Armstrong 2.5 & 5.0 IPC 셋업 가이드

Date	Contents	Version	writer
2022.09.23	Initial version	1.0	

Contents

1. IPC1 Setting

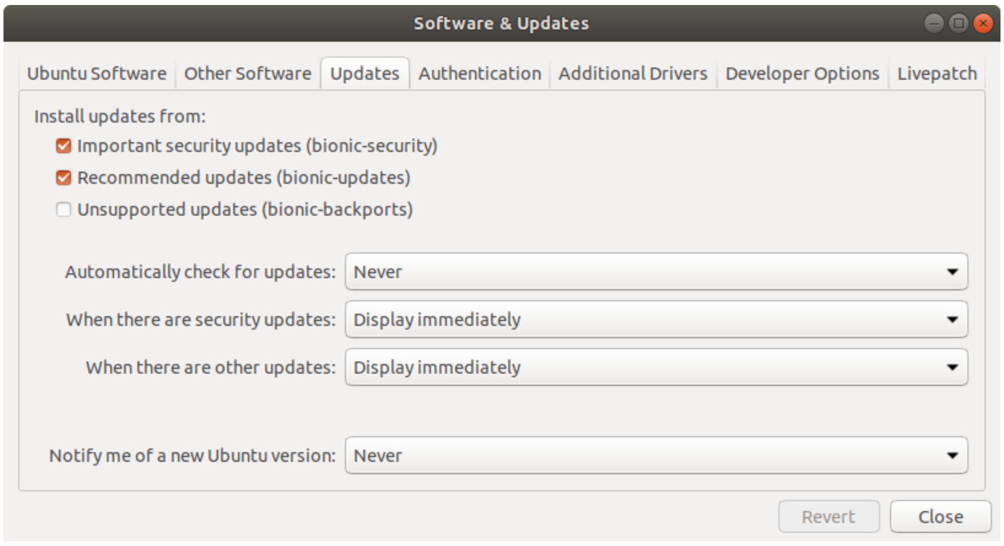
1.1 Install Ubuntu Linux

1.1.1 Install ubuntu 18.04.6

a) Download link : http://mirror.kakao.com/ubuntu-releases/18.04.6/ubuntu-18.04.6-desktop-amd64.iso
b) 언어는 기본 English로 설정
c) User Name : ARMSTRONG, ID : armstrong, PW : swm.ai
d) 한글 키보드 설치
    i) Ubuntu18.04 : https://greedywyatt.tistory.com/105

10.2 Update ubuntu and install generic kernel

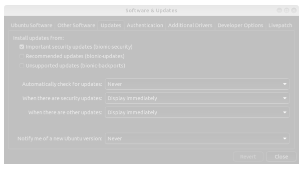
a) Software & Updates 설정창에서 Update 탭은 아래와 같이 설정한다.



Install updates from:
☑ Important security updates (bionic-security)
☑ Recommended updates (bionic-updates)
☐ Unsupported updates (bionic-backports)

Automatically check for updates: Never
When there are security updates: Display immediately
When there are other updates: Display immediately

Notify me of a new Ubuntu version: Never


0ㅏ

b) 터미널에서 다음 명령어들을 실행
    i) 커널을 패치하기 전에 최신 상태로 유지하기 위해 update, upgrade 실행

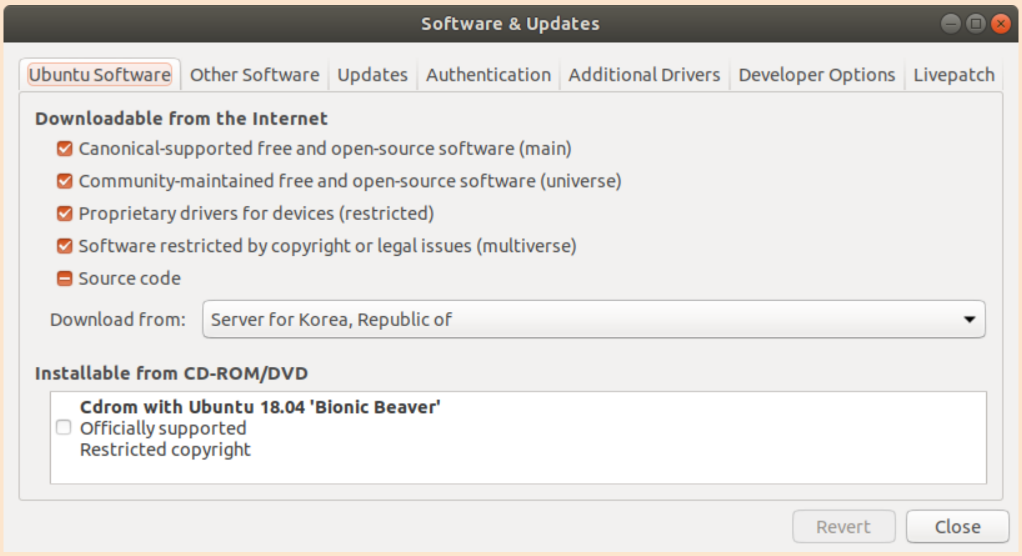
```
$ sudo apt update

$ sudo apt upgrade -y
```

    ii) 현재 시스템의 최신 커널 버전 확인

```
$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.6 LTS       <-- (Current lsb_release version)
Release:        18.04
Codename:       bionic

$ cat /proc/version_signature
Ubuntu 5.4.0-89.100~18.04.1-generic 5.4.143  <-- (Current kernel version)
```

iii)   리눅스 커널 빌드 환경 준비

```
$ sudo apt-get install git git-lfs vim mc

$ sudo apt-get build-dep linux apt-get install git git-lfs vim mc


Reading package lists... Done
E: You must put some 'source' URIs in your sources.list  <-- 해당 에러 발생되면 아래 과정 확인

>> Open Software & Updates. in the Ubuntu Software menu tick the ‘Source code’ box
```



Software & Updates

Downloadable from the Internet

- [✓] Canonical-supported free and open-source software (main)
- [✓] Community-maintained free and open-source software (universe)
- [✓] Proprietary drivers for devices (restricted)
- [✓] Software restricted by copyright or legal issues (multiverse)
- [–] Source code

Download from: Server for Korea, Republic of

Installable from CD-ROM/DVD

- [ ] **Cdrom with Ubuntu 18.04 'Bionic Beaver'** Officially supported Restricted copyright

```
$ sudo apt-get build-dep linux
$ sudo apt-get install libncurses-dev flex bison openssl libssl-dev dkms libelf-dev libudev-dev
libpci-dev libiberty-dev autoconf fakeroot
```

```
$
```

1.2 Install Real Time Kernel

참고 : https://docs.ros.org/en/foxy/Tutorials/Building-Realtime-rt_preempt-kernel-for-ROS-2.html

~~1.2.1 Patch된 Real Time Kernel Source Down~~

~~1) RT 패치된 커널 소스를 다운로드~~

~~Linux-stable-rt-5.4.143-rt64-basa_220122.tgz~~

~~1.2.2 Build real time generic kernel~~

~~➤ We simply want to use the config of our Ubuntu installation, so we get the Ubuntu config with~~

```
~~$ cd ~;~~
~~$ mkdir kernel~~
~~$ cp ~/Downloads/linux-stable-rt-5.4.143-rt64-basa_220122.tgz ~/kernel      #Aleady Downloaded kernel source~~
~~$ cd ~/kernel~~
~~$ tar vzxf linux-stable-rt-5.4.143-rt64-basa_220122.tgz~~
~~$ cd linux-stable-rt-5.4.143-rt64~~


~~Cd~~









~~`.~~
```

```
~~$ vi .config~~
~~    CONFIG_DEBUG_INFO_BTF=n 으로 변경~~
```

~~➤ Now we're going to build the kernel which will take quite some time. (about 50 min)~~

```
~~$ make -j `nproc` deb-pkg LOCALVERSION=-armstrong~~
~~... ...~~
~~... ...~~
~~  INSTALL ./debian/headertmp/usr/include~~
~~dpkg-deb: building package 'linux-headers-5.4.143-rt64' in '../linux-headers-5.4.143-rt64_5.4.143-rt64-1_amd64.deb'.~~
~~dpkg-deb: building package 'linux-libc-dev' in '../linux-libc-dev_5.4.143-rt64-1_amd64.deb'.~~
~~dpkg-deb: building package 'linux-image-5.4.143-rt64' in '../linux-image-5.4.143-rt64_5.4.143-rt64-1_amd64.deb'.~~
~~dpkg-deb: building package 'linux-image-5.4.143-rt64-dbg' in '../linux-image-5.4.143-rt64-dbg_5.4.143-rt64-1_amd64.deb'.~~
~~ dpkg-genbuildinfo~~
~~ dpkg-genchanges  >../linux-5.4.143-rt64_5.4.143-rt64-1_amd64.changes~~
~~dpkg-genchanges: info: including full source code in upload~~
~~ dpkg-source -i.git --after-build linux-stable-rt-5.4.143-rt64~~
~~dpkg-buildpackage: info: full upload (original source is included)~~
~~$~~

~~만약 dpk-pkg 관련으로 빌드 에러가 발생하면…~~
~~   -      ./vmlinux-gdb.py 를 삭제~~
~~   -~~
```

~~► After the build is finished check the debian packages~~

```
~~$ ls -lh ../*deb~~
~~-rw-r--r-- 1 ipc1 ipc1  12M 11월 13 18:45 ../linux-headers-5.4.143-rt64_5.4.143-rt64-1_amd64.deb~~
~~-rw-r--r-- 1 ipc1 ipc1  59M 11월 13 18:46 ../linux-image-5.4.143-rt64_5.4.143-rt64-1_amd64.deb~~
~~-rw-r--r-- 1 ipc1 ipc1 817M 11월 13 18:54 ../linux-image-5.4.143-rt64-dbg_5.4.143-rt64-1_amd64.deb~~
~~-rw-r--r-- 1 ipc1 ipc1 1.1M 11월 13 18:45 ../linux-libc-dev_5.4.143-rt64-1_amd64.deb~~
```

~~1.2.3 Install the build kernel package and Reboot~~

~~► Install all kernel debian packages~~

```
~~$ sudo dpkg -i ../*.deb~~
~~Selecting previously unselected package linux-headers-5.4.143-rt64.~~
~~(Reading database ... 201690 files and directories currently installed.)~~
~~Preparing to unpack .../linux-headers-5.4.143-rt64_5.4.143-rt64-1_amd64.deb ...~~
~~Unpacking linux-headers-5.4.143-rt64 (5.4.143-rt64-1) ...~~
~~Selecting previously unselected package linux-image-5.4.143-rt64.~~
~~Preparing to unpack .../linux-image-5.4.143-rt64_5.4.143-rt64-1_amd64.deb ...~~
~~Unpacking linux-image-5.4.143-rt64 (5.4.143-rt64-1) ...~~
~~Selecting previously unselected package linux-image-5.4.143-rt64-dbg.~~
~~Preparing to unpack .../linux-image-5.4.143-rt64-dbg_5.4.143-rt64-1_amd64.deb ...~~
~~Unpacking linux-image-5.4.143-rt64-dbg (5.4.143-rt64-1) ...~~
~~Preparing to unpack .../linux-libc-dev_5.4.143-rt64-1_amd64.deb ...~~
~~Unpacking linux-libc-dev:amd64 (5.4.143-rt64-1) over (4.15.0-162.170) ...~~
~~Setting up linux-headers-5.4.143-rt64 (5.4.143-rt64-1) ...~~
~~Setting up linux-image-5.4.143-rt64 (5.4.143-rt64-1) ...~~
~~ * dkms: running auto installation service for kernel 5.4.143-rt64~~
~~       [ OK ]~~
~~update-initramfs: Generating /boot/initrd.img-5.4.143-rt64~~
~~Sourcing file `/etc/default/grub'~~
~~Generating grub configuration file ...~~
~~Found linux image: /boot/vmlinuz-5.4.143-rt64~~
~~Found initrd image: /boot/initrd.img-5.4.143-rt64~~
~~Found linux image: /boot/vmlinuz-5.4.0-90-generic~~
~~Found initrd image: /boot/initrd.img-5.4.0-90-generic~~
~~Found linux image: /boot/vmlinuz-5.4.0-84-generic~~
~~Found initrd image: /boot/initrd.img-5.4.0-84-generic~~
~~Adding boot menu entry for EFI firmware configuration~~
```

```
~~done~~
~~Setting up linux-image-5.4.143-rt64-dbg (5.4.143-rt64-1) ...~~
~~Setting up linux-libc-dev:amd64 (5.4.143-rt64-1) ...~~
```

▸ ~~Now the real time kernel should be installed~~

▸ ~~Reboot the system and check the new kernel version~~

```
~~$ sudo reboot~~
~~Rebooting… … … …~~

~~$ uname -r~~
~~5.4.XXX-rtYY~~
```

1.2.1 Patch된 Real Time Kernel Source Down

2) RT kernel 이 패치된 Kernel을 다운로드
   Linux-stable-rt-5.4.218.tar.gz

1.2.2 Build real time generic kernel

▸ We simply want to use the config of our Ubuntu installation, so we get the Ubuntu config with

```
$ cd ~;
$ mkdir kernel
$ cp ~/Downloads/linux-stable-rt-5.4.218.tar.gz ~/kernel    #Aleady Downloaded kernel source
$ cd ~/kernel
$ tar vzxf linux-stable-rt-5.4.218.tar.gz
$ cd linux-stable-rt-5.4.218
```

▸ Now we're going to build the kernel which will take quite some time. (about 50 min)

```
$ make -j `nproc` deb-pkg LOCALVERSION=-armstrong
… …
… …
  INSTALL ./debian/headertmp/usr/include
dpkg-deb: building package 'linux-headers-5.4.143-rt64' in '../linux-headers-5.4.143-rt64_5.4.143-rt64-1_amd64.deb'.
dpkg-deb: building package 'linux-libc-dev' in '../linux-libc-dev_5.4.143-rt64-1_amd64.deb'.
dpkg-deb: building package 'linux-image-5.4.143-rt64' in '../linux-image-5.4.143-rt64_5.4.143-rt64-1_amd64.deb'.
dpkg-deb: building package 'linux-image-5.4.143-rt64-dbg' in '../linux-image-5.4.143-rt64-dbg_5.4.143-rt64-1_amd64.deb'.
 dpkg-genbuildinfo
 dpkg-genchanges  >../linux-5.4.143-rt64_5.4.143-rt64-1_amd64.changes
dpkg-genchanges: info: including full source code in upload
 dpkg-source -i.git --after-build linux-stable-rt-5.4.143-rt64
dpkg-buildpackage: info: full upload (original source is included)
$

만약 dpk-pkg 관련으로 빌드 에러가 발생하면…
  -   ./vmlinux-gdb.py 를 삭제
  -
```

‣ After the build is finished check the debian packages

```
$ ls -lh ../*deb
-rw-r--r-- 1 ipc1 ipc1  12M 11월 13 18:45 ../linux-headers-5.4.143-rt64_5.4.218-1_amd64_amd64.deb
-rw-r--r-- 1 ipc1 ipc1  59M 11월 13 18:46 ../linux-image-5.4.143-rt64_5.4.218-1_amd64_amd64.deb
-rw-r--r-- 1 ipc1 ipc1 817M 11월 13 18:54 ../linux-image-5.4.143-rt64-dbg_5.4.218-1_amd64_amd64.deb
-rw-r--r-- 1 ipc1 ipc1 1.1M 11월 13 18:45 ../linux-libc-dev_5.4.218-1_amd64_amd64.deb
```

1.2.3 Install the build kernel package and Reboot

‣ Install all kernel debian packages

```
$ sudo dpkg -i ../*.deb
Selecting previously unselected package linux-headers-5.4.143-rt64.
(Reading database ... 201690 files and directories currently installed.)
Preparing to unpack .../linux-headers-5.4.143-rt64_5.4.143-rt64-1_amd64.deb ...
Unpacking linux-headers-5.4.143-rt64 (5.4.143-rt64-1) ...
Selecting previously unselected package linux-image-5.4.143-rt64.
Preparing to unpack .../linux-image-5.4.143-rt64_5.4.143-rt64-1_amd64.deb ...
Unpacking linux-image-5.4.143-rt64 (5.4.143-rt64-1) ...
Selecting previously unselected package linux-image-5.4.143-rt64-dbg.
Preparing to unpack .../linux-image-5.4.143-rt64-dbg_5.4.143-rt64-1_amd64.deb ...
Unpacking linux-image-5.4.143-rt64-dbg (5.4.143-rt64-1) ...
Preparing to unpack .../linux-libc-dev_5.4.143-rt64-1_amd64.deb ...
Unpacking linux-libc-dev:amd64 (5.4.143-rt64-1) over (4.15.0-162.170) ...
Setting up linux-headers-5.4.143-rt64 (5.4.143-rt64-1) ...
Setting up linux-image-5.4.143-rt64 (5.4.143-rt64-1) ...
 * dkms: running auto installation service for kernel 5.4.143-rt64
        [ OK ]
update-initramfs: Generating /boot/initrd.img-5.4.143-rt64
Sourcing file `/etc/default/grub'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.4.143-rt64
Found initrd image: /boot/initrd.img-5.4.143-rt64
Found linux image: /boot/vmlinuz-5.4.0-90-generic
Found initrd image: /boot/initrd.img-5.4.0-90-generic
Found linux image: /boot/vmlinuz-5.4.0-84-generic
Found initrd image: /boot/initrd.img-5.4.0-84-generic
Adding boot menu entry for EFI firmware configuration
```

```
done
Setting up linux-image-5.4.143-rt64-dbg (5.4.143-rt64-1) ...
Setting up linux-libc-dev:amd64 (5.4.143-rt64-1) ...
```

‣ Now the real time kernel should be installed

‣ Reboot the system and check the new kernel version

```
$ sudo reboot
Rebooting… … … …

$ uname -r
5.4.XXX-rtYY
```

1.3 Install NVIDIA Driver for Realtime Kernel

~~1.3.1 처음 설치~~

~~- Link : https://drive.google.com/file/d/1k5RhVegObTI7lmACxw84teBmYxrvWv7e/view?usp=sharing 다운로드~~

```
~~$ sudo service lightdm stop ⇐ 시스템에 lightdm 서비스가 없어도 무방함~~
~~$ sudo vi /etc/modprobe.d/blacklist-nouveau.conf  ⇐ 파일에 추가(blacklist nouveau \ options nouveau modset=0~~
~~$ sudo update-initramfs -u~~
~~$ sudo reboot~~
~~System rebooting…~~

~~$ sudo IGNORE_PREEMPT_RT_PRESENCE=1 bash ./Downloads/NVIDIA-Linux-x86_64-460.39.run~~
~~# 설치 과정에서 다음과 같이 진행.~~
~~# 1. DKMS 설치 ⇒ No 선택~~
~~# 2. Warning... --compat32-libdir option ⇒ OK 넘어감~~
~~# 3, Warning... --glvnd-egl-config-path ⇒ OK 넘어감~~
~~# 4. nvidia-xconfig utility to automatically update your X configuration... ⇒ Yes 선택~~
~~#~~
~~$ sudo reboot~~
```

1.3.1 처음 설치

Link : NVIDIA-Linux-x86_64-460.91.03.run 다운로드

```
$ sudo service lightdm stop ⇐ 시스템에 lightdm 서비스가 없어도 무방함
$ sudo vi /etc/modprobe.d/blacklist-nouveau.conf  ⇐ 파일에 추가(blacklist nouveau \ options nouveau
modset=0
$ sudo update-initramfs -u
$ sudo reboot
System rebooting…

$ sudo IGNORE_PREEMPT_RT_PRESENCE=1 bash ./Downloads/NVIDIA-Linux-x86_64-460.91.03.run
# 설치 과정에서 다음과 같이 진행.
# 1. DKMS 설치 ⇒ No 선택
# 2. Warning... --compat32-libdir option ⇒ OK 넘어감
# 3, Warning... --glvnd-egl-config-path ⇒ OK 넘어감
# 4. nvidia-xconfig utility to automatically update your X configuration... ⇒ Yes 선택
#
$ sudo reboot
```

1.3.2 기존설치된 Nvidia driver 제거하는 방법

기존에 설치한 버전을 확인후, 기존에 설치했던 `NVIDIA-Linux-x86_64-460.91.03.run(예시시)` **이 있는 폴더로 이동**

```
$ sudo bash ./NVIDIA-Linux-x86_64-460.91.03.run  - uninstall

$ sudo reboot
```

1.4 Git client 설치

1.4.1 Install Git & Git-LFS

```
$ sudo apt install git
$ sudo apt install git-lfs
```

1.4.2 Git configuration

```
$ git config --global http.sslVerify false
$ git config --global core.editor vim
$ git config --global merge.tool vim -d
$ git config --global push.default simple
```

1.5 AV Source code Build Environment Setting

1.5.1 AV Source code Download from the git repository

a) For Armstrong 5.0

```
:$ cd ~/
:~$ mkdir Armstrong && cd Armstrong
:~/Armstrong$ git clone https://adtc.swm.ai/gitlab/armstrong/a50.git &&  cd a50
:~/Armstrong/a50$       <= working directory(~/Armstrong/a50)
```

b) For Armstrong 2.5

```
:$ cd ~/
:~$ mkdir Armstrong && cd Armstrong
:~/Armstrong$ git clone https://adtc.swm.ai/gitlab/bak-armstrong/apollo60.git && cd apollo60/Armstrong25
:~/Armstrong/apollo60/Armstrog25$       <= working directory(~/Armstrong/apollo60/Armstrong25)
```

1.5.2 Build Environment Setting

a) .bashrc 설정

```
$ echo "export APOLLO_HOME=$(pwd)" >> ~/.bashrc
```

b) setup_host.sh 설정

```
$ ./docker/setup_host/setup_host.sh
                           ^--- /proc/sys/kernel/core_pattern = /apollo/core/data/core_%e.%p
                           ^--- /etc/crontab + ntpdate
                           ^--- /etc/udev/rules.d/ ← ./docker/setup_host/etc/udev/rules.d/*
                           ^--- /etc/modules ← uvcvideo clock=realtime
```

1.6 Docker and NVIDIA-Docker Installation

1.6.1 Install docker using the installation scripts of the armstrong packages

~~$ chmod +x ./docker/setup_host/install_docker.sh~~

```
$ ./docker/setup_host/install_docker.sh
```

1.6.2 Install Nvidia Container Toolkit

~~Install_nvidia_docker.sh 를 다운로드 하고, docker/setup_host 아래 copy~~

```
$ ~~chmod +x ./docker/setup_host/install_nvidia_docker.sh~~
$ ./docker/setup_host/install_nvidia_docker.sh
$ sudo systemctl restart docker
```

1.6.3 Check Nvidia Container Toolkit

a) docker run gpu

```
$ sudo docker run --rm --gpus all nvidia/cuda:11.0.3-base-ubuntu 18.04 nvidia-smi
[sudo] password for armstrong:
Unabin
54ee1f796a1e: Pull complete
f7bfea53ad12: Pull complete
46d371e02073: Pull complete
b66c17bbf772: Pull complete
3642f1a6dfb3: Pull complete
e5ce55b8b4b9: Pull complete
155bc0332b0a: Pull complete
Digest: sha256:774ca3d612de15213102c2dbbba55df44dc5cf9870ca2be6c6e9c627fa63d67a
Status: Downloaded newer image for nvidia/cuda:11.0-base
Tue Jan 25 04:44:06 2022
+-----------------------------------------------------------------------------+
¦ NVIDIA-SMI 460.39       Driver Version: 460.39       CUDA Version: 11.2     ¦
¦-------------------------------+----------------------+----------------------+
¦ GPU  Name        Persistence-M¦ Bus-Id        Disp.A ¦ Volatile Uncorr. ECC ¦
¦ Fan  Temp  Perf  Pwr:Usage/Cap¦         Memory-Usage ¦ GPU-Util  Compute M. ¦
¦                               ¦                      ¦               MIG M. ¦
¦===============================+======================+======================¦
¦   0  GeForce GTX 108...  Off  ¦ 00000000:01:00.0  On ¦                  N/A ¦
¦  9%   54C   P0    61W / 250W ¦   729MiB / 11175MiB ¦   2%      Default ¦
¦                               ¦                      ¦                  N/A ¦
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
¦ Processes:                                                                  ¦
¦  GPU   GI   CI    PID   Type   Process name                  GPU Memory ¦
¦        ID   ID                                               Usage   ¦
¦=============================================================================¦
+-----------------------------------------------------------------------------+
```

b) nvidia-docker

```
$ sudo nvidia-docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
```

```
   buildx: Docker Buildx (Docker Inc., v0.7.1-docker)
   scan: Docker Scan (Docker Inc., v0.12.0)

 Server:
  Containers: 0
   Running: 0
   Paused: 0
   Stopped: 0
  Images: 0
  Server Version: 20.10.12
  Storage Driver: overlay2
   Backing Filesystem: extfs
   Supports d_type: true
   Native Overlay Diff: true
   userxattr: false
  Logging Driver: json-file
  Cgroup Driver: cgroupfs
  Cgroup Version: 1
  Plugins:
   Volume: local
   Network: bridge host ipvlan macvlan null overlay
   Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
  Swarm: inactive
  Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux nvidia runc
  Default Runtime: runc
  Init Binary: docker-init
  containerd version: 7b11cfaabd73bb80907dd23182b9347b4245eb5d
  runc version: v1.0.2-0-g52b36a2
  init version: de40ad0
  Security Options:
   apparmor
   seccomp
    Profile: default
  Kernel Version: 5.4.143-rt64
  Operating System: Ubuntu 18.04.6 LTS
  OSType: linux
  Architecture: x86_64
  CPUs: 8
  Total Memory: 31.22GiB
  Name: armstrong-Nuvo-6108GC
  ID: AQTF:5S2G:TNUW:7ATY:INDH:Z37W:JSGX:XSBI:VKZH:7W3D:KZM3:7EBJ
  Docker Root Dir: /var/lib/docker
  Debug Mode: false
  Registry: https://index.docker.io/v1/
  Labels:
  Experimental: false
  Insecure Registries:
   127.0.0.0/8
  Live Restore Enabled: false

 WARNING: No swap limit support
```

1.7 ESD CAN Driver & Library Installation

1.7.1 Install ESD CAN PCIE402 Driver (리부팅 필요)

a) "esdcan-pcie402-linux-x86_64-4.1.1.tgz" 을 준비한다.

```
[]@host:~/$ cp ~/Downloads/esdcan-pcie402-linux-x86_64-4.1.1.tgz ~/kernel/ && cd ~/kernel/
[]@host:~/kernel$ tar xzf esdcan-pcie402-linux-x86_64-4.1.1.tgz
[]@host:~/kernel$ cd esdcan-pcie402-linux-x86_64-4.1.1
[]@host:~/<esdcan-...>$ sudo dkms add ./src
Creating symlink /var/lib/dkms/esdcan-pcie402-linux-x86_64/4.1.1/source ->
                 /usr/src/esdcan-pcie402-linux-x86_64-4.1.1

DKMS: add completed.
[]@host:~/<esdcan-...>$ sudo dkms build esdcan-pcie402-linux-x86_64/4.1.1

Kernel preparation unnecessary for this kernel.  Skipping...

Building module:
cleaning build area...
make -j8 KERNELRELEASE=5.4.143-rt64 -C /lib/modules/5.4.143-rt64/build
M=/var/lib/dkms/esdcan-pcie402-linux-x86_64/4.1.1/build...
Signing module:
 - /var/lib/dkms/esdcan-pcie402-linux-x86_64/4.1.1/5.4.143-rt64/x86_64/module/esdcan-pcie402.ko
EFI variables are not supported on this system
Secure Boot not enabled on this system.
cleaning build area...

DKMS: build completed.
----------------------------------------------------------------------------------------------------
$ sudo dkms build esdcan-pcie402-linux-x86_64/4.1.1

Kernel preparation unnecessary for this kernel.  Skipping...

Building module:
cleaning build area...
make -j8 KERNELRELEASE=5.4.143-rt64 -C /lib/modules/5.4.143-rt64/build
M=/var/lib/dkms/esdcan-pcie402-linux-x86_64/4.1.1/build...
Signing module:
Generating a new Secure Boot signing key:
Can't load /var/lib/shim-signed/mok/.rnd into RNG     ← Check~~~
140329077428672:error:2406F079:random number generator:RAND_load_file:Cannot open
file:../crypto/rand/randfile.c:88:Filename=/var/lib/shim-signed/mok/.rnd
Generating a RSA private key
.+++++
.+++++
writing new private key to '/var/lib/shim-signed/mok/MOK.priv'
-----
 - /var/lib/dkms/esdcan-pcie402-linux-x86_64/4.1.1/5.4.143-rt64/x86_64/module/esdcan-pcie402.ko
EFI variables are not supported on this system
Secure Boot not enabled on this system.
cleaning build area...
```

```
DKMS: build completed.



[]@host:~/<esdcan-...>$ sudo dkms install esdcan-pcie402-linux-x86_64/4.1.1

esdcan-pcie402:
Running module version sanity check.
 - Original module
 - Installation
   - Installing to /lib/modules/5.4.143-rt64/updates/dkms/

depmod...

DKMS: install completed.
[]@host:~/<esdcan-...>$ sudo cp udev/esdcan-pcie402-dev.rules /etc/udev/rules.d/99-esdcan.rules
[]@host:~/<esdcan-...>$ sudo reboot

(tips!!) dkms remove esdcan-pcie402-linux-x86_64/4.1.1  - all
```

1.7.2 Install ESD CAN Library

a) Host library installation

```
[]@host:~/$ cd kernel/esdcan-pcie402-linux-x86_64-4.1.1/
[]@host:~/<esdcan-...>$ sudo cp include/ntcan.h /usr/local/include
[]@host:~/<esdcan-...>$ sudo cp lib64/libntcan.so.4.2.3 /usr/local/lib
[]@host:~/<esdcan-...>$ cd ./usr/local/lib/libntcan.so.4
[]@host:~/<esdcan-...>$ sudo ldconfig
```

~~b) AV Source Code build library installation ($APOLLO_HOME basis, install version > v4.2.3)~~

~~[]@host:~/$ cp -pr ~/kernel/esdcan-pcie402-linux-x86_64-4.1.1/include/~~
~~$APOLLO_HOME/third_party/can_card_library/esd_can/~~
~~[]@host:~/$ mkdir -p $APOLLO_HOME/third_party/can_card_library/esd_can/lib/~~
~~[]@host:~/$ cp -p ~/kernel/esdcan-pcie402-linux-x86_64-4.1.1/lib64/libntcan.so.4.2.3~~
~~$APOLLO_HOME/third_party/can_card_library/esd_can/lib/~~
~~[]@host:~/$ ln -s libntcan.so.4 $APOLLO_HOME/third_party/can_card_library/esd_can/lib/libntcan.so~~
~~[]@host:~/$ ln -s libntcan.so.4.2.3~~
~~$APOLLO_HOME/third_party/can_card_library/esd_can/lib/libntcan.so.4~~

1.7.3 ESD CAN Card 커널모듈 동작 확인

a) 커널모듈이 정상적으로 로드되었는지 확인

```
[]@host:~/$ lsmod | grep esdcan
esdcan_pcie402         131072  0

[]@host:~/$ modinfo esdcan-pcie402
filename:       /lib/modules/5.4.143-rt64/updates/dkms/esdcan-pcie402.ko
```

```
license:        Proprietary
description:    esd CAN driver
author:         esd electronics gmbh, support@esd.eu
alias:          pci:v000012FEd00000402sv000012FEsd00001403bc*sc*i*
alias:          pci:v000012FEd00000402sv000012FEsd00001402bc*sc*i*
alias:          pci:v000012FEd00000402sv000012FEsd00000403bc*sc*i*
alias:          pci:v000012FEd00000402sv000012FEsd00000402bc*sc*i*
alias:          pci:v000012FEd00000402sv000012FEsd00000401bc*sc*i*
depends:
retpoline:      Y
name:           esdcan_pcie402
vermagic:       5.4.143-rt64 SMP preempt_rt mod_unload modversions
signat:         PKCS#7
signer:
sig_key:
sig_hashalgo:   md4
parm:           major:major number to be used for the CAN card (uint)
parm:           mode:activate certain driver modes, e.g. LOM (int)
parm:           verbose:change verbose level of driver (uint)
parm:           errorinfo:enable/disable extended error info (default: on) (int)
parm:           compat32:disable 32-Bit compatibility on 64-Bit systems (int)
parm:           pcimsg:enable/disable pcimsg interface on firmware drivers (default: on) (int)
parm:           clock:LEAVE THIS ONE ALONE, works with special hardware, only (iln -s
libntcan.so.4.2.3 $APOLLO_HOME/third_party/can_card_library/esd_can/lib/libntcan.so.4nt)
parm:           txtswin:override default TX-TS-window size (in ms) (int)
```

b) /etc/udev/rules.d/99-esdcan.rules 에서 can node 생성함 (장착된 캔카드의 채널 만큼 노드 생성)

```
 []@host:~/$ ls /dev/can* (예, 4채널 캔카드 2개 장착)
/dev/can0  /dev/can1  /dev/can2  /dev/can3  /dev/can4  /dev/can5  /dev/can6  /dev/can7
```

1.7.4 ESD CAN Card 동작 확인

a) library 에 포함된 cantest 프로그램 사용

```
[]@host:~/$ cd ~/kernel/esdcan-pcie402-linux-x86_64-4.1.1/
[]@host:~/kernel/esdcan-pcie402-linux-x86_64-4.1.1/$ ./bin64/cantest
CAN Test FD Rev 3.0.13  -- (c) 1997 - 2020 esd electronics gmbh

Available CAN-Devices:
Net   0: ID=CAN_PCIE402 (4 ports) Serial no.: HD002736
      Versions (hex): Lib=4.2.03 Drv=4.1.01 HW=1.0.1E FW=0.0.47 (0.0.00)
      Baudrate=00000002 (500 KBit/s) Status=0000 Features=00008ffa
      Ctrl=esd Advanced CAN Core @ 80 MHz (Error Passive / REC:0 / TEC:128)
      Transceiver=TI SN65HVD265  TDC=Automatic (0 / 0)
      TimestampFreq=80.000000 MHz Timestamp=000000188CB65B3A
Net   1: ID=CAN_PCIE402 (4 ports) Serial no.: HD002736
      Versions (hex): Lib=4.2.03 Drv=4.1.01 HW=1.0.1E FW=0.0.47 (0.0.00)
      Baudrate=7fffffff (Not set) Status=0000 Features=00008ffa
      Ctrl=esd Advanced CAN Core @ 80 MHz (Error Active / REC:0 / TEC:0)
      Transceiver=TI SN65HVD265  TDC=Automatic (0 / 0)
      TimestampFreq=80.000000 MHz Timestamp=000000188CB69A7B
Net   2: ID=CAN_PCIE402 (4 ports) Serial no.: HD002736
      Versions (hex): Lib=4.2.03 Drv=4.1.01 HW=1.0.1E FW=0.0.47 (0.0.00)
      Baudrate=7fffffff (Not set) Status=0000 Features=00008ffa
      Ctrl=esd Advanced CAN Core @ 80 MHz (Error Active / REC:0 / TEC:0)
      Transceiver=TI SN65HVD265  TDC=Automatic (0 / 0)
      TimestampFreq=80.000000 MHz Timestamp=000000188CB6C76B
Net   3: ID=CAN_PCIE402 (4 ports) Serial no.: HD002736
      Versions (hex): Lib=4.2.03 Drv=4.1.01 HW=1.0.1E FW=0.0.47 (0.0.00)
      Baudrate=7fffffff (Not set) Status=0000 Features=00008ffa
      Ctrl=esd Advanced CAN Core @ 80 MHz (Error Active / REC:0 / TEC:0)
      Transceiver=TI SN65HVD265  TDC=Automatic (0 / 0)
      TimestampFreq=80.000000 MHz Timestamp=000000188CB6FC44
Net   4: ID=CAN_PCIE402 (2 ports) Serial no.: HF003286
      Versions (hex): Lib=4.2.03 Drv=4.1.01 HW=1.0.16 FW=0.0.44 (0.0.00)
      Baudrate=7fffffff (Not set) Status=0000 Features=00000ffa
      Ctrl=esd Advanced CAN Core @ 80 MHz (Error Active / REC:0 / TEC:0)
      Transceiver=TI SN65HVD265
      TimestampFreq=80.000000 MHz Timestamp=000000188C8A9FC5
Net   5: ID=CAN_PCIE402 (2 ports) Serial no.: HF003286
      Versions (hex): Lib=4.2.03 Drv=4.1.01 HW=1.0.16 FW=0.0.44 (0.0.00)
      Baudrate=7fffffff (Not set) Status=0000 Features=00000ffa
      Ctrl=esd Advanced CAN Core @ 80 MHz (Error Active / REC:0 / TEC:0)
      Transceiver=TI SN65HVD265
      TimestampFreq=80.000000 MHz Timestamp=000000188C8AD05B

Syntax: cantest test-Nr [net id-1st id-last count
      txbuf rxbuf txtout rxtout baud[:data_baud] testcount data0 data1 ...]
Test   0:  canSend()
Test  20:  canSendT()
Test  50:  canSend() with incrementing ids
Test  60:  canSendX()
Test  90:  canSend() after using NTCAN_IOCTL_GET_TX_MSG_COUNT
Test   1:  canWrite()
Test  21:  canWriteT()
Test  51:  canWrite() with incrementing ids
Test  61:  canWriteX()
Test   2:  canTake()
Test  12:  canTake() with time-measurement for 10000 can-frames
Test  22:  canTakeT()
Test  32:  canTake() in Object-Mode
Test  42:  canTakeT() in Object-Mode
Test  62:  canTakeX()
Test  72:  canTakeX() with time-measurement for 10000 can-frames
Test  82:  canTakeX() in Object-Mode
Test  92:  canTake() after using NTCAN_IOCTL_GET_RX_MSG_COUNT
Test   3:  canRead()
Test  13:  canRead() with time-measurement for 10000 can-frames
Test  23:  canReadT()
Test  63:  canReadX()
Test  73:  canReadX() with time-measurement for 10000 can-frames
Test   4:  canReadEvent()
Test  64:  Retrieve bus statistics (every tx timeout)
Test  74:  Reset bus statistics
Test  84:  Retrieve bitrate details (every tx timeout)
Test   5:  canSendEvent()
Test   8:  Create auto RTR object
Test   9:  Wait for RTR reply
Test  19:  Wait for RTR reply without text-output
Test 100:  Object Scheduling test
Test 110:  Object Scheduling test with cmsg_x
Test  -2:  Overview without syntax help
Test  -3:  Overview without syntax help but with feature flags details
Test  -4:  Overview without syntax help but with bit rate index table
```

1.8 ASU System(Basa Driver & adv tools Build) Installation

1.8.1 Check Basa Driver

a) IPC에 장착된 ASU PCIe Card & Baidu Sensor Box 가 리눅스에서 인식되는지 확인

```
[]@host:~/$ lspci -tv
-[0000:00]-+-00.0  Intel Corporation Xeon E3-1200 v5/E3-1500 v5/6th Gen Core Processor Host
Bridge/DRAM Registers
             +-01.0-[01]--+-00.0  NVIDIA Corporation GP102 [GeForce GTX 1080 Ti]
             ¦              \-00.1  NVIDIA Corporation GP102 HDMI Audio Controller
             +-14.0  Intel Corporation 100 Series/C230 Series Chipset Family USB 3.0 xHCI Controller
             +-16.0  Intel Corporation 100 Series/C230 Series Chipset Family MEI Controller #1
             +-16.3  Intel Corporation 100 Series/C230 Series Chipset Family KT Redirection
             +-17.0  Intel Corporation Q170/Q150/B150/H170/H110/Z170/CM236 Chipset SATA Controller
[AHCI Mode]
             +-1b.0-[02]----00.0  Device 1d22:2083  ⇐ ASU PCIe Card & Baidu Sensor Box
             +-1c.0-[03]----00.0  ESD Electronic System Design GmbH Device 0402
             +-1d.0-[04]----00.0  Intel Corporation I210 Gigabit Network Connection
             +-1f.0  Intel Corporation C236 Chipset LPC/eSPI Controller
             +-1f.2  Intel Corporation 100 Series/C230 Series Chipset Family Power Management
Controller
             +-1f.3  Intel Corporation 100 Series/C230 Series Chipset Family HD Audio Controller
             +-1f.4  Intel Corporation 100 Series/C230 Series Chipset Family SMBus
             \-1f.6  Intel Corporation Ethernet Connection (2) I219-LM
```

b) 리눅스에서 Basa Kernel Module 이 로딩되는지 확인

```
[]@host:~/$ modinfo basa
filename:     /lib/modules/5.4.143-rt64/kernel/drivers/baidu/basa/basa.ko
version:      3.0.0.5
description:  basa: Baidu Sensor Aggregator Driver
author:       BAIDU USA, LLC
license:      Dual BSD/GPL
srcversion:   D91370593554AA59745D919
alias:        pci:v00001D22d00002083sv*sd*bc*sc*i*
depends:      videobuf2-v4l2,videodev,videobuf2-common,videobuf2-vmalloc
retpoline:    Y
intree:       Y
name:         basa
vermagic:     5.4.143-rt64 SMP preempt_rt mod_unload modversions
signat:       PKCS#7
signer:
sig_key:
sig_hashalgo:   md4
parm:         gpssynctime:GPS sync interval in seconds (uint)
parm:         gpssmoothstep:GPS sync smoothing step length in usec (uint)
parm:         gpssmoothmax:GPS sync smoothing max tolerance in msec (uint)
parm:         gpschecksum:GPRMC checksum validation (uint)
parm:         gpshifactor:high water mark factor for GPS time check (uint)
parm:         gpslofactor:low water mark factor for GPS time check (uint)
parm:         videofmt:video format (uint)
```

```
parm:           videobufnum:video buffer number (uint)
parm:           videozerocopy:video zero copy (uint)
parm:           videotstype:video timestamp type (uint)
parm:           videoflash16:camera flash 16b address width (int)
parm:           videopinswap:camera pin swap (int)
parm:           videoerrtol:video error tolerant (uint)
parm:           videoerrthr:video error threshold (uint)
parm:           canrxbuf:CAN Rx buffer number (uint)
parm:           nocanhwts:disable CAN Rx H/W timestamp (uint)
parm:           canrxdma:enable CAN Rx DMA mode (uint)
parm:           cantxdma:enable CAN Tx DMA mode (uint)
parm:           trace:Trace level bitmask (uint)
parm:           bringup:bringup mode (uint)
parm:           fwupdate:card firmware update enabling (uint)
parm:           dbgregdump:enable debug register dump (uint)
parm:           statslog:statistics log interval (uint)

[]@host:~/$ cd kernel
[]@host:~/kernel$ git clone https://github.com/ApolloAuto/apollo-contrib.git
[]@host:~/kernel$ cd apollo-contrib/baidu
[]@host:~/kernel/apollo-contrib/baidu$ ./build.sh
[]@host:~/kernel/apollo-contrib/baidu$ cd output
[]@host:~/kernel/apollo-contrib/baidu/output$ unzip plat-sw-3.0.1.zip
[]@host:~/kernel/apollo-contrib/baidu/output$ cd kernel/drivers/baidu/basa
[]@host:~/kernel/apollo-contrib/baidu/output/kernel/drivers/baidu/basa$ sudo cp basa.ko
/lib/modules/5.4.143-rt64/kernel/drivers/baidu/basa/.
[]@host:~/kernel/apollo-contrib/baidu/output/kernel/drivers/baidu/basa$ sudo reboot
```

b) v4l2 tool 설치

```
[]@host:~/$ sudo apt install v4l-utils
```

1.9 ESD Novatel Driver & Library Installation

1.9.1 https://www.novatel.com/support/info/documents/809 접속 Or USB Drivers | NovAtel

1.9.2 USB Drivers => Linux USB Driver (Ver 1.1.0) 다운로드

1.9.3 cd ~/Downloads

1.9.4 tar xzvf ngpsusbpackage.tar.gz

1.9.5 cd ngpsusbpackage

1.9.6 sudo dpkg -i ngpsusb.deb

1.9.7 IPC 재부팅 후 터미널에 ls /dev/novatel* 입력 후 출력 확인

1.10 PTP 설치 (IPC 2개 이상 사용시 IPC1에 설치)

1.10.1 linuxptp 설치

a) 시작 (처음 도커 이미지를 받는 경우는 많은 시간이 걸림)

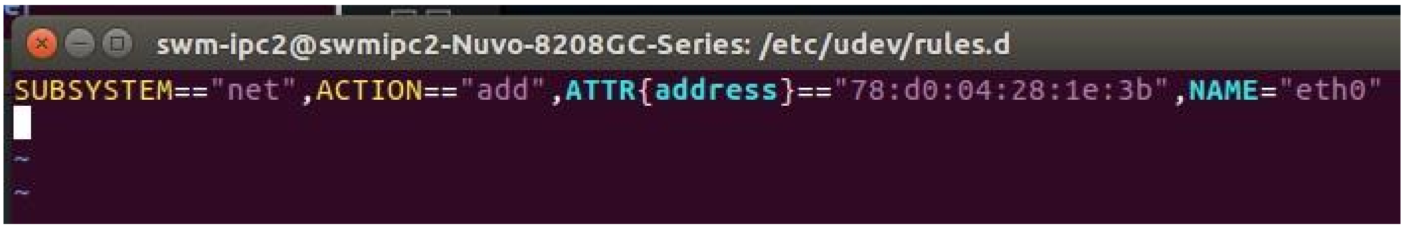
```
$ sudo apt install linuxptp
```

1.10.2 eth0 변경방법

a) ifconfig 명령으로 각각의 ethernet device name 을 확인한다.
예를들어 device name 이 "enp0s31f6" 이고, hardware address (mac address – 78:d0:04:28:1e:3b) 인경우

b) `$ cd /etc/udev/rules.d`

c) "sudo vim 10-rename-network.rules" 명령으로 rules 파일을 만들어서 아래내용을 기입한다.



```
swm-ipc2@swmipc2-Nuvo-8208GC-Series: /etc/udev/rules.d
SUBSYSTEM=="net",ACTION=="add",ATTR{address}=="78:d0:04:28:1e:3b",NAME="eth0"

~
~
```

d) `$ sync`
`$ reboot`

e) ifconfig 로 "eth0" ethernet device name 을 확인한다.

1.10.3 IPC 에서 PTP 실행

# IPC1 :

```
$ sudo ./ptp4l -i eth0 -m &

# IPC2 :
$ sudo ./ptp4l -i eth0 -m -s &
$ sudo ./phc2sys -a -r &
```

1.11 Start Armstrong Docker and Build Armstrong Source

1.11.1 Docker 시작 & 진입

a) 시작 (처음 도커 이미지를 받는 경우는 많은 시간이 걸림)

```
$ ./docker/scripts/dev_start.sh
```

a) 진입

```
$ ./docker/scripts/dev_into.sh    <---- in_dev_docker 쉘모드로 변경됨.
```

1.11.2 Clean and Build Source Code

```
[]@in_dev_docker:/apollo$ ./apollo.sh clean
[]@in_dev_docker:/apollo$ ./apollo.sh build_opt_gpu
```

1.12 Dreamview Start

1.12.1 Start bootstrap

```
[]@in_dev_docker:/apollo$ ./scripts/bootstrap.sh
```

1.12.2 브라우저에서 dreamview 접속

```
http://localhost:8888
```

1.13 control-platform-node-bridge 설치

1.13.1 control-platform-node-bridge 설치

```
$ cd /home/armstrong/Armstrong
$ git clone https://adtc.swm.ai/gitlab/control-platform/control-platform-node-bridge.git
$ echo export BRIDGE_HOME=/home/armstrong/Armstrong/control-platform-node-bridge >> ~/.bashrc
$ source ~/.bashrc
$ cd /home/armstrong/Armstrong/control-platform-node-bridge
$ docker-compose up -dote: HTTP Basic: Access denied. The provided password or token is incorrect or
your account has 2FA enabled and you must use a personal access token instead of a password. See
https://adtc.swm.ai/gitlab/help/topics/git/troublesh
```

참고

1.14 시작 및 종료 서비스 설치

1.14.1 작업 위치

do

```
$ cd /home/armstrong/Armstrong/apollo60/Armstrong25
$ git remote -v
origin  https://adtc.swm.ai/gitlab/bak-armstrong/apollo60.git (fetch)
origin  https://adtc.swm.ai/gitlab/bak-armstrong/apollo60.git (push)
$ ls WORKSPACE .gitignore
.gitignore  WORKSPACE
```

혹은

```
$ cd /home/armstrong/Armstrong/a50
$ git remote -v
origin  https://adtc.swm.ai/gitlab/armstrong/a50.git (fetch)
origin  https://adtc.swm.ai/gitlab/armstrong/a50.git (push)
$ ls WORKSPACE .gitignore
.gitignore  WORKSPACE
```

1.14.2 Armstrong 서비스 설치

```
$ cd services/armstrong
$ head install.sh ⇐ 쉘 스크립트 앞부분 표시용
#!/bin/bash

#Tested on Ubuntu 18.04
#일반 사용자 계정으로 수행해야 함

$ head uninstall.sh ⇐ 쉘 스크립트 앞부분 표시용
#!/bin/bash

#Tested on Ubuntu 18.04
#일반 사용자 계정으로 수행해야 함
```

먼저 uninstall 을 하고 그 다음에 install 을 하도록 한다. 수행할 스크립트에는 맨 위 자리에 간단한 안내 메세지가 주석으로 되어 있다. 위와 같이 살펴보면 install/uninstall 모두 일반 사용자 계정으로 수행해야 한다고 되어있다. 즉, sudo 를 붙여서 수행하지 말며 su 를 사용해서 다른 사용자로 전환해서도 수행하지 말라는 것이다. 이는 중요한데, 왜냐하면 시스템 시작시 지금 설치를 행한 사용자 계정으로 암스트롱을 시작시키기 때문이다. 아폴로의 시작스크립트는 작업을 수행한 사용자가 누구인가에 민감하게 동작하게 되어있다.

```
$ whoami
armstrong

$ ./uninstall.sh
3734c8c9207f
d82822e81ccd
dd8e23a3d044
57f2df1424ff
e95effcf83c0
3734c8c9207f
```

```
[sudo] password for armstrong:
Removed /etc/systemd/system/multi-user.target.wants/armstrong.service.
```

이전에 설치했던 이력이 없다면 이렇게 제거를 먼저 할 필요는 없다.

```
$ whoami
armstrong

$ ./install.sh
Created symlink /etc/systemd/system/multi-user.target.wants/armstrong.service →
/etc/systemd/system/armstrong.service.
```

이렇게 설치가 완료된다.

```
$ tree /etc/armstrong_starter/
/etc/armstrong_starter/
├── armstrong_starter.sh
└── logs

1 directory, 1 file

$ ls -alh /etc/systemd/system/armstrong.service
-rw-r--r-- 1 root root 376 10월 12 09:24 /etc/systemd/system/armstrong.service
```

설치가 된 파일들은 위와 같다.

1.14.3 Terminator 서비스 설치

```
$ cd services/terminator
$ head install.sh ⇐ 쉘 스크립트 앞부분 표시용
#!/bin/bash

#Tested on Ubuntu 18.04
#일반 사용자 계정으로 수행해야 함
```

수행할 스크립트에는 맨 위 자리에 간단한 안내 메세지가 주석으로 되어 있다. 위와 같이 살펴보면 일반 사용자 계정으로 수행해야 한다고 되어있다. 즉, sudo 를 붙여서 수행하지 말며 su 를 사용해서 다른 사용자로 전환해서도 수행하지 말라는 것이다.

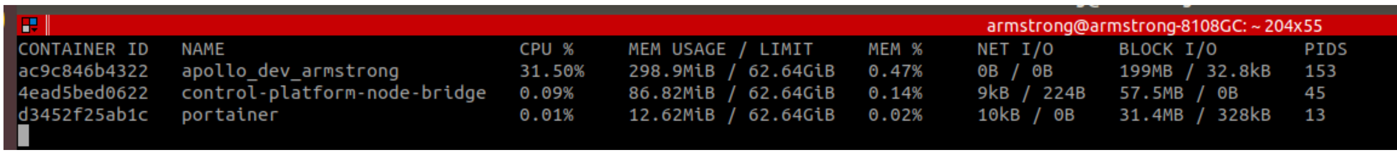
```
$ whoami
armstrong

$ sudo apt install inotify-tools

$ ./install.sh
```

```
[sudo] password for armstrong:
```

1.14.4 재부팅 및 설치 확인

```
Dreamview 에 접속 후 poweroff 버튼 클릭
```

시스템 종료 뒤 전원을 다시 켜서 로그인을 해보자. 로그인을 하면 자동 시작된 암스트롱 도커 컨테이너와 기타 컨테이너들이 보여진다.



```
armstrong@armstrong-8108GC: ~ 204x55
CONTAINER ID   NAME                          CPU %    MEM USAGE / LIMIT      MEM %    NET I/O       BLOCK I/O          PIDS
ac9c846b4322   apollo_dev_armstrong          31.50%   298.9MiB / 62.64GiB    0.47%    0B / 0B       199MB / 32.8kB     153
4ead5bed0622   control-platform-node-bridge  0.09%    86.82MiB / 62.64GiB    0.14%    9kB / 224B    57.5MB / 0B        45
d3452f25ab1c   portainer                     0.01%    12.62MiB / 62.64GiB    0.02%    10kB / 0B     31.4MB / 328kB     13
```

아래 명령으로 출력되는 내용이다.

```
$ docker stats -a
```

또한 systemd의 journalctl 로 보여지는 암스트롱 서비스의 시작로그 창도 보여진다.

1.15 IPC System reboot

```
$ sudo reboot
```

1.16 IPC Bios 설정 변경

While starting up the computer, press F2 to enter BIOS setup menu.

Advanced 탭 => Fan Control Configuration
 Fan Start Trip Point: 30 => 20으로 변경
 Fan Max. Trip Point: 75 => 50으로 변경

Advanced 탭 => System Agent (SA) Configuration => Graphics Configuration
 Primary Display: AUTO => PEG로 변경

Power 탭 => Power & Performance
 SKU Power Config: 35W => MAX. TDP로 변경

Exit 탭 => Exit Saving Changes 눌러 변경사항 저장 후 IPC 재부팅

Shko

1.17 차량별 Setting 상태

1) IPC 이더넷 포트 설정

		Ethernet	
		ETH 0 (인터넷, V2X, GPS)	ETH 1 (라이다 센서)
G80 #1 G80 #2 —---------- KA4 #1 KA4 #2 —---------- IGFL #2 IGFL #3	IPC	IP : 192.168.20.100 Subnet mask : 255.255.255.0 Gateway : 192.168.20.1 DNS : 8.8.8.8	IP : 192.168.10.100 Subnet mask : 255.255.255.0 Gateway : DNS :
IGFL #1	IPC 1	IP : 192.168.20.100 Subnet mask : 255.255.255.0 Gateway : 192.168.20.1 DNS : 8.8.8.8	IP : 192.168.10.100 Subnet mask : 255.255.255.0 Gateway : DNS :
	IPC 2	IP : 192.168.20.101 Subnet mask : 255.255.255.0 Gateway : 192.168.20.1 DNS : 8.8.8.8	

2) 라이다 센서 이더넷 포트 설정

종류	제품	설정	장착 차량	비고
Roof Lider	Velodyne VLS-128		G80 #1 / G80 #2 KA4 #1 / KA4 #2	
	Robsense Ruby		IGFL #3	
	Robsense Ruby lite	IP : 192.168.10.128 Dest : 192.168.10.100	IGFL #1 / IGFL #2	
Left Lider	Robosense Bpearl	IP : 192.168.10.110 Dest : 192.168.10.100	IGFL #1	
Front Lider	Robosense Bpearl	IP : 192.168.10.111 Dest : 192.168.10.100	IGFL #1	
Right Lider	Robosense Bpearl	IP : 192.168.10.112 Dest : 192.168.10.100	IGFL #1	

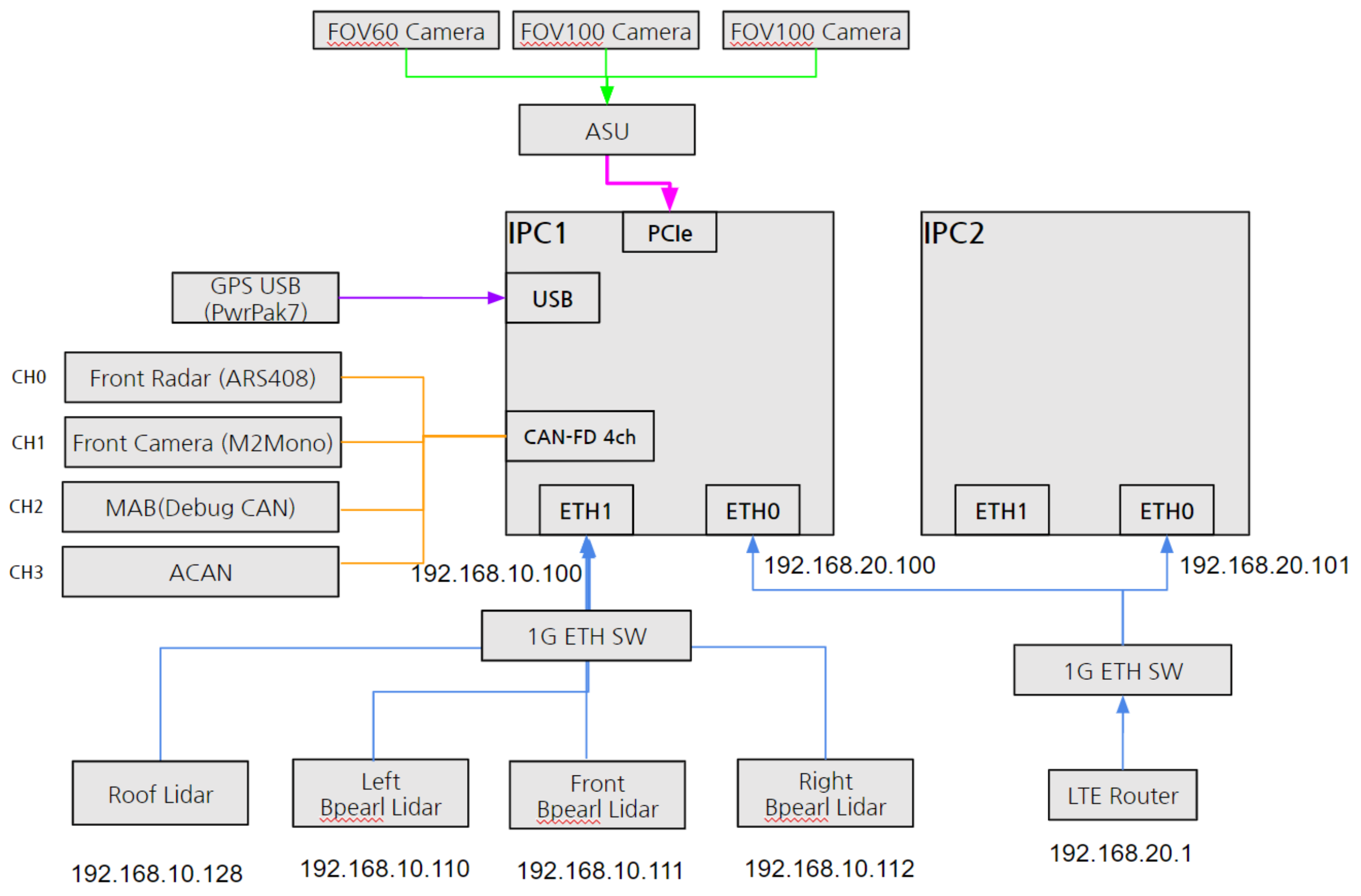
3) IPC CAN 장치 설정

		CAN			
		CAN 0	CAN 1	CAN 2	CAN 3
G80 #1	IPC				
G80 #2	IPC				
KA4 #1	IPC				
KA4 #2	IPC				
IGFL #1	IPC 1	Front Radar (ARS408)	Front camera (M2Mono)	Debug CAN (MAB)	A-CAN (ACU)
IGFL #2	IPC				
IGFL #2	IPC				

4) IPC ASU/Camera 장치 설정

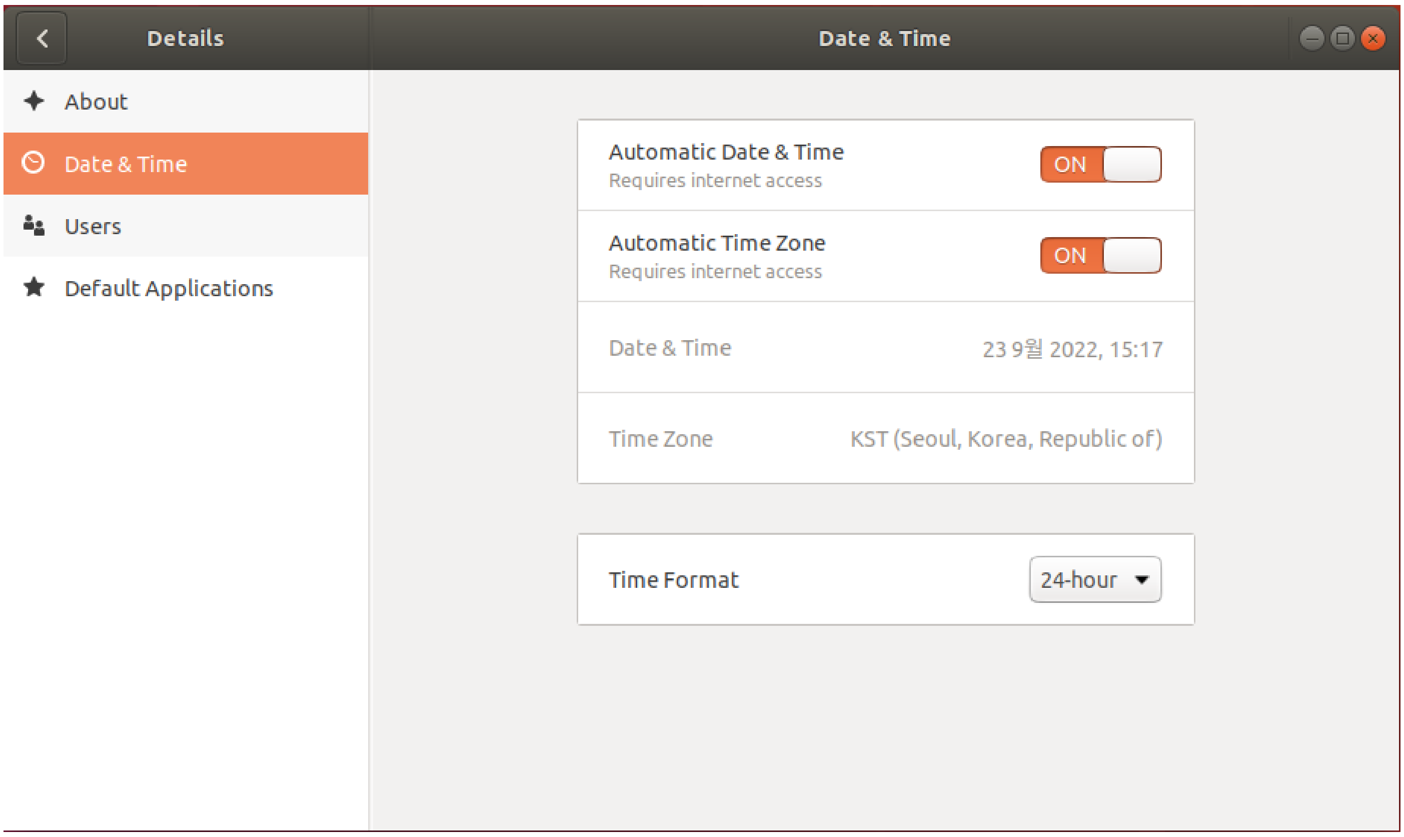
		ASU				
		Video0	Video1	Video2	Video3	Video4
G80 #1	IPC					
G80 #2	IPC					
KA4 #1	IPC					
KA4 #2	IPC					
IGFL #1	IPC 1	차량 신호등 FOV60	Left 보행자 FOV100	Right 보행자 FOV100		
IGFL #2	IPC					
IGFL #2	IPC					

1.17.1 IGFL 1호차 시스템 구조



1.18 Ubuntu IPC 시간 설정

ubutu booting 시 시간 정보를 인터넷에서 받아서 설정하도록 되어 있고, 만일 인터넷이 연결되어 있지 않으면 내부 RTC시간 정보 값으로 동작되도록 되어 있음.
Ubuntu 설정 화면에서 다음과 같이 설정되어 있는지 확인.



Automatic Date & Time — Requires internet access — ON

Automatic Time Zone — Requires internet access — ON

Date & Time: 23 9월 2022, 15:17

Time Zone: KST (Seoul, Korea, Republic of)

Time Format: 24-hour

1.19 Trouble case

1.19.1 IPC 네트워크 이상동작 시

< 방화벽 상태 확인 >

방화벽 동작 확인 => `sudo ufw status verbose`

방화벽 비활성화 => `sudo ufw disable`