Build and install the QT5 toolchain

In order to compile the required libraries, a full OpenEmbedded build environment needs to first be in place.

In order to build Qt applications targeted for the embedded Yocto image, we need to have a full Qt toolchain which provides (besides cross GCC and GDB) the Qt headers to include and libraries to link against.

The toolchain can be built using sources and a bitbake recipe included the yocto repository. If you've closed your terminal since building Yocto, you may need to first run the setup_environment.sh script as described in Build Yocto release.

From the Yocto Jethro build output directory (eg: 'build_fb', 'build_x11' or 'build_wayland'), the bitbake target providing the toolchain is:

```
$ bitbake meta-toolchain-qt5
```

After building the bitbake target, run the script to install the SDK:

```
$ <BUILD_DIR_PATH>/tmp/deploy/sdk/poky-glibc-x86_64-meta-toolchain-qt5-cortexa9hf-vfp-neon-toolchain-2.0.sh
```

Assuming you accept the default settings, the SDK will be installed in /opt/poky/2.0.

The toolchain is updated from time to time and may need to be rebuilt and reinstalled using these same steps.

Build and install the QT5 toolchain

Before Qt creator can be used to build projects targeted for our embedded Yocto images, the Qt5 toolchain must be built and installed on your development machine.

Install QT creator

In order to use QT 5.5, QT creator version must be 3.1.1 or above.

In CentOS, the package qt-creator is in the epel.repo

In Ubuntu >= 15.04, qtcreator package can be directly installed via "sudo apt-get install qtcreator".

In Ubuntu <= 14.10, please execute following instructions:

```
$ sudo add-apt-repository ppa:ubuntu-sdk-team/ppa
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install qtcreator
```

Configure QT creator

Qt uses special configuration files to describe the build environment called mkspecs (they specify which compiler, linker or other tools to use). However, this configuration files still need to know where the compiler (or cross-compiler in our case) is located.

The SDK created a script which does all the hard work (see below).

On systems where we want to build for both x86 and imx6, it may be cleaner to allow a custom build script to invoke the environment-setup script when needed, rather than doing it beforehand (more on this later in the guide):

```
$ cd /opt/poky/2.0
$ . environment-setup-cortexa9hf-vfp-neon-poky-linux-gnueabi
$ qtcreator
```

Add the Target Device

Go to Tools => Options. In the left panel, click on Devices and add a new device representing the target "mydevice":

- Press Add and choose Generic Linux Device
- Specifiy a name (e.g. mydevice)
- Fill in the device's IP address
- Authentication on our modules by default: Password, User "root", empty password

Add the debugger

Go to Tools => Options. In the left panel, click on Build&Run.

- Click on the Debuggers tab
- Press Add
- Specifiy a name (e.g. mydevice GDB)
- Specify the path: `/opt/poky/2.0/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-gdb`

Add the compiler

Go to Tools => Options. In the left panel, click on Build&Run.

- Click on the Compilers tab
- Press Add
- Specifiy a name (e.g. mydevice GCC)
- Specify the path: `/opt/poky/2.0/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-`

```
linux-gnueabi/arm-poky-linux-gnueabi-cpp
```

Add the QT version

Go to Tools => Options. In the left panel, click on Build&Run.

- Click on the QT Version tab
- Press Add
- Specifiy a name (e.g. mydevice QT 5.5.1)
- Specify the path: `/opt/poky/2.0/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/qmake`

Add the kit

Go to Tools => Options. In the left panel, click on Build & Run.

- Click on the Compilers tab
- Press Add
- Name: mydevice
- Device: mydevice
- Sysroot: /opt/poky/2.0/sysroots/cortexa9hf-vfp-neon-poky-linux-gnueabi
- Compiler: mydevice GCC
- Debugger: mydevice GDB
- Qt version: mydevice QT 5.5.1
- Qt mkspec: leave empty

Deploying Applications

When you run the application, Qt Creator copies the necessary files to the device and starts the application on it.

In your .pro file, remember to add the following lines to allow QT creator knowing where executable will be deployed:

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