Raspberry Pi with Qt

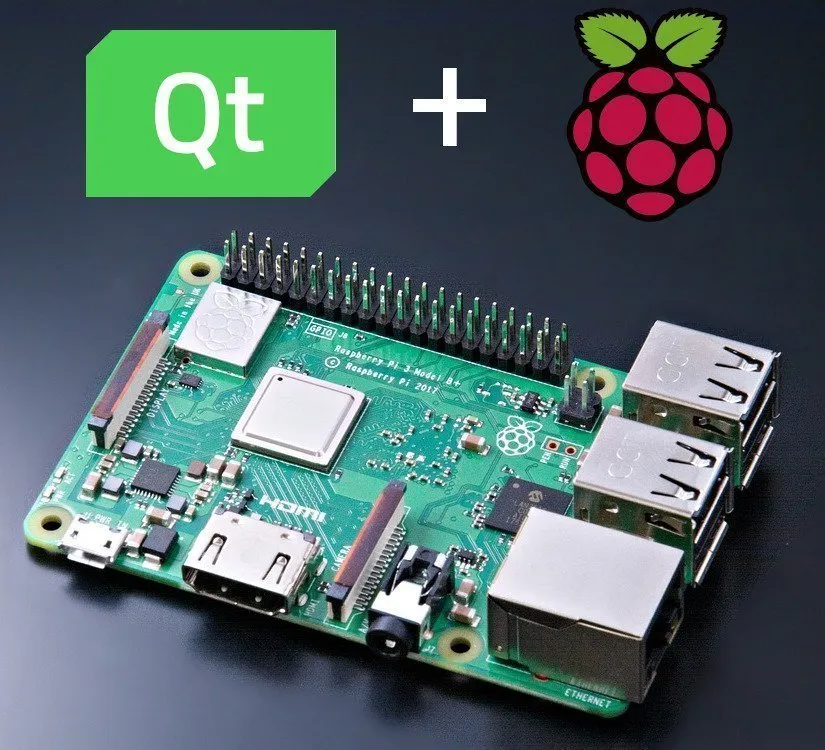


Table of Contents

[Introduction 3](#__RefHeading___Toc193_2907706026)

[What is Cross-Compilation 3](#__RefHeading___Toc195_2907706026)

[What are the prerequisite before follow this documentation 3](#__RefHeading___Toc197_2907706026)

[Steps for configure Raspberry Pi 4](#__RefHeading___Toc199_2907706026)

[Install Following Things 4](#__RefHeading___Toc213_2907706026)

[Make Following Directory 5](#__RefHeading___Toc215_2907706026)

[Steps for configure host machine 5](#__RefHeading___Toc201_2907706026)

[First update and upgrade your host machine 5](#__RefHeading___Toc217_2907706026)

[Test your Raspberry Pi connection 5](#__RefHeading___Toc219_2907706026)

[Go to root 5](#__RefHeading___Toc221_2907706026)

[Install following things 5](#__RefHeading___Toc223_2907706026)

[Install GCC with Ubuntu via Toolchain PPA 5](#__RefHeading___Toc261_2907706026)

[Configure the priority of each version 6](#__RefHeading___Toc263_2907706026)

[Create new Directory and give permission to that 6](#__RefHeading___Toc265_2907706026)

[Get Linaro toolchain 6](#__RefHeading___Toc267_2907706026)

[Add above path to .bashrc 6](#__RefHeading___Toc269_2907706026)

[Download Qt 7](#__RefHeading___Toc271_2907706026)

[Create sysroot directory 7](#__RefHeading___Toc273_2907706026)

[Open terminal in current directory and run follow step one by one 7](#__RefHeading___Toc275_2907706026)

[Get sysroot-relativelinks.py 7](#__RefHeading___Toc277_2907706026)

[Check again sync all file with raspberry pi 7](#__RefHeading___Toc279_2907706026)

[Create qt5build directory 8](#__RefHeading___Toc281_2907706026)

[Compile All things 8](#__RefHeading___Toc283_2907706026)

[Make it 8](#__RefHeading___Toc285_2907706026)

[Install Qt Creator 8](#__RefHeading___Toc203_2907706026)

[Configure Qt Creator 8](#__RefHeading___Toc205_2907706026)

[Create New Kit 8](#__RefHeading___Toc287_2907706026)

[Add Raspberry Pi Device 11](#__RefHeading___Toc207_2907706026)

[Finish 11](#__RefHeading___Toc209_2907706026)

[Reference 11](#__RefHeading___Toc211_2907706026)

Raspberry-PI with Qt 5.14.2

# Introduction

Creating applications with Qt for Raspberry Pi is a rewarding and fun experience that allows you to unleash your creativity and innovation. Qt is a cross-platform framework that enables you to develop graphical user interfaces (GUIs) and applications that run on various hardware and operating systems. Raspberry Pi is a low-cost, credit-card-sized computer that can run Linux and other operating systems, and can be used for a variety of projects, such as robotics, gaming, smart home, and education. In this documentation, you will learn how to set up your development environment, how to create a cross compiler for Raspberry Pi, how to design and build GUIs with Qt Creator, and how to deploy and run your applications on Raspberry Pi. By the end of this documentation, you will have the skills and knowledge to create your own amazing applications with Qt for Raspberry Pi.

# What is Cross-Compilation

Cross compilation in Qt for Raspberry Pi is a process of building Qt applications on a host machine (such as a PC) and then transferring them to a target device (such as a Raspberry Pi) for execution. This way, you can take advantage of the faster CPU and RAM of the host machine, and avoid the long compilation time and limited resources of the Raspberry Pi. Cross compilation also allows you to use Qt Creator, a powerful IDE for Qt development, on your host machine, and debug your applications remotely on the Raspberry Pi. To cross-compile Qt for Raspberry Pi, you need to have a cross-compiler toolchain, a sysroot directory that contains the target device's file system, and a CMake toolchain file that specifies the configuration options for Qt.

# What are the prerequisite before follow this documentation

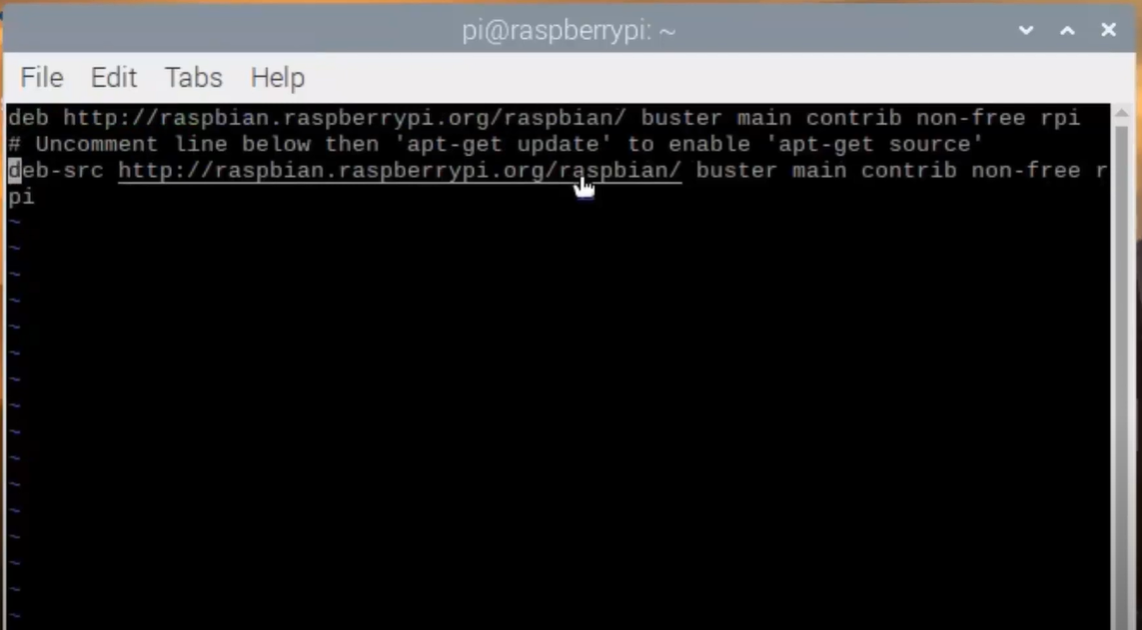
* Raspberry-Pi (This documentation is tested for Raspberry-Pi 2 model B. You should need to install raspbian OS to raspberry Pi. Next you need to install RealVNC server to connect Raspberry Pi to your host computer. Next you need to turn on ssh connection in Raspberry Pi.
* Host Machine (This documentation is tested for Ubuntu version is 22.04 – Ubuntu-22.04-desktop-amd64. After install Ubuntu you need to install RealVNC client to your machine. Turn on ssh connection in your host machine)
* Ethernet cable ( for connect Raspberry Pi and Host machine )
* Power cable ( for give power supple to Raspberry Pi )

*Note :- After you connect your Raspberry Pi to Host machine via Ethernet you have to change your Raspberry Pi Ip Address. Host machine IP and Raspberry Pi IP should be same network*

*Follow this video URL :-* [*Watch*](https://youtu.be/LCJtUbRlIXE)

# Steps for configure Raspberry Pi

sudo vi /etc/apt/sources.list

Figure 1: Uncomment Last Line

sudo apt update

sudo apt full-upgrade

sudo reboot

sudo rpi-update

sudo reboot

## **Install Following Things**

sudo apt-get build-dep qt5-qmake

sudo apt-get build-dep libqt5webengine-data

sudo apt-get install libboost1.58-all-dev libudev-dev libinput-dev libts-dev libmtdev-dev libjpeg-dev libfontconfig1-dev

sudo apt-get install libssl-dev libdbus-1-dev libglib2.0-dev libxkbcommon-dev libegl1-mesa-dev libgbm-dev libgles2-mesa-dev mesa-common-dev

sudo apt-get install libasound2-dev libpulse-dev gstreamer1.0-omx libgstreamer1.0-dev libgstreamer-plugins- base1.0-dev gstreamer1.0-alsa

sudo apt-get install libvpx-dev libsrtp0-dev libsnappy-dev libnss3-dev

sudo apt-get install "^libxcb.\*"

sudo apt-get install flex bison libxslt-dev ruby gperf libbz2-dev libcups2-dev libatkmm-1.6-dev libxi6 libxcomposite1

sudo apt-get install libfreetype6-dev libicu-dev libsqlite3-dev libxslt1- dev libavcodec-dev libavformat-dev libswscale-dev

sudo apt-get install libgstreamer0.10-dev gstreamer-tools libraspberrypi-dev libx11-dev libglib2.0-dev

sudo apt-get install freetds-dev libsqlite0-dev libpq-dev libiodbc2-dev firebird-dev libjpeg9-dev libgst-dev libxext-dev libxcb1 libxcb1-dev libx11-xcb1

sudo apt-get install libx11-xcb-dev libxcb-keysyms1 libxcb-keysyms1-dev libxcb-image0 libxcb-image0-dev libxcb-shm0 libxcb-shm0-dev libxcb-icccm4 libxcb-icccm4-dev

sudo apt-get install libxcb-sync1 libxcb-sync-dev libxcb-render-util0 libxcb-render-util0-dev libxcb-xfixes0- dev libxrender-dev libxcb-shape0-dev libxcb-randr0-dev

sudo apt-get install libxcb-glx0-dev libxi-dev libdrm-dev libssl-dev libxcb-xinerama0 libxcb-xinerama0-dev

sudo apt-get install libatspi-dev libssl-dev libxcursor-dev libxcomposite-dev libxdamage-dev libfontconfig1- dev

sudo apt-get install libxss-dev libxtst-dev libpci-dev libcap-dev libsrtp0-dev libxrandr-dev libnss3-dev libdirectfb-dev libaudio-dev

## **Make Following Directory**

sudo mkdir /usr/local/qt5pi

sudo chown <username>:<username> /usr/local/qt5pi

# Steps for configure host machine

## **First update and upgrade your host machine**

sudo apt-get update

sudo apt-get upgrade

## **Test your Raspberry Pi connection**

ping <ip address of raspberry>

## **Go to root**

sudo bash

## **Install following things**

apt-get install gcc git bison python gperf pkg-config

apt install make

apt install libclang-dev

apt install build-essential

*Note :- build-essential install gcc and g++ latest compilers. But this tutorial work with only gcc – 9 and g++ -9. for that we have to install gcc previous compilers.*

## **Install GCC with Ubuntu via Toolchain PPA**

sudo add-apt-repository ppa:ubuntu-toolchain-r/ppa -y

sudo apt update

sudo apt install gcc-9 g++-9 gcc-10 g++-10 gcc-11 g++-11 g++-12 gcc-12 g++-13 gcc-13

## **Configure the priority of each version**

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-12 40 -- slave /usr/bin/g++ g++ /usr/bin/g++-13 --slave /usr/bin/gcov gcov /usr/bin/gcov-13

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-13 90 -- slave /usr/bin/g++ g++ /usr/bin/g++-12 --slave /usr/bin/gcov gcov /usr/bin/gcov-12

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-11 80 -- slave /usr/bin/g++ g++ /usr/bin/g++-11 --slave /usr/bin/gcov gcov /usr/bin/gcov-11

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-10 60 -- slave /usr/bin/g++ g++ /usr/bin/g++-10 --slave /usr/bin/gcov gcov /usr/bin/gcov-10

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 100 -- slave /usr/bin/g++ g++ /usr/bin/g++-9 -- slave /usr/bin/gcov gcov /usr/bin/gcov-9

## **Create new Directory and give permission to that**

mkdir /opt/qt5pi

chown <username>:<username> /opt/qt5pi

cd /opt/qt5pi/

## **Get Linaro toolchain**

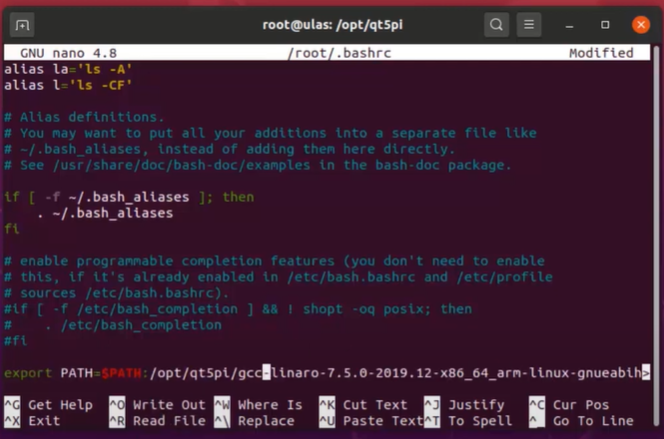
Wget <https://releases.linaro.org/components/toolchain/binaries/latest-7/> [arm-linux-](https://releases.linaro.org/components/toolchain/binaries/latest-7/arm-linux-)gnueabihf/gcc- linaro-7.5.0-2019.12-x86\_64\_arm-linux- gnueabihf.tar.xz

tar xf gcc-linaro-7.5.0-2019.12-x86\_64\_arm-linux- gnueabihf.tar.xz

export PATH=$PATH:/opt/qt5pi/gcc-linaro-7.5.0-2019.12- x86\_64\_arm- linux-gnueabihf/bin

## **Add above path to .bashrc**

nano ~/.bashrc

Figure 2: Add path to .bashrc

## **Download Qt**

wget <https://download.qt.io/official_releases/qt/5.15/5.15.2/single/qt-everywhere-src-5.15.2.tar.xz>

tar xf qt-everywhere-src-5.15.2.tar.xz

## **Create sysroot directory**

mkdir sysroot

chown <username>:<username> sysroot

## **Open terminal in current directory and run follow step one by one**

rsync -avz --rsync-path="sudo rsync" <username>@<rasp-IP>:/usr/include sysroot/usr

rsync -avz --rsync-path="sudo rsync" <username>@<rasp-IP>:/lib sysroot

rsync -avz --rsync-path="sudo rsync" <username>@<rasp-IP>:/usr/lib sysroot/usr

rsync -avz --rsync-path="sudo rsync" <username>@<rasp-IP>:/opt/vc sysroot/opt

ls sysroot/usr/lib/arm-linux-gnueabihf/libEGL.so.1.1.0

mv sysroot/usr/lib/arm-linux-gnueabihf/libEGL.so.1.1.0 sysroot/usr/lib/arm-linux- gnueabihf/libEGL.so.1.1.0\_backup

ln -s sysroot/opt/vc/lib/libEGL.so sysroot/usr/lib/arm-linux-gnueabihf/libEGL.so.1.1.0

mv sysroot/usr/lib/arm-linux-gnueabihf/libGLESv2.so.2.1.0 sysroot/usr/lib/arm-linux-gnueabihf/ libGLESv2.so.2.1.0\_backup

ln -s sysroot/opt/vc/lib/libGLESv2.so sysroot/usr/lib/arm-linux-gnueabihf/libGLESv2.so.2.1.0

mv sysroot/usr/lib/arm-linux-gnueabihf/libGLESv2.so.2.1.0 sysroot/usr/lib/arm-linux-gnueabihf/ libGLESv2.so.2.1.0\_backup

ls sysroot/usr/lib/arm-linux-gnueabihf/libGLESv2.so.2.1.0

ln -s sysroot/opt/vc/lib/libGLESv2.so sysroot/usr/lib/arm-linux-gnueabihf/libGLESv2.so.2.1.0

ln -s sysroot/opt/vc/lib/libEGL.so sysroot/opt/vc/lib/libEGL.so.1

ln -s sysroot/opt/vc/lib/libGLESv2.so sysroot/opt/vc/lib/libGLESv2.so.2

## **Get sysroot-relativelinks.py**

wget <https://raw.githubusercontent.com/riscv/riscv-poky/master/scripts/>sysroot-relativelinks.py

chmod +x sysroot-relativelinks.py

./sysroot-relativelinks.py sysroot

## Check again sync all file with raspberry pi

rsync -avz --rsync-path="sudo rsync" <username>@<rasp-IP>:/lib sysroot

rsync -avz --rsync-path="sudo rsync" <username>@<rasp-IP>:/usr/include sysroot/usr

rsync -avz --rsync-path="sudo rsync" <username>@<rasp-IP>:/usr/lib sysroot/usr

rsync -avz --rsync-path="sudo rsync" <username>@<rasp-IP>:/opt/vc sysroot/opt/

./sysroot-relativelinks.py sysroot

## Create qt5build directory

mkdir qt5build

cd qt5build/

## Compile All things

../qt-everywhere-src-5.15.2/configure -opengl es2 -device linux-rasp-pi2-g++ - device-option CROSS\_COMPILE=/opt/qt5pi/gcc-linaro-7.5.0-2019.12-x86\_64\_arm-linux- gnueabihf/bin/arm-linux-gnueabihf- -sysroot /opt/qt5pi/sysroot -prefix /usr/local/qt5pi -opensource -confirm-license -skip qtscript -skip qtwayland -skip qtdatavis3d -nomake examples -make libs -pkg-config -no-use-gold-linker -v

*Note :- Check again above command paths according to your host machine. Stay one or two hour for complete this task*

## Make it

make -j8

make install

cd /opt/qt5pi/

rsync -avz --rsync-path="sudo rsync" sysroot/usr/local/qt5pi <username>@<rasp-IP>:/usr/local

*Note:- Now cross-compiler is ready. After that you have to install Qt creator*

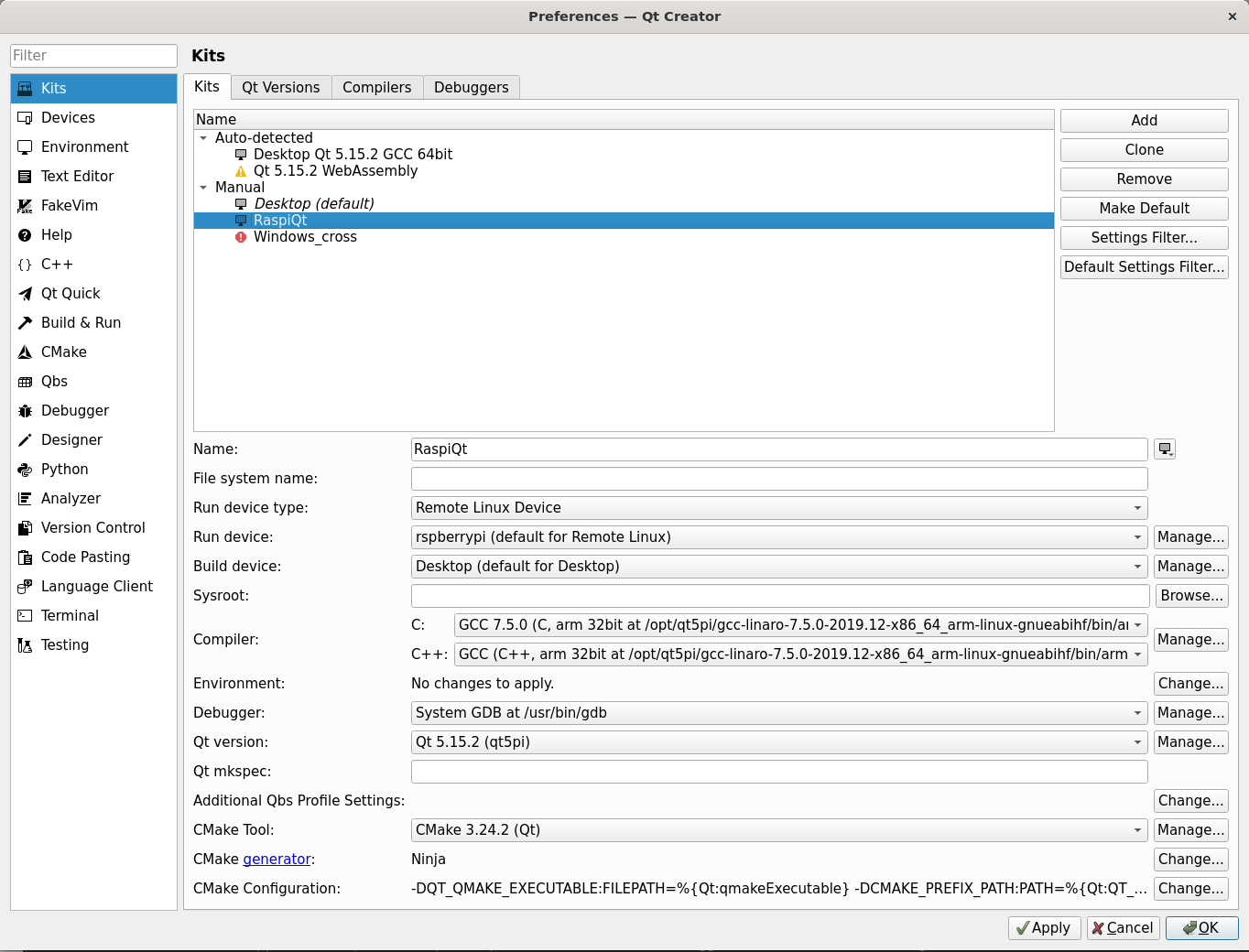
# Install Qt Creator

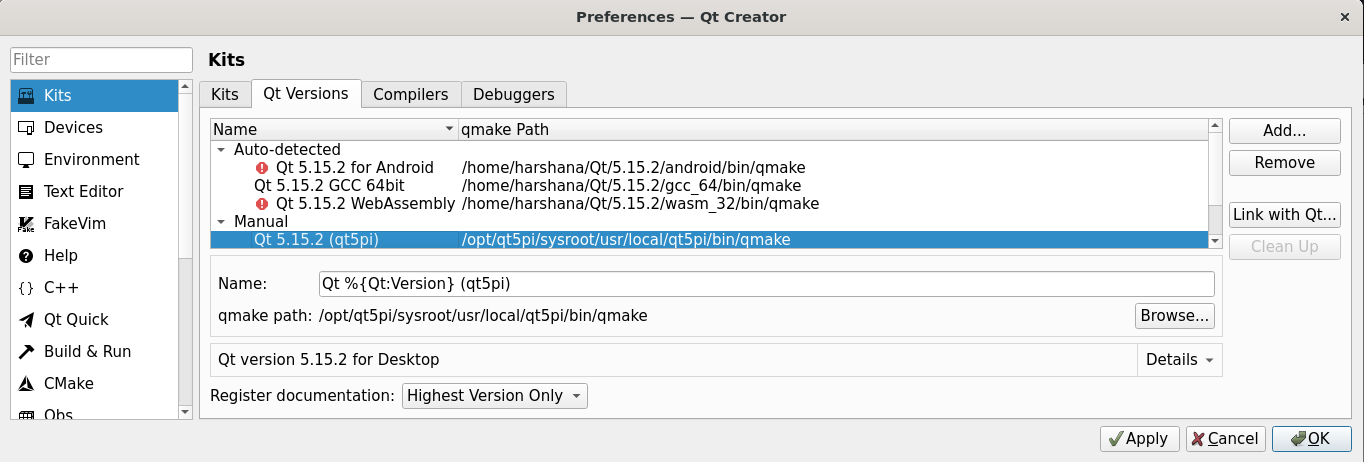
Download Qt creator online installer and install it. Use qt 5.xx.xx version. Because in this we use qt5.

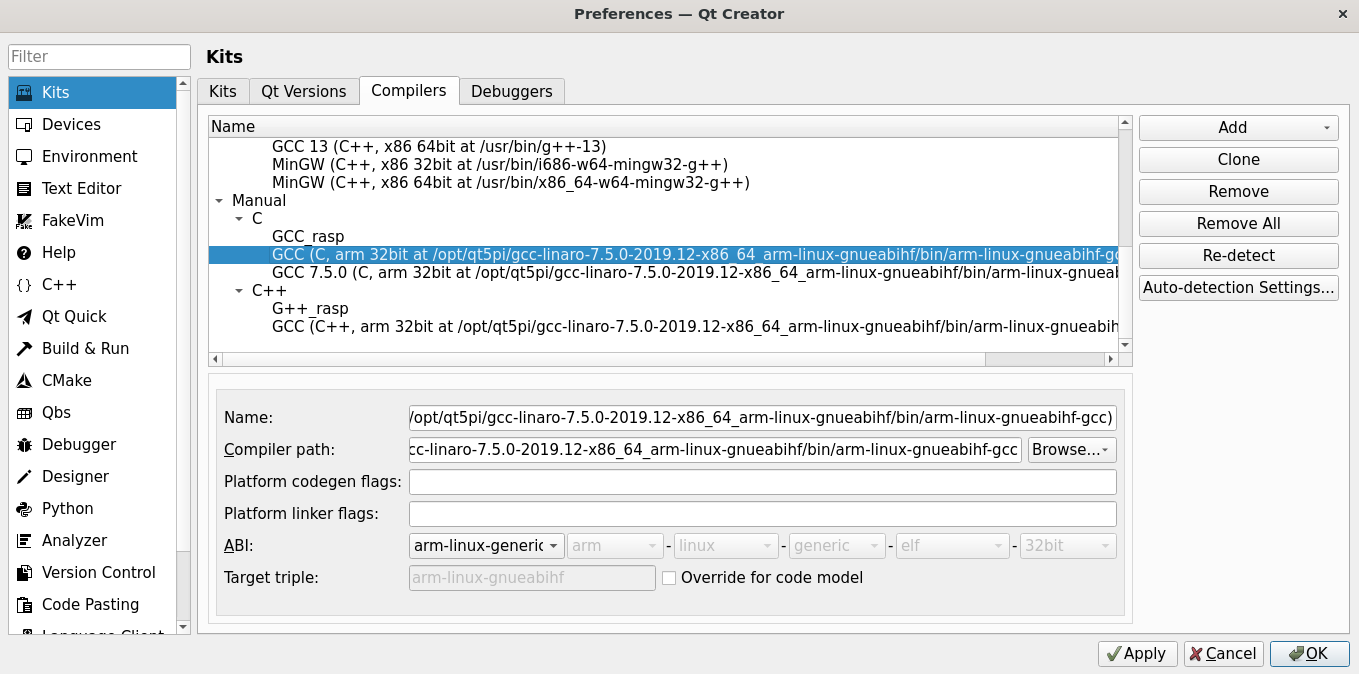
# Configure Qt Creator

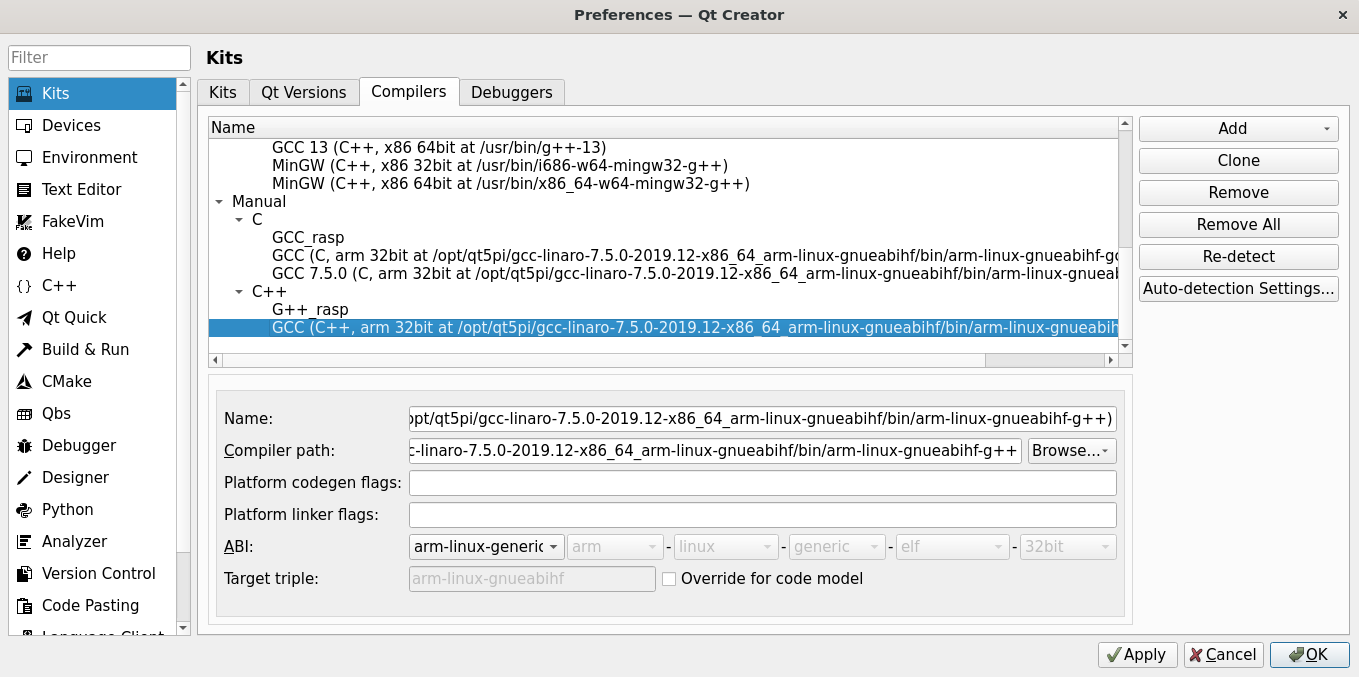
## **Create New Kit**

*Note:- See follow screenshots and use that paths for your configurations*

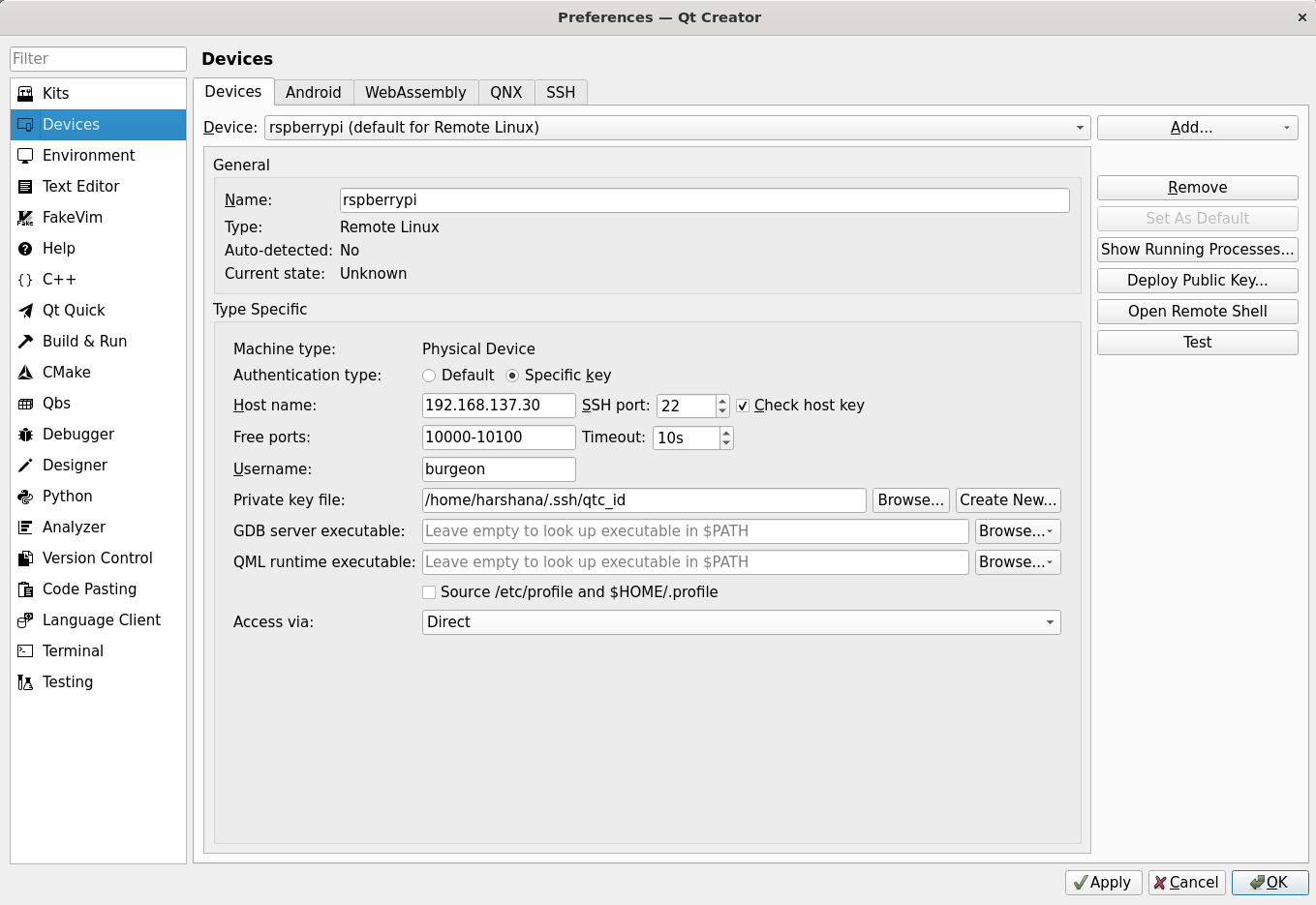
Figure 3: Kit Configuration

Figure 4: Qt Version Configuration

Figure 5: Add GCC Compiler - Linaro Toolchain

Figure 6: Add G++ Compiler - Linaro Toolchain

# Add Raspberry Pi Device



*Note :- This step is necessary one. In this step you have to create private key file when you are going to add the device. Otherwise you will have errors when you are going to compile Qt applications*

# Finish

Now all configurations are complete. Now you can create application for raspberry pi. When you are going to create new project choose newly created kit. Then create project.

After build your project you can copy that build folder to raspberry pi. Then you can execute that application on raspberry pi

# Reference

Tutorial Video -> [Video](https://youtu.be/TmtN3Rmx9Rk)

All Commands -> [All Commands](https://github.com/PhysicsX/QTonRaspberryPi/tree/main/QtRaspberryPi5.14.2)

Linaro toolchain path -> [Toolchain](https://releases.linaro.org/components/toolchain/binaries/latest-7/arm-linux-gnueabihf/gcc-linaro-7.5.0-2019.12-x86_64_arm-linux-gnueabihf.tar.xz)