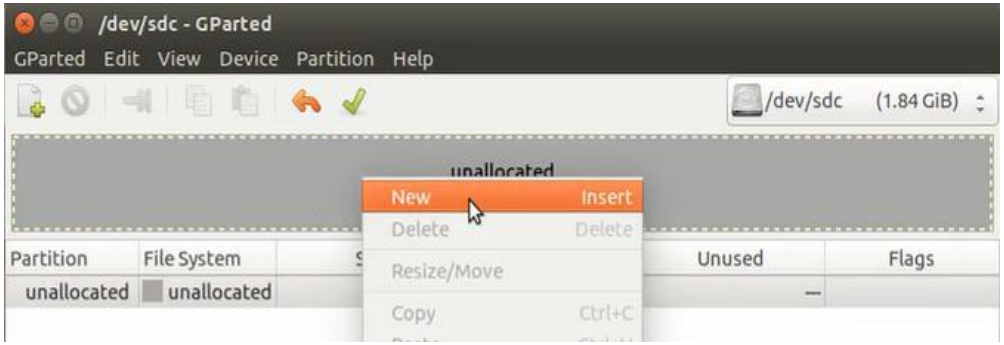
HYUNDAI AUTRON

사용중인 SD 카드를 초기화 하고 다시 Flashing 하는 방법 (3/5)

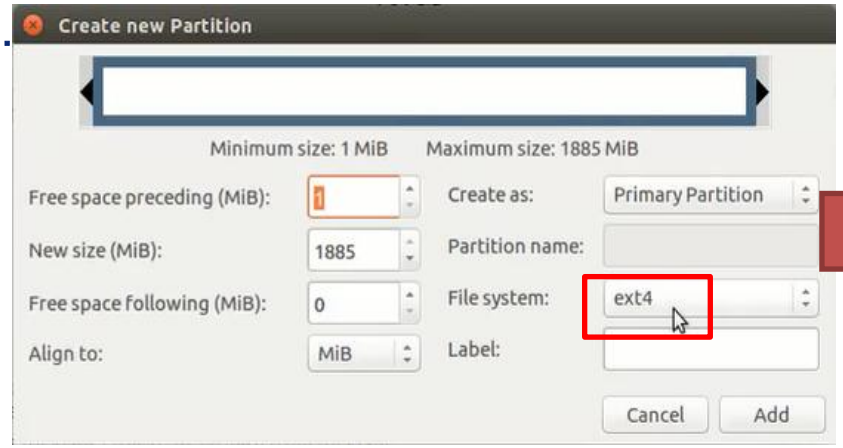
별첨

■ GParted 실행 → SD 카드 드라이버 선택 → unmount → 파티션 제거(delete) → 새로운 파티션 생성(new)

5.



6.



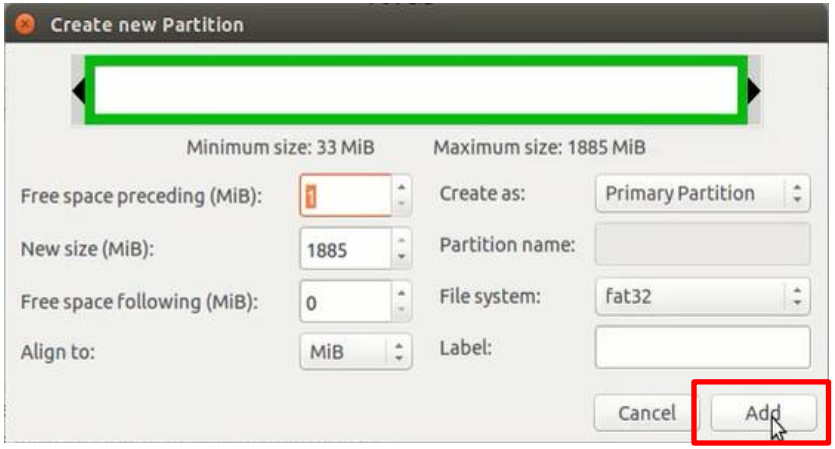
HYUNDAI AUTRON

사용중인 SD 카드를 초기화 하고 다시 Flashing 하는 방법 (4/5)

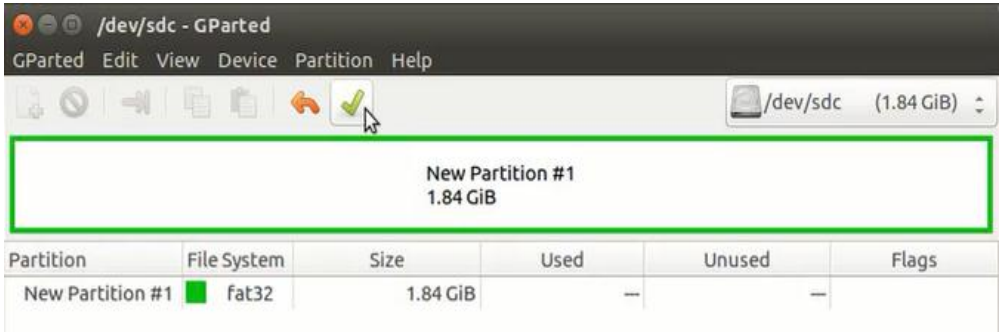
별첨

■ GParted 실행 → SD 카드 드라이버 선택 → unmount → 파티션 제거(delete) → 새로운 파티션 생성(new)

7.



8.

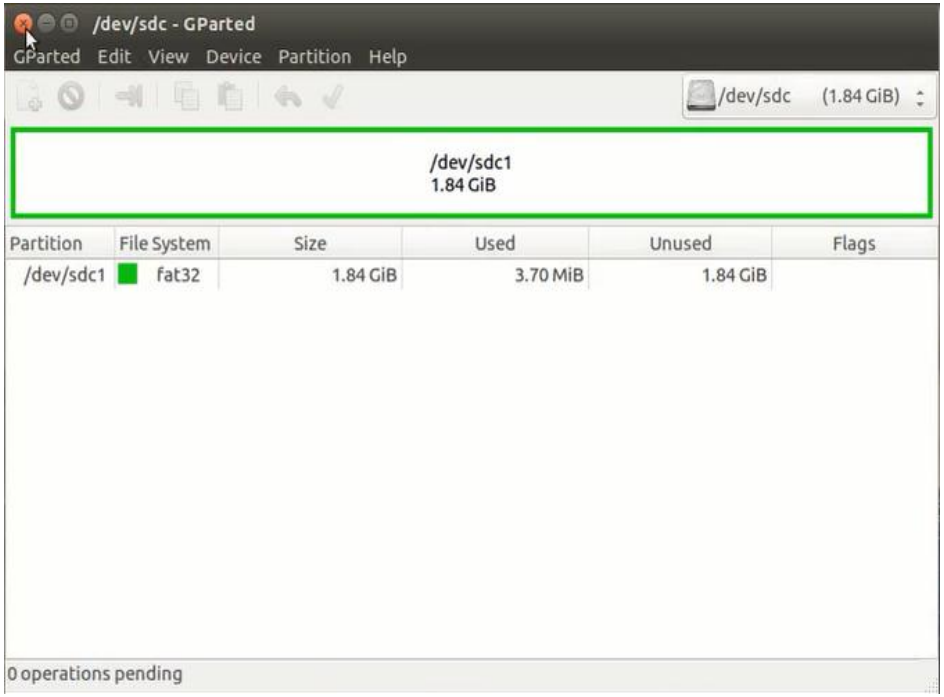


HYUNDAI AUTRON

사용중인 SD 카드를 초기화 하고 다시 Flashing 하는 방법 (5/5)

별첨

■ GParted 이용하여 새로운 단일 파티션 생성되고 완료된 모습



■ Gparted 종료 시킨 후 Flashing 과정은 동일

- CarSDK/Flashing 폴더로 이동

  `$./flashingimage.sh /dev/sdc`

  → 'sdc'는 host PC에서 잡히는 SD 카드 드라이브명 확인 'sdb', 'sdd'과 같이 다른 이름 될 수 있음

- SD 카드 target board 장착 후 전원 On

HYUNDAI AUTRON

SD카드를 초기화 하고 다시 Flashing 했을 때 주의사항

별첨

■ 초기화 하기 전 SD카드와 scp 로 데이터를 보낸 기록이 남아있다면 기존 암호화 키와 충돌

- 해결 방법은 warning 메시지에 써 있음.