

QT 설치

이 파일은 ti-processor-sdk-linux-am57xx-evm-03.02.00.05 을 기반으로 작성 했으며 ti-processor-sdk-linux-am57xx-evm-03.02.00.05 설치와 minicom이 설치 안돼 있을시 카페에서 설치 한후 보자

qt 설치를 하기전에 되었어야 하는게 sd 카드에 sdk 설치완료 , 컴퓨터와 하드웨어 제품 연결, qt 설치완료 설치 하는방법들은 원문사이트 들어가면 볼 수 있다.

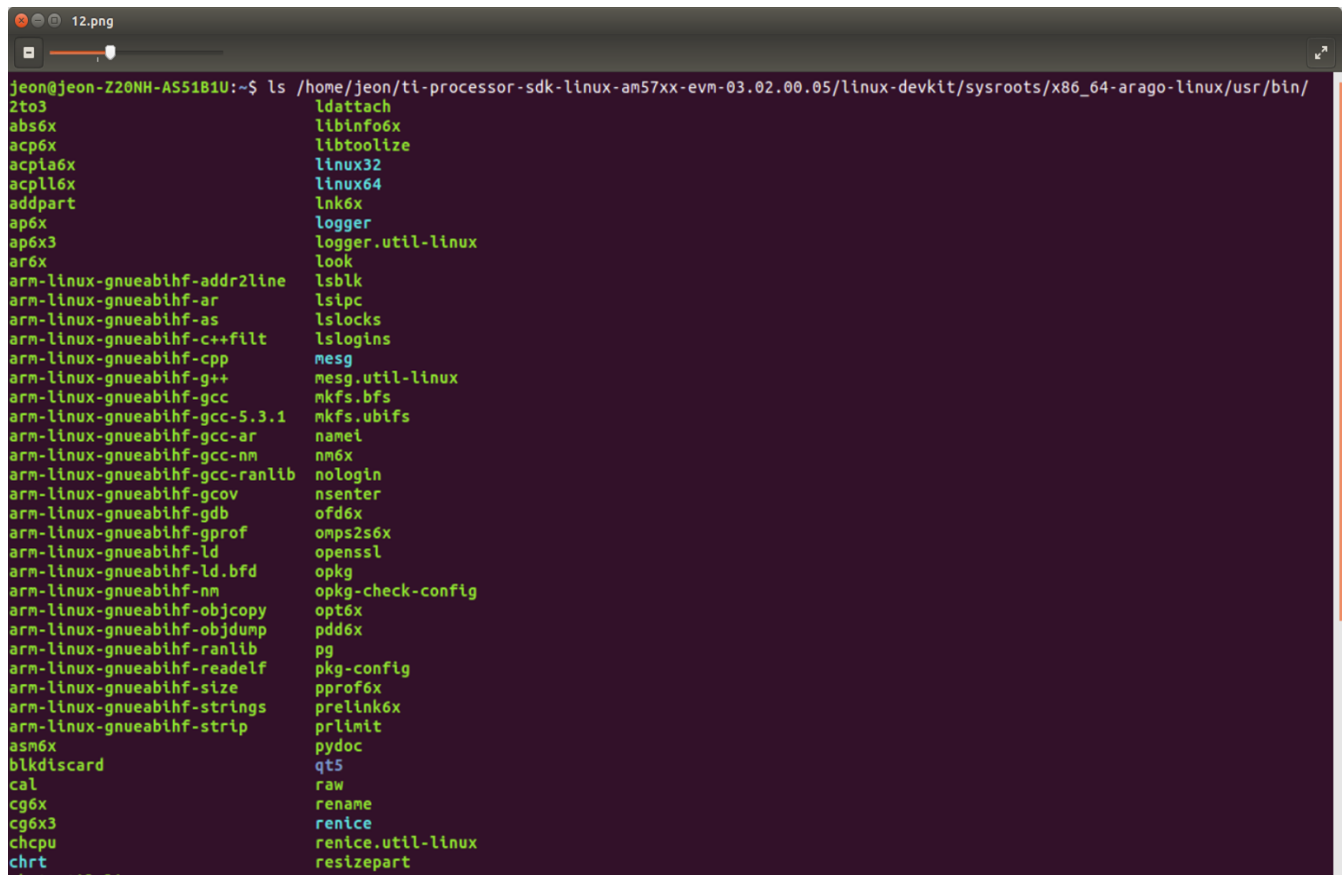
원문 사이트 :

http://processors.wiki.ti.com/index.php/Sitara_Linux_Training:_Hands_on_with_QT

설치를 하면서 컴퓨터는 제품(여기서는 am5728) 이 연결돼 있어야 하며 제품은 인터넷이 연결 돼 있어야 한다.

home/koitt/tisdk3.02.00.05/linux-devkit/sysroots/armv7ahf-neon-linux-gnueabi/usr/include~~~~~ 여기서 koitt 는 우분투 만들때 적은 이름으로 사용자가 정한 이름으로 적자

1. ls 목록에 <사진 1> 처럼 아래와 같이 교차 컴파일 도구 목록들이 있는지 확인



```
jeon@jeon-Z20NH-A551B1U:~$ ls /home/jeon/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/
2to3          ldattach
abs6x         libinfo6x
acp6x         libtoolize
acpla6x       linux32
acpll6x       linux64
addpart       lnk6x
ap6x          logger
ap6x3         logger.util-linux
ar6x          look
arm-linux-gnueabihf-addr2line  lsblk
arm-linux-gnueabihf-ar         lsipc
arm-linux-gnueabihf-as         lslocks
arm-linux-gnueabihf-c++filt    lslogins
arm-linux-gnueabihf-cpp        mesg
arm-linux-gnueabihf-g++        mesg.util-linux
arm-linux-gnueabihf-gcc        mkfs.bfs
arm-linux-gnueabihf-gcc-5.3.1  mkfs.ubifs
arm-linux-gnueabihf-gcc-ar     namei
arm-linux-gnueabihf-gcc-nm     nm6x
arm-linux-gnueabihf-gcc-ranlib nologin
arm-linux-gnueabihf-gcov       nsenter
arm-linux-gnueabihf-gdb        ofd6x
arm-linux-gnueabihf-gprof      omps2s6x
arm-linux-gnueabihf-ld         openssl
arm-linux-gnueabihf-ld.bfd     opkg
arm-linux-gnueabihf-nm         opkg-check-config
arm-linux-gnueabihf-objcopy    opt6x
arm-linux-gnueabihf-objdump    pdd6x
arm-linux-gnueabihf-ranlib     pg
arm-linux-gnueabihf-readelf    pkg-config
arm-linux-gnueabihf-size       pprof6x
arm-linux-gnueabihf-strings    prelink6x
arm-linux-gnueabihf-strip      prlimit
asm6x          pydoc
blkdiscard     qt5
cal            raw
cg6x           rename
cg6x3          renice
chcpu          renice.util-linux
chrt           resizepart
```

<그림1>

```
ls/home/jeon/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-  
devkit/sysroots/x86_64-arago-linux/usr/bin/
```

참고: 제품 버전에 따라 ti-processor-sdk-linux-am57xx-evm-03.02.00.05 부분이 다를수있음

미리 빌드 된 ARM 라이브러리를 찾으려면 다음 명령을 수행하십시오.

```
cd/home/jeon/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-
devkit/sysroots/armv7ahf-neon-linux-gnueabi/usr/lib
로 들어가서 ls libQt *로 정렬해서 Qt관련 파일들이 있는지 확인한다.
```

| | | | | |
|-----------------------------|-------------------------------------|---------------------------------|------------------------------------|---------------------------|
| libQt5Quick.so.5.6.2 | libavahi-glib.so.1 | libgmpxx.so.4.5.0 | libopencv_face.so.3.1 | libstdc++.so.6 |
| libQt5Quick.so.5.6.2 | libavahi-glib.so.1.0.2 | libgnutls.so.3 | libopencv_face.so.3.1.0 | libstdc++.so.6 |
| libQt5QuickInput-prl | libavahi-gobject.so | libgnutls.so.30.6.1 | libopencv_features2d.so | libtag.so |
| libQt5QuickInput.so | libavahi-gobject.so.0 | libgobject-2.0.so | libopencv_features2d.so.3.1 | libtag.so.1 |
| libQt5QuickInput.so.5.6.2 | libavahi-gobject.so.0.0.4 | libgobject-2.0.so.0 | libopencv_features2d.so.3.1.0 | libtag.so.14.0 |
| libQt5QuickInput.so.5.6.2 | libbfd-2.25.2.so | libgobject-2.0.so.0.4600.2 | libopencv_flann.so | libtag_c.so |
| libQt5QuickInput.so.5.6.2 | libbfd.so | libgomp.so | libopencv_flann.so.3.1 | libtag_c.so.0 |
| libQt5QuickRender-prl | libbkltd.so | libgphoto2 | libopencv_flann.so.3.1.0 | libtag_c.so.0.0.0 |
| libQt5QuickRender.so | libbluetooth.so | libgphoto2.so | libopencv_fuzzy.so | libtbb.so |
| libQt5QuickRender.so.5 | libbluetooth.so.3 | libgphoto2.so.6 | libopencv_fuzzy.so.3.1 | libtbb.so.2 |
| libQt5QuickRender.so.5.6 | libbluetooth.so.3.13.0 | libgphoto2.so.6.0.0 | libopencv_fuzzy.so.3.1.0 | libtbbmalloc.so |
| libQt5QuickRender.so.5.6.2 | libboost_atomic-nt.so | libgphoto2_port | libopencv_highgui.so | libtbbmalloc.so.2 |
| libQt5QuickRender-prl | libboost_atomic.so | libgphoto2_port.so | libopencv_highgui.so.3.1 | libtbbmalloc_proxy.so |
| libQt5QuickRender.so.5 | libboost_atomic.so.1.60.0 | libgphoto2_port.so.12 | libopencv_imgcodecs.so | libtbbmalloc_proxy.so.2 |
| libQt5QuickRender.so.5 | libboost_chrono-nt.so | libgphoto2_port.so.12.0.0 | libopencv_imgcodecs.so.3.1 | libtbcap.so |
| libQt5QuickRender.so.5.6.2 | libboost_chrono.so | libgstadaptivedenux-1.0.so | libopencv_imgcodecs.so.3.1.0 | libtheora.so |
| libQt5QuickRender.so.5.6.2 | libboost_chrono.so.1.60.0 | libgstadaptivedenux-1.0.so.0 | libopencv_imgcodecs.so.3.1.0 | libtheora.so.0 |
| libQt5QuickRender-prl | libboost_date_time-nt.so | libgstadaptivedenux-prl | libopencv_improc.so | libtheora.so.0.3.10 |
| libQt5Compositor-prl | libboost_date_time.so | libgstallocators-1.0.so | libopencv_improc.so.3.1 | libtheoradec.so |
| libQt5Compositor.so | libboost_date_time.so.1.60.0 | libgstallocators-1.0.so.0 | libopencv_improc.so.3.1.0 | libtheoradec.so.1 |
| libQt5Compositor.so.5.6 | libboost_filesystem-nt.so | libgstallocators-1.0.so.0.603.0 | libopencv_line_descriptor.so | libtheoradec.so.1.1.4 |
| libQt5Compositor.so.5.6.2 | libboost_filesystem.so | libgstapp-1.0.so | libopencv_line_descriptor.so.3.1 | libtheorenc.so |
| libQt5Compositor.so.5.6.2 | libboost_filesystem.so.1.60.0 | libgstapp-1.0.so.0.603.0 | libopencv_line_descriptor.so.3.1.0 | libtheorenc.so.1 |
| libQt5Concurrent-prl | libboost_graph-nt.so | libgstapp-1.0.so.0.603.0 | libopencv_ml.so | libtheorenc.so.1.1.2 |
| libQt5Concurrent.so | libboost_graph.so | libgstaudio-1.0.so | libopencv_ml.so.3.1 | libthread_db-1.0.so |
| libQt5Concurrent.so.5 | libboost_graph.so.1.60.0 | libgstaudio-1.0.so.0 | libopencv_ml.so.3.1.0 | libthread_db.so |
| libQt5Concurrent.so.5.6.2 | libboost_locale-nt.so | libgstaudio-1.0.so.0.603.0 | libopencv_objdetect.so | libt13dft.so |
| libQt5Concurrent.so.5.6.2 | libboost_locale.so | libgstbtdbase-1.0.so | libopencv_objdetect.so.3.1 | libt13dft.so.0.6.6 |
| libQt5Core-prl | libboost_locale.so.1.60.0 | libgstbtdbase-1.0.so.0 | libopencv_objdetect.so.3.1.0 | libt13dft.so.34 |
| libQt5Core.so | libboost_log-nt.so | libgstbtdbase-1.0.so.0.603.0 | libopencv_optflow.so | libtbc.so |
| libQt5Core.so.5 | libboost_log.so | libgstbtdvideo-1.0.so | libopencv_optflow.so.3.1 | libtbc.so.5 |
| libQt5Core.so.5.6 | libboost_log.so.1.60.0 | libgstbtdvideo-1.0.so.0 | libopencv_optflow.so.3.1.0 | libtbc.so.5.9 |
| libQt5Core.so.5.6.2 | libboost_log_setup.so | libgstbtdvideo-1.0.so.0.603.0 | libopencv_photo.so | libtbcnsm.so |
| libQt5DBus-prl | libboost_log_setup.so.1.60.0 | libgstbthse-1.0.so | libopencv_photo.so.3.1 | libtbcnsm.so.1 |
| libQt5DBus.so | libboost_prg_exec_monitor.so | libgstbthse-1.0.so.0 | libopencv_photo.so.3.1.0 | libtbcnsm.so.1.0.0 |
| libQt5DBus.so.5 | libboost_prg_exec_monitor.so.1.60.0 | libgstbtdecoders-1.0.so | libopencv_plot.so | libtbcw.so |
| libQt5DBus.so.5.6.2 | libboost_program_options-nt.so | libgstbtdecoders-1.0.so.0.603.0 | libopencv_plot.so.3.1 | libtbcw.so.5.9 |
| libQt5DBus.so.5.6.2 | libboost_program_options.so | libgstbtdecoders-1.0.so.0 | libopencv_plot.so.3.1.0 | libtbcw.so.5.9 |
| libQt5Declarative-prl | libboost_program_options.so.1.60.0 | libgstbtdecoders-1.0.so.0.603.0 | libopencv_reg.so | libtbfrr.so |
| libQt5Declarative-prl | libboost_random-nt.so | libgstbtdecoders-1.0.so.0 | libopencv_reg.so.3.1 | libtbfrr.so.5.2.4 |
| libQt5Declarative-prl | libboost_random.so | libgstbtdecoders-1.0.so.0 | libopencv_reg.so.3.1.0 | libtbfrr.so.5.2.4 |
| libQt5Declarative.so.5.6 | libboost_random.so.1.60.0 | libgstbtdecoders-1.0.so.0.603.0 | libopencv_rgbd.so | libtbfrrfx.so |
| libQt5Declarative.so.5.6.2 | libboost_regex-nt.so | libgstcontroller-1.0.so | libopencv_rgbd.so.3.1 | libtbfrrfx.so.5 |
| libQt5Declarative.so.5.6.2 | libboost_regex.so | libgstcontroller-1.0.so.0 | libopencv_rgbd.so.3.1.0 | libtbfrrfx.so.5.2.4 |
| libQt5SegDevIntegration-prl | libboost_regex.so.1.60.0 | libgstcontroller-1.0.so.0.603.0 | libopencv_saliency.so | libtbtipc.so |
| libQt5SegDevIntegration-prl | libboost_serialization-nt.so | libgstdrm-1.0.so | libopencv_saliency.so.3.1 | libtbtipc.so.1 |
| libQt5SegDevIntegration-prl | libboost_serialization.so | libgstdrm-1.0.so.0 | libopencv_saliency.so.3.1.0 | libtbtipc.so.1.0 |
| libQt5SegDevIntegration-prl | libboost_serialization.so.1.60.0 | libgstdrm-1.0.so.0.603.0 | libopencv_shape.so | libtbtipcutils.so |
| libQt5SegDevIntegration-prl | libboost_signals-nt.so | libgstfft-1.0.so | libopencv_shape.so.3.1 | libtbtipcutils.so.1 |
| libQt5SegDevIntegration-prl | libboost_signals.so | libgstfft-1.0.so.0 | libopencv_shape.so.3.1.0 | libtbtipcutils.so.1.0 |
| libQt5SegDevIntegration-prl | libboost_signals.so.1.60.0 | libgstfft-1.0.so.0.603.0 | libopencv_stereo.so | libtbtipcutils_lad.so |
| libQt5SegDevIntegration-prl | libboost_system-nt.so | libgstintserbtin-1.0.so | libopencv_stereo.so.3.1 | libtbtipcutils_lad.so.1 |
| libQt5SegDevIntegration-prl | libboost_system.so | libgstintserbtin-1.0.so.0 | libopencv_stereo.so.3.1.0 | libtbtipcutils_lad.so.1.0 |
| libQt5SegDevIntegration-prl | libboost_system.so.1.60.0 | libgstintserbtin-1.0.so.0.603.0 | libopencv_stitching.so | libtbtinfo.so |
| libQt5SegDevIntegration-prl | libboost_thread-nt.so | libgstnpegs-1.0.so | libopencv_stitching.so.3.1 | |

파일들이 많아서 원문을 보는걸 추천

2.

cd/home/jeon/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-
devkit/sysroots/armv7ahf-neon-linux-gnueabi/usr/include/qt5/
로 들어가서 파일목록들이 궁금하면 ls로 확인하고 안해도된다.

이제 할려는 거에 대한 간단한 설명으로는

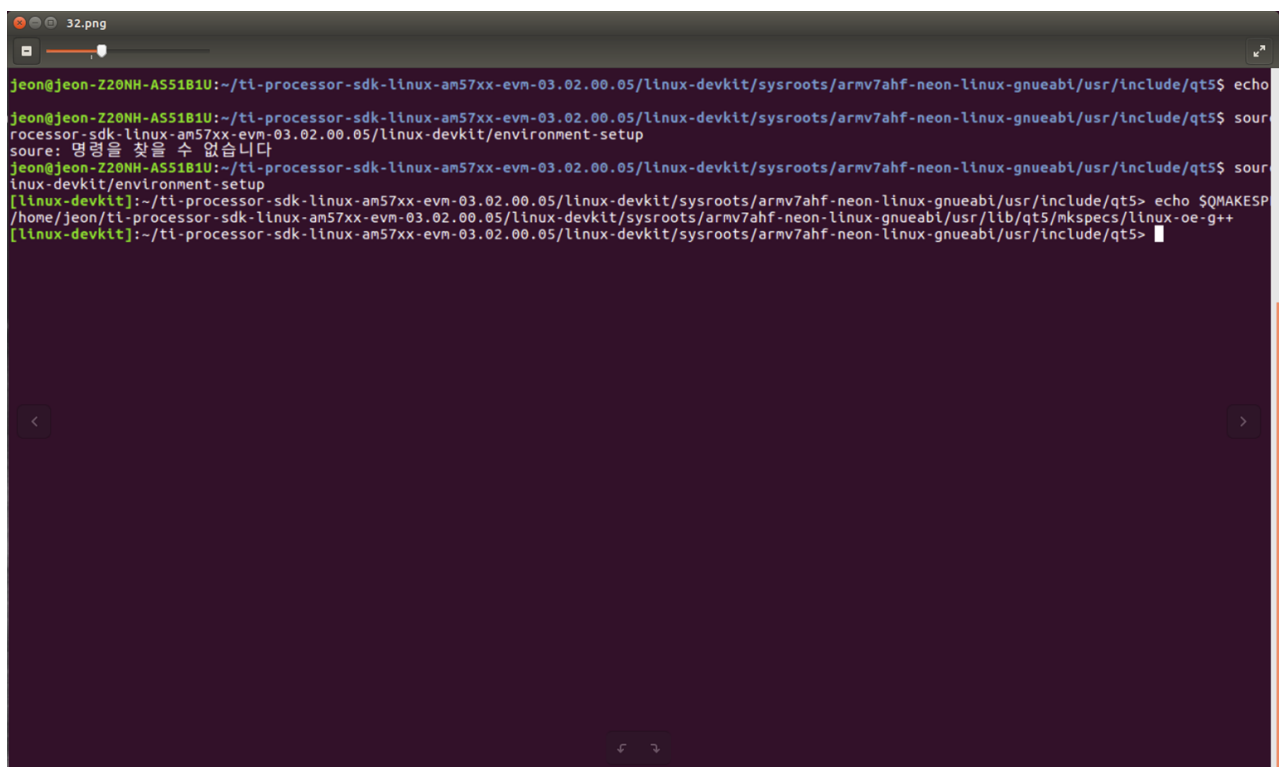
'크로스 컴파일을 쉽게하고 호스트 시스템 라이브러리 대신 적절한 컴파일 된 라이브러리와 링크되도록하기 위해 환경 설정 스크립트가 linux-devkit 디렉토리에 생성되어있다. 이 스크립트는 CC와 같은 많은 표준 변수를 구성하여 교차 컴파일

도구 체인을 사용하고 PATH에 도구 체인을 추가하고 라이브러리 위치에 대한 경로를 구성합니다. 환경 - 설치 스크립트에 의해 제공되는 설정을 사용하려면 필요합니다 소스 스크립트를. 환경 설정 스크립트를 제공하고 QMAKESPEC 변수의 변경 사항을 관찰하려면 다음 명령을 수행하십시오.'

라고 원문에 번역이 되어있다. 이제 시작해보자.

- `echo $ QMAKESPEC`
- `source / home / sitara / AM335x / ti-processor-sdk-linux- <machine> - <sdk version> / linux-devkit / environment-setup`
- `echo $ QMAKESPEC`

순차적으로 입력



```
jeon@jeon-Z20NH-A551B1U:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/armv7ahf-neon-linux-gnueabi/usr/include/qt5$ echo
jeon@jeon-Z20NH-A551B1U:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/armv7ahf-neon-linux-gnueabi/usr/include/qt5$ source
processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/environment-setup
source: 명령을 찾을 수 없습니다
jeon@jeon-Z20NH-A551B1U:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/armv7ahf-neon-linux-gnueabi/usr/include/qt5$ source
linux-devkit/environment-setup
[linux-devkit]:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/armv7ahf-neon-linux-gnueabi/usr/include/qt5> echo $QMAKESPEC
/home/jeon/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/armv7ahf-neon-linux-gnueabi/usr/lib/qt5/mkspecs/linux-oe-g++
[linux-devkit]:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/armv7ahf-neon-linux-gnueabi/usr/include/qt5>
```

source 를 하고 echo를 했을때 `/home/jeon/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/armv7ahf-neon-linux-gnueabi/usr/lib/qt5/mkspecs/linux-oe-g++` 비슷하게 뜨면 성공!

이제 helloworld를 만들러 가보자

`[linux-devkit]:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/example-applications/helloworld>`

example-applications 폴더에서 helloworld 폴더를 만들고 들어가자
(만드는건 mkdir로 만들면된다.)

gedit helloworld.cpp 입력하고 뜨는 메모장 에

```
#include <QApplication>
#include <QLabel>
int main(int argc, char **argv)
{
    QApplication app(argc,argv);

    QLabel label("Hello World");

    label.show();

    return app.exec();
}
```

를 입력

그다음 **gedit helloworld.pro** 입력하고 파일에

```
QT += core gui widgets
SOURCES += helloworld.cpp
```

입력하고 저장

다음

qmake helloworld.pro

해서 Makefile를 만들고

make입력

여기서 주의 해야하는게

make 하면 많은 라인들이 나오는데 cross-compiler-arm-linux-gnueabi-g++ 문장이 출력되는지 확인

```
sitara@ubuntu: ~/ti-sdk-am335x-evm-06.00.00.00/linux-devkit/sysroots/armv7ahf-vfp-ne
[linux-devkit]:~/ti-sdk-am335x-evm-06.00.00.00/example-applications/helloworld>
make
arm-linux-gnueabi-g++ -c -pipe -pipe -pipe -march=armv7-a -marm -mthumb-in
terwork -mfloat-abi=hard -mfp=neon -mtune=cortex-a8 --sysroot=/home/sitara/ti-s
dk-am335x-evm-06.00.00.00/linux-devkit/sysroots/armv7ahf-vfp-neon-3.2-oe-linux-g
nueabi -march=armv7-a -marm -mthumb-interwork -mfloat-abi=hard -mfp=neon -m
tune=cortex-a8 --sysroot=/home/sitara/ti-sdk-am335x-evm-06.00.00.00/linux-devkit
/sysroots/armv7ahf-vfp-neon-3.2-oe-linux-gnueabi -O2 -O2 -Wall -W -Wall -W -Wall
-W -D_REENTRANT -DQT_QWS_CLIENTBLIT -DQT_NO_DEBUG -DQT_GUI_LIB -DQT_CORE_LIB -D
QT_SHARED -I../linux-devkit/sysroots/armv7ahf-vfp-neon-3.2-oe-linux-gnueabi/u
sr/share/qt5/mkspecs/linux-g++ -I. -I/home/sitara/ti-sdk-am335x-evm-06.00.00.
00/linux-devkit/sysroots/armv7ahf-vfp-neon-3.2-oe-linux-gnueabi/usr/include/Qt5
QtCore -I/home/sitara/ti-sdk-am335x-evm-06.00.00.00/linux-devkit/sysroots/arm
v7ahf-vfp-neon-3.2-oe-linux-gnueabi/usr/include/Qt5QtGui -I/home/sitara/ti-s
dk-am335x-evm-06.00.00.00/linux-devkit/sysroots/armv7ahf-vfp-neon-3.2-oe-linux-g
nueabi/usr/include/Qt5 -I. -o helloworld.o helloworld.cpp
arm-linux-gnueabi-g++ --sysroot=/home/sitara/ti-sdk-am335x-evm-06.00.00.00/li
nux-devkit/sysroots/armv7ahf-vfp-neon-3.2-oe-linux-gnueabi -Wl,-rpath-link,/home
/sitara/ti-sdk-am335x-evm-06.00.00.00/linux-devkit/sysroots/armv7ahf-vfp-neon-3.
2-oe-linux-gnueabi/usr/lib -o helloworld helloworld.o -L/home/sitara/ti-sdk-a
m335x-evm-06.00.00.00/linux-devkit/sysroots/armv7ahf-vfp-neon-3.2-oe-linux-gnea
bi/usr/lib -lQtGuiE -lEGL -lIMGegl -lsrv_um -lQtNetworkE -lQtCoreE -lpthread
[linux-devkit]:~/ti-sdk-am335x-evm-06.00.00.00/example-applications/helloworld>
```

새로운 터미널에 minicom을 실행하고 제품 ip를 기존에 실행하고 있던 터미널에 **scp helloworld root@xx.xx.xx.xx : / home / root** 명령을 실행 한다. 아래 xx.xx.xx.xx를 제품 IP 주소로 작성.

나는

scp helloworld3 [root@xx.xxx.xx](#):/home/root 하니까

The authenticity of host 'xx.xxx.xx(xx.xxx.xx)' can't be established.

RSA key fingerprint is

SHA256:x0Qw1SFLqvqmeTTn1dTHknWo3PbUGw+hM1eeEhxn+gw.

Are you sure you want to continue connecting (yes/no)? y

Please type 'yes' or 'no': yes

Warning: Permanently added '192.168.0.22' (RSA) to the list of known hosts.

helloworld3 100% 12KB 12.2KB/s 00:00

문장이 출력 되었고 yes를 눌러서 계속 실행 하였다. (만약에 ssh 키가 보안이 안좋다 어쩌구 뜰시에는 sudo scp ~~~~~ 를 해서 하자 아니면 ssh.host인가? 를 삭제해 하자

무사히 마치고 minicom에 ls를 치면 내 파일이 올라간걸 확인 할 수 있다.

'./올린 파일이름' 을 실행하면 제품에 실행 되는걸 확인 할 수 있다.

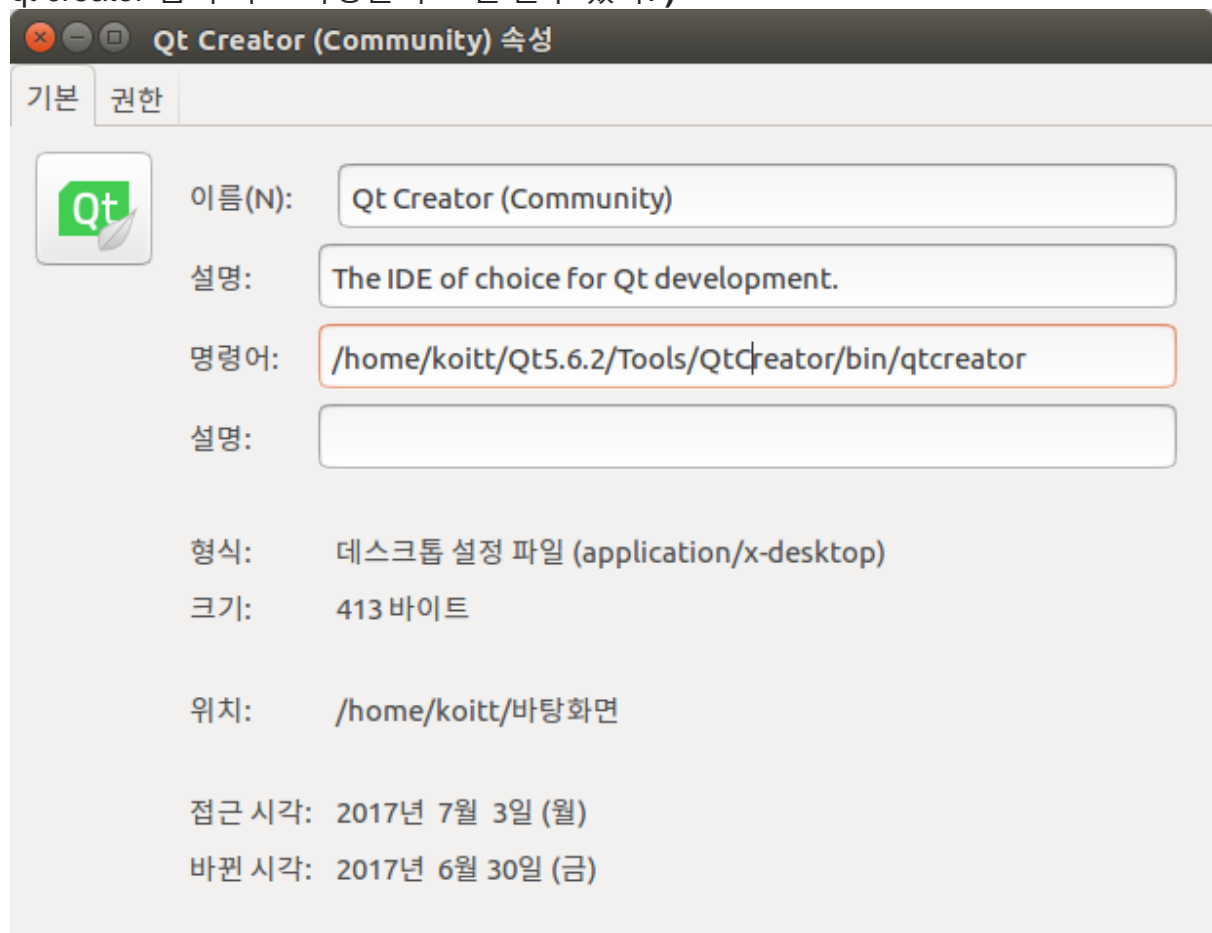
QT creator

참고사항 : 확실한건 아니지만 qt 버전이랑 "/home/koitt/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/qt5/" qt5폴더에 있는 qmake 버전이랑 같아야 한다고 한다. 만약 컴파일이 안될시에는 qmake 버전이랑 qt 파일버전이랑 같게 해보고 하자

cmd에서 source 를 통해 qt 우리가 원하는 설정을 할 수 있다. 꼭 source를 통해 실행 시킬 것

먼저

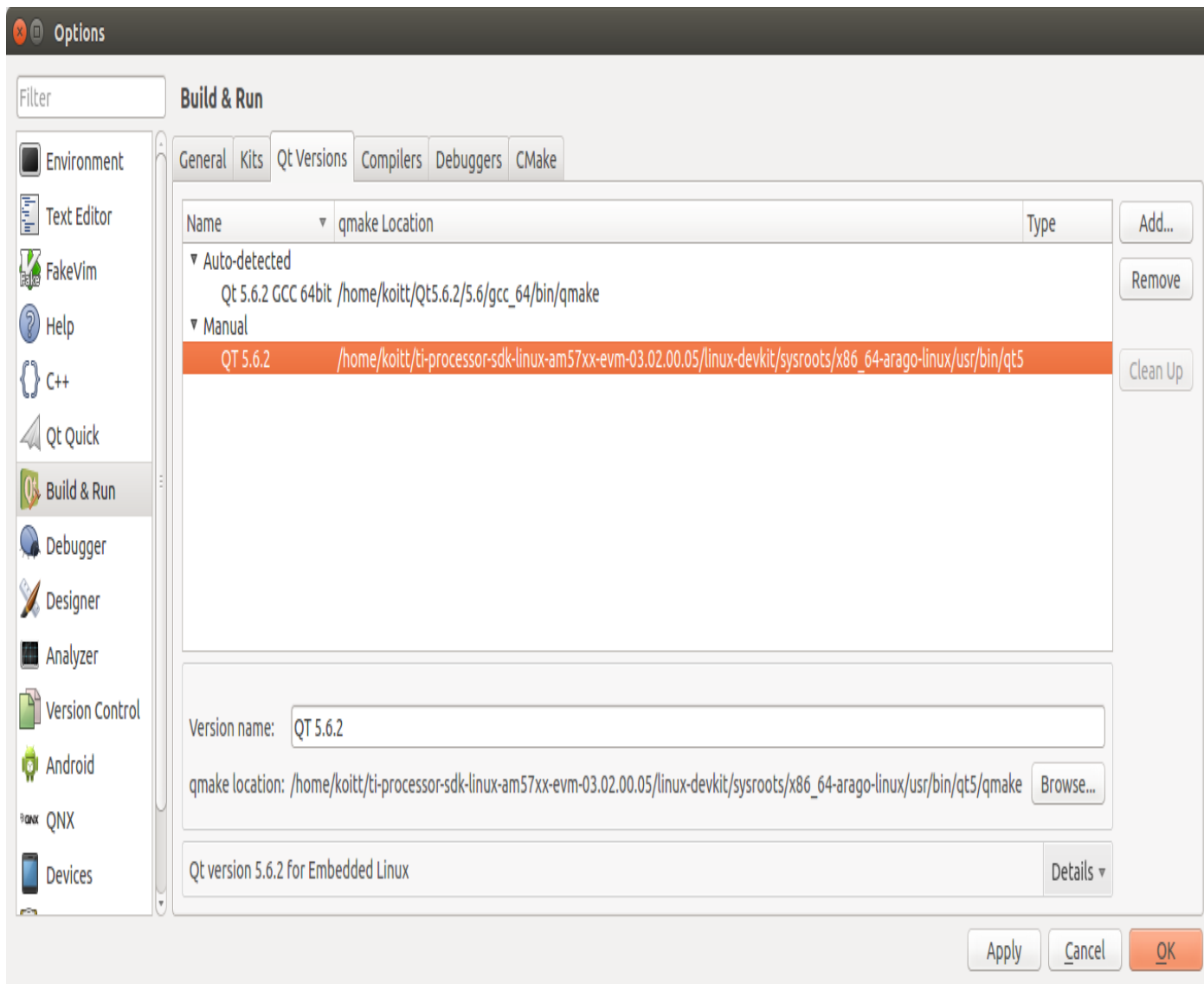
source / home / 사용자 / AM335x / ti-processor-sdk-linux- <machine> - <sdk version> / linux-devkit / environment-setup 를 치고 qt 를 실행해야한다. (홈에서 qt creator 검색 하고 속성을 누르면 볼수 있다.)



명령어를 복사해 cmd 에 치면 qt creator 가 실행 된다.

qt 에서 tool -> options 에 들어가서 qt versions 를 들어간다.

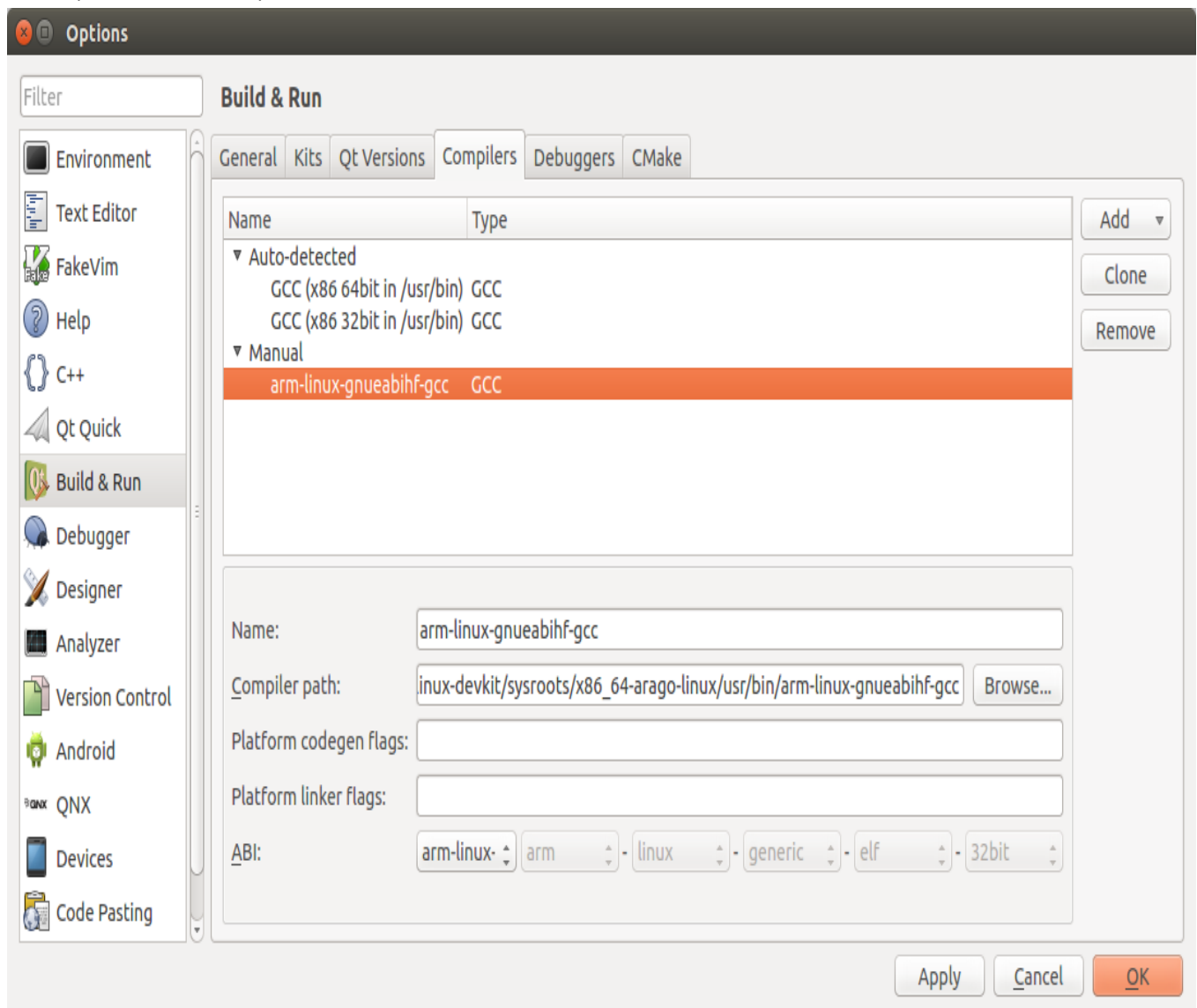
add를 눌러서 ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/qt5/ 를 들어가 qmake 를 클릭 그러면 아래그림과 같이 뜨는데 그림 맨아래 보면 qt 버전을 확인 할 수있다.



<qt versions>

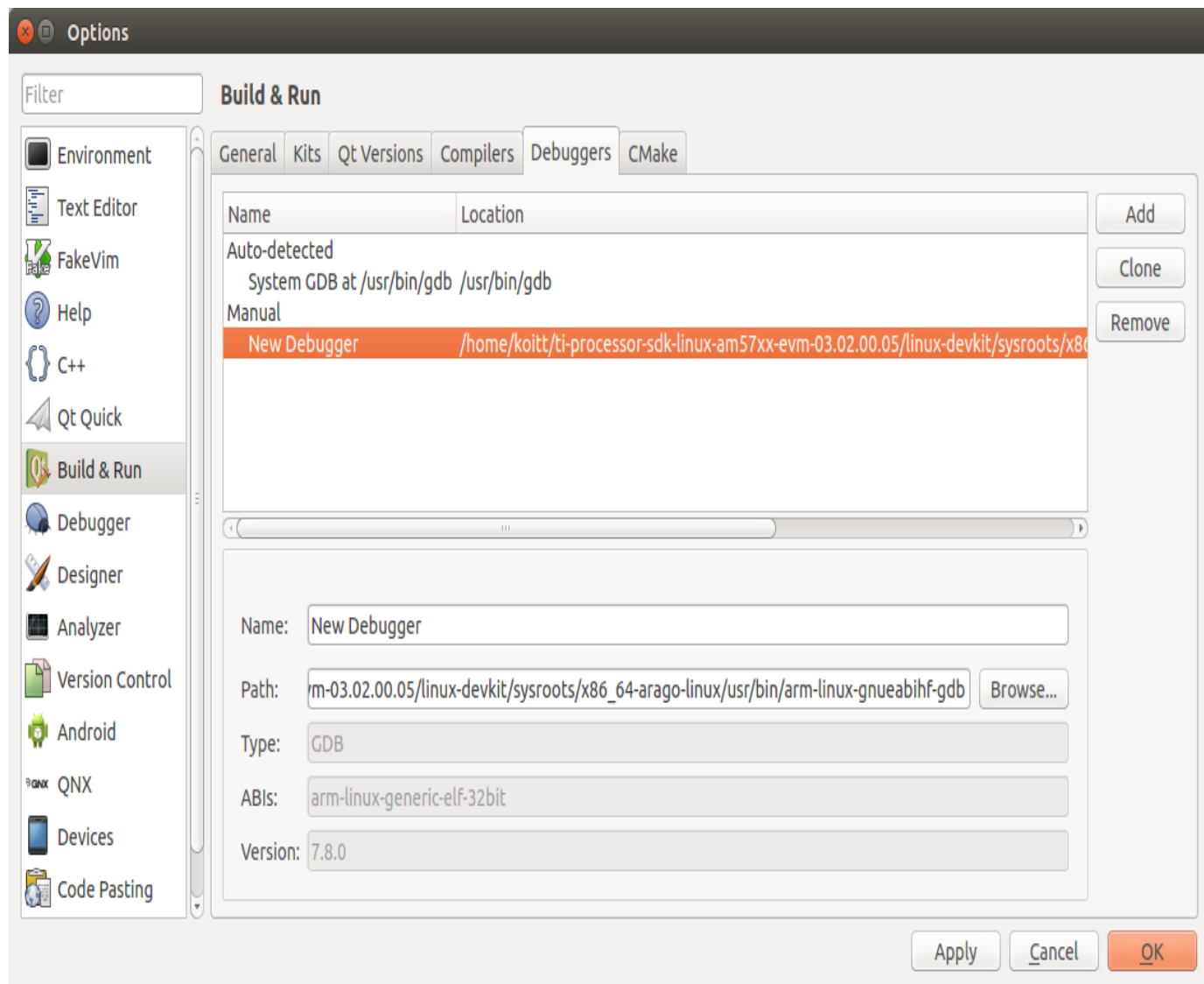
compilers로 가서 add-> gcc를 클릭 그다음 browse 클릭 /home/koitt/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/ 에서 arm-linux-gnueabihf-gcc 를 클릭 이름은 안헛갈리게 arm-linux-gnueabihf-gcc 로

하자 (아래그림 참조)



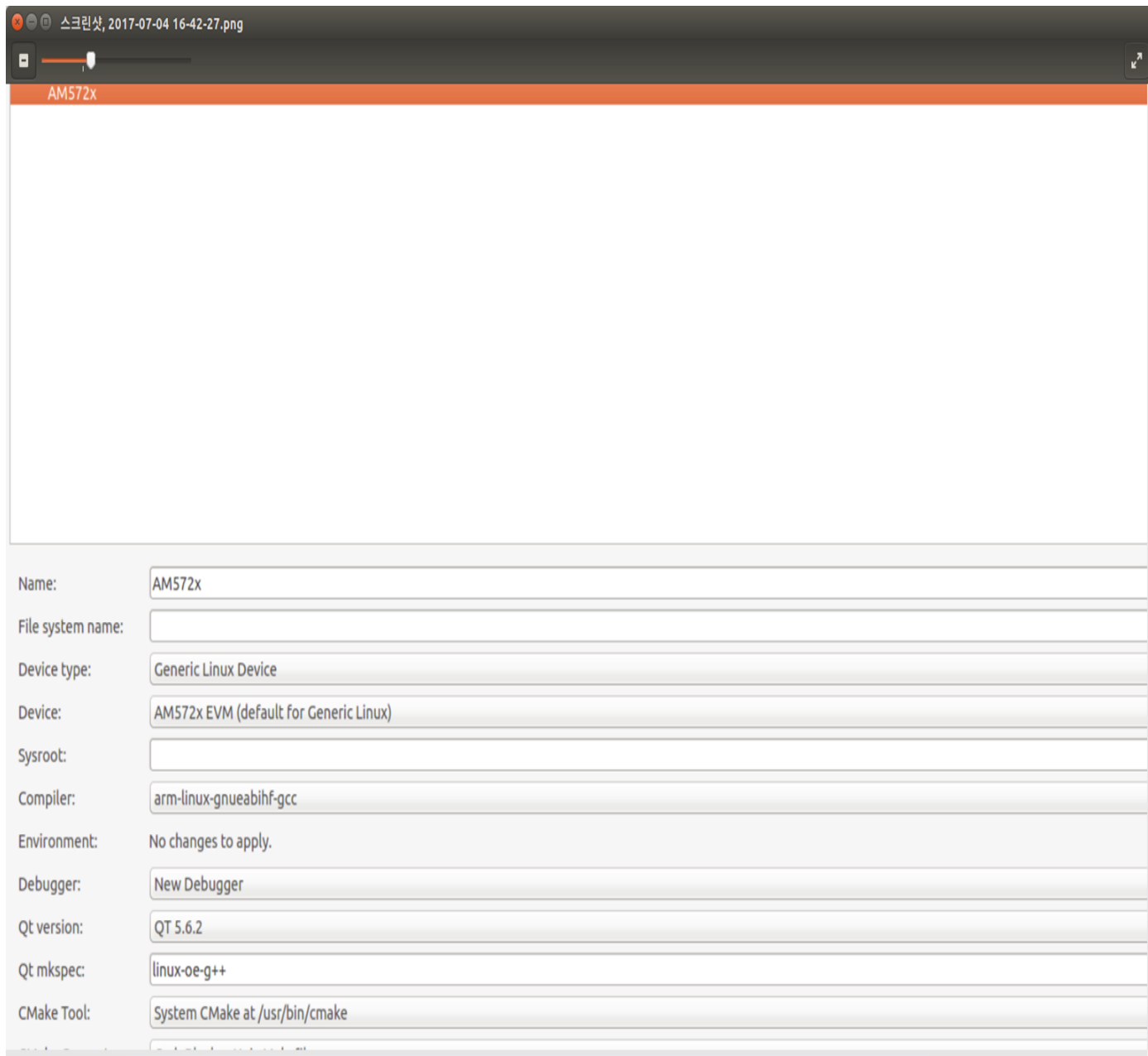
<comilers>

Debuggers 에 들어가 add 클릭 후 browser를 눌러 /home/koitt/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/ 에 있는 **arm-linux-gnueabi-gdb** 선택 이름 은 GDB Engine 로 하자 이름은 마음대로 해도 상관없다.



<Debuggers>

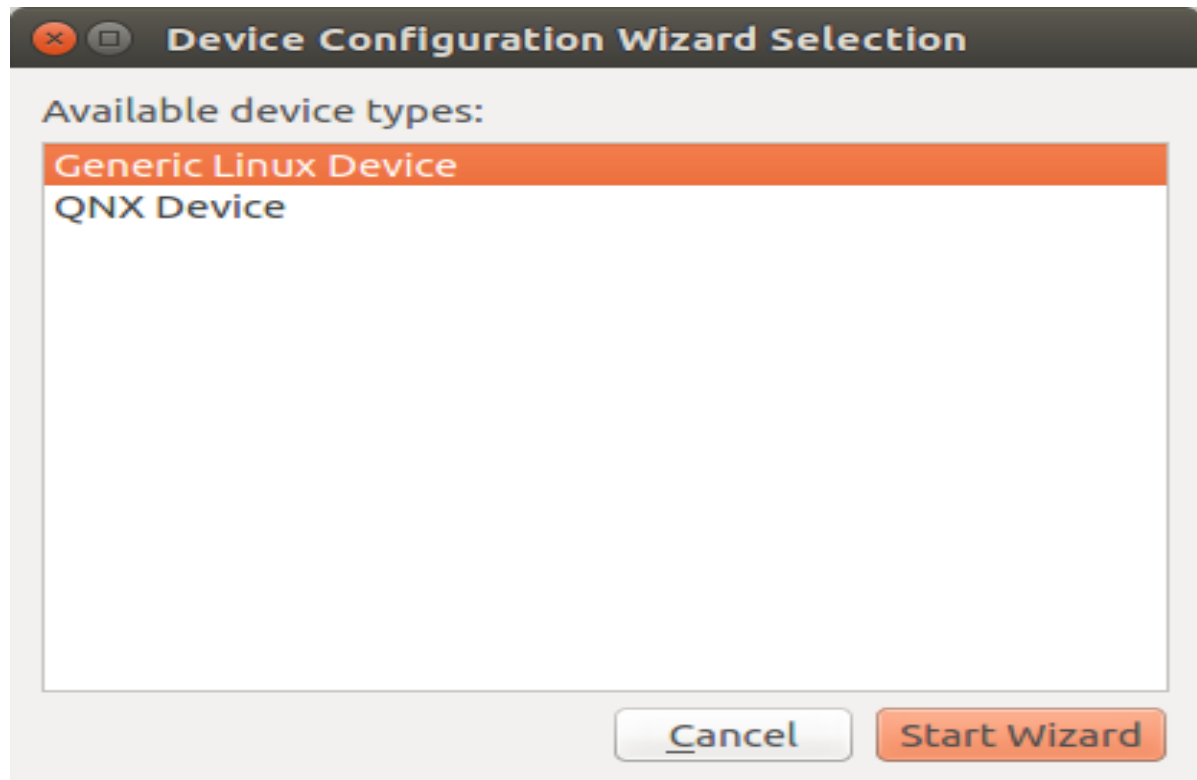
다음 Kits 로 간다. 그리고 add 를 해서 name설정(나는 제품 이름으로 설정 am572x) device type을 **Generic Linux Device** 로 설정 compiler는 **arm-linux-gnueabi-gcc**(우리가 설정했던 이름) **Debugger** 는 **GDB Engine**.(우리가 설정했던이름) QT VERSION 은 QT 5.6.2(QMAKE에서 설정했던이름) 으로 설정하고 APPLY를 한다.



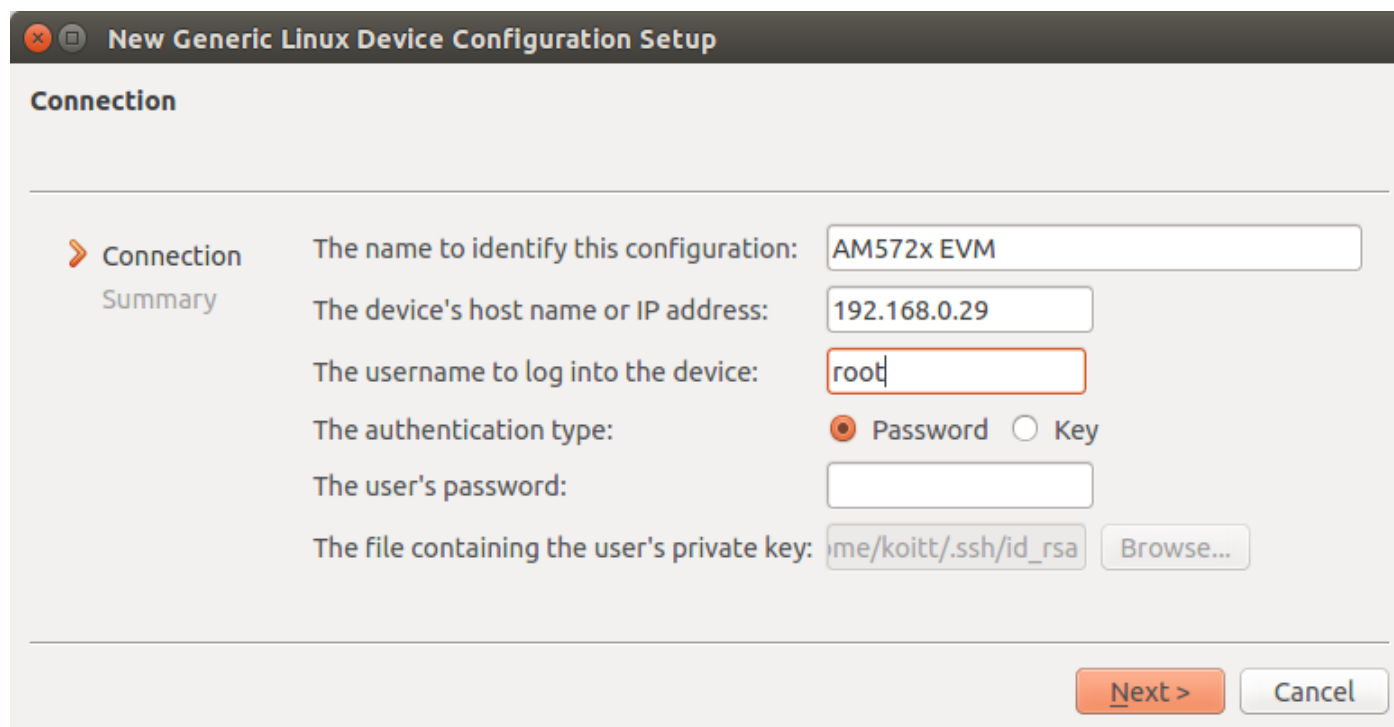
<Klts>

주의 사항 : 설정을 완료하면 am572x 옆에 빨강색 이 떠있을 거다. 신경쓰지말고 다음 설정을 해주면 빨강색이 사라질거다.

왼쪽 탭에 devices 를 눌러서 우측에 있는 add를 클릭한다. 그러면 아래 사진처럼 뜰거고

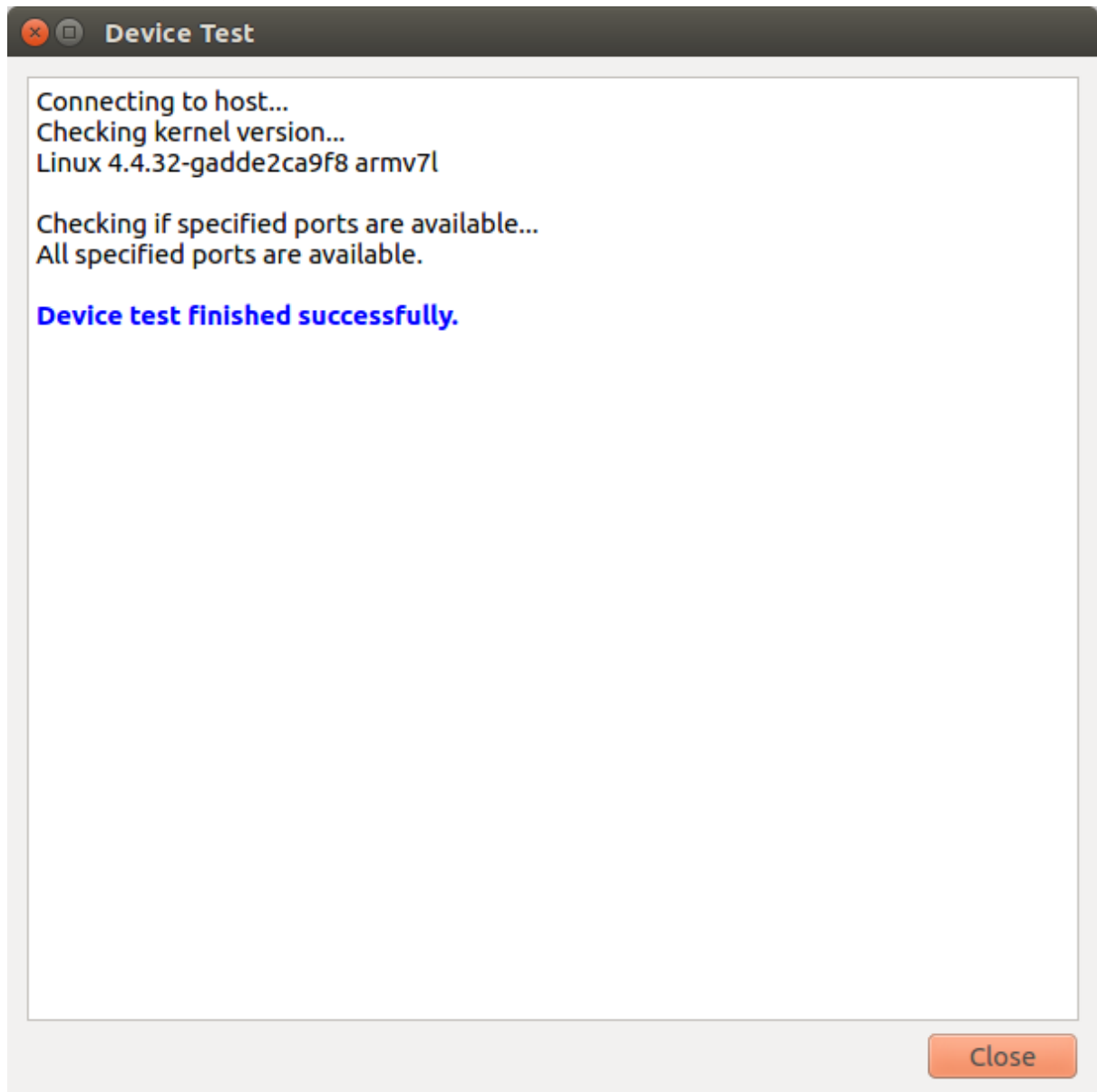


generic linux device를 누르고 start wizard를 누른다.

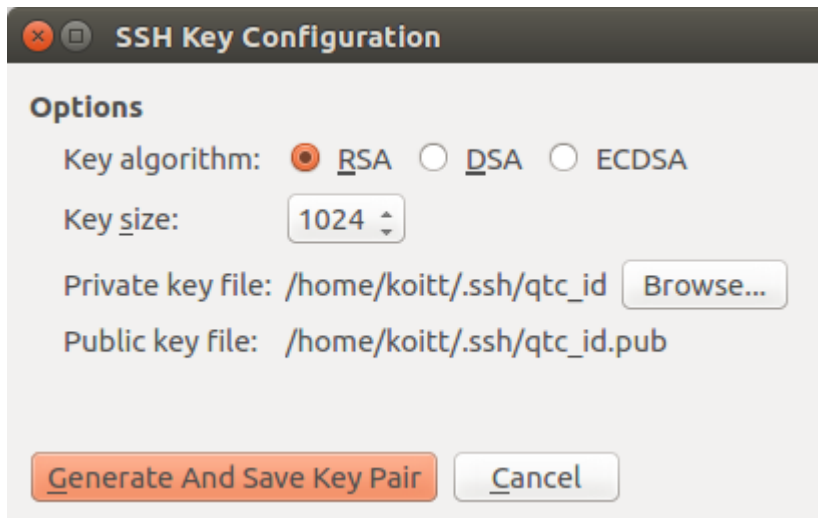


그럼 위 사진처럼 뜰거고(처음엔 name 쪽말고 아무것도 안적혀 있을거다.) name이랑 제품 ip주소(제품에서 settings 를 들어가 network settings 를 들어가면 확인 할 수 있다. 아니면

미니컴에서 ifconfig 를 쳐보자) log into the device 는 root로 하고 password는 적지말자 그리고 next summary 가 뜨는데 그냥 finish 결과는 아래 사진처럼 뜨면 완료



닫고 devices 창에서 아래 보면 private key file 옆에 create new... 를 클릭

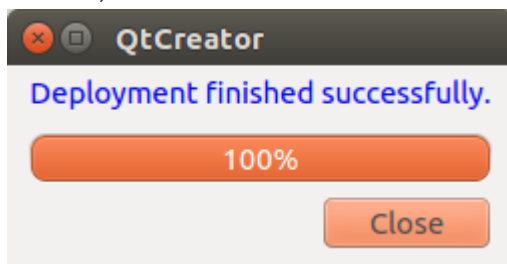


generate and save key pair를 누르자 .(누르면 아래사진처럼 나올거다.)



Do NOT Encrypt key file 클릭

마지막으로 우측에 deploy public key... 를 클릭해 qtc_id.pub를 선택하자. (아래와 같이 나오면 끝난다.)

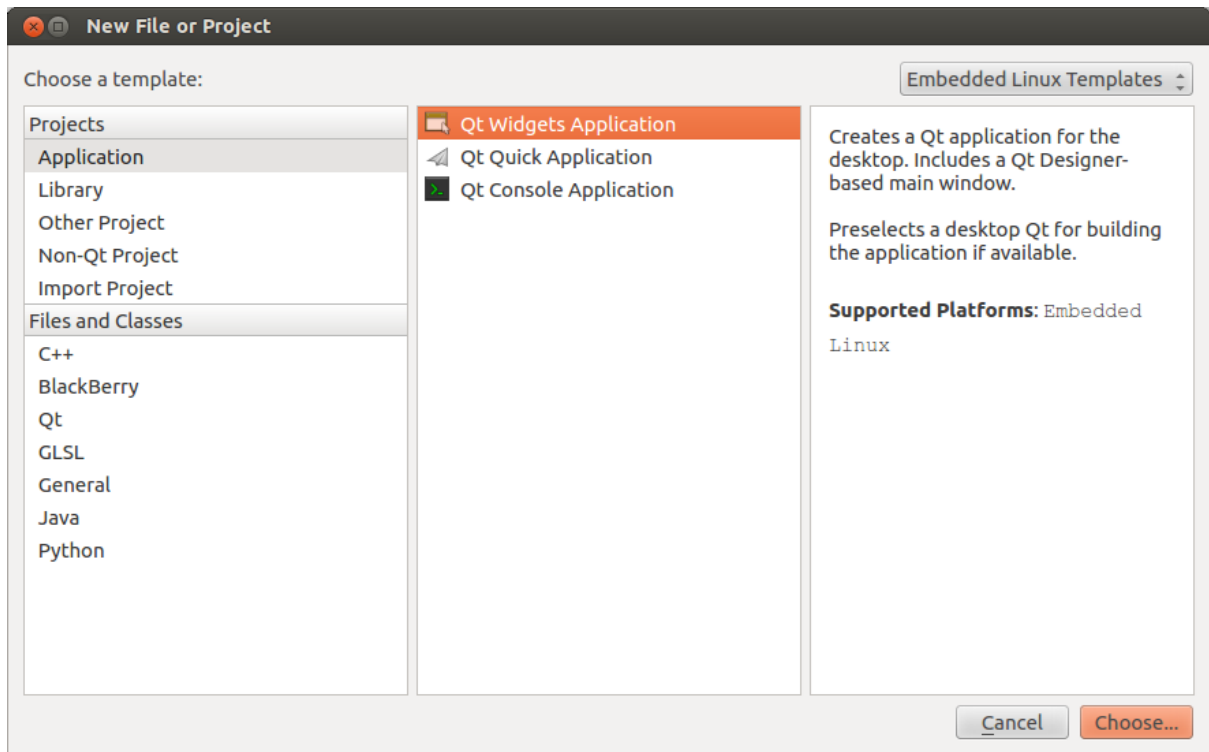


ok를 눌러서 options 를 끝내자

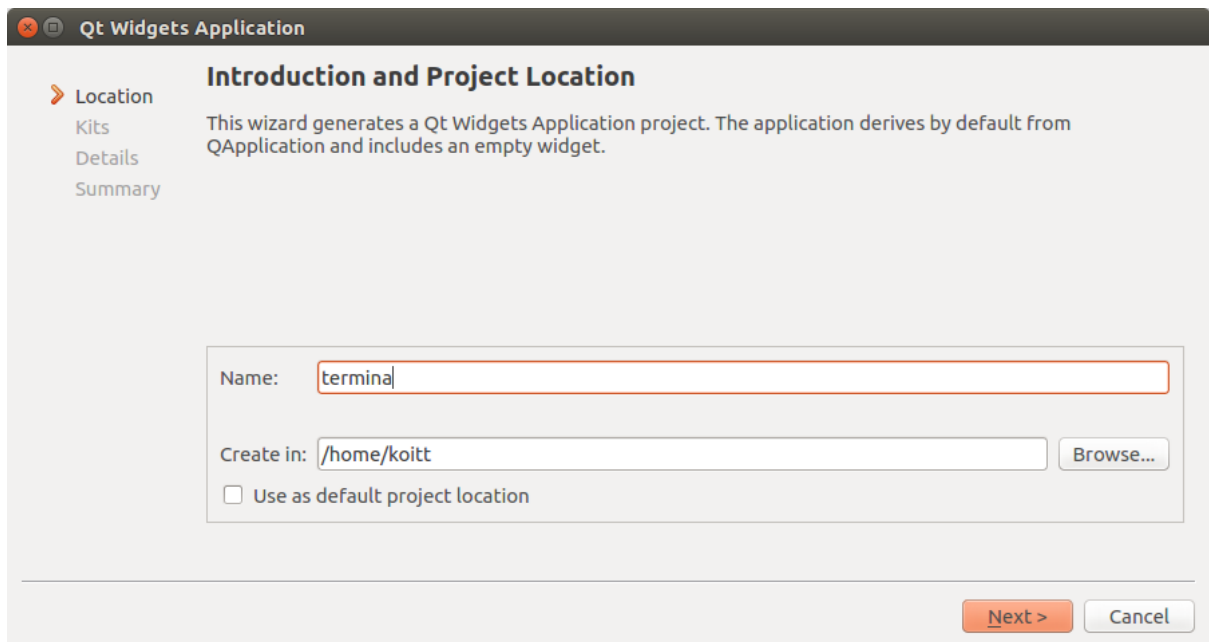
프로젝트를 만들어 보자.

상단 메뉴 File -> New File or Project클릭

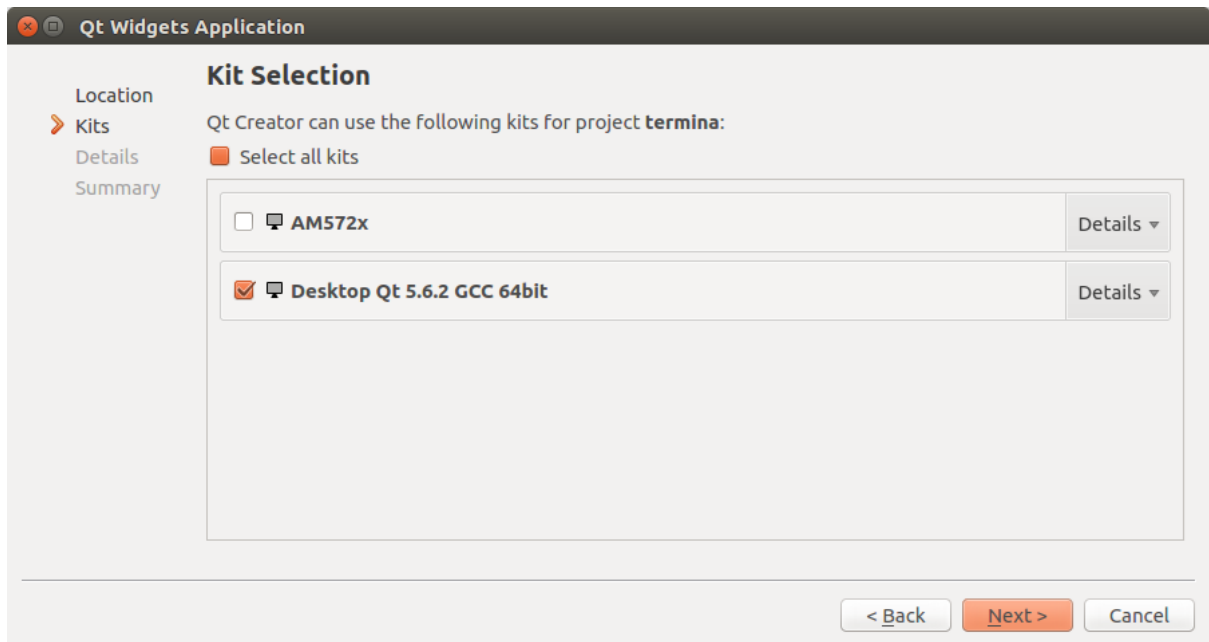
Applications 에서 QT Widgets Applicaton 선택



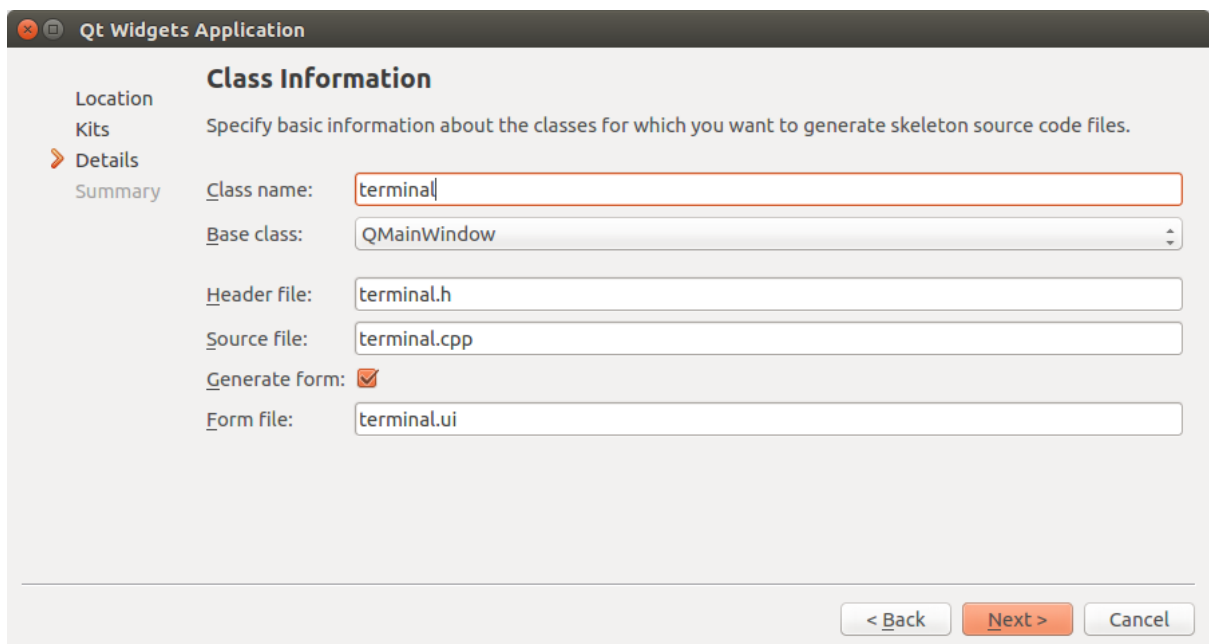
이름을 설정해주자 그다음 next

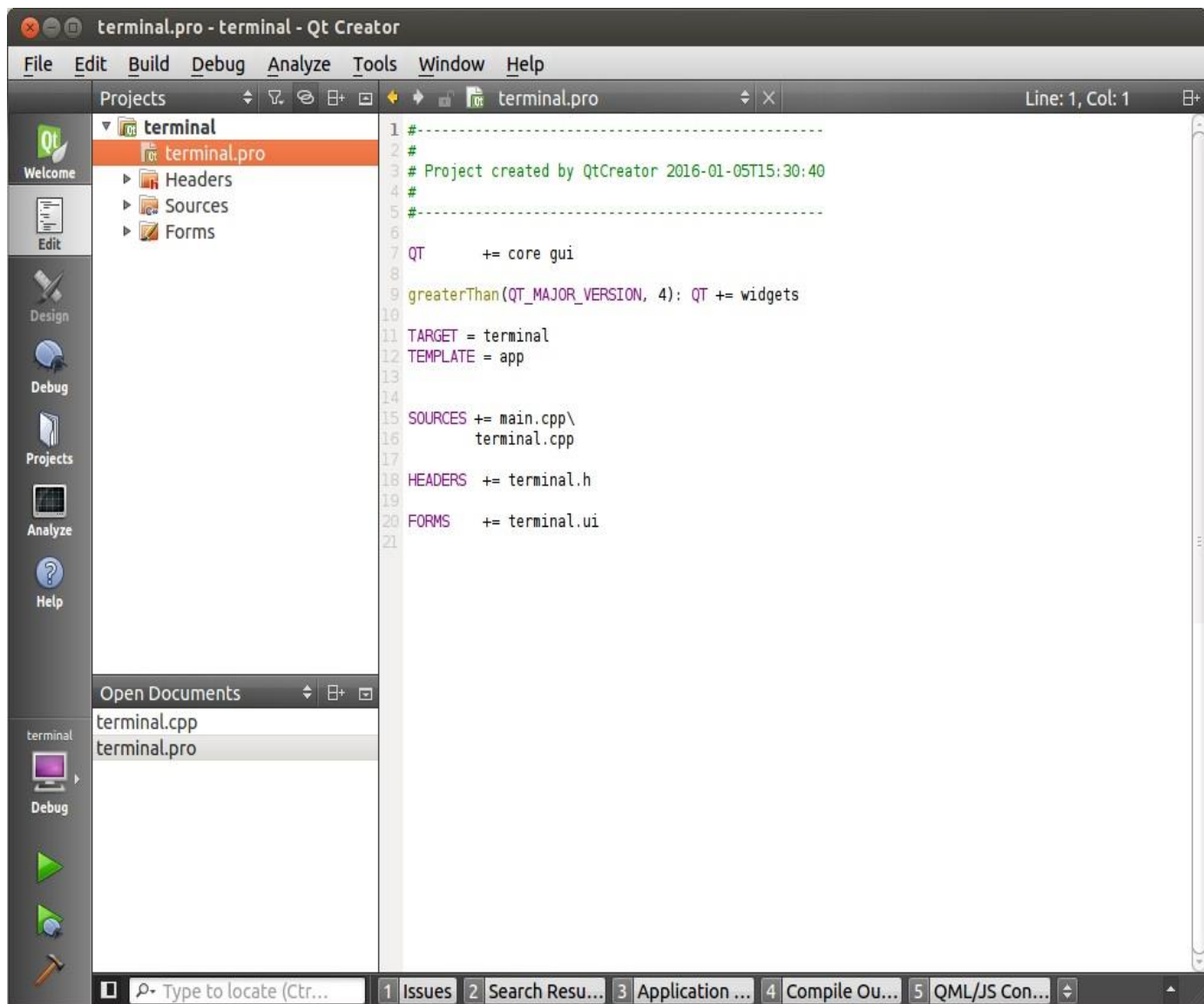


kit Selection 이 나올거고 우리는 제품으로 화면을 출력할거기 때문에 am572x 를 선택하자 (둘다 선택해도 상관없다. 아래desktop은 컴퓨터에 출력시켜주는 거다.)

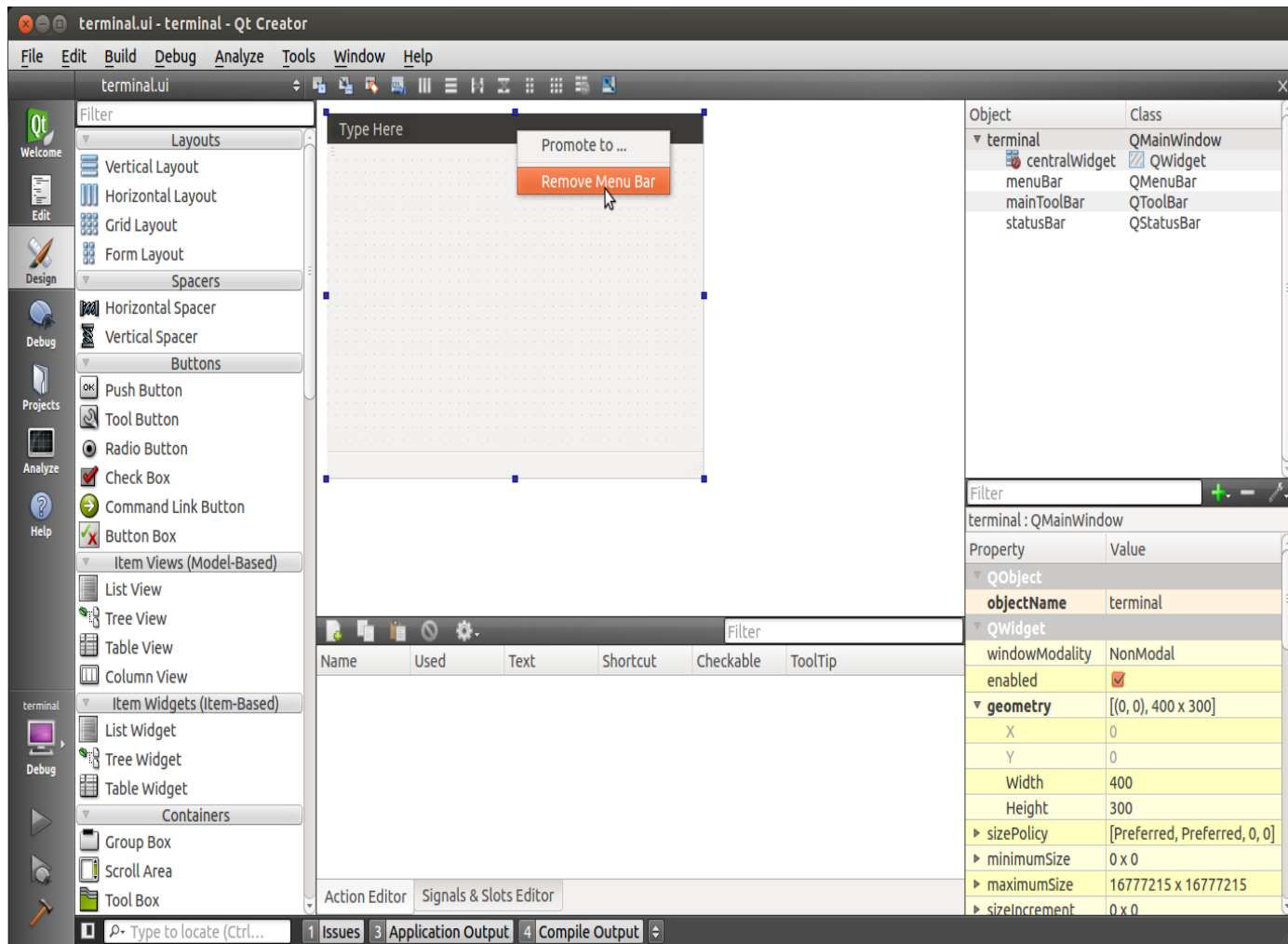


이름 설정을 해주고 next 다음 finish를 하면 프로젝트가 생성 된다.

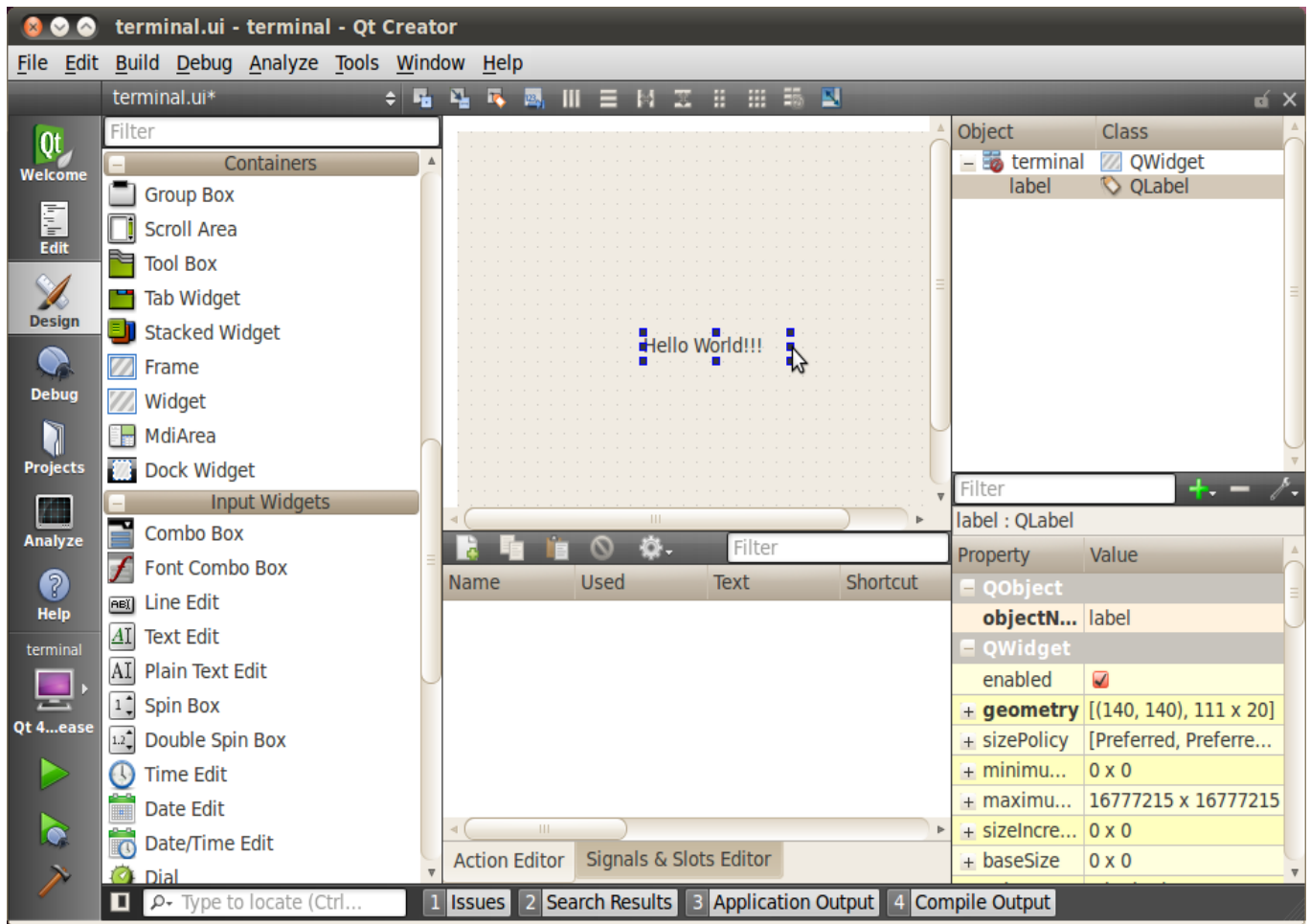




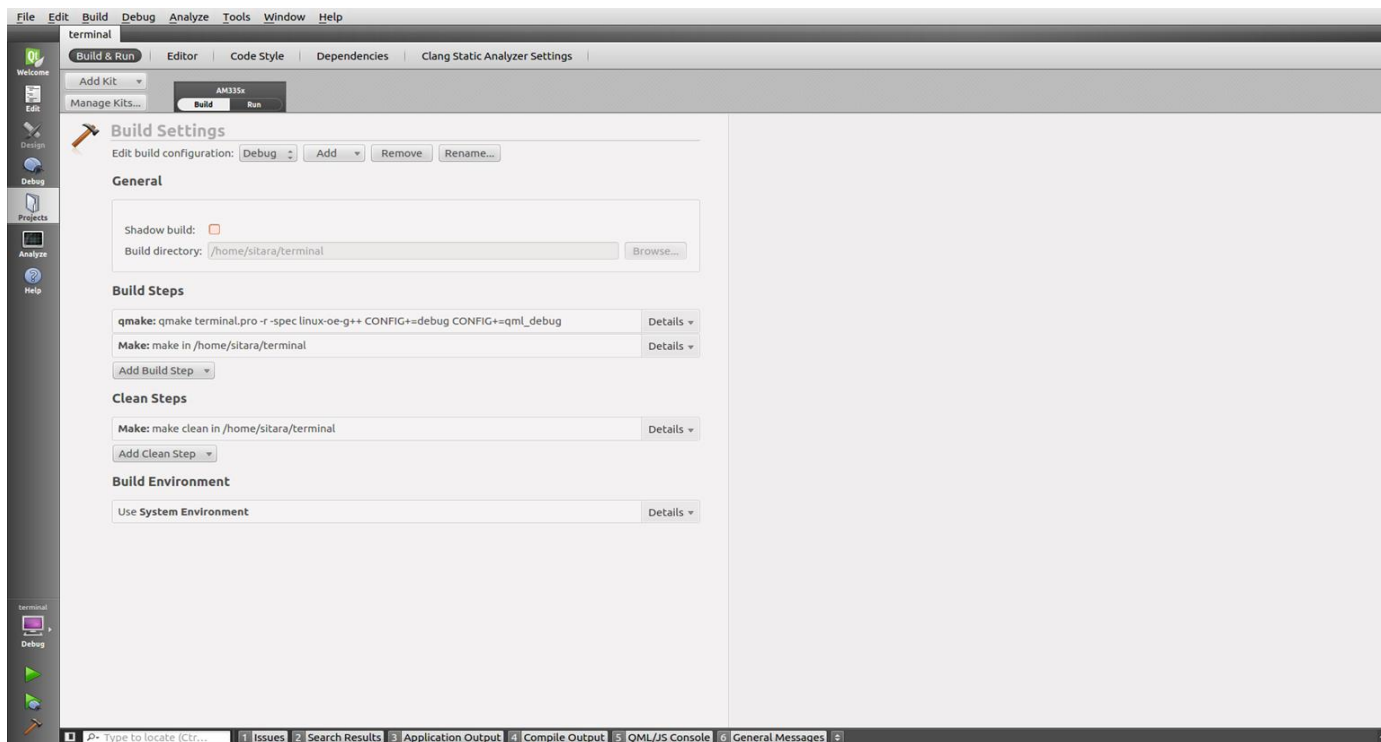
좌측에 froms 를 들어가서



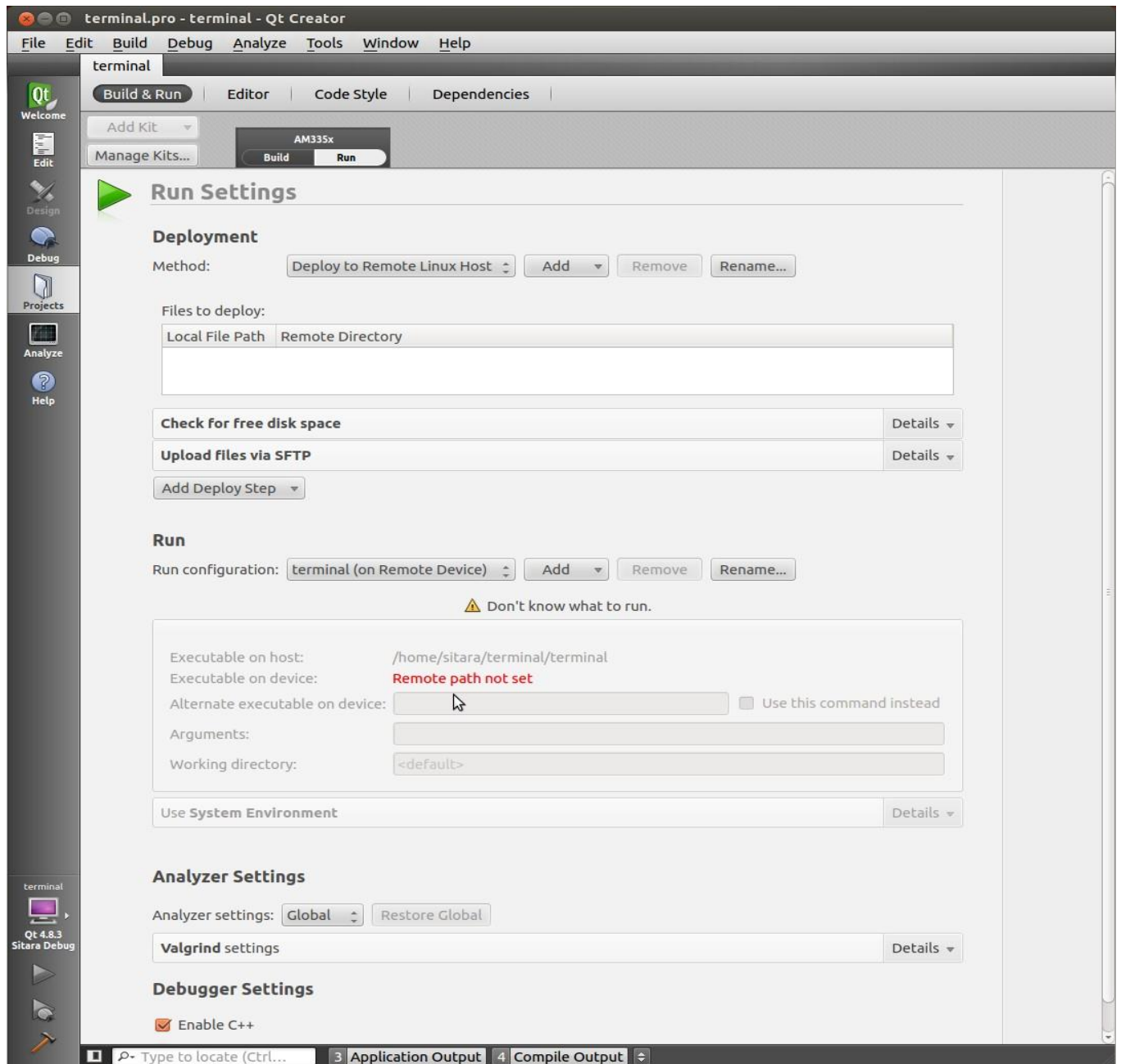
라벨을 끌고와서 적고싶은거 적자



Projects를 들어가서



bulid에 가면 shadow build 가 체크 되어있을텐데 해제해주자. 다음 run을 클릭

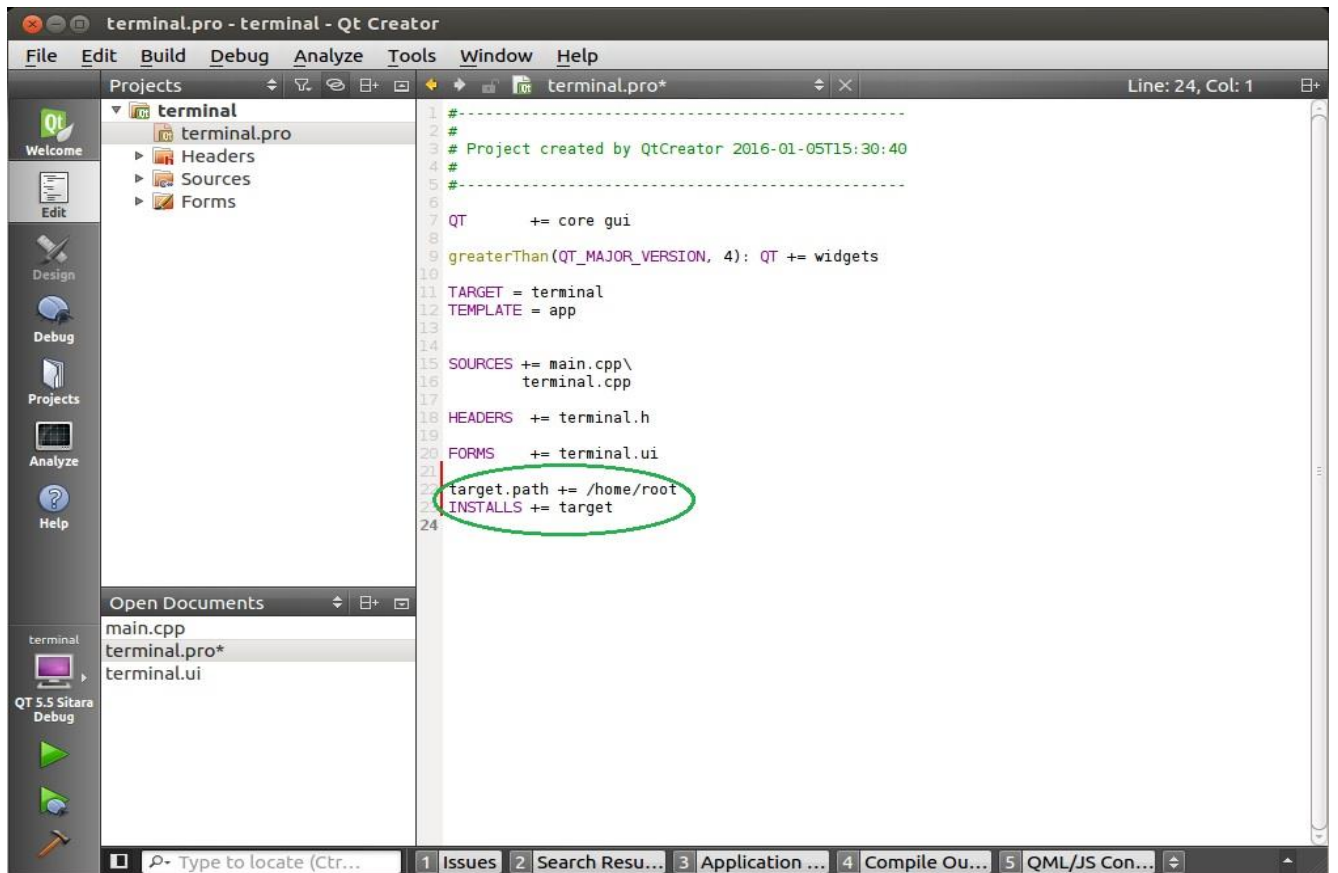


exectable on device 에 <Remote path not set> 라고 떠있을거다. 이걸 해결 하기 위해서는

terminal.pro 에 가서

```
target.path += /home/root
INSTALLS += target
```

이 줄을 쳐주자



다시 run 에 가보면 빨강글이 사라져있는걸 확인 할 수 있다. 그리고 초록색 재생버튼을 누르자.

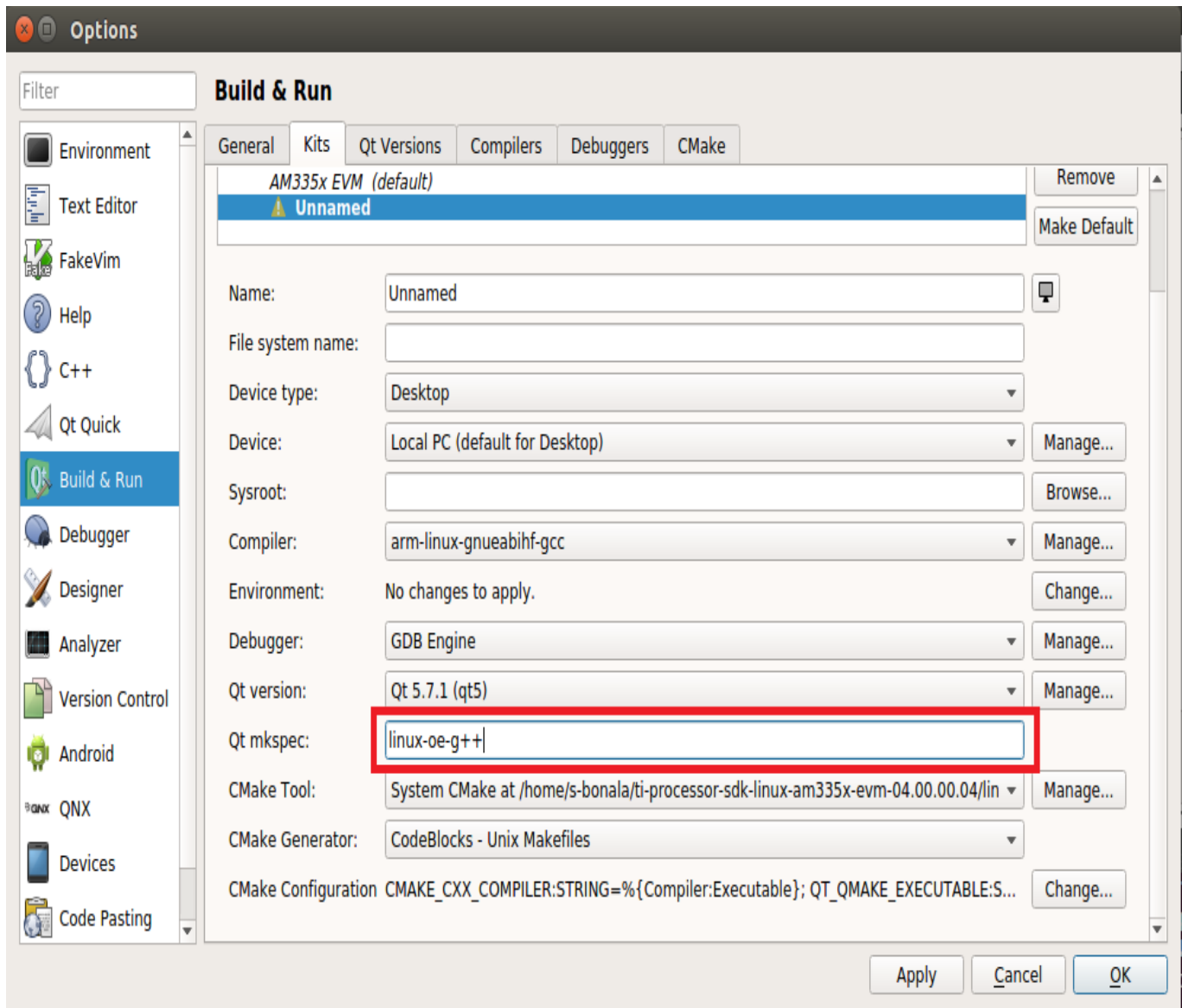
그러면 실행이 된다고 될텐데 실행된 사람들은 다행이고 아니면 에러가 뜰거다.

에러에 대해서 해결을 해보자.

/home/koitt/tisdk3.02.00.05/linux-devkit/sysroots/armv7ahf-neon-linux-gnueabi/usr/include/gnu/stubs.h:7: error: gnu/stubs-soft.h: No such file or directory
include <gnu/stubs-soft.h> 에러가 뜰시에는
cmd 에서

touch /home/koitt/tisdk3.02.00.05/linux-devkit/sysroots/armv7ahf-neon-linux-gnueabi/usr/include/gnu/stubs-soft.h 명령어로 경로를 정해서 stubs-soft.h를 생성을 해주자.

다시 실행을 하면 엄청나게 에러가 또 많이 뜰 텐데



qt mkspec에다가 linux-oe-g++ 를 넣어주자.