# Yocto

## Qt Application Development

**Andreas Burghart**

**6 October 2014**

**v1.0**

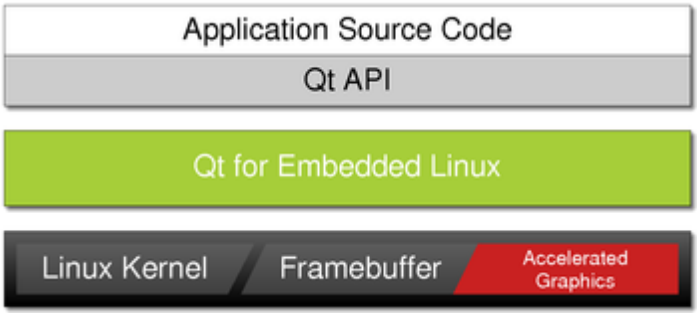
# Contents

1.0 Introduction ..... 3
1.1 Qt for Embedded Linux ..... 3
1.2 Outline ..... 4
1.3 Assumptions ..... 5
1.4 Corrections ..... 5
1.5 Version ..... 5
2.0 Yocto Qt Toolchain Generation and Installation ..... 6
2.1 Yocto Qt Toolchain Generation ..... 6
2.2 Yocto Qt SDK Installation ..... 6
3.0 Qt Creator Installation and Setup ..... 7
3.1 Qt Creator Installation ..... 7
4.0 Configure Target Connection and Toolchain ..... 8
4.1 Create New Device Configuration ..... 8
4.2 Setup New Build & Run Configuration ..... 9
5.0 Create Qt Project and Application ..... 12
5.1 Create New Qt Project ..... 12
5.2 Add Your Application Source Code ..... 13
6.0 Build Your Project and Deploy Application ..... 16
6.1 Build Your Application ..... 16
6.2 Deploy Application Binary to Target ..... 17
7.0 Run and Debug Your Application on the Target ..... 19
7.1 Run Your Application ..... 19
7.2 Debug Your Application ..... 20

# 1.0 INTRODUCTION

## 1.1 Qt for Embedded Linux

Qt for Embedded Linux is a C++ framework for GUI and application development for embedded devices. It runs on a variety of processors, usually with Embedded Linux. Qt for Embedded Linux provides the standard Qt API for embedded devices with a lightweight window system.




Qt for Embedded Linux applications write directly to the framebuffer, eliminating the need for the X Window System and saving memory. The Linux framebuffer is enabled by default on all modern Linux distributions and provides an efficient way to develop GUI applications for embedded Linux devices.

## 1.2 Outline

The aim of this application note is to guide the user through the steps to set up a Qt application development environment for Yocto systems using Qt Designer.

The application note will guide the user through the setup of a Qt sample project, deployment of the build results to a Yocto system and required configurations necessary for application debugging.

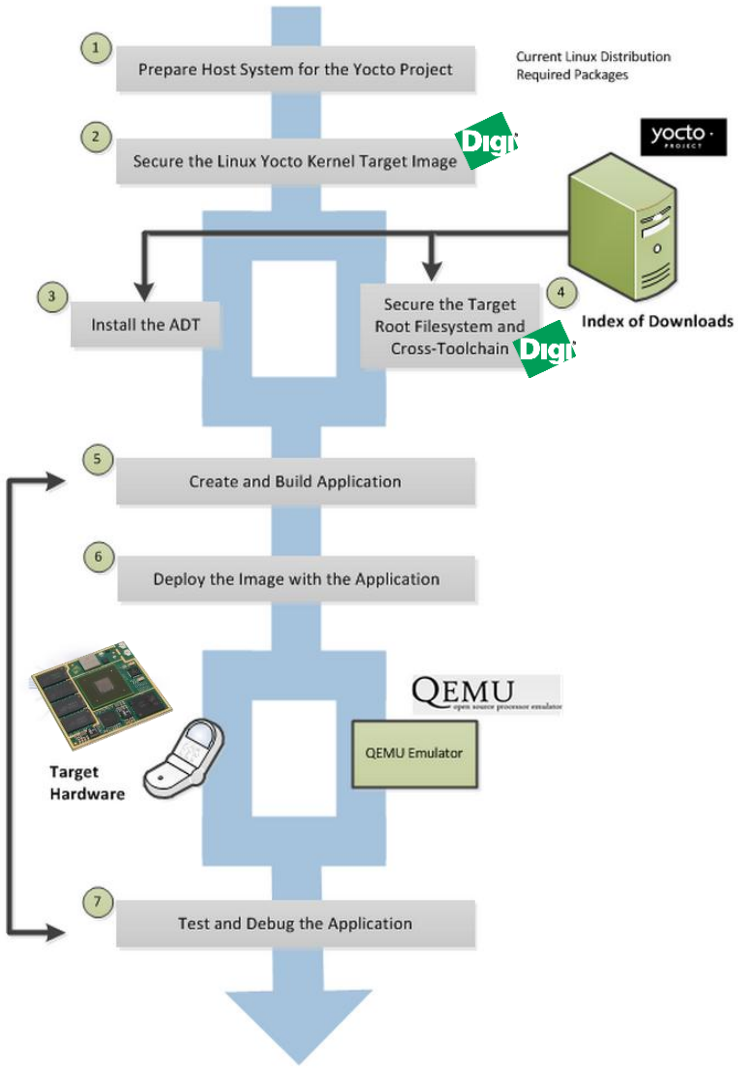



**Figure 1-1: Yocto Application Development Workflow Overview**

## 1.3 Assumptions

This application note has been written for use by technically competent personnel with a good understanding of the usage of a Linux desktop operating system and common Linux commands.

This application note applies to all Digi ARM embedded modules running Digi Embedded Yocto (DEY).

A DEY 1.6 installation on the development host and a Digi ConnectCore 6 SBC target hardware platform is assumed.

It's further assumed the Digi ConnectCore 6 SBC is running a previously compiled dey-image-graphical operating system image with the following extra image features included:

- EXTRA_IMAGE_FEATURES = "debug-tweaks tools-debug eclipse-debug dey-qt"

Follow the "Digi Embedded Yocto 1.6 First Steps Guide (SBC)", P/N: 90001423_A for detailed instructions about the installation of DEY and the building of the dey-image-graphical image.

## 1.4 Corrections

Requests for corrections or amendments to this application note are welcome and should be addressed to: embedded.presales@digi.com. Requests for new application notes can be sent to the same address.

## 1.5 Version

| | |
|---|---|
| Version Number: | 0.1 |
| Status: | DRAFT |

# 2.0 YOCTO QT TOOLCHAIN GENERATION AND INSTALLATION

## 2.1 Yocto Qt Toolchain Generation

To build a Qt enabled image the dey-qt extra image feature needs to be included in your project. Make sure the project configuration (conf/local.conf) is configured with the following extra image features:

- EXTRA_IMAGE_FEATURES = "debug-tweaks tools-debug eclipse-debug dey-qt"

Initialize the build environment for the ccimx6sbc platform and build the matching toolchain with the following commands:

- source /usr/local/dey-1.6/mkproject.sh -p ccimx6sbc
- bitbake meta-toolchain-qt

The resulting SDK file is located in your project folder in the tmp/deploy/sdk directory:

- dey-eglibc-i686-meta-toolchain-qt-cortexa9hf-vfp-neon-toolchain-qt-1.6.1.sh

## 2.2 Yocto Qt SDK Installation

Run the .sh file to install the toolchain in a directory of choice (this example assumes installation into /opt/dey-1.6/graphical/qt/ccimx6sbc):

- ./dey-eglibc-i686-meta-toolchain-qt-cortexa9hf-vfp-neon-toolchain-qt-1.6.1.sh

```
Enter target directory for SDK (default: /opt/poky/1.6):  /opt/dey-1.6/graphical/qt/ccimx6sbc/

You are about to install the SDK to
"/opt/dey-1.6/graphical/qt/ccimx6sbc/". Proceed[Y/n]?  Y

[sudo] password for <user>:  ****

Extracting SDK...done

Setting it up...done

SDK has been successfully set up and is ready to be used.
```

# 3.0 QT CREATOR INSTALLATION AND SETUP

## 3.1 Qt Creator Installation

Download the standalone Qt Creator IDE from the Qt Website: http://qt-project.org/downloads.

Install Qt Creator by running the install file:

- ./qt-creator-opensource-linux-x86-3.2.1.run

Patch the qtcreator.sh start script to source the Yocto environment setup script at Qt Creator start-up, with the following patch code:

```
diff --git a/qtcreator.sh b/qtcreator.sh
index aca30f5b9ecf..114f69717877 100755
--- a/qtcreator.sh
+++ b/qtcreator.sh
@@ -1,3 +1,4 @@
+source /opt/dey-1.6/graphical/qt/ccimx6sbc/environment-setup-cortexa9hf-vfp-neon-dey-linux-gnueabi
 #! /bin/sh

 makeAbsolute() {
```

You can copy & paste the above patch code into a new file e.g. named “0001-Source-Yocto-environment-script-setup.patch” and apply the patch with the following command:

- patch qtcreator.sh < 0001-Source-Yocto-environment-script-setup.patch

Make sure the path given in the patch file matches the path to the dey environment setup script of your development system.

Start Qt Creator:

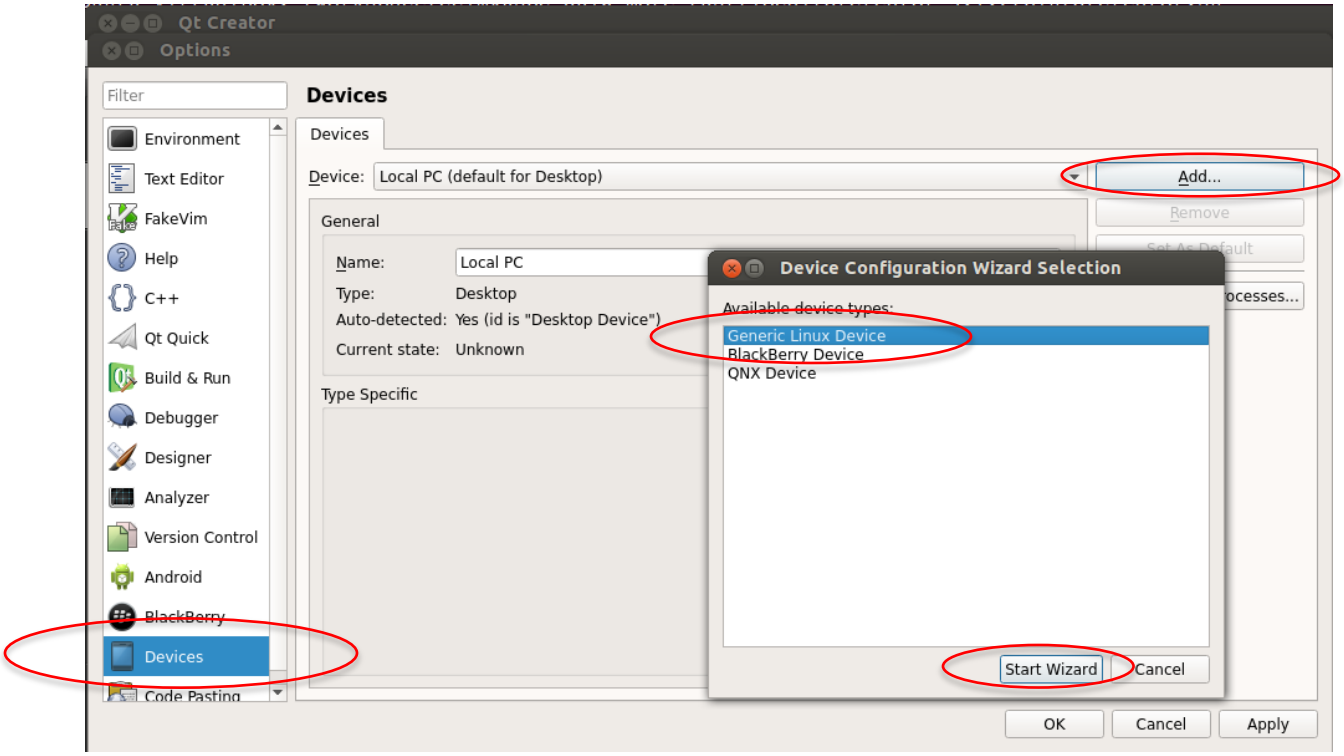
- /usr/local/qtcreator-3.2.1/bin/qtcreator.sh &

# 4.0 CONFIGURE TARGET CONNECTION AND TOOLCHAIN

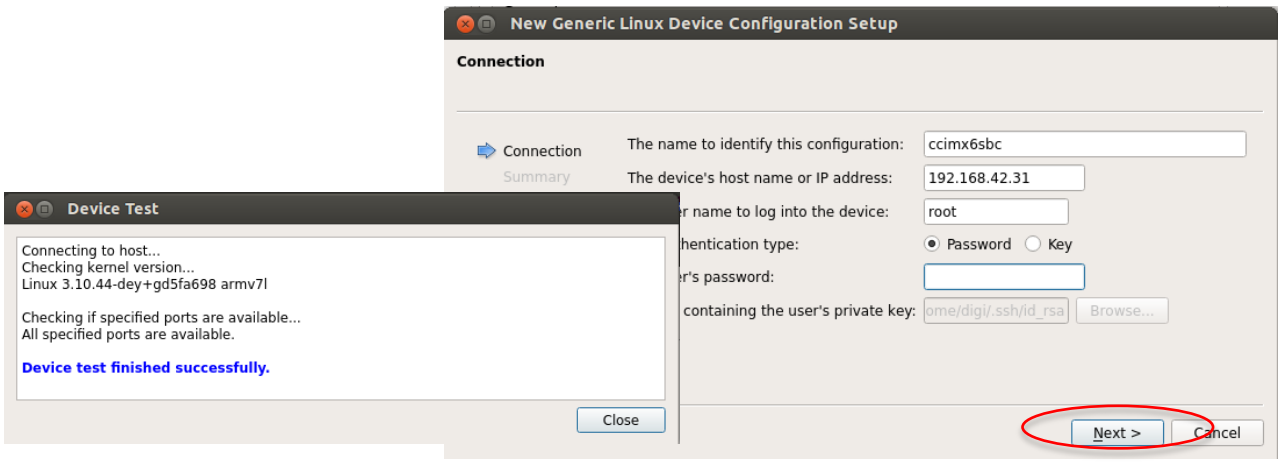
## 4.1 Create New Device Configuration

In Qt Creator menu go to the “Tools” -> “Options"

Select “Devices” and “Add” a new “Generic Linux Device”:




Configure the target connection with your targets IP address, login and password and run the optional “Device Test”:

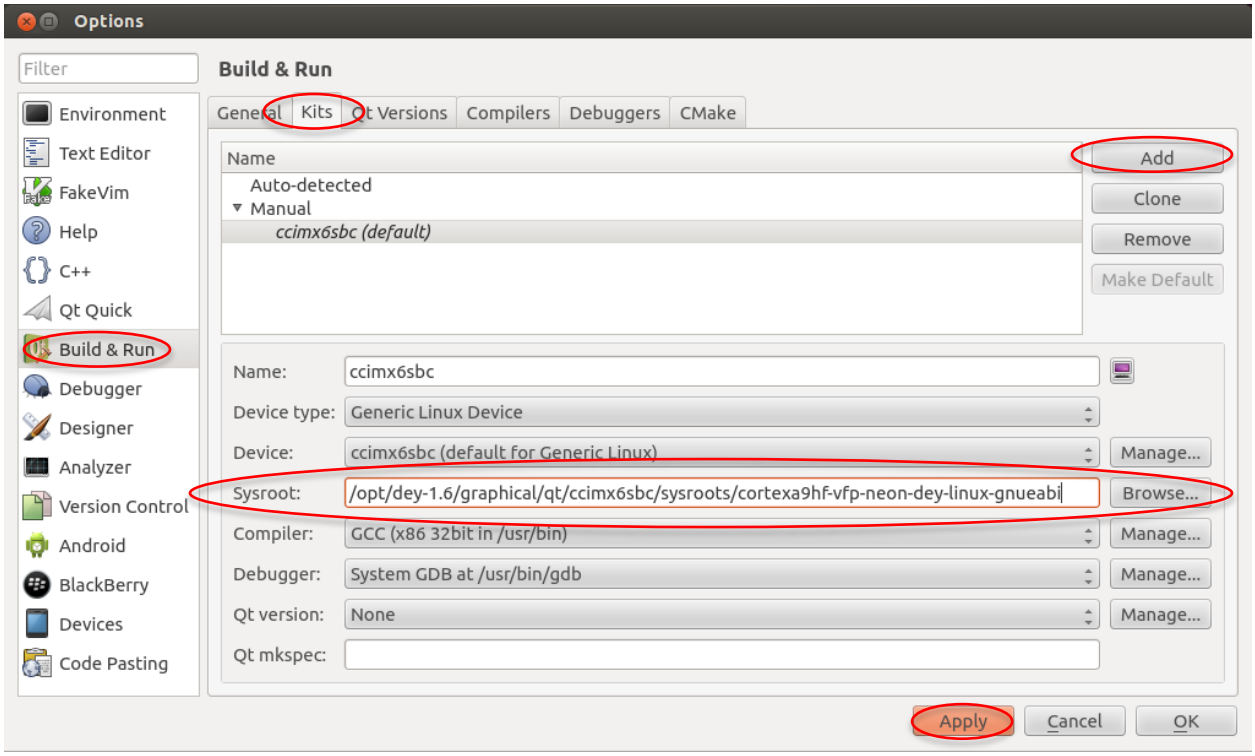


## 4.2 Setup New Build & Run Configuration

Select “Build & Run”, go to the “Kits” tab and “Add” a new kit.

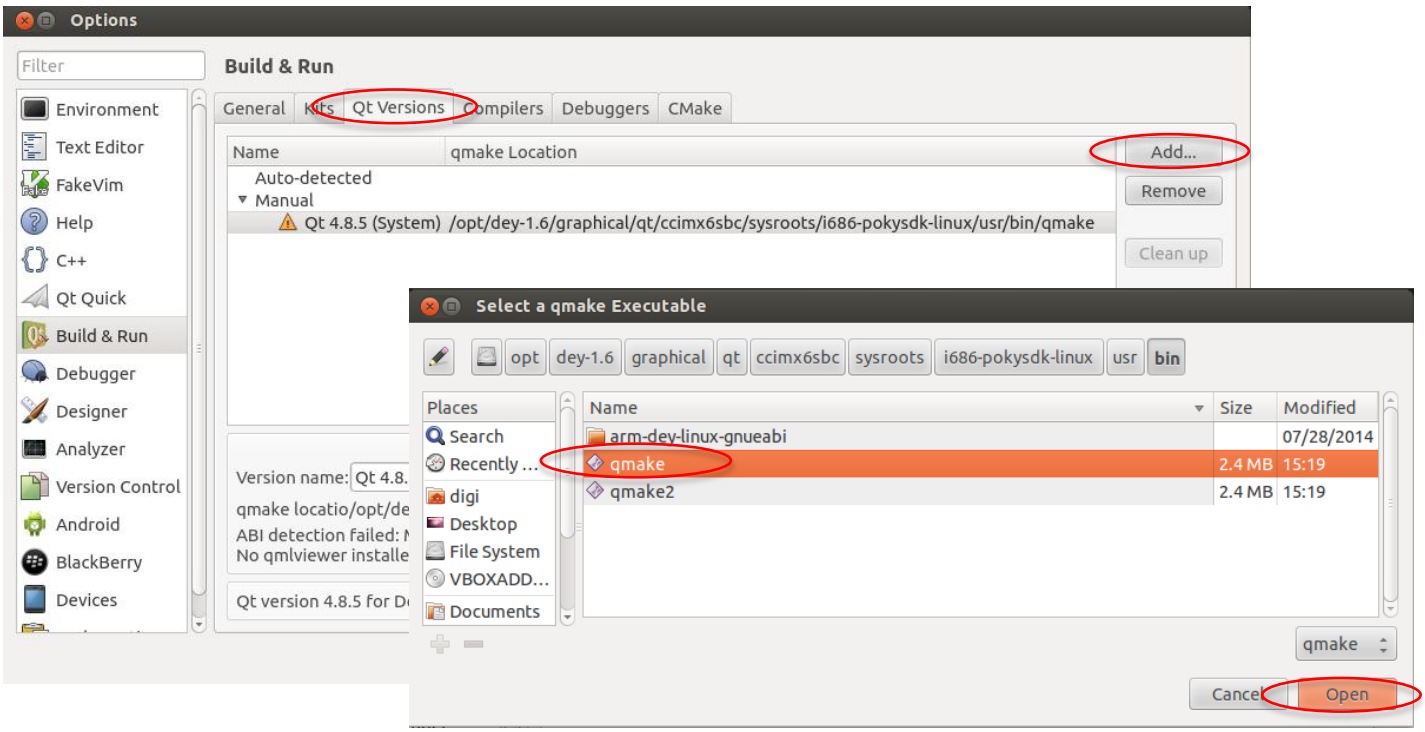
In the “Sysroot:” field, browse to the sysroot folder of your Qt toolchain installation directory and point to cortexa9hf-vfp-neon-dey-linux-gnueabi:

- /opt/dey-1.6/graphical/qt/ccimx6sbc/sysroots/cortexa9hf-vfp-neon-dey-linux-gnueabi



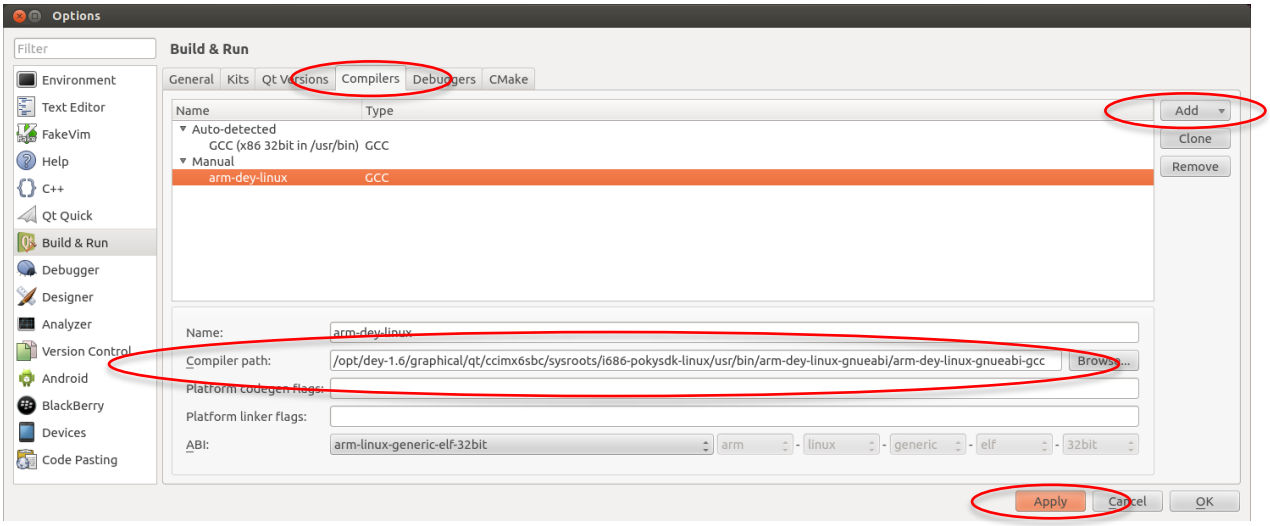
Move to the “Qt Version” tab and make sure your toolchain’s qmake is selected. If not Auto-detected, add it manually:

- /opt/dey-1.6/graphical/qt/ccimx6sbc/sysroots/i686-pokysdk-linux/usr/bin/qmake



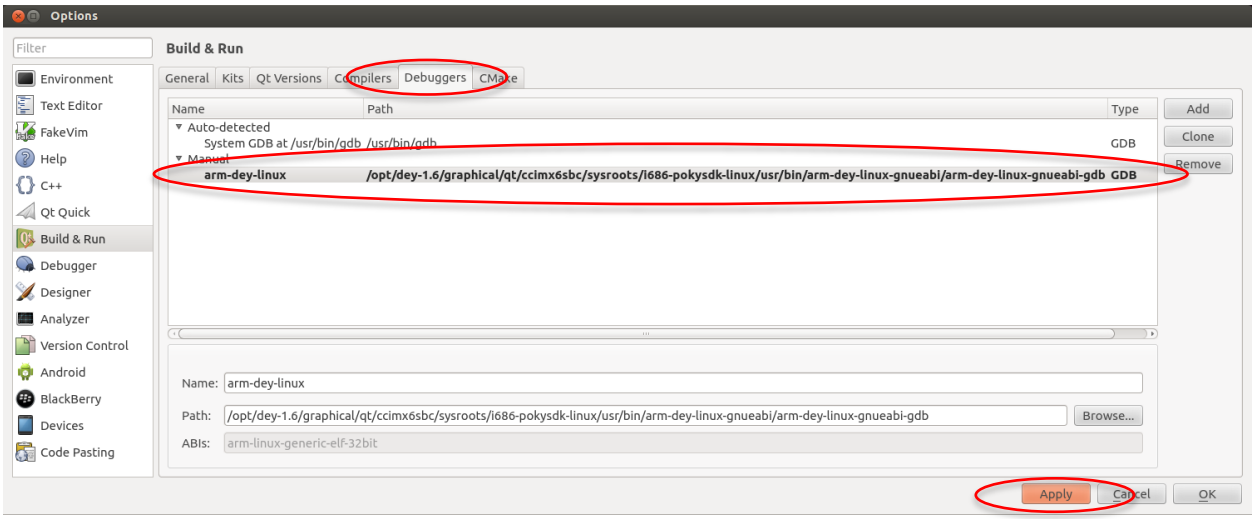

Move to the “Compilers” tab and “Add” you gcc compiler from your toolchain installation folder:

- /opt/dey-1.6/graphical/qt/ccimx6sbc/sysroots/i686-pokysdk-linux/usr/bin/arm-dey-linux-gnueabi/arm-dey-linux-gnueabi-gcc

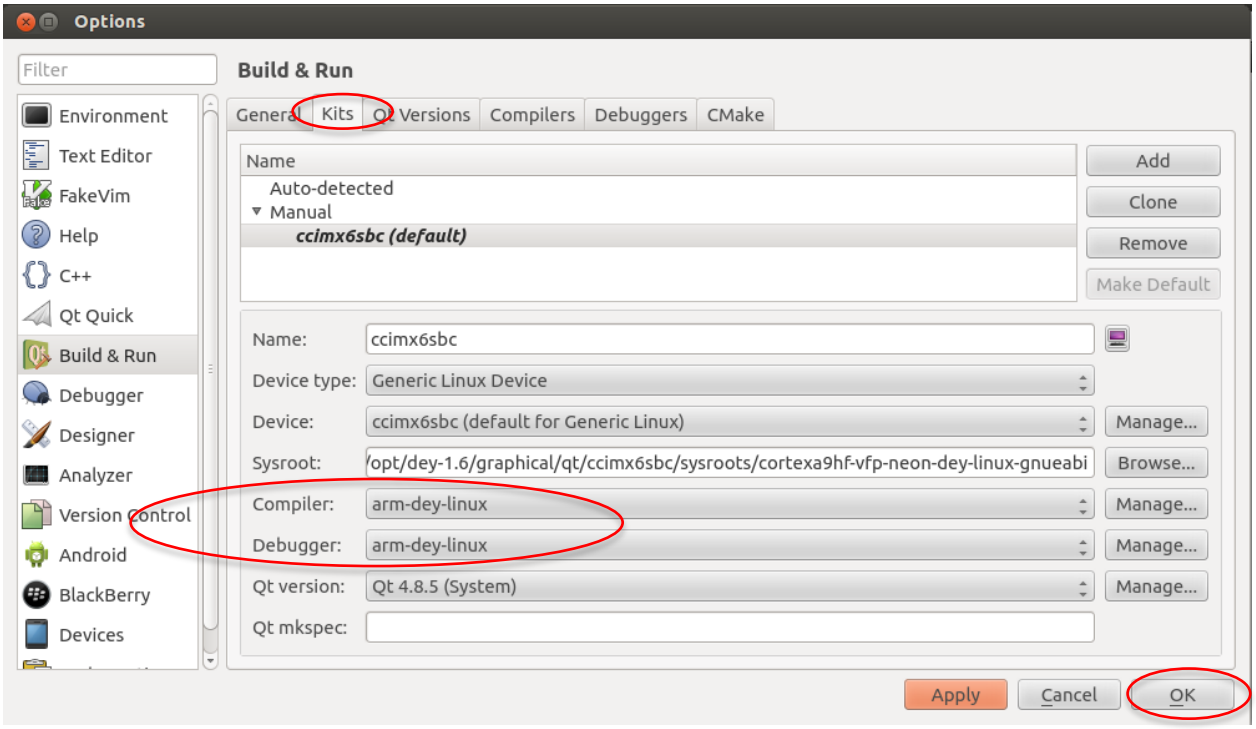


Move to the “Debuggers” tab and “Add” your gdb debugger from your toolchain installation folder:

- /opt/dey-1.6/graphical/qt/ccimx6sbc/sysroots/i686-pokysdk-linux/usr/bin/arm-dey-linux-gnueabi/arm-dey-linux-gnueabi-gdb



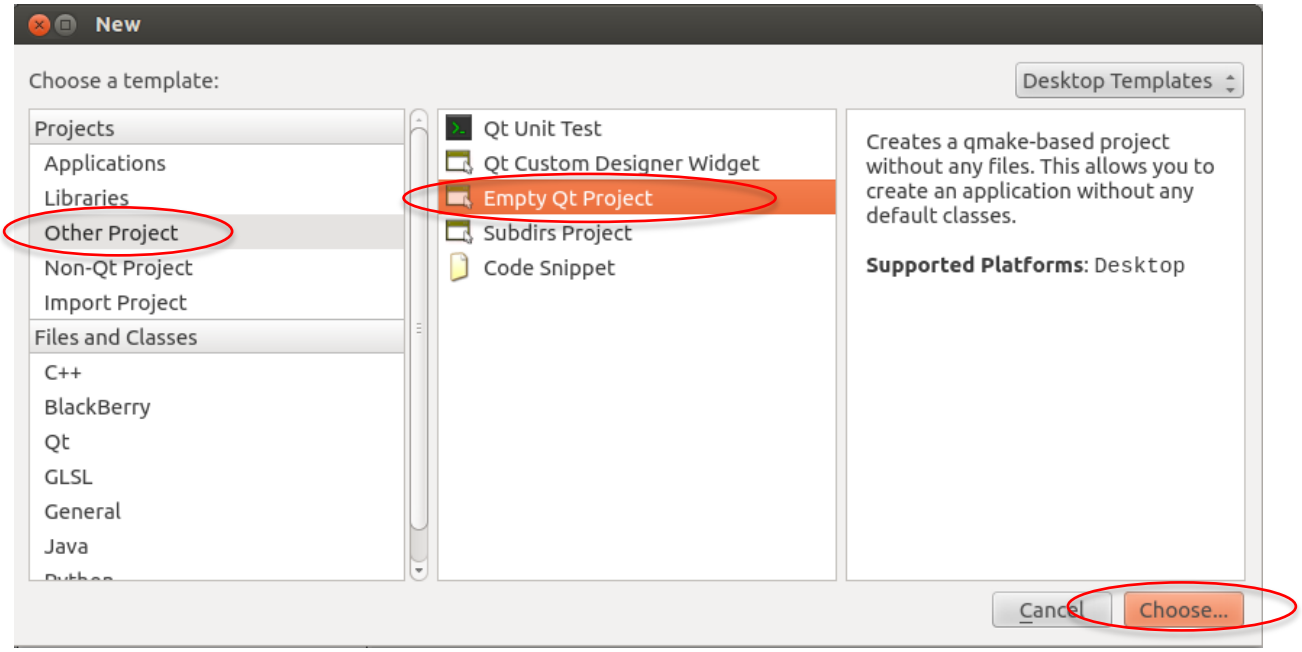

Go back to the “Kits” tab and chose the just configured arm-dey-linux “Compiler” and “Debugger”. Finalize your settings by pressing the “Ok” button:



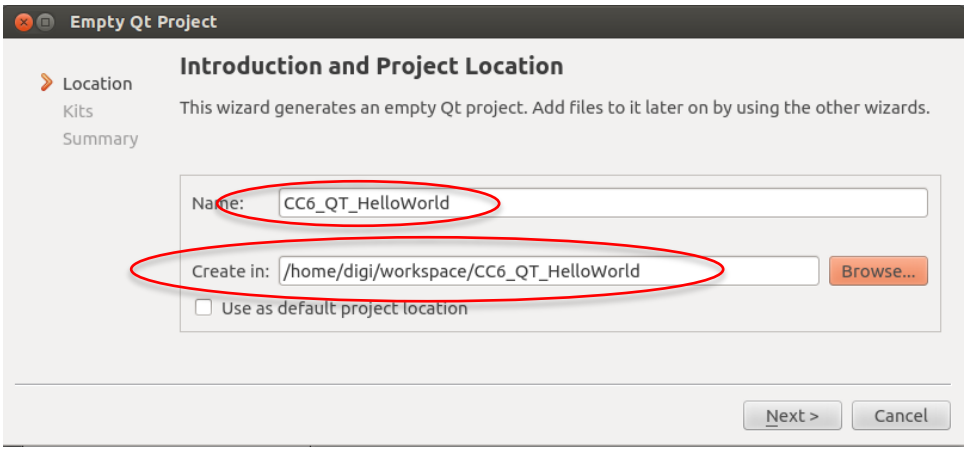
# 5.0 CREATE QT PROJECT AND APPLICATION

## 5.1 Create New Qt Project

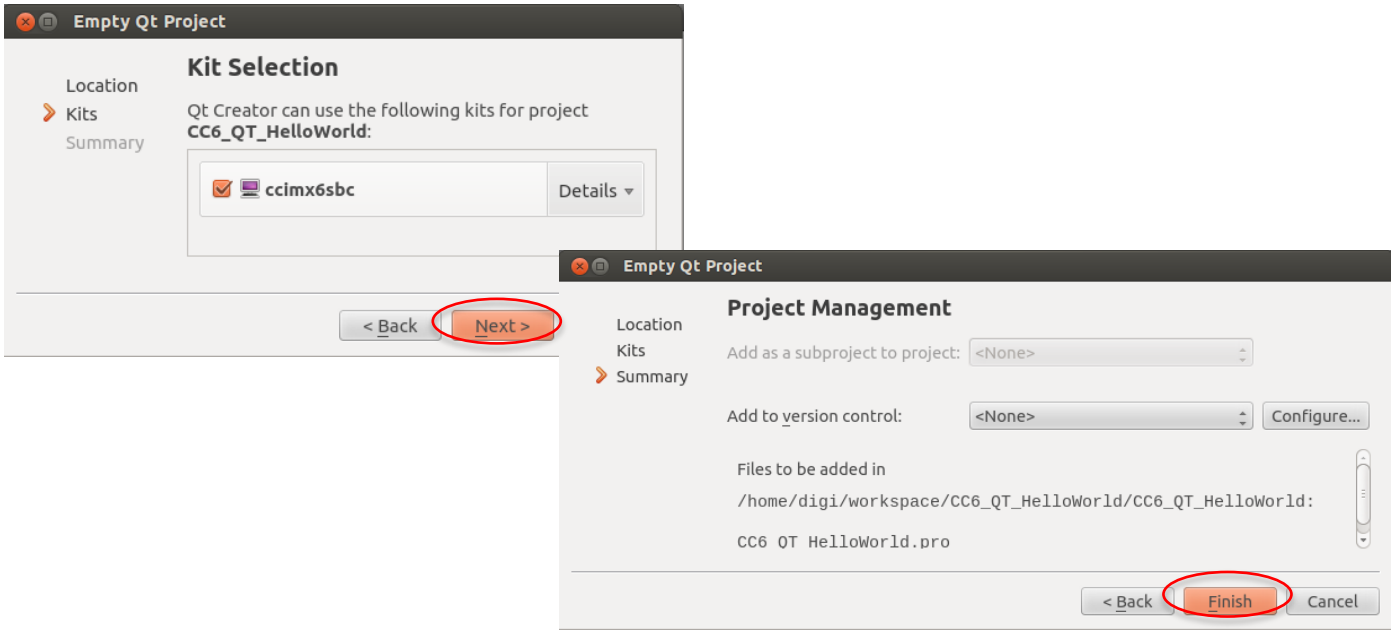
In Qt Creator menu go to the “File” -> “New File  or Project…" and “Choose” an “Empty Qt Project” from the “Other Project” category:




Enter a project “Name” and select the “Create in” folder within your workspace directory:

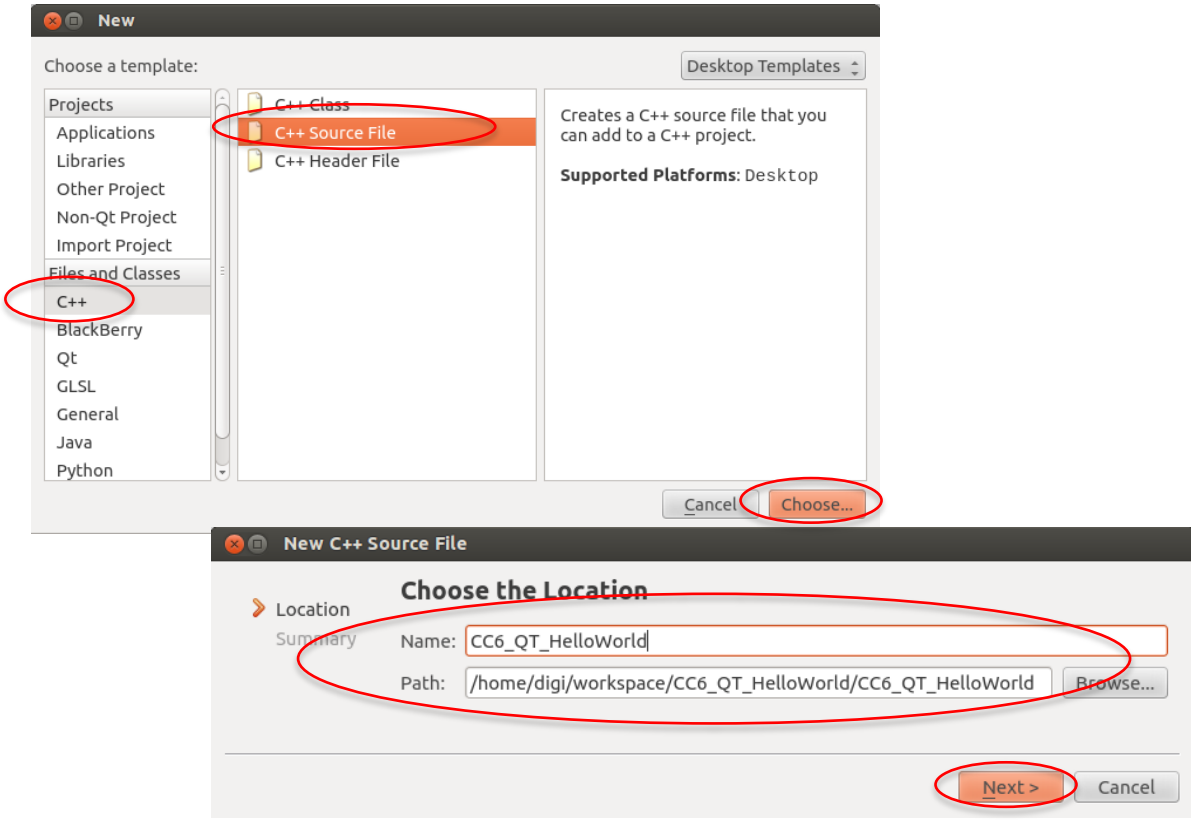


Select the just created “ccimx6sbc” kit and finish the project setup:

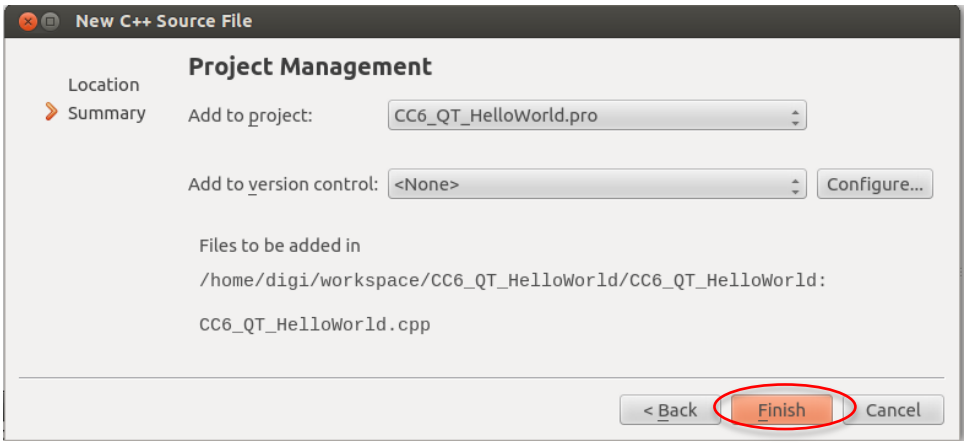



## 5.2 Add Your Application Source Code

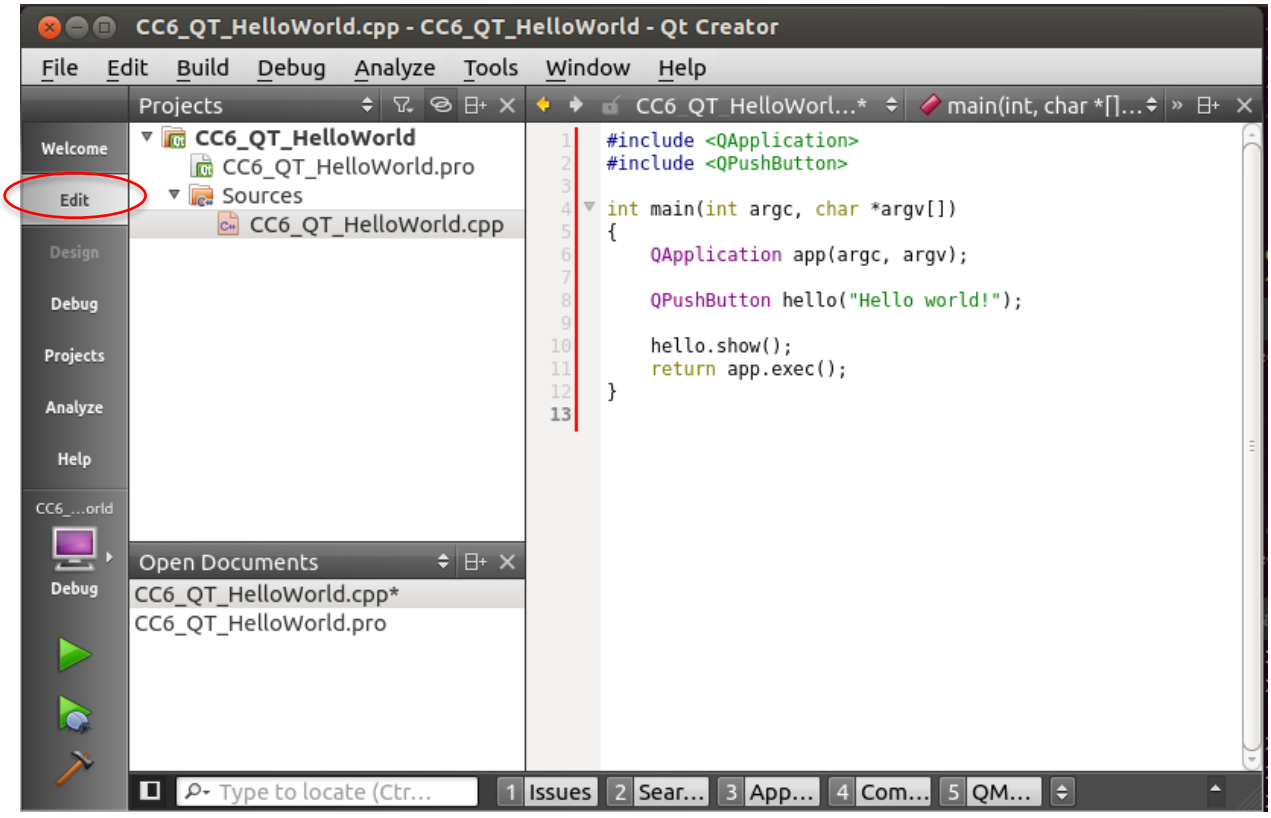
Go to the “File” -> “New File or Project…" to add your new application “C++ Source File”:



Complete the new project setup by pressing the “Finish” button on the “Project Management” dialog box:




Write your application source code into the newly added source code file (or copy and paste the hello world example code from above):



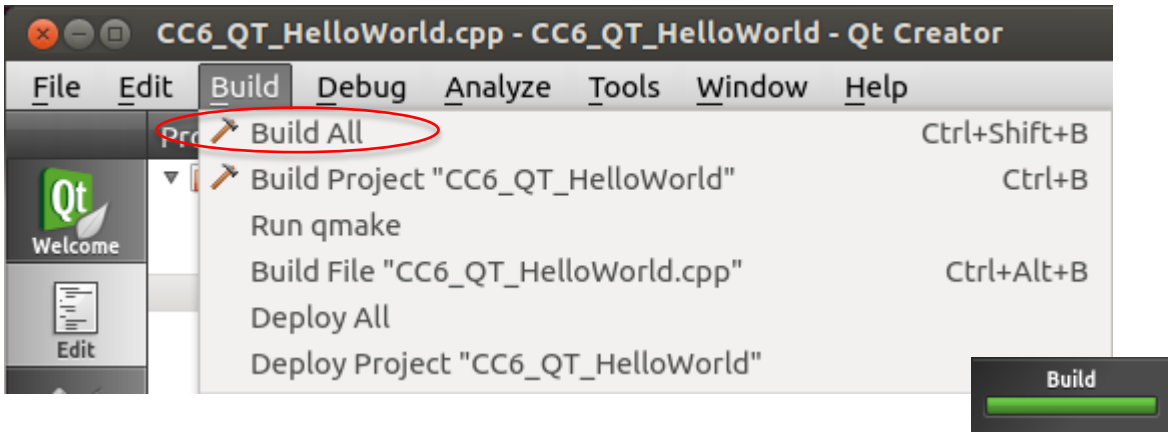
"Hello World" Qt example code:

```
#include <QApplication>
#include <QPushButton>
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QPushButton hello("Hello world!");
    hello.show();
    return app.exec();
}
```

# 6.0 BUILD YOUR PROJECT AND DEPLOY APPLICATION

## 6.1 Build Your Application

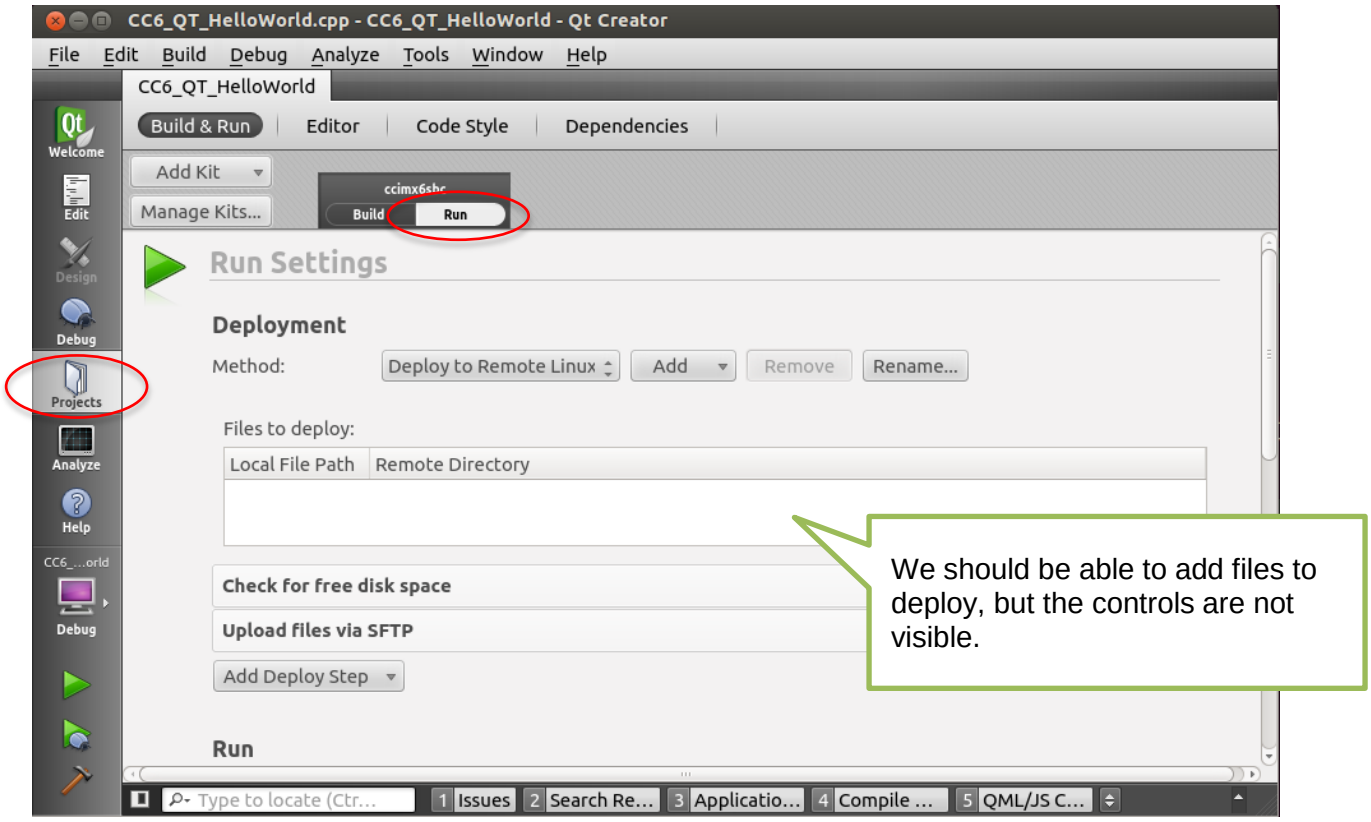
In Qt Creator menu go to the “Build” -> “Build All”:




The application source code will be guild using your Yocto Qt toolchain configuration.

## 6.2 Deploy Application Binary to Target

In the left side tool bar in Qt Creator go to the “Projects” and select the “Run” configuration of your kit:



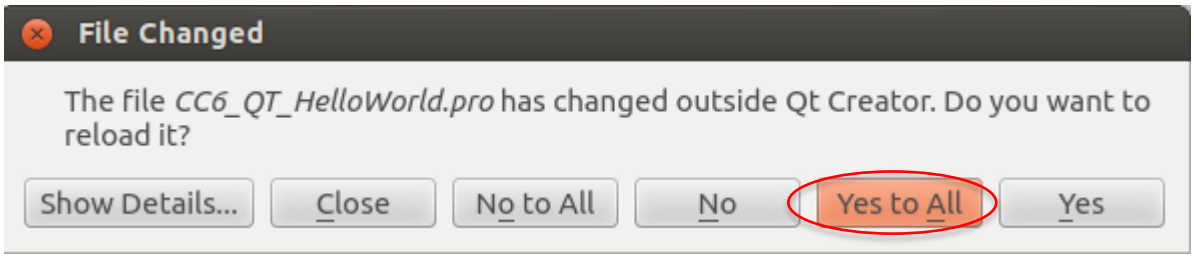

Manually add files to deploy in your .pro project configuration file located in your project directory:

- nano CC6_QT_HelloWorld/CC6_QT_HelloWorld.pro

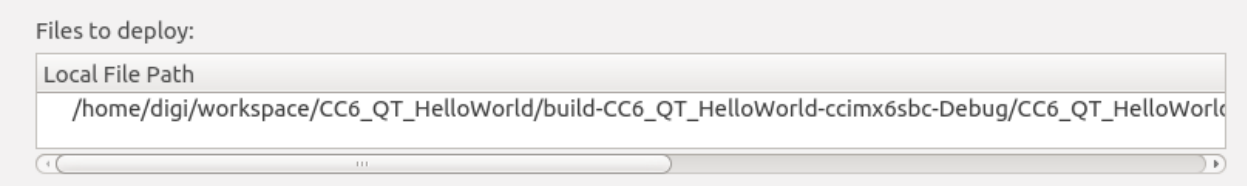
Add the following code to the .pro file and save it:

```
TARGET = CC6_QT_HelloWorld
target.files = CC6_QT_HelloWorld
target.path = /
INSTALLS += target
```

Qt Creator may notify you that the .pro file changed. Accept the changes by pressing the “Yes to All” button:



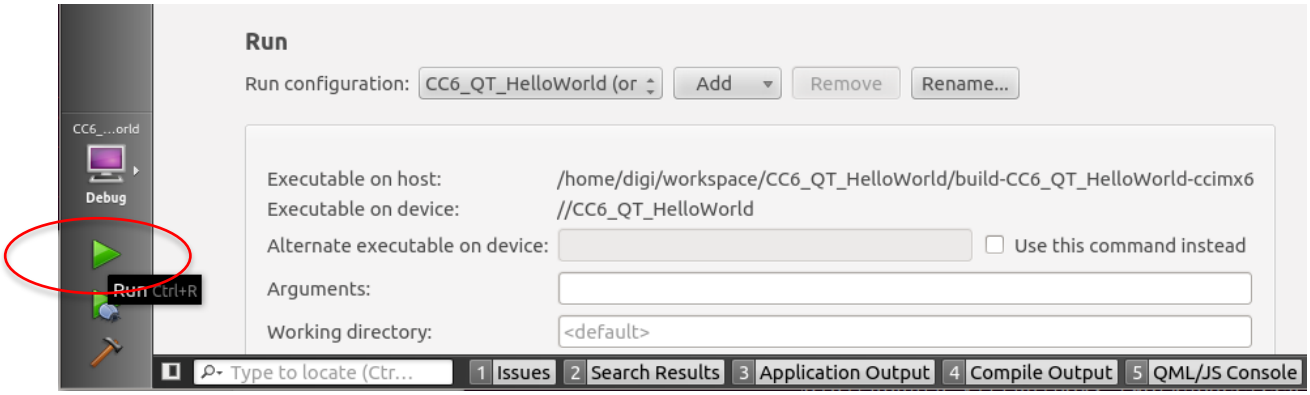

Your “Files to Deploy” list in Qt Creator should now get updated and show the added files:



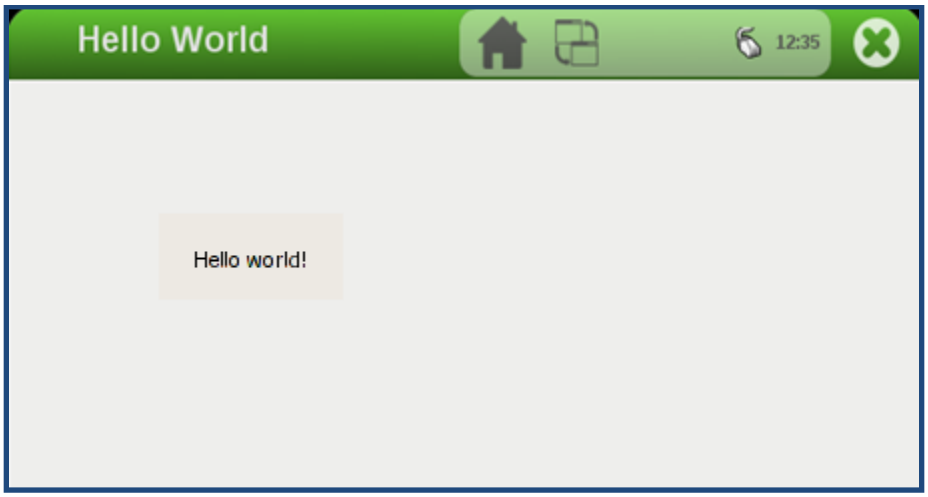
# 7.0 RUN AND DEBUG YOUR APPLICATION ON THE TARGET

## 7.1 Run Your Application

In the left side tool bar in Qt Creator press the green arrow “Run” button to execute the application on your target:




You should see the “Hello World” application come up on your target’s display:



## 7.2 Debug Your Application

Similar to running the application you can press the debug arrow on the left side tool bar in Qt creator to start debugging: