Embedded Linux Customer Workshop

Develop your Qt based UI with Atmel SAMA5D3

Hands-On Training Modules References

1. AN-8995_SAMA5D3-Xplained_demo_Programming
2. AN-8498_SAMA5D3-Xplained_u-boot_Introduction
3. AN-8503_SAMA5D3-Xplained_Device_tree_basics
4. AN-8513_SAMA5D3-Xplained_Qt_SDK_GetStart
5. AN-8518_SAMA5D3-Xplained_Develop_Qt_UI
6. AN-8523_SAMA5D3-Xplained_QML_Intro

Prerequisites

- **Hardware Prerequisites**
  - Atmel® SAMA5D3 Xplained XSTK
  - Micro USB to USB-A cable
  - USB FTDI cable
    - FTDI TTL-232R-3V3 USB to TTL serial cable
- **Software Prerequisites**
  - Ubuntu 12.04 LTS (32 or 64 bit)
- **Estimated Completion Time:** 10 minutes

Introduction

In order to start the different hands-on sessions, we need to ensure that you've installed the necessary SW & Tools on your computer. It includes cross compilers, Atmel SAM-BA and various Linux applications like vim, git, tftp server, etc.

Please follow the instructions described in this document to help you.

Table of Contents

Icon Key Identifiers

INFO	Delivers contextual information about a specific topic
TIPS	Highlights useful tips and techniques
TO DO	Highlights objectives to be completed on the Linux computer
RESULT	Highlights the expected result of an assignment step
WARNING	Indicates important information
EXECUTE	Highlights actions to be executed out of the SAMA5D3 Xplained board

1. Hardware Prerequisites

- Atmel® SAMA5D3 Xplained XSTK
- Two USB host ports
- Ethernet cable
- USB FTDI cable

2. Software Prerequisites

- Workstation or laptop with Ubuntu 12.04 LTS installed (32 or 64 bit)
- User account with “sudoer” rights
- The Prerequisites installation MUST be done BEFORE the session (full details in the Prerequisites section)

3. Hands-On set-up

During this SAMA5D3 Linux workshop we will provide you a USB key with the whole materials including presentations, hands-on manual and the whole required software components.

To prevent any network bandwidth issue you will use the software from the USB key content to your local at91data folder. This means, that you will not have to make any ftp download (wget command) during hands-on deployment.

At the beginning of the session please copy the whole USB key content on your laptop:

- `at91data` folder content into your local ~/`at91data` folder
- `AN-11811_SAMA5D3_RSA_Customer_Workshop_2015.tar.tgz`

During all hands-on pay attention to the Icons as some commands needs be run out of the SAMA5D3 Xplained thanks to the serial console, and others needs to be run on your computer:

EXECUTE Highlights actions to be executed out of the SAMA5D3 Xplained board

TO DO Highlights objectives to be completed on the Linux computer

Develop your Qt based UI with the Atmel SAMA5D3

AN-11811

Introduction

On behalf of the Atmel Applications Support Teams, we would like to thank you for attending this training.

A SAMA5D3x-XSTK and others hardware required for the workshop will be provided to you during the training session.

In order to start the various hands-on sessions, we need to ensure that the **Ubuntu 12.04 LTS** is already installed on your computer.
Please follow the instructions described in this document and
Install the Ubuntu 12.04 in your Windows XP or Win7 machine BEFORE the training.
You need to have several tools and software components installed on your computer, so follow up and complete the whole prerequisites steps BEFORE the training.

Thanks!

Table of Contents

Icon Key Identifiers

INFO	Delivers contextual information about a specific topic
TIPS	Highlights useful tips and techniques
TO DO	Highlights objectives to be completed
RESULT	Highlights the expected result of an assignment step
WARNING	Indicates important information
EXECUTE	Highlights actions to be executed out of the target when necessary

1. Hardware Prerequisites

Personal Computer with at least:

- Two USB host ports
- One Ethernet port

2. Software Prerequisites

- OS: WinXP or Win7 with 1GB DDR
- A Windows drive with **18GB** free disk space, we will install Ubuntu OS in this drive
- Administrator rights on the laptop
- Fast Internet connection to retrieve material from an Ubuntu website

Estimated Completion Time: 50 minutes

2.1 Material Download

TO DO Create a working folder

Create an "`at91data`" directory directly under your `C:\`

TO DO Download WUBI Installer for Ubuntu 12.04 version only and save it under "`at91data`" :
http://releases.ubuntu.com/12.04/wubi.exe

TO DO Download the Ubuntu 12.04 image file and save it under "`at91data`"

For 32-bit machine, use this link:: http://releases.ubuntu.com/12.04/ubuntu-12.04.4-desktop-i386.iso

This Ubuntu image (ubuntu-12.04.4-desktop-i386.iso) includes most machines with Intel/AMD/etc type processors and almost all computers that run Microsoft Windows, as well as newer Apple Macintosh systems based on Intel processors. Choose this if you are at all unsure

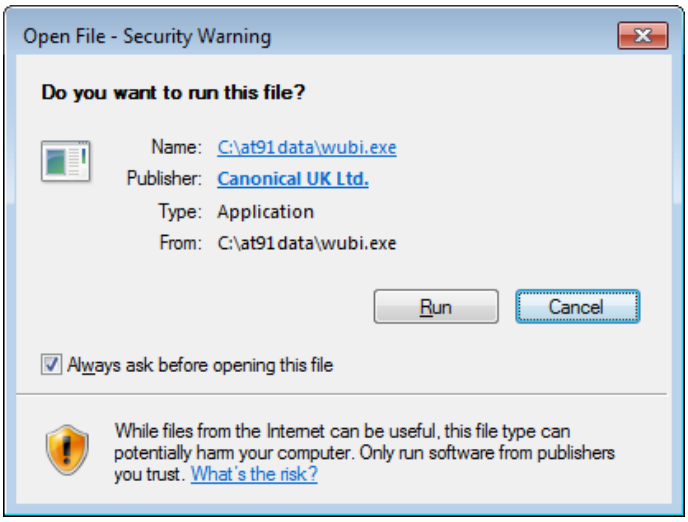
For 64-bit machine, use this link: http://releases.ubuntu.com/12.04/ubuntu-12.04.4-desktop-amd64.iso

Choose this Ubuntu image (ubuntu-12.04.4-desktop-amd64.iso) to take full advantage of computers based on the AMD64 or EM64T architecture (e.g., Athlon64, Opteron, EM64T Xeon, Core 2).

INFO By setting the Ubuntu image into the same folder that wubi.exe, allows a local installation on machine. This prevents to have any installation issue due to firewall, proxy settings…

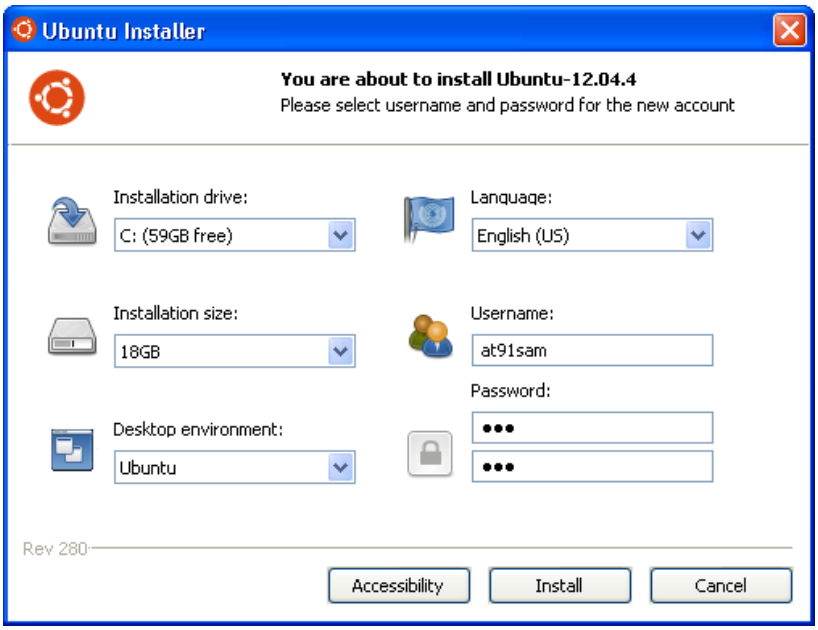
2.2 Install Ubuntu 12.04 inside Windows XP or Win7

TO DO Double click on wubi.exe

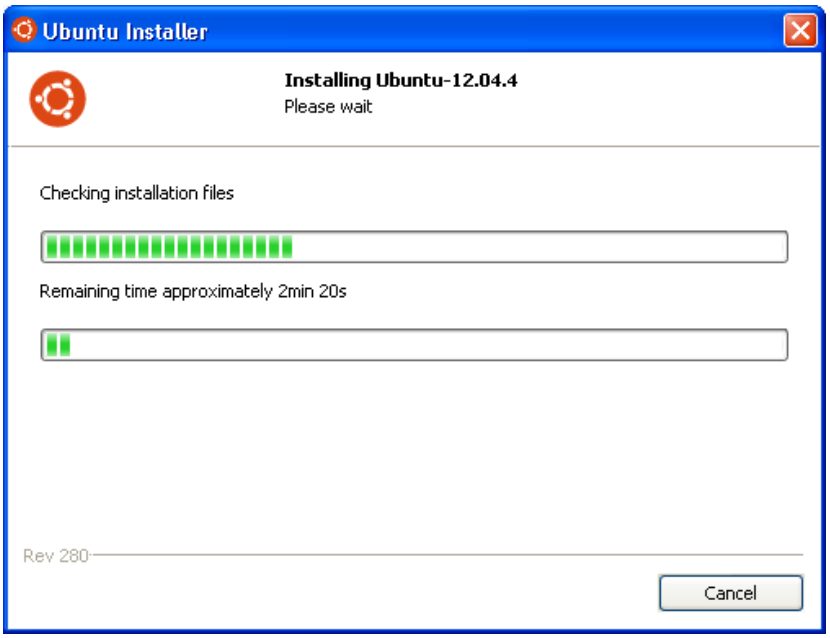


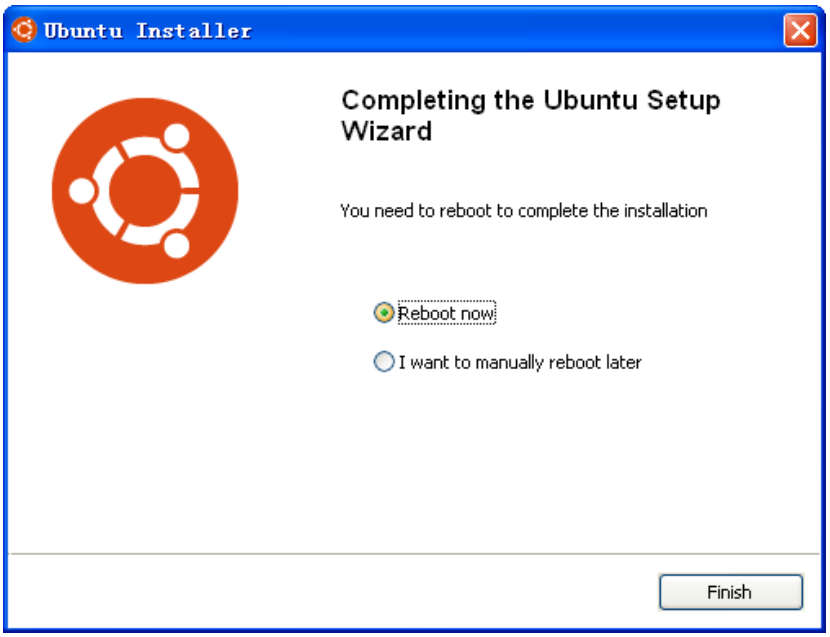
TO DO Set-up Ubuntu 12.04 parameters

- Choose your installation drive which has at least 18Gbytes of free space
- Configure your account
  - Username: at91sam
  - Password: sam

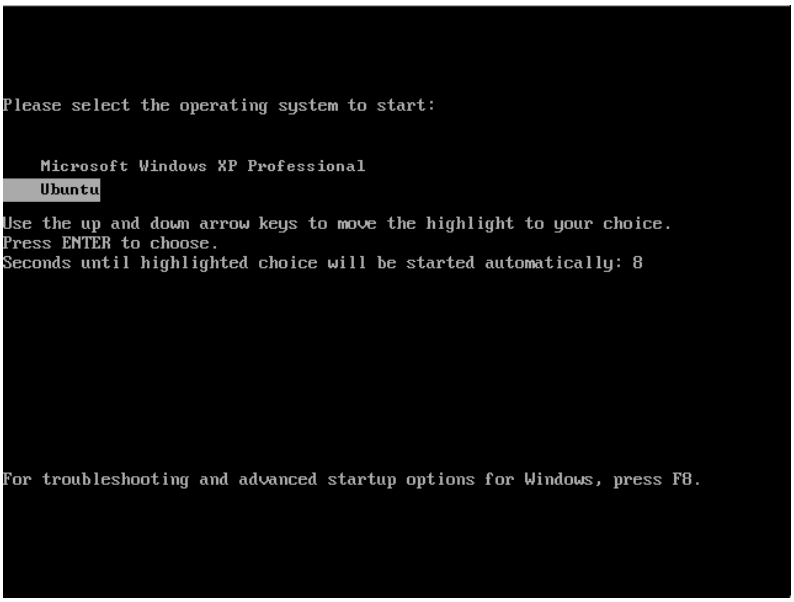


TO DO Check that your parameters are correct, then click Install button start to install





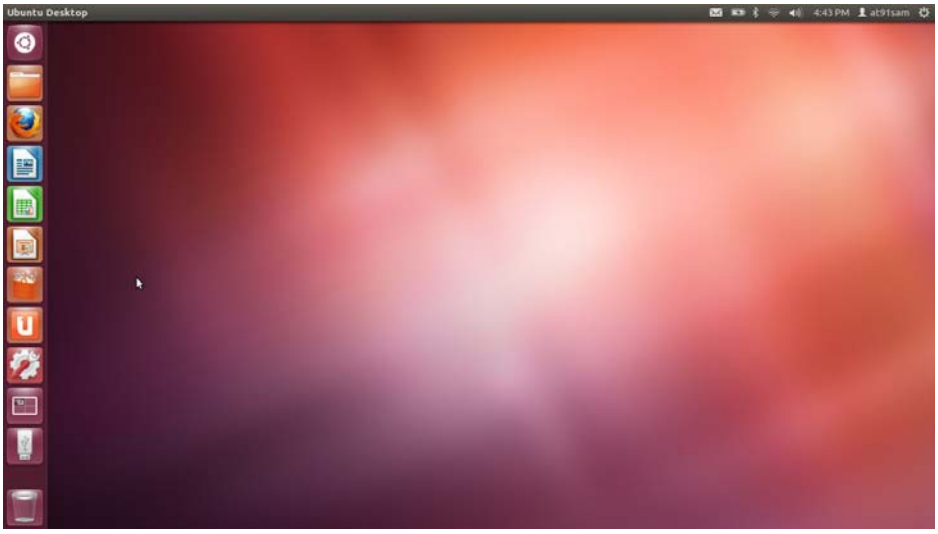
TO DO After reboot, select Ubuntu OS to continue the installation



RESULT Installation will take about 10 minutes, depending on the user's machine

TO DO When the login screen is shown,

- change the language to English, if a different language is used (see the icon on the up right of the screen
- Input your password: `sam` to log in



RESULT Now Ubuntu 12.04 is installed,

WARNING DO NOT updates the Ubuntu distribution to a newer version if you connect to Internet. In our course, we support only this particular 12.04 version. However, applying security and software updates without changing the distribution version is fine.

INFO Here is an introduction to useful Linux commands

- press CTRL+ALT+T to open a terminal
- List all files and directories

```
ls
```

- List all files and directories with details and human readable size information

```
ls -lh
```

- Change the current directory to `/opt`

```
cd /opt/
```

- Change current directory to a specified directory.

```
cd ~    # go to user folder. In this training it means: /home/at91sam
cd .    # go to current directory
cd ..   # go to parent directory
cd -    # go to previous directory
```

- Copy file from source to destination

```
cp src dest
```

- Copy folder/file from source to destination

cp -r src dest

- Move or rename folder/file from source to destination

```
mv src dest
```

- Remove file

```
rm xxx.txt
```

- Remove a directory

```
rm -rf folder
```

- Uncompress a tar ball to current directory

```
tar xvjf xxx.tar.bz2
```

- Uncompress a tar ball to a specified directory

```
tar xvjf xxx.tar.bz2 -C dest-folder
```

- Display the contents of a file

  ```
  cat xxx.txt
  ```

- Edit a file, in a similar way as in Windows with `gedit` tool.

  ```
  gedit xxx.txt &
  ```

3. Software components and tools

Computer requirements:

- Workstation or laptop with Ubuntu 12.04 LTS installed (32 or 64 bit)
- User account with “sudoer” rights

TO DO Open a terminal: Press “CRTL+ATL+T”

TO DO Install following software needed for the training:
vim-gtk: Vi IMproved - enhanced vi editor - with GTK2 GUI (aka gvim)
picocom: minimal dumb-terminal emulation program
lrzsz: Tools for zmodem/xmodem/ymodem file transfer
gitk: fast, scalable, distributed revision control system
meld: graphical tool to diff and merge files
tftpd-hpa: HPA's tftp server (Trivial file transfer protocol)

```
# sudo apt-get update
# sudo apt-get install vim-gtk libncurses5-dev picocom lrzsz gitk meld tftpd-hpa
    [sudo] password for at91sam:
    Reading package lists... Done
    Building dependency tree
    Reading state information... Done
    […]
    Do you want to continue [Y/n]?
    The following NEW packages will be installed:
      gitk lrzsz meld picocom
    […]
    Setting up gitk (1:1.7.9.5-1) ...
    Setting up lrzsz (0.12.21-5) ...
    Setting up meld (1.5.3-1ubuntu1) ...
    Setting up picocom (1.4-3) ...
```

3.1 Working directory setup

TO DO Create your working folder “ `at91data`” in your home directory:

```
# mkdir ~/at91data
```

TO DO All the training material needs to be available on `at91data` local folder. So you need to download the required contents to run correctly the different hands-on

```
# cd ~/at91data
# wget ftp://www.at91.com/pub/Trainings/at91data/*
# mkdir ipk
# cd ipk
# wget ftp://www.at91.com/pub/Trainings/at91data/ipk/*
```

RESULT You should have the following contents on `at91data` and `ipk` folders

```
training@ubuntu:~$ ls -al at91data
total 244232
drwxrwxr-x  3 training training      4096 May 14 09:40 .
drwxr-xr-x 35 training training      4096 May 14 09:43 ..
-rw-r--r--  1 training training     21016 Apr  9 16:11 greenLED.png
-rw-r--r--  1 training training   6001292 May 14 09:25 home-automation-1.5.tar.gz
drwxrwxr-x  2 training training      4096 May 14 09:40 ipk
-rw-r--r--  1 training training 111350389 May 14 09:26 linux.tar.gz
-rw-r--r--  1 training training  64093032 Apr  4 16:17 qt-creator-linux-x86_64-opensource-2.8.1.run
-rw-r--r--  1 training training  65125377 Apr  4 16:17 qt-creator-linux-x86-opensource-2.8.1.run
-rw-r--r--  1 training training     20569 Apr  9 16:11 redLED.png
-rw-r--r--  1 training training   3439320 Apr 18 14:21 zImage
training@ubuntu:~$ ls -al at91data/ipk/
total 17356
drwxrwxr-x 2 training training     4096 May 14 09:40 .
drwxrwxr-x 3 training training     4096 May 14 09:40 ..
-rw-r--r-- 1 training training  1303586 Apr  4 16:17 binutils_2.23.2-r4_cortexa5hf-vfp.ipk
-rw-r--r-- 1 training training     1878 Apr  4 16:17 binutils-symlinks_2.23.2-r4_cortexa5hf-vfp.ipk
-rw-r--r-- 1 training training     3364 Apr  4 16:17 devmem2_1.0-r7_cortexa5hf-vfp.ipk
-rw-r--r-- 1 training training    46552 Apr  4 16:17 eglibc-extra-nss_2.18-r0_cortexa5hf-vfp.ipk
-rw-r--r-- 1 training training    12772 Apr  4 16:17 fb-test_0.0-r0_cortexa5hf-vfp.ipk
-rw-r--r-- 1 training training 15116806 Apr  4 16:18 gcc_4.8.1-r0_cortexa5hf-vfp.ipk
-rw-r--r-- 1 training training     1568 Apr  4 16:17 gcc-symlinks_4.8.1-r0_cortexa5hf-vfp.ipk
-rw-r--r-- 1 training training   523506 Apr  4 16:18 libc6-dev_2.18-r0_cortexa5hf-vfp.ipk
-rw-r--r-- 1 training training    56460 Apr  4 16:18 libcidn1_2.18-r0_cortexa5hf-vfp.ipk
-rw-r--r-- 1 training training   463662 Apr  4 16:18 libgcc-s-dev_4.8.1-r0_cortexa5hf-vfp.ipk
-rw-r--r-- 1 training training    34788 Apr  4 16:18 libmpc3_1.0.1-r0_cortexa5hf-vfp.ipk
-rw-r--r-- 1 training training   165638 Apr  4 16:18 libmpfr4_3.1.2-r0_cortexa5hf-vfp.ipk
-rw-r--r-- 1 training training     5908 Apr  4 16:18 python-smbus_3.0.3-r0_cortexa5hf-vfp.ipk
training@ubuntu:~$
```

TO DO Download the different SAMA5D3 Linux demos

```
# cd ~/at91data
# wget ftp://www.at91.com/pub/demo/linux4sam_4.3/linux4sam-poky-sama5d3_xplained-4.3.zip
# wget ftp://www.at91.com/pub/demo/linux4sam_4.4/linux4sam-poky-qte-sama5d3_xplained-4.4.zip
```

3.2 Download QtCreator

Right now you need to download the Qt Creator tool as all the required settings and tools are available thanks to the Atmel Qt SDK. Installer has to be executed with super user rights. We will install qt-creator in "/opt" directory during the Qt getting started hands-on.

INFO You are able to check your machine architecture using the following command

```
# uname –-machine
x86_64 → "64" means you are using a 64 bit distribution
```

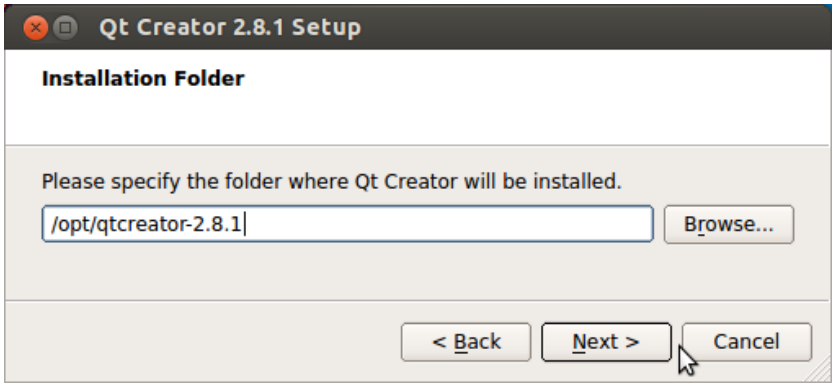
WARNING For Ubuntu 64 bit installation

```
# cd ~/at91data
# wget http://download.qt-project.org/official_releases/qtcreator/2.8/2.8.1/qt-creator-linux-x86_64-opensource-2.8.1.run
# chmod a+x ./qt-creator-linux-x86_64-opensource-2.8.1.run
# sudo ./qt-creator-linux-x86_64-opensource-2.8.1.run
```

WARNING For Ubuntu 32 bit installation

```
# cd ~/at91data
# wget http://download.qt-project.org/official_releases/qtcreator/2.8/2.8.1/qt-creator-linux-x86-opensource-2.8.1.run
# chmod a+x ./qt-creator-linux-x86-opensource-2.8.1.run
# sudo ./qt-creator-linux-x86-opensource-2.8.1.run
```

- Set the installation folder to `:/opt/qtcreator-2.8.1/`



RESULT Qt Creator is installed but not yet configured for SAMA5D3 application.

3.3 Install Yocto SDK on Ubuntu 12.04

TO DO Download and Install Atmel Yocto QTE SDK.

- This SDK has been built with Yocto-1.5.1 (http://www.yoctoproject.org/). It includes ARM linux gcc, ARM linux gdb and also cross complied libraries.
  This SDK is available on Linux4SAM ftp server for 32 and 64 bit Ubuntu installation:
  ftp://ftp.linux4sam.org/pub/demo/yocto-qte-sdk/

WARNING For Ubuntu 64 bit installation

```
# cd ~/at91data
# wget ftp://ftp.linux4sam.org/pub/demo/yocto-qte-sdk/poky-eglibc-x86_64-meta-toolchain-qte-cortexa5hf-vfp-toolchain-qte-1.5.1.sh
# chmod a+x ./poky-eglibc-x86_64-meta-toolchain-qte-cortexa5hf-vfp-toolchain-qte-1.5.1.sh
# ./poky-eglibc-x86_64-meta-toolchain-qte-cortexa5hf-vfp-toolchain-qte-1.5.1.sh
Enter target directory for SDK (default: /opt/poky/1.5.1):
You are about to install the SDK to "/opt/poky/1.5.1". Proceed[Y/n]?
[sudo] password for at91sam:
Setting it up...
SDK has been successfully set up and is ready to be used.
```

WARNING For Ubuntu 32 bit installation

```
# cd ~/at91data
# wget ftp://ftp.linux4sam.org/pub/demo/yocto-qte-sdk/poky-eglibc-i686-meta-toolchain-qte-cortexa5hf-vfp-toolchain-qte-1.5.1.sh
# chmod a+x ./poky-eglibc-i686-meta-toolchain-qte-cortexa5hf-vfp-toolchain-qte-1.5.1.sh
# ./poky-eglibc-i686-meta-toolchain-qte-cortexa5hf-vfp-toolchain-qte-1.5.1.sh
Enter target directory for SDK (default: /opt/poky/1.5.1):
You are about to install the SDK to "/opt/poky/1.5.1". Proceed[Y/n]?
[sudo] password for at91sam:
Extracting SDK...done
SDK has been successfully set up and is ready to be used.
```

3.4 Serial device access Setup

With Ubuntu distributions, a user has to be member of the `dialout` group to access serial devices like RS232 serial port, usb serial, serial modem, etc.

As SAM-BA uses the USB serial class driver to connect to the SAMA5D3 Xplained board, it is important to enable serial devices access to the Ubuntu user.

TO DO Enable access to serial devices

- Add current "user" to `dialout` group:

```
# cd ~
# sudo adduser at91sam dialout
# cat /etc/group | grep dialout
dialout:x:20:at91sam
```

- **Log out from Ubuntu** and log in again to make the changes effective and access the serial port.

WARNING Logging out from Ubuntu is mandatory to have the user added to the `dialout` group.

RESULT Current user has now access to serial devices including the USB serial port.

3.5 Install SAM-BA on Ubuntu 12.04

TO DO Download SAM-BA for linux archives.
SAM-BA is also available on atmel.com:
http://www.atmel.com/tools/ATMELSAM-BAIN-SYSTEMPROGRAMMER.aspx

```
# cd ~/at91data
# wget ftp://ftp.linux4sam.org/pub/sam-ba/sam-ba_2.14_cdc_linux.zip
# cd ~
```

WARNING For Ubuntu 64 bit only. SAM-BA requires 32 bit libraries

```
# uname –-machine
x86_64 → "64" means you are using a 64 bit distribution
# sudo apt-get install ia32-libs
[sudo] password for at91sam: → type your password (sam)
```

INFO User is `at91sam` for that hands-on with `sam` as password. Use your login/password whenever required if different.

TO DO Uncompress SAM-BA archive.

```
# cd /opt
# sudo unzip ~/at91data/sam-ba_2.14_cdc_linux.zip
```

- Add SAM-BA directory to PATH variable:

```
# export PATH=$PATH:/opt/sam-ba_cdc_linux
```

- Edit `.bashrc` file to permanently add SAM-BA directory to the PATH variable:

```
# gedit ~/.bashrc
```

- Add at the end of the file the following line to update the PATH:

```
# export PATH=$PATH:/opt/sam-ba_cdc_linux
```

- Activate the `.bashrc` file with the updated PATH using the source command:

```
# source ~/.bashrc
```

RESULT SAM-BA for Linux has been successfully installed.

4. Conclusion

You have successfully setup your computer for SAMA5D3 Linux training.
Following configuration should be up and running on your working station:

- Ubuntu 12.04 LST
- Ubuntu software packages:
    - vim-gtk: Vi IMproved - enhanced vi editor - with GTK2 GUI (aka gvim)
    - picocom: minimal dumb-terminal emulation program
    - lrzsz: Tools for zmodem/xmodem/ymodem file transfer
    - gitk: fast, scalable, distributed revision control system
    - meld: graphical tool to diff and merge files
    - tftpd-hpa: HPA's tftp server (Trivial file transfer protocol)
    - ia32-libs for 64-bit machine only
- SAM-BA 2.14.
- Atmel Yocto QTE SDK.
- Qt Creator is installed but not yet configured.
- ~/at91data folder contents all the required software components

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370

Using SAM-BA for Linux on SAMA5D3 Xplained

AN-8995

Prerequisites

- **Hardware Prerequisites**
  - Atmel® SAMA5D3 Xplained
  - USB serial TTL adapter (**optional**)
    - FTDI TTL-232R-3V3 USB to TTL serial cable
- **Software Prerequisites**
  - Linux® Ubuntu® 12.04 LTS
- **Estimated completion time**: 15min

Introduction

The goal of this hands-on is to:

- Get Started on SAM-BA® for Linux, the Atmel In System Programming tool for Atmel ARM®-based MPUs
- Use SAM-BA Graphical UI to connect to the SAMA5D3 Xplained board
- Use SAM-BA Command Line mode to program a SAMA5D3 Linux demo

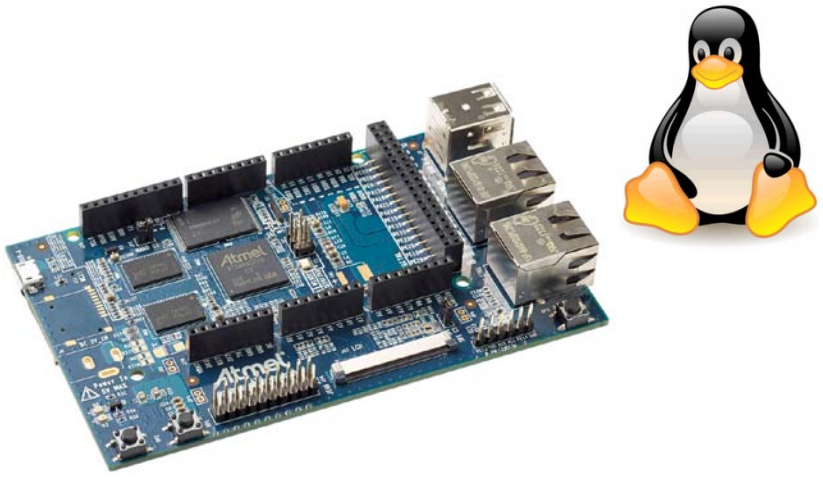


Table of Contents

Icon Key Identifiers

INFO	Delivers contextual information about a specific topic.
TIPS	Highlights useful tips and techniques.
TO DO	Highlights objectives to be completed.
RESULT	Highlights the expected result of an assignment step.
WARNING	Indicates important information.
EXECUTE	Highlights actions to be executed out of the target when necessary.

1. Introduction to SAM-BA

The SAM-BA GUI tool provides a means of easily programming various Atmel ARM processor-based microcontrollers.

SAM-BA GUI Key Features:

- Performs in-system programming through JTAG, RS232 or USB interfaces
    - On-device and on-board memories programming solutions
- May be used via a Graphical User Interface or started in batch mode from a command line
- Memories and peripherals display content
- User scripts executable from SAM-BA GUI or a shell
- Runs under Windows® or Linux OS

WARNING SAM-BA for Linux supports only the USB interface.

INFO SAM-BA and its patches are available from Atmel web site at the following address: http://www.atmel.com/tools/ATMELSAM-BAIN-SYSTEMPROGRAMMER.aspx.

SAM-BA has been validated on different Linux distributions such as Ubuntu, Debian®, Fedora®, openSUSE® but it will certainly work on other distributions as well.

Tested distributions are referenced on Linux4SAM web site, which is the main starting point for Linux OS on the Atmel ARM-based MPUs, at the following link:
http://www.at91.com/linux4sam/bin/view/Linux4SAM/SoftwareTools#Tested_Linux_distributions.

INFO Ubuntu 12.04 LTS will be used as the Linux distribution reference for the next assignments.

2. Assignment 1: Install SAM-BA on Ubuntu 12.04 LTS

2.1 Serial device access Setup

With Ubuntu distributions, a user has to be member of the `dialout` group to access serial devices like RS232 serial port, usb serial, serial modem, etc.

As SAM-BA uses the USB serial class driver to connect to the SAMA5D3 Xplained board, it is important to enable serial devices access to the Ubuntu user.

TO DO Enable access to serial devices

- Add current "user" to `dialout` group:

```
# cd ~
# sudo adduser at91sam dialout
# cat /etc/group | grep dialout
dialout:x:20:at91sam
```

- **Log out from Ubuntu** and log in again to make the changes effective and access the serial port.

WARNING Logging out from Ubuntu is mandatory to have the user added to the `dialout` group.

RESULT Current user has now access to serial devices including the USB serial port.

2.2 Install SAM-BA on Ubuntu 12.04

TO DO Download SAM-BA for linux archives.
SAM-BA is also available on atmel.com:
http://www.atmel.com/tools/ATMELSAM-BAIN-SYSTEMPROGRAMMER.aspx

```
# cd ~/at91data
# wget ftp://ftp.linux4sam.org/pub/sam-ba/sam-ba_2.14_cdc_linux.zip
# cd ~
```

WARNING For Ubuntu 64 bit only. SAM-BA requires 32 bit libraries

```
# uname –-machine
x86_64 → "64" means you are using a 64 bit distribution
# sudo apt-get install ia32-libs
[sudo] password for at91sam: → type your password (sam)
```

INFO User is `at91sam` for that hands-on with `sam` as password. Use your login/password whenever required if different.

TO DO Uncompress SAM-BA archive.

```
# cd /opt
# sudo unzip ~/at91data/sam-ba_2.14_cdc_linux.zip
```

- Add SAM-BA directory to PATH variable:

```
# export PATH=$PATH:/opt/sam-ba_cdc_linux
```

- Edit `.bashrc` file to permanently add SAM-BA directory to the PATH variable:

```
# gedit ~/.bashrc
```

- Add at the end of the file the following line to update the PATH:

```
# export PATH=$PATH:/opt/sam-ba_cdc_linux
```

- Activate the `.bashrc` file with the updated PATH using the source command:

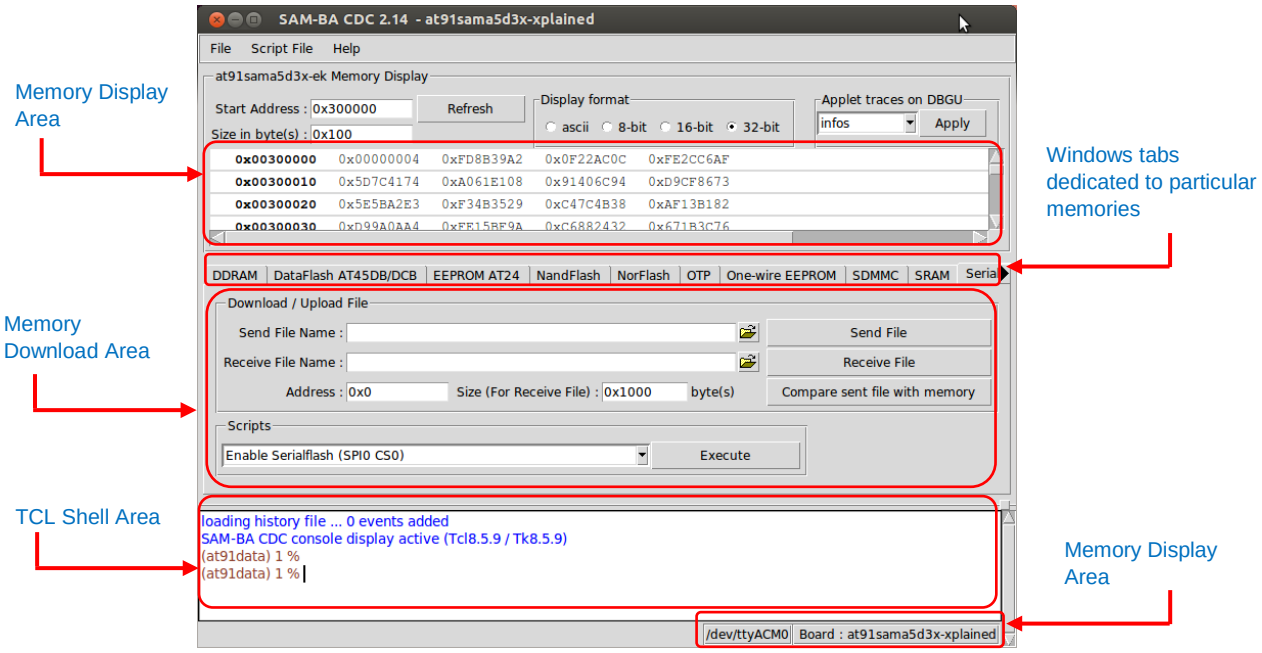
```
# source ~/.bashrc
```

RESULT SAM-BA for Linux has been successfully installed.

3. Assignment 2: Connect SAM-BA GUI to SAMA5D3 Xplained

3.1 Understand SAM-BA GUI Main Window

SAM-BA GUI Main window is composed of three different areas as described below:



- The Memory Display area is used to dump the internal memories as well as the external RAM-based memories
- The Memory Download area is used to program the different on-board and on-board memories. There is one memory tab per supported memory such as:
  - EEPROM
  - DataFlash
  - Serial Flash
  - NAND Flash
  - DDR2
- The TCL Shell area which can be used to execute TCL commands as well as displaying information about the on-going connection/transfers

3.2 Connect to the Board

To connect the SAMA5D3 Xplained to SAM-BA UI using USB serial port, SAMA5D3 must execute a monitor called SAM-BA Monitor (aka SAM-BA Boot) which is part of the Boot ROM (Standard Boot mode sequence).

This monitor is executed only if no valid code is found from the external non-volatile memories supported by the Boot ROM.

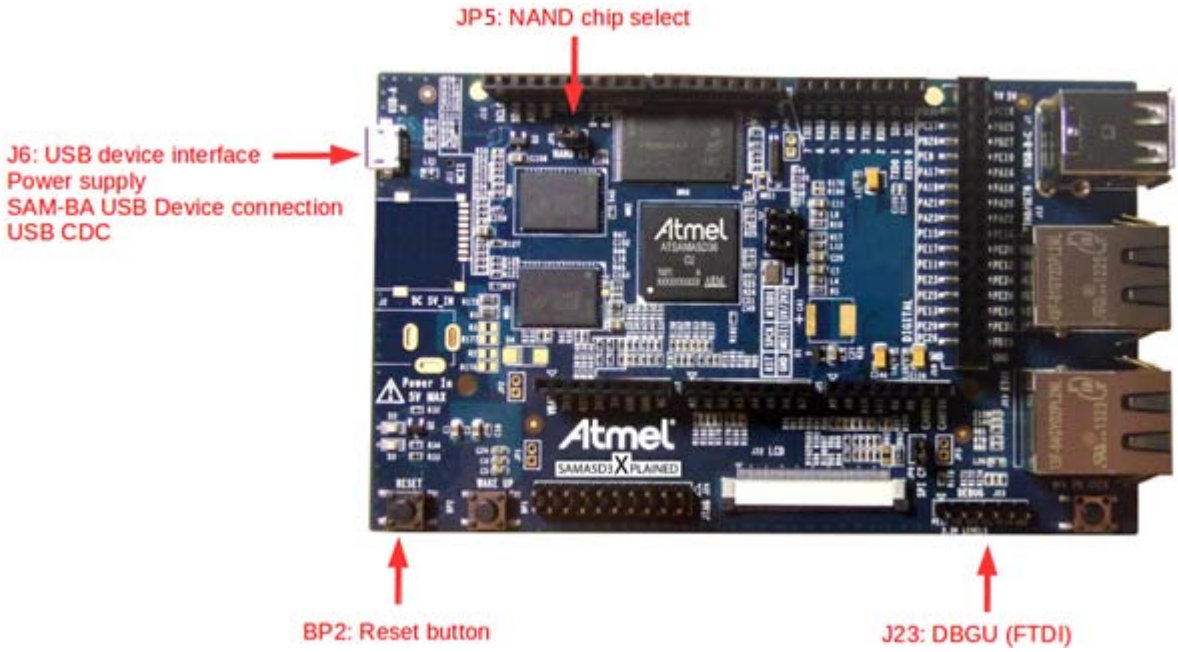
INFO More information on the Standard Boot mode can be found in the section called “Standard Boot Strategies” from SAMA5D3 datasheet.

The goal is, in a first step, to disable NAND Flash access in case a valid code is programmed in NAND which would prevent SAM-BA Monitor execution.

Then, SAM-BA UI will be able to connect to the board.

TO DO Run SAM-BA Monitor.

- Open JP5 to disable NAND Flash memory access
- Plug the USB micro cable between J6 and your computer to boot from the on-chip Boot ROM
- Close JP5 to enable NAND Flash memory access

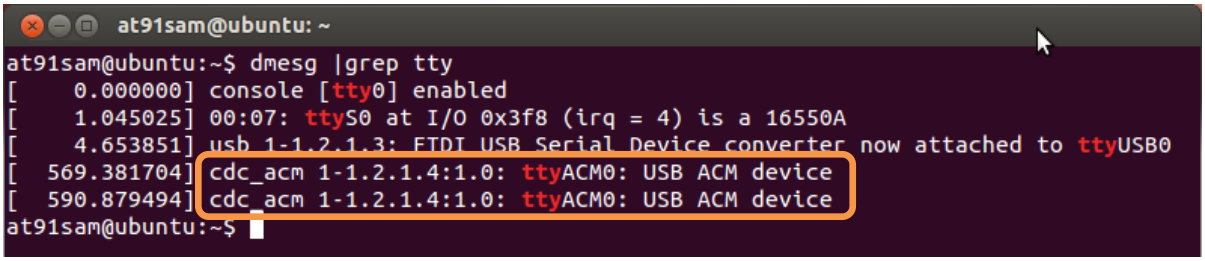


RESULT SAM-BA (Boot) Monitor is executed.

TO DO Connect the SAMA5D3 Xplained to SAM-BA.

- Identify the USB connection by monitoring the last lines of `dmesg` command using `tail` command:
  - `/dev/ttyACMx` number will be used to configure the terminal emulator

```
# dmesg | grep tty
```

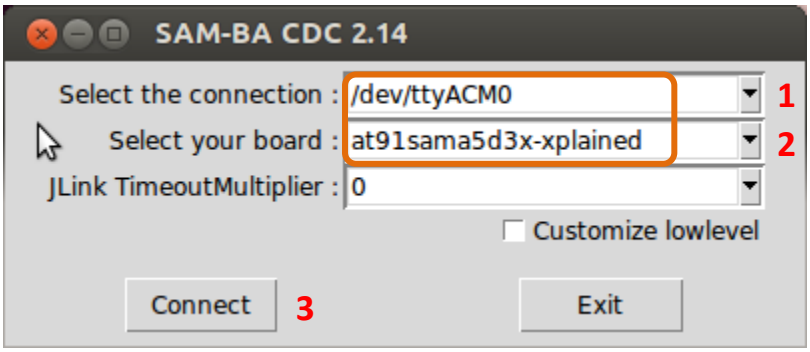


- Run SAM-BA application:

```
# /opt/sam-ba_cdc_linux/sam-ba
```

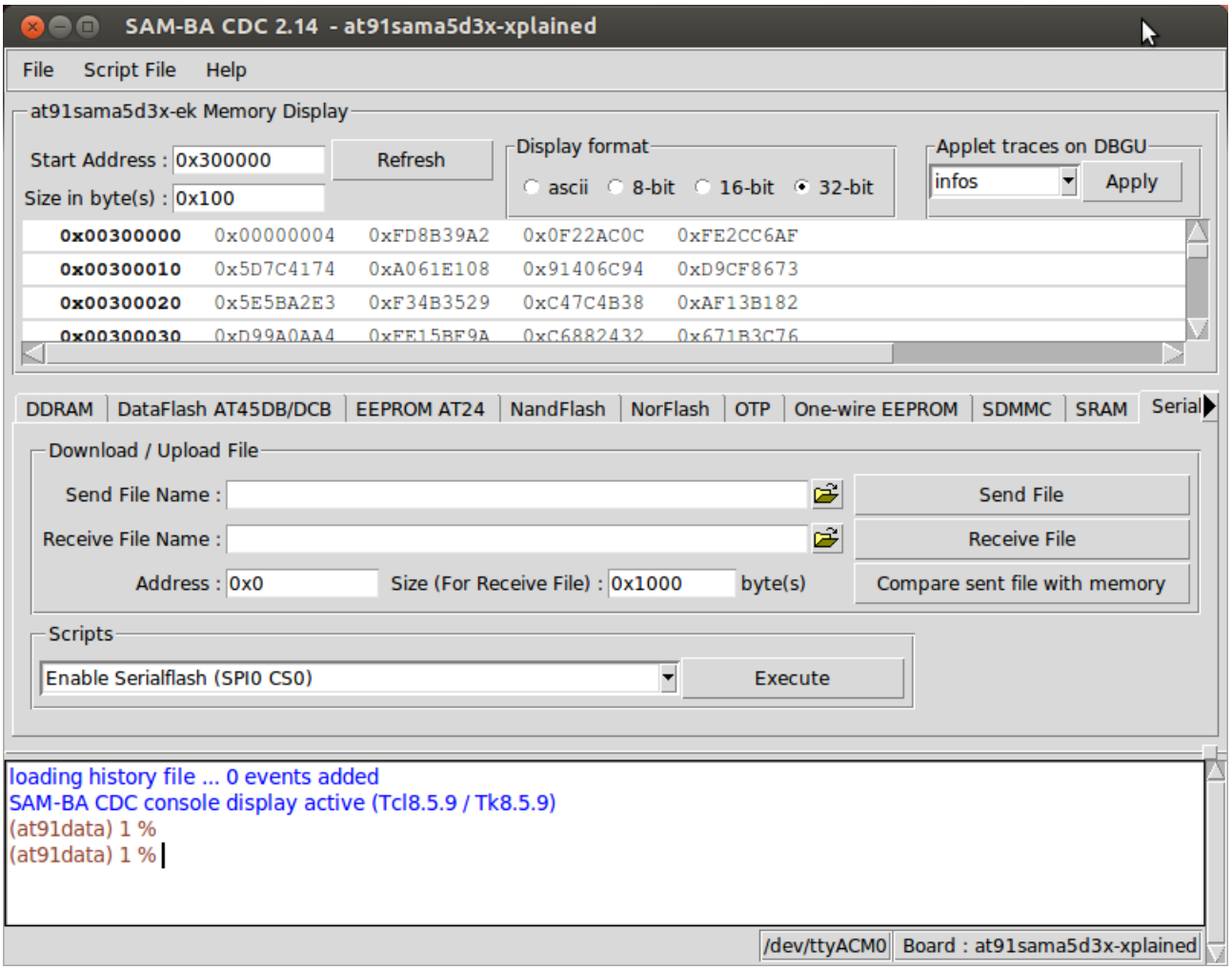
INFO Typing directly `sam-ba` should also work as its path has been added to the PATH variable.

The following pop-up should appear:



- Select **`/dev/ttyACMx`** as connection
- Select **at91sama5d3x-ek** as board
- Click on `Connect`

RESULT SAM-BA Main window should appear which proves the successful connection to the SAMA5D3 Xplained board:



4. Assignment 3: Use SAM-BA in Command Line Mode to Program a Demo

The goal of this assignment is to program one of the SAMA5D3 demos available on Linux4SAM ftp site using SAM-BA in command line mode.

SAM-BA Command Line mode uses TCL scripts to program the different SAMA5D3 Xplained on-board memories without the need to launch a graphical interface.

INFO Linux4SAM.org demos are delivered as archives containing all the binaries that must be flashed on the board.

Pre-built demo binaries are available on Linux4SAM ftp server: ftp://ftp.linux4sam.org/pub/demo/.

TO DO Get SAMA5D3 Demo Archive and Identify the right demo script.

- Close SAM-BA
- Download demo archive:

```
# cd ~/at91data
# wget ftp://ftp.linux4sam.com/pub/demo/linux4sam_4.4/linux4sam-poky-sama5d3_xplained-4.4.zip
# unzip linux4sam-poky-sama5d3_xplained-4.4.zip
```

- Once uncompressed, look at demo content:

```
# cd ~/at91data/linux4sam-poky-sama5d3_xplained-4.4
# ls
```

Different OS-based scripts are available:

- `.bat` scripts are to be used with Windows
- `.sh` scripts are to be used with Linux

There is one OS-based script for each board configuration:

- SAMA5D3 Xplained without any display: `demo_linux_nandflash`
- SAMA5D3 Xplained with PDA 4.3'' capacitive display: `demo_linux_nandflash_pda4`
- SAMA5D3 Xplained with PDA 7'' capacitive display: `demo_linux_nandflash_pda7`

Each Windows (.bat) or Linux (.sh) script executes a single TCL script (.tcl) that is responsible for programming the NVM memory with the different Linux binary components (kernel, device tree, root file system…).

INFO We will use the SAMA5D3 Xplained without any display extension board.

RESULT demo_linux_nandflash.sh is the script to execute in order to program the demo on the SAMA5D3 Xplained without any display support:

```
at91-sama5d3_xplained.dtb                          → regular xplained dtb file (no LCD)
at91-sama5d3_xplained_pda4.dtb                     → xplained + LCD pda 4.3" dtb file
at91-sama5d3_xplained_pda7.dtb                     → xplained + LCD pda 7" dtb file

sama5d3_xplained-nandflashboot-uboot-3.6.1.bin     → u-boot 2013.07
u-boot-sama5d3_xplained-v2013.07-at91-r1.bin       → at91bootstrap (nandflash boot)
zImage-sama5d3_xplained.bin                        → kernel zImage (Linux 3.10-at91)
atmel-xplained-demo-image-sama5d3_xplained.ubi     → root file system (UBIFS)

demo_linux_nandflash.bat                           → Windows SAM-BA scripts
demo_linux_nandflash_pda4.bat                        one for each board variant
demo_linux_nandflash_pda7.bat

demo_linux_nandflash.sh                            → Linux SAM-BA scripts
demo_linux_nandflash_pda4.sh                         one for each board variant
demo_linux_nandflash_pda7.sh

demo_linux_nandflash.tcl                           → TCL SAM-BA scripts for configuration
demo_linux_nandflash_pda4.tcl                        one for each board variant
demo_linux_nandflash_pda7.tcl
demo_script_linux_nandflash.tcl                    → main TCL SAM-BA script
```

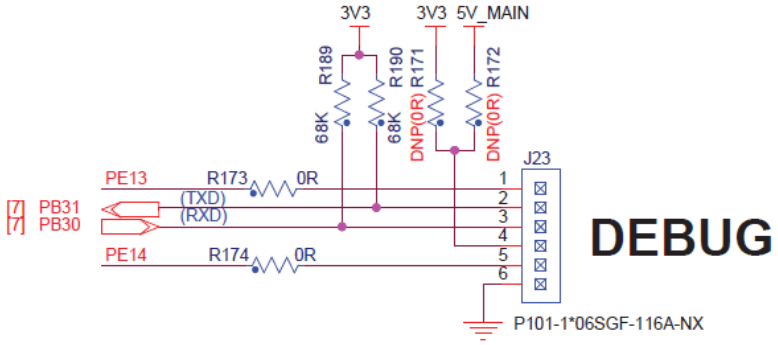
TO DO Monitor Demo Programming Progress using SAMA5D3 DEBUG port.

WARNING This section is optional as you need to have a USB serial TTL adapter to do it.

SAM-BA programming operations can be monitored by tracing the SAMA5D3 DBGU serial port peripheral.

The SAMA5D3 Xplained board has a dedicated serial port for debugging, which is accessible through the 6-pin male header J23.

Various interfaces can be used as USB/Serial DBGU port bridge, such as a FTDI TTL-232R-3V3 USB to TTL serial cable or a basic breakout board for the 232/USB converter.



INFO We will use a USB serial TTL adapter to connect the SAMA5D3 Xplained board to the computer.

- Plug the USB serial TTL adapter to your **computer**

INFO The USB-serial driver is already installed on a Linux OS and should appear as a `/dev/ttyUSBx` device when plugged.

- Identify the USB connection by monitoring the last lines of `dmesg` command using `tail` command:
  - `/dev/ttyUSBx` number will be used to configure the terminal emulator

```
# dmesg | grep tty
```

```
at91sam@ubuntu: ~/at91data/linux4sam-poky-sama5d3_xplained-4.3
at91sam@ubuntu:~/at91data/linux4sam-poky-sama5d3_xplained-4.3$ dmesg | grep tty
[    0.000000] console [tty0] enabled
[    4.653851] usb 1-1.2.1.3: FTDI USB Serial Device converter now attached to ttyUSB0
[  590.879494] cdc_acm 1-1.2.1.4:1.0: ttyACM0: USB ACM device
at91sam@ubuntu:~/at91data/linux4sam-poky-sama5d3_xplained-4.3$
```

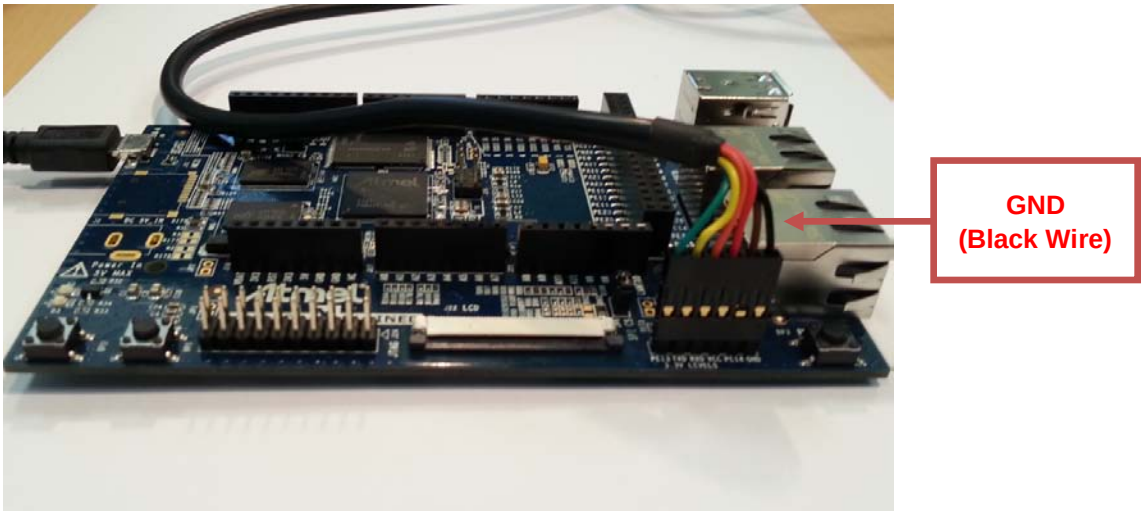
- Open **another** terminal (CRTL+ATL+T) that will be dedicated to trace the SAMA5D3 DBGU serial port

- Open a serial terminal using `picocom` program with the required settings (115200 bauds, 8-N-1 with no hardware flow control):

```
# picocom -b 115200 /dev/ttyUSB0
```

```
at91sam@ubuntu: ~
at91sam@ubuntu:~$ picocom -b 115200 /dev/ttyUSB0
picocom v1.4

port is        : /dev/ttyUSB0
flowcontrol    : none
baudrate is    : 115200
parity is      : none
databits are   : 8
escape is      : C-a
noinit is      : no
noreset is     : no
nolock is      : no
send_cmd is    : ascii_xfr -s -v -l10
receive_cmd is : rz -vv

Terminal ready
```

- Plug the 6-pin female connector of your cable on the J23 male from SAMA5D3 Xplained:
    - The ground pin indicated on the silkscreen "GND" has to be connected with the black wire of the FTDI cable:



RESULT Demo programming progress can be now monitored on the SAMA5D3 DBGU serial port.

It is also now possible to trace the different messages sent to the SAMA5D3 DBGU serial port by the Boot ROM and other applications such as u-boot, Linux…

TO DO Program the Demo.

- If the /dev/ttyACMx previously chosen for SAM-BA is different from /dev/ttyACM0, edit the demo_linux_nandflash.sh file and modify /dev/ttyACMx device number accordingly. Change the board name, replace at91sama5d3x-ek by at91sama5d3x-xplained:

```
# gedit demo_linux_nandflash.sh
```

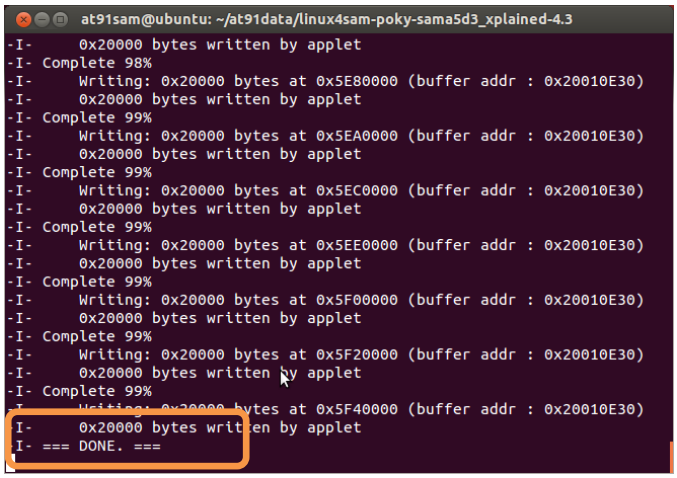
- Add “execution” rights to the script:

```
# chmod a+x ./demo_linux_nandflash.sh
```

- Execute the script to start demo programming:

```
#./demo_linux_nandflash.sh
```

- When the logfile.log appears on the terminal (this will take a few minutes), check that === DONE. === is written at the end of the file:



INFO The demo that is programmed is based on Yocto™ 1.5.

TIPS Monitor Demo Programming Progress using SAM-BA log file.

It is possible to follow the progress of the demo programming, in case the user does not have a serial DBGU connection available to trace SAM-BA operations, by displaying the content of SAM-BA log file:

- Open **another** terminal (CRTL+ATL+T) that will be dedicated to display SAM-BA log file content

- Type the following commands to monitor SAM-BA log file:

```
# cd ~/at91data/linux4sam-poky-sama5d3_xplained-4.4
# tail –f logfile.log
```

RESULT The demo is programmed.

TO DO Run the Demo.

- Press BP2 reset button to boot from the NAND Flash memory and run the demo

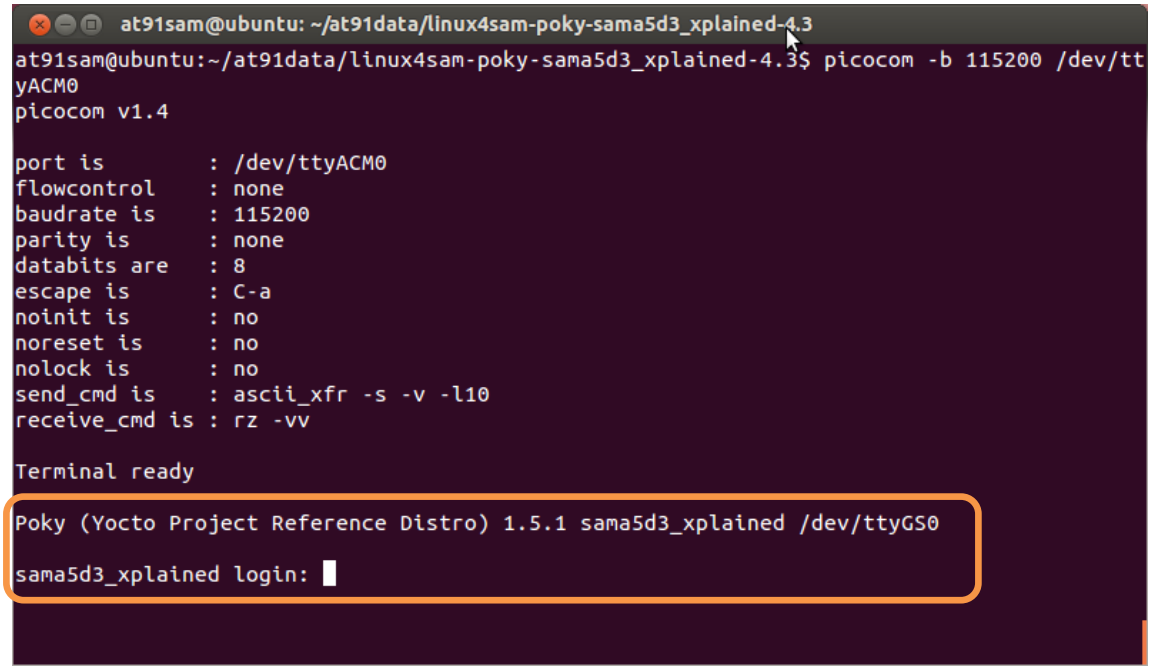
INFO If you do not have any serial DBGU debug connection (using for example a USB serial TTL adapter), you can get Linux Kernel debug information by opening a serial terminal on the USB connection (ttyACMx):

- Open a serial terminal using `picocom` program with the required settings (115200 bauds, 8-N-1 with no hardware flow control):

```
# picocom -b 115200 /dev/ttyACM0
```

RESULT The demo has booted successfully.

You should see the Linux `sama5d3_xplained_login` prompt:



5. Conclusion

The hands-on taught you how to install and use SAM-BA to program one of the SAMA5D3 demos which is available on Linux4SAM ftp site.

Same processes can be followed to program any other Linux demos hosted on Linux4SAM.org web site.

6. Revision History

Doc. Rev.	Date	Comments
42328A	06/2014	Initial document release

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370

Prerequisites

- **Hardware Prerequisites**
  - SAMA5D3 Xplained board
  - Ethernet cable
  - Micro USB to USB-A cable
  - USB FTDI cable
    - FTDI TTL-232R-3V3 USB to TTL serial cable
- **Software Prerequisites**
  - Ubuntu 12.04 LTS
  - Serial terminal emulator (picocom, minicom, putty…)
- **Estimated completion time**: 45 min

Introduction

U-Boot utility is an open-source, universal boot-loader with comprehensive support for loading and managing boot images, such as the Linux kernel.
The goal of this hands-on is to:

- Configure a TFTP server,
- Set up U-Boot network configuration,
- Load a kernel through TFTP.

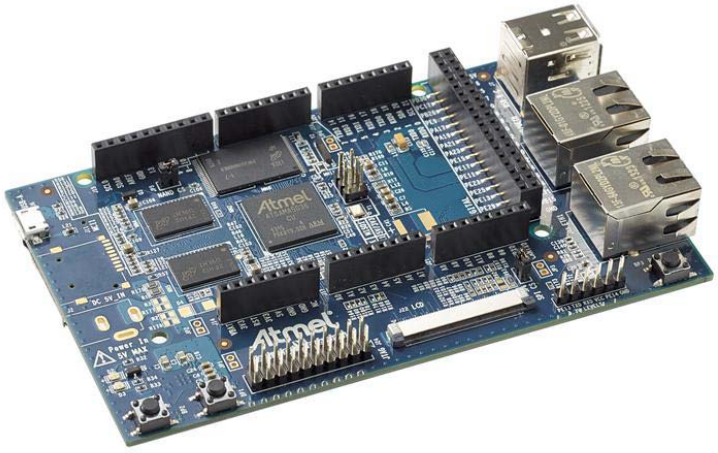


Table of Contents

Icon Key Identifiers

INFO	Delivers contextual information about a specific topic
TIPS	Highlights useful tips and techniques
TO DO	Highlights objectives to be completed on the Linux computer
RESULT	Highlights the expected result of an assignment step
WARNING	Indicates important information
EXECUTE	Highlights actions to be executed out of the SAMA5D3 Xplained board

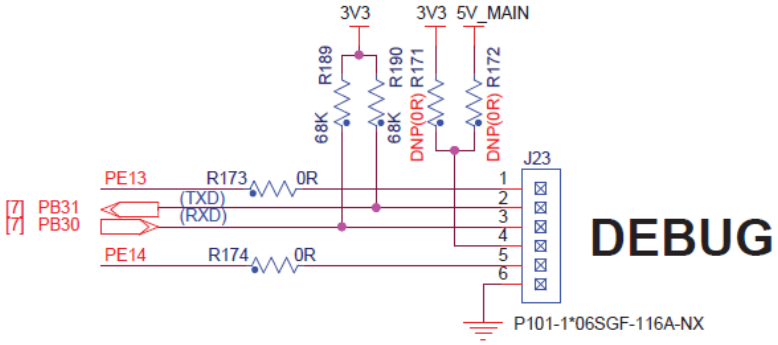
1. Getting started with SAMA5D3 Xplained demo

TO DO Setup USB serial TTL adapter

SAM-BA programming operations can be monitored by tracing the SAMA5D3 DBGU serial port peripheral.

The SAMA5D3 Xplained board has a dedicated serial port for debugging, which is accessible through the 6-pin male header J23.

Various interfaces can be used as USB/Serial DBGU port bridge, such as a FTDI TTL-232R-3V3 USB to TTL serial cable or a basic breakout board for the 232/USB converter.

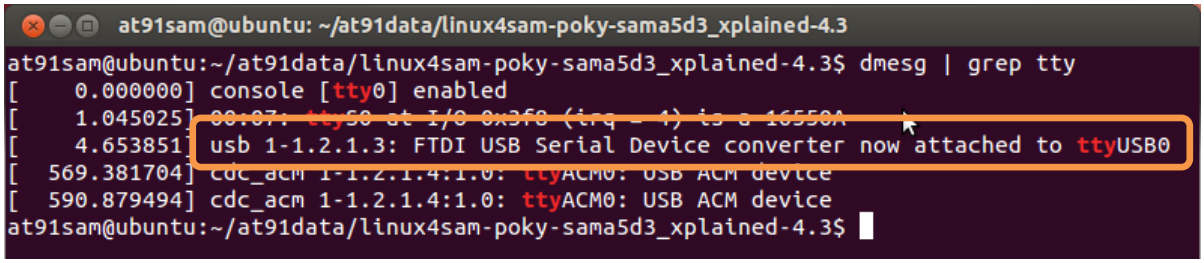


TO DO Plug the USB serial TTL adapter to your **computer**.

INFO The USB-serial driver is already installed on a Linux OS and should appear as a `/dev/ttyUSBx` device when plugged.

- Identify the USB connection by monitoring the last lines of `dmesg` command:
  - `/dev/ttyUSBx` number will be used to configure the terminal emulator.

```
# dmesg | grep tty
```



TO DO Connect to the SAMA5D3 Xplained demo

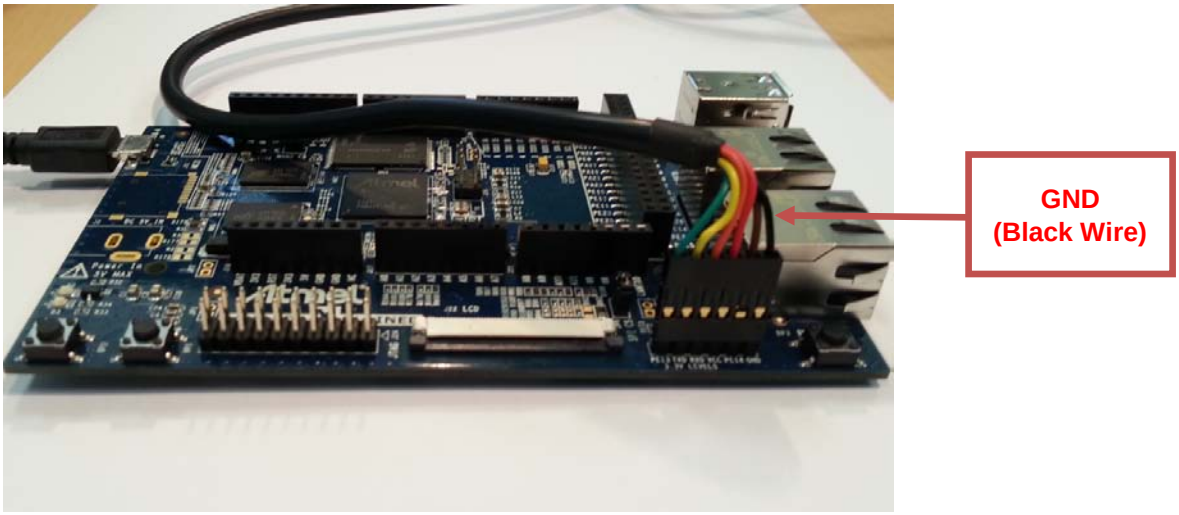
Open **another** terminal (CRTL+ATL+T) that will be dedicated to trace the SAMA5D3 DBGU serial port.

Open a serial terminal using `picocom` program with the required settings (115200 bauds, 8-N-1 with no hardware flow control):

```
# picocom -b 115200 /dev/ttyUSB0
```

```
at91sam@ubuntu: ~
at91sam@ubuntu:~$ picocom -b 115200 /dev/ttyUSB0
picocom v1.4

port is        : /dev/ttyUSB0
flowcontrol    : none
baudrate is    : 115200
parity is      : none
databits are   : 8
escape is      : C-a
noinit is      : no
noreset is     : no
nolock is      : no
send_cmd is    : ascii_xfr -s -v -l10
receive_cmd is : rz -vv

Terminal ready
```

- Plug the 6-pin female connector of your cable on the J23 male from SAMA5D3 Xplained:
  - The ground pin indicated on the silkscreen "GND" has to be connected with the black wire of the FTDI cable.



RESULT Reset the board by pressing BP2 button. You will notice that you have the full log while the system is booting.

EXECUTE .To access to the SAMA5D3 Xplained Linux system, hit the enter key and login as `root`: you have now access to a Linux shell.

```
sama5d3_xplained login: root
#
```

If you reboot the board, you will see that each and every step of the system boot process is displayed:

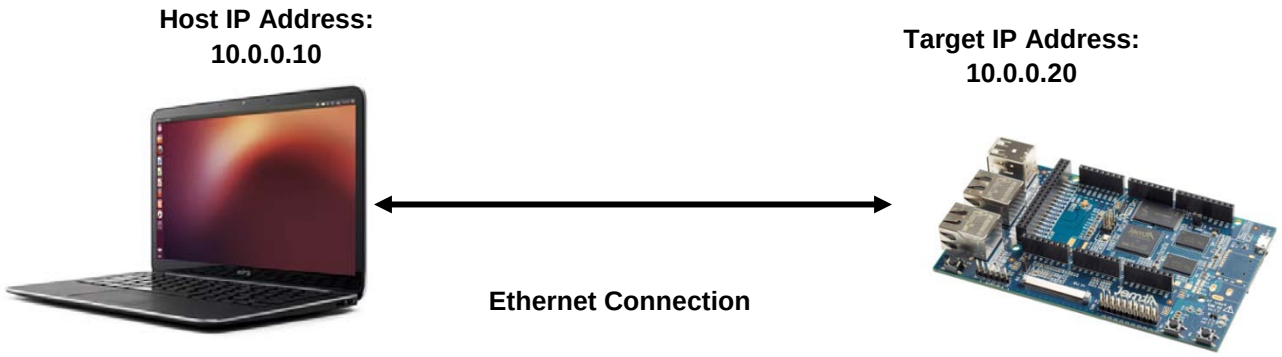
```
# reboot
```

2. Computer Network settings

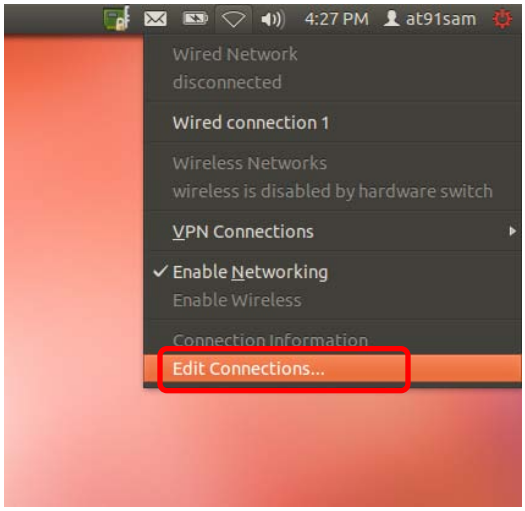
2.1 Wired connection configuration

The goal is to establish an Ethernet connection with the crossed cable provided. The following picture shows the Ethernet connection and IP addresses of the PC and the board.

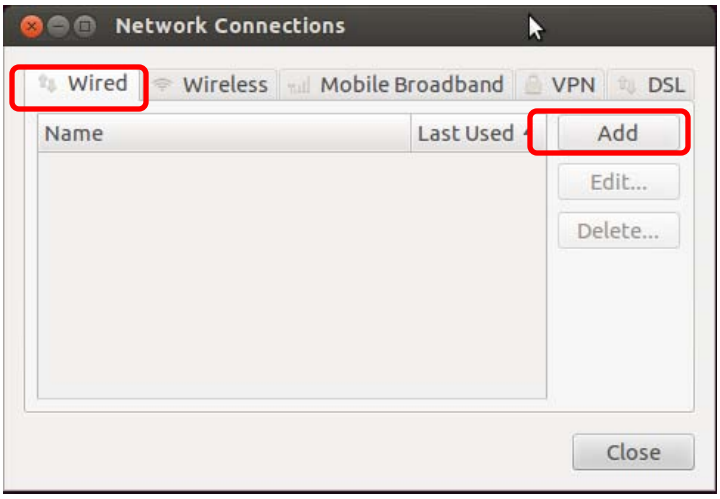
EXECUTE Connect the Ethernet cable between your computer and the Ethernet connector marked `ETH0/GETH` on the SAMA5D3 Xplained



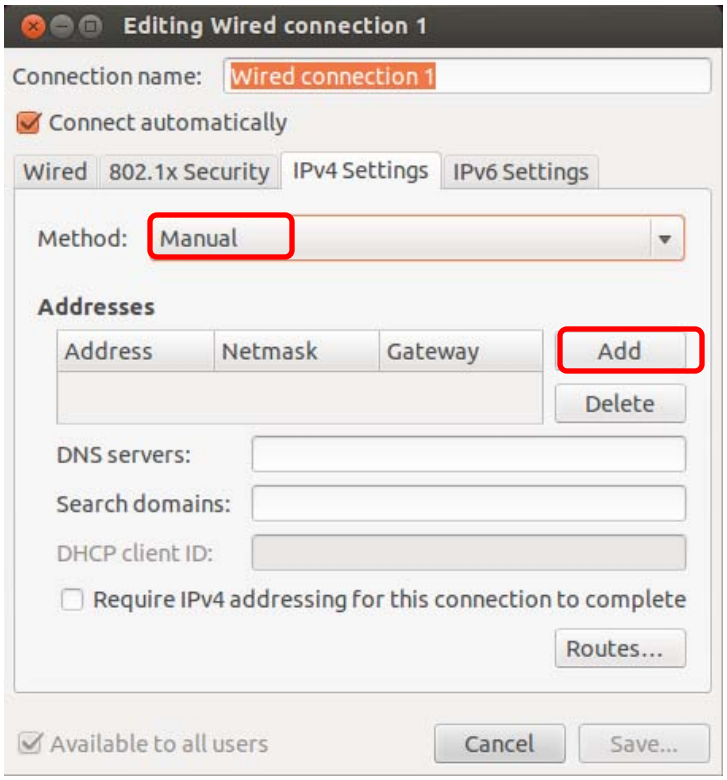
TO DO Set up laptop's IP Address by using the drop-down menu as shown on the screenshot below and click on "Edit Connections":



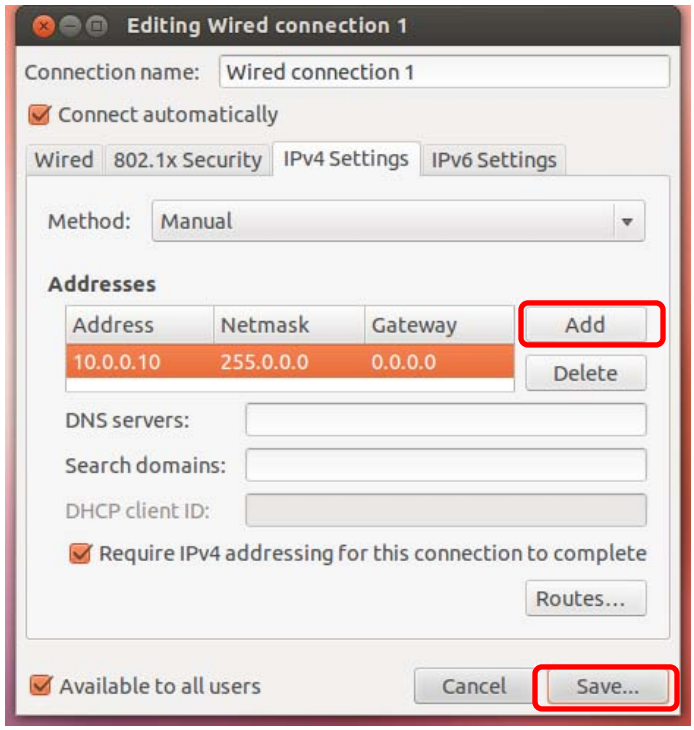
TO DO Select "Wired connection 1" and click Add button



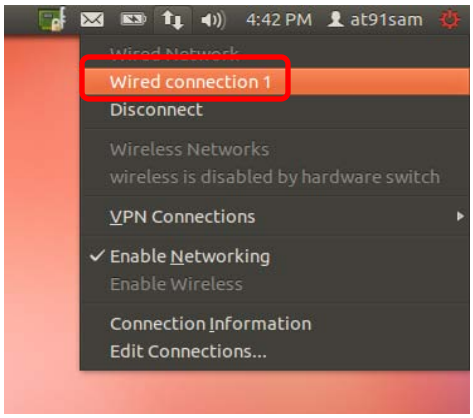
TO DO Choose the “IPv4 Settings” tab, and select Method as “Manual”.



TO DO Click “Add”, and set the IP address to 10.0.0.10, the Netmask to 255.0.0 as follow. Finally, click the “Save…” button.



TO DO Click on “Wired connection 1” to activate this network interface



2.2 TFTP server configuration

TO DO Check whether TFTP server is installed. We use HPA's TFTP server, make sure that the below command returns something

```
dpkg –l | grep tftpd-hpa
```

RESULT

```
~/linux$ dpkg -l | grep tftpd-hpa
ii  tftpd-hpa          5.2-1ubuntu1              HPA's tftp server
```

TO DO Check the TFTP server status:

```
netstat –a | grep tftp
```

RESULT

```
udp        0      0 *:tftp                  *:*
```

WARNING If you cannot see the above result, you need to restart TFTP server like following:

```
# sudo service tftpd-hpa restart
```

TO DO Check the TFTP server's configuration

```
# cat /etc/default/tftpd-hpa
# /etc/default/tftpd-hpa
```

RESULT

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/var/lib/tftpboot"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure"
```

INFO `/var/lib/tftpboot` is the directory containing the files served by the TFTP server.
If you change any configuration of the TFTP server, then you need to use the following command to restart the server: `sudo service tftpd-hpa restart`.
You may also have to restart the TFTP server after booting your PC (that's an Ubuntu bug).

3. SAMA5D3 Xplained Network configuration

- Use the DBGU console (`/dev/ttyUSB0`) to get early boot logs. Power on or reset the board and interrupt U-Boot. You should be in this state:

EXECUTE

```
RomBOOT


AT91Bootstrap 3.6.1-00078-g5415d4e (Tue Feb  4 15:36:46 CET 2014)

NAND: ONFI flash detected
NAND: Manufacturer ID: 0x2c Chip ID: 0x32
NAND: Disable On-Die ECC
NAND: Initialize PMECC params, cap: 0x4, sector: 0x200
NAND: Image: Copy 0x80000 bytes from 0x40000 to 0x26f00000
NAND: Done to load image


U-Boot 2013.07 (Feb 04 2014 - 15:36:32)

CPU: SAMA5D36
Crystal frequency:       12 MHz
CPU clock        :      528 MHz
Master clock     :      132 MHz
DRAM:  256 MiB
NAND:  256 MiB
MMC:   mci: 0, mci: 1
In:    serial
Out:   serial
Err:   serial
Net:   gmac0
Warning: failed to set MAC address
, macb0
Warning: failed to set MAC address
Hit any key to stop autoboot:  0
U-Boot>
```

EXECUTE You can change the automatic boot delay by modifying the bootdelay variable:

```
U-Boot> setenv bootdelay 4
U-Boot> saveenv
```

"`saveenv`" command allows to save in NAND Flash the u-boot environment variables.

"`printenv`" or "`print`" commands allow you to list all the defined u-boot variables.

EXECUTE Set up the TFTP server's IP address which is installed on your laptop

```
U-Boot> setenv serverip 10.0.0.10
```

INFO `serverip` is the IP address of your laptop.

EXECUTE Set up the board's IP address

```
U-Boot> setenv ipaddr 10.0.0.20
```

EXECUTE Set up the board's Ethernet address

```
U-Boot> setenv ethaddr 3a:1f:34:08:54:54
```

INFO If U-Boot prints "`Can't overwrite "ethaddr"`", it means `ethaddr` has already been set. No need to set it again. The ethaddrr is the u-boot variable that allows to store the Ethernet PHY MAC address.

EXECUTE Save the new settings in NAND Flash memory

```
U-Boot> saveenv
U-Boot> print
```

The "`printenv`" or "`print`" commands allow you to list all the defined u-boot variables.

EXECUTE Reset the board to apply new settings and interrupt the boot process

```
U-Boot> reset
```

EXECUTE Test the network connection to the server

```
U-Boot> ping 10.0.0.10
```

RESULT

```
gmac0: PHY present at 7
gmac0: Starting autonegotiation...
gmac0: Autonegotiation complete
gmac0: link up, 100Mbps full-duplex (lpa: 0x45e1)
Using gmac0 device
host 10.0.0.10 is alive
```

4. Loading a kernel through TFTP

4.1 Check U-Boot environment

EXECUTE You can see the U-Boot environment as its configuration. To view it, use the following command:

```
U-Boot> print
```

RESULT

```
baudrate=115200
bootargs=console=ttyS0,115200
mtdparts=atmel_nand:256k(bootstrap)ro,512k(uboot)ro,256k(env),256k(env_redundant
),256k(spare),512k(dtb),6M(kernel)ro,-(rootfs) rootfstype=ubifs ubi.mtd=7
root=ubi0:rootfs rw
bootcmd=nand read 0x21000000 0x00180000 0x00005047; nand read 0x22000000
0x00200000 0x0033BE28; bootz 0x22000000 - 0x21000000
bootdelay=1
ethact=gmac0
ethaddr=3a:1f:34:08:54:54
ipaddr=10.0.0.20
serverip=10.0.0.10
stderr=serial
stdin=serial
stdout=serial

Environment size: 497/131067 bytes
```

The `bootcmd` variable is important since it is the command executed after `bootdelay`. You can easily figure out that it gets two items from the NAND Flash memory: the Linux kernel image and the device tree binary image.
The syntax of the commands is `nand read <dst addr> <src addr> <size>` and

`bootz <kernel zImage addr> - <device tree binary addr>`.

The `bootz` variable allows booting from a kernel compressed image (zImage), this command support device tree.

4.2 Resume the boot process

- We will start the kernel stored in NAND Flash since we need some information contained in its boot logs. The following command will call `bootcmd`:

EXECUTE

```
U-Boot> boot
```

RESULT

```
NAND read: device 0 offset 0x180000, size 0x5047
 20551 bytes read: OK

NAND read: device 0 offset 0x200000, size 0x33be28
 3391016 bytes read: OK
Kernel image @ 0x22000000 [ 0x000000 - 0x33be28 ]
## Flattened Device Tree blob at 21000000
   Booting using the fdt blob at 0x21000000
   Loading Device Tree to 2bb12000, end 2bb1a046 ... OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
Booting Linux on physical CPU 0x0
Initializing cgroup subsys cpuset
Initializing cgroup subsys cpu
Initializing cgroup subsys cpuacct
Linux version 3.10.0-yocto-standard (nferre@tenerife) (gcc version 4.8.1 (GCC) )
#1 Wed Feb 5 10:03:20 CET 2014
CPU: ARMv7 Processor [410fc051] revision 1 (ARMv7), cr=50c5387d
CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
Machine: Atmel SAMA5 (Device Tree), model: SAMA5D3 Xplained
Memory policy: ECC disabled, Data cache writeback
AT91: Detected soc type: sama5d3
AT91: Detected soc subtype: sama5d36
```

INFO Please notice information about the kernel, see the red bold line above.

4.3 Load the kernel image from the TFTP server

When we are developing or debugging, it is too long to flash the kernel each time we want to do a new try so a common way is to load it from a TFTP server. Of course, it's also true for the device tree binary.
We provide you another kernel image built on another machine you will deploy it on the SAMA5D3 Xplained board.

TO DO Copy the kernel image "zImage" located under at91data folder file to the TFTP default server folder:" /var/lib/tftpboot"

```
sudo chmod 777 /var/lib/tftpboot
cp ~/at91data/zImage /var/lib/tftpboot/
```

Now you are able to update u-boot variables to download this new kernel image on the SAMA5D3 Xplained.

EXECUTE Reset the board and interrupt boot sequence

Create a new u-boot variable called 'go' whose purpose is to load the kernel image from the TFTP server and the device tree binary from the NAND Flash memory. tftp command syntax is tftp <dst addr> <file>.

```
U-Boot> setenv go 'nand read 0x21000000 0x00180000 0x00005047; tftp 0x22000000
zImage; bootz 0x22000000 - 0x21000000'
U-Boot> saveenv
```

EXECUTE Execute the go command

```
U-Boot> run go
```

RESULT

```
NAND read: device 0 offset 0x180000, size 0x5047
 20551 bytes read: OK
gmac0: PHY present at 7
gmac0:7 is connected to gmac0.  Reconnecting to gmac0
gmac0: Starting autonegotiation...
gmac0: Autonegotiation complete
gmac0: link up, 100Mbps full-duplex (lpa: 0x45e1)
Using gmac0 device
TFTP from server 10.0.0.10; our IP address is 10.0.0.20
Filename 'zImage'.
Load address: 0x22000000
Loading: T #################################################################
      #################################################################
      #################################################################
      #################################################################
      #################################################################
      #################################################################
      #################################################################
      #################################################################
      #################################################################
      ######################
      470.7 KiB/s
done
Bytes transferred = 3439240 (347a88 hex)
```

```
Kernel image @ 0x22000000 [ 0x000000 - 0x347a88 ]
## Flattened Device Tree blob at 21000000
   Booting using the fdt blob at 0x21000000
   Loading Device Tree to 2bb12000, end 2bb1a046 ... OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
Booting Linux on physical CPU 0x0
Linux version 3.10.0+ (ldesroches@ibiza) (gcc version 4.7.3 (Sourcery CodeBench
Lite 2013.05-24) ) #4 Tue Apr 8 10:42:24 CEST 2014
CPU: ARMv7 Processor [410fc051] revision 1 (ARMv7), cr=10c5387d
CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
Machine: Atmel SAMA5 (Device Tree), model: SAMA5D3 Xplained
debug: ignoring loglevel setting.
Memory policy: ECC disabled, Data cache writeback
AT91: Detected soc type: sama5d3
AT91: Detected soc subtype: sama5d36
```

INFO Notice that kernel information have changed, it's the proof we have successfully used the kernel image from the TFTP server.

5. Conclusion

You have learned how to use U-Boot to load a Linux kernel image from TFTP.

The main advantages of using TFTP to load a kernel image:

- You can put different kernel images in the TFTP server folder and write U-Boot commands to load the image you need.
- You do not have to flash the kernel image into NAND flash every time. After using the TFTP to test the image, you can flash it into the target.
- Through U-Boot, you can use many commands (that helps during the developing phase).

During this hands-on, we have seen why having U-Boot as a second stage bootloader is useful, especially when developing or debugging.
It speeds up the test of a new kernel, allows to change parameters given to the kernel, etc

http://www.at91.com/linux4sam/bin/view/Linux4SAM/U-Boot

https://github.com/linux4sam/u-boot-at91

http://www.denx.de/wiki/U-Boot/Documentation

6. Revision History

Doc. Rev.	Date	Comments
Rev0.1	04/2014	Initial document release
Rev 0.2	05/2014	Update for standalone set-up

Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370

Prerequisites

- **Hardware Prerequisites**
  - SAMA5D3 Xplained board
  - Ethernet cable
  - Micro USB to USB-A cable
    - FTDI TTL-232R-3V3 USB to TTL serial cable
  - PDA 4.3 display module + flex cable
- **Software Prerequisites**
  - Ubuntu 12.04 LTS
- **Estimated completion time**: 60 min

Introduction

The goal of this hands-on is to add the support of the PDA 4.3” display module by loading a new DTB fil. We will introduce you some tools that allows dumping the content of a DTB file.

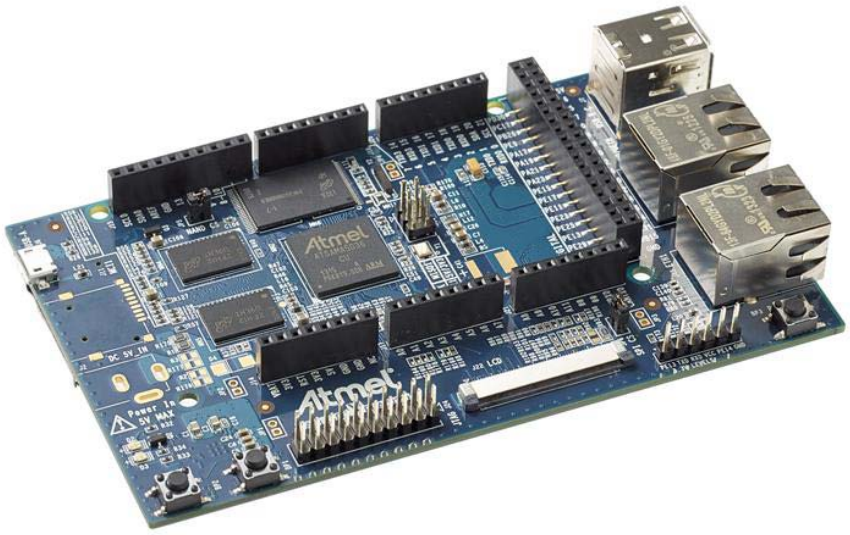


Table of Contents

Icon Key Identifiers

INFO	Delivers contextual information about a specific topic
TIPS	Highlights useful tips and techniques
TO DO	Highlights objectives to be completed on the Linux computer
RESULT	Highlights the expected result of an assignment step
WARNING	Indicates important information
EXECUTE	Highlights actions to be executed out of the SAMA5D3 Xplained board

1. Learn about Device Tree implementation for SAMA5D3

Since Linux kernel 3.x, the Linux community has implemented device tree support for ARM platforms. The Device Tree is a data structure that describes the on-board and on-chip components in a system/target. In other words, it describes the hardware and its peripherals in total. Rather than hard coding every detail of a device into Linux kernel, many aspect of the hardware can be described in a data structure that is passed to the operating system at boot time.
The data structure is basically a tree of named nodes and its properties.

2. Device Tree source code

2.1 Getting Linux-at91 source code

The main commun and official way is to download the zip archive from linux4sam github:

https://github.com/linux4sam/linux-at91 and click on the Download ZIP button on the right side of the page.

You could do it with command lines.

```
wget https://github.com/linux4sam/linux-at91/archive/master.zip
```

```
unzip master.zip –d ../linux/ && cd ../linux/linux-at91-master
```

The other way (and preferred one) to get linux-at91 is to use git:

`git clone https://github.com/linux4sam/linux-at91.git`.
The first checkout is quite long since it will download all the history of the Linux kernel.

The easiest and fastest way is to copy the zip archive provided usb key:

TO DO

The linux-at91-master archive is located under `at91data` folder in the `linux.tar.gz` archive.
You need to create a folder: “linux-at91-master” before uncompressed the archive on it

```
# cd ~
# mkdir linux-at91-master
# cd linux-at91-master
# tar zxvf ~/at91data/linux.tar.gz
```

RESULT

The entire Linux kernel source for AT91 devices is now uncompressed and available on your machine.

2.2 Finding device tree source code

- The source code of device trees is part of the Linux kernel. It's located in the `boot/dts` directory of your device architecture. So in our case, the path is `arch/arm/boot/dts`. To list only Atmel stuff, use the following command:

TO DO Find the SAMA5 device tree files

```
ls arch/arm/boot/dts | egrep "at91-sama5"
```

RESULT

```
at91sam9x5cm.dtsi
at91sam9x5.dtsi
at91sam9x5ek.dtsi
at91-sama5d3_xplained_dm_pda4.dtsi
at91-sama5d3_xplained_dm_pda7.dtsi
at91-sama5d3_xplained.dts
at91-sama5d3_xplained_pda4.dts
at91-sama5d3_xplained_pda7.dts
sama5d31ek.dts
sama5d31ek_pda4.dts
sama5d31ek_pda7.dts
```

3. Device Tree Usage

The Device Tree Compiler (DTC) is the toolchain that convert the human-readable device tree source into the machine-readable binary that both U-Boot and the Linux kernel understand.
The compiler is found in the "`scripts/dtc`" directory in the kernel source tree. The DTC also allows you to dump the content of the dtb file.

The device tree files are compiled per default with the main compilation of the kernel.
You are also able to compile directly the device tree file.

3.1 Accessing DT via Linux User Space

EXECUTE On the SAMA5D3 target running Linux, you can also get the contents of the DT using the `device-tree` entry in the `/proc` filesystem. Type the following command on the SAMA5D3 serial terminal:

- Press BP2 reset button and log as root.

EXECUTE When using Linux, you can also get the content of the DT with the `device-tree` entry in the `/proc` filesystem

```
root@sama5d3_xplained:~# ls /proc/device-tree/
```

RESULT

```
#address-cells    ahb    chosen  compatible  gpio_keys            leds    model
#size-cells aliases  clocks  cpus        interrupt-parent  memory  name
```

INFO One the device tree architecture, usually directories are nodes and files nodes properties. So here you have the content of the root node. You can have a look to the source code to see how it matches.

EXECUTE Use the cat command to print the node information

```
root@sama5d3_xplained:~# cat /proc/device-tree/model
```

RESULT Display the name of the root node of the system

```
SAMA5D3 Xplainedroot@sama5d3_xplained:
```

EXECUTE Print the status of i2c0

```
root@sama5d3_xplained:~# cat /proc/device-tree/ahb/apb/i2c\@f0014000/status
```

RESULT "okay" status means that the node is active and available into the running Linux system

```
okayroot@sama5d3_xplained
```

EXECUTE Print the status of usart2

```
root@sama5d3_xplained:~# cat /proc/device-tree/ahb/apb/serial\@f8020000/status
```

RESULT "disabled "status means that the node is disabled so you are not able to accede to its functionalities.

```
disabledroot@sama5d3_xplained
```

EXECUTE Parse the device tree architecture.

All the embedded peripherals connected via the APB bridge are listed under*/proc/device-tree/ahb/apb/.*

If you want to have dedicated information on a node you need to parse until the node level.

INFO All the access on these directories and files are READ-ONLY and EXE. So you will not able to modify the device tree information on a running Linux kernel.

3.2 Accessing via U-Boot

The u-boot manages directly the device tree blob (dtb) using a dedicated library called “libfdt”, that allows“ blob” support on u-boot.
All the commands used to improve, manage a current device tree blob use the fdt prefix command.
The list / command allows printing information on the current nodes architecture

The print / command allows to go down into the whole node structure.

As we have already launched the demo once and not power off the board, the device tree blob is still available in DDRAM. In other case we need to load again the dtb file into DDR2.

EXECUTE Load the DTB in DDR2 memory and pass to u-boot the location of the DTB (fdt addr)

```
U-Boot> nand read 0x21000000 0x00180000 0x00005047

U-Boot> fdt addr 0x21000000
```

Thanks to the `fdt addr` command U-Boot has been informed about the DTB location. Once this command has been issued, all subsequent `fdt` handling commands will use the DTB stored at the given address

EXECUTE List the content of the root node

```
U-Boot> fdt list /
```

RESULT

```
/ {
    #address-cells = <0x00000001>;
    #size-cells = <0x00000001>;
    model = "SAMA5D3 Xplained";
    compatible = "atmel,sama5d3-xplained", "atmel,sama5d3", "atmel,sama5";
    interrupt-parent = <0x00000001>;
    chosen {
    };
    aliases {
    };
    memory {
    };
    cpus {
    };
    ahb {
    };
    clocks {
    };
    gpio_keys {
    };
    leds {
    };
};
```

INFO To get the full dump of the DTB

```
U-Boot> fdt print /
```

EXECUTE To get the content of a specific node:

```
U-Boot> fdt print /ahb/apb/i2c
```

RESULT

```
i2c@f0014000 {
    compatible = "atmel,at91sam9x5-i2c";
    reg = <0xf0014000 0x00004000>;
    interrupts = <0x00000012 0x00000004 0x00000006>;
    pinctrl-names = "default";
    pinctrl-0 = <0x0000000d>;
    #address-cells = <0x00000001>;
    #size-cells = <0x00000000>;
    status = "okay";
};
```

INFO U-Boot can be used to modify the DT before booting the Linux kernel. Have a look to the `fdt` command with `help fdt`.

In this hands on we will not modify the device tree into u-boot but this can be done easily by updating the node thanks to the set command. For example, to disable the USB:

```
fdt set /ahb/ehci@00700000 status disabled
```

```
fdt set /ahb/ohci@00600000 status disabled
```

4. Update and Compile a Device Tree file

We will add the 4.3" pda support to the SAMA5D3 Xplained board. Fos this new system we will add the "heartbeat" on the led D3 and disable some interfaces such as the MCI and i2c0.

If you don't want to compile the kernel, you can build only the DTB you want. This is what we are going to do in order to build the DTB currently stored in the NAND flash memory of the board.
So we will modify and build the "`at91-sama5d3_xplained_pda4.dts`". The 4.3" pda touch screen is supported into this device tree file, we will update it to integrate ours changes:
MCI0 and i2c0 peripherals disabled, and led D3 blinking.

TO DO Open a Terminal and go into the Linux kernel main directory "`linux-at91-master`" type the following command to open the "`at91-sama5d3_xplained_pda4.dts`"file:

```
gedit arch/arm/boot/dts/at91-sama5d3_xplained_pda4.dts &
```

Modify the file dts file to have: MCI0 and i2c0 peripherals disabled, and led D3 blinking

```
mmc0: mmc@f0000000 {
                    pinctrl-names = "default";
                    pinctrl-0 = <&pinctrl_mmc0_clk_cmd_dat0 &pinctrl_mmc0_dat1_3
&pinctrl_mmc0_dat4_7 &pinctrl_mmc0_cd>;
                    status = "disabled";
                    slot@0 {
                          reg = <0>;
                          bus-width = <8>;
                          cd-gpios = <&pioE 0 GPIO_ACTIVE_LOW>;
                    };
              };
```

```
i2c0: i2c@f0014000 {
                     status = "disabled";
              };
```

```
leds {
      compatible = "gpio-leds";

      d2 {
            label = "d2";
            gpios = <&pioE 23 GPIO_ACTIVE_LOW>; /* PE23, conflicts with A23,
CTS2 */
            linux,default-trigger = "heartbeat";
      };

      d3 {
            label = "d3";
            gpios = <&pioE 24 GPIO_ACTIVE_LOW>;
            linux,default-trigger = "heartbeat";
      };
    };
```

Save and Close "`at91-sama5d3_xplained_pda4.dts`" file.

TO DO Compile the modified device tree file: “at91-sama5d3_xplained_pda4.dts

```
make ARCH=arm sama5d3_xplained_defconfig
```

```
make ARCH=arm CROSS_COMPILE=/opt/poky/1.5.1/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi- at91-sama5d3_xplained_pda4.dtb
```

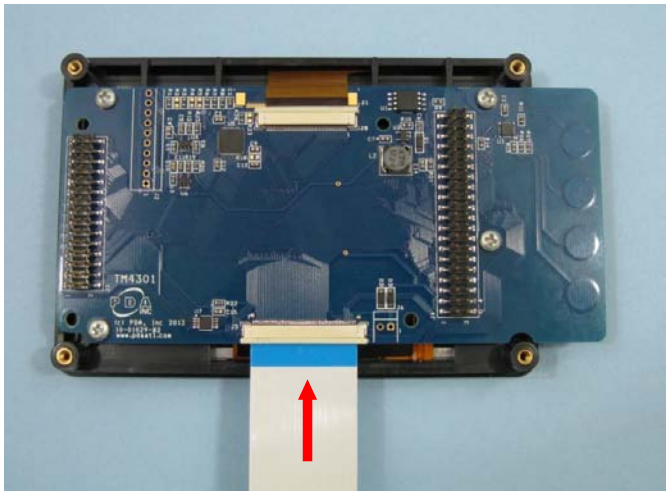
RESULT You have successfully compiled the DTB

```
  CC      scripts/mod/devicetable-offsets.s
  GEN     scripts/mod/devicetable-offsets.h
  HOSTCC  scripts/mod/file2alias.o
  HOSTLD  scripts/mod/modpost
  DTC     arch/arm/boot/dts/at91-sama5d3_xplained_pda4.dtb
```

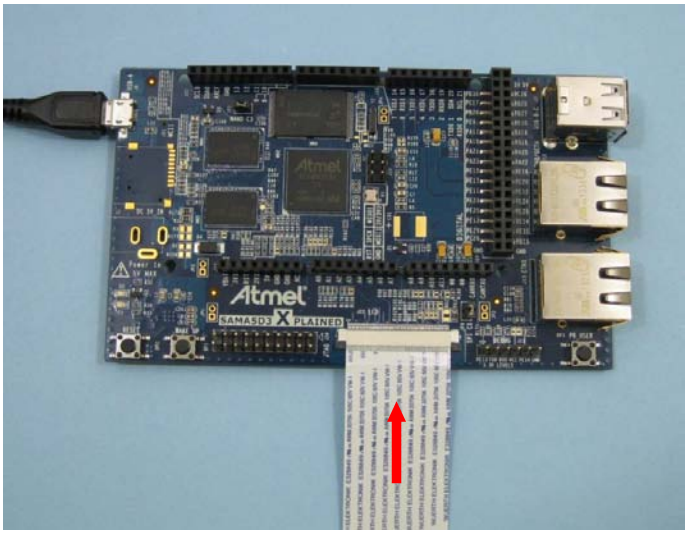
5. Deployment on the SAMA5D3 Xplained

5.1 Connect the 4.3” PDA to the SAMA5D3 Xplained

EXECUTE Power OFF the board and Connect the flex cable to the PDA Touch Module Contact on the flex face down toward PCB



EXECUTE Connect flex cable to SAMA5D3 Xplained LCD connector (J22) Contact on flex up away from PCB



EXECUTE Re-position the setup with the Touch module oriented as shown below and power ON the board:

5.2 U-boot configuration

The easiest way is to deploy the net DTB file on the target is get it with U-Boot (as we have done for the kernel) then to write it in NAND Flash memory. Load the device tree binary from the TFTP server

TO DO Copy the new at91-sama5d3_xplained_pda4.dtb file in the TFTP server folder

```
sudo cp arch/arm/boot/dts/at91-sama5d3_xplained_pda4.dtb /var/lib/tftpboot/
```

- Create a new variable whose purpose is to load the kernel image from the TFTP server and the device tree binary from the NAND Flash memory.
    - tftp command syntax is tftp <dst addr> <file>.

EXECUTE

```
U-Boot> setenv go_dtb 'tftp 0x21000000 at91-sama5d3_xplained_pda4.dtb; nand read
0x22000000 0x200000 0x33BE28; bootz 0x22000000 - 0x21000000'
U-Boot> saveenv
```

EXECUTE Execute the go_dtb command

```
U-Boot> run go_dtb
```

WARNING If you have timeout, you need to restart TFTP server like following:

```
# sudo service tftpd-hpa restart
```

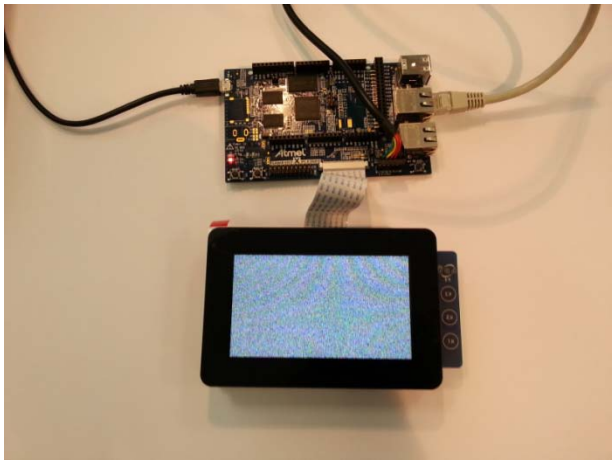
RESULT The led D3 is blinking, and check the modifications you have done on the device tree by parsing it: /proc/device-tree

```
root@sama5d3_xplained:~# cat /proc/device-tree/ahb/apb/mmc\@f0000000/status

root@sama5d3_xplained:~# cat /proc/device-tree/ahb/apb/i2c\@f0014000/status

root@sama5d3_xplained:~# cat /proc/device-tree/leds/d3/linux,default-trigger
```

EXECUTE Fill the LCD buffer with random values

```
root@sama5d3_xplained:~# cat /dev/urandom > /dev/fb0
cat:write error: No space left on device
# root@sama5d3_xplained:~#
```

WARNING The error "cat:write error: No space left on device" is because we fill the frame buffer fb0 with " zero" until there is no anymore space left.



EXECUTE You need to configure your board network interface on your running Linux application:

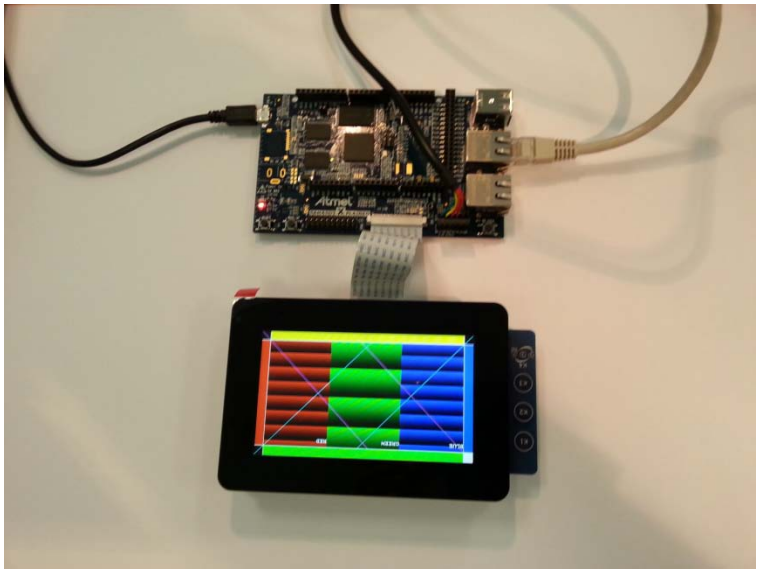
```
root@sama5d3_xplained: ifconfig eth0 10.0.0.20
```

TO DO Download the fb-test package on the SAMA5D3 Xplained using ssh connection

```
scp ~/at91data/ipk/fb-test* root@10.0.0.20:/home/root/
```

EXECUTE Install and run fb-test package on the target

```
root@sama5d3_xplained: cd /home/root
root@sama5d3_xplained: opkg install fb-test_0.0-r0_cortexa5hf-vfp.ipk
root@sama5d3_xplained:fb-test
```



5.3 Optional: Flash the new DTB on the SAMA5D3 Xplained

EXECUTE On the target download the new dtb file into DDR2 memory (location 0x21000000)

```
U-Boot> tftp 0x21000000 at91-sama5d3_xplained_pda4.dtb
```

RESULT

```
gmac0: PHY present at 7
gmac0:7 is connected to gmac0.  Reconnecting to gmac0
gmac0: Starting autonegotiation...
gmac0: Autonegotiation complete
gmac0: link up, 100Mbps full-duplex (lpa: 0x45e1)
Using gmac0 device
TFTP from server 10.0.0.10; our IP address is 10.0.0.20
Filename 'at91-sama5d3_xplained.dtb'.
Load address: 0x21000000
Loading: T #####
     3.9 KiB/s
done
Bytes transferred = 21113 (5279 hex)
```

INFO The file size can be different, you need to us the correct one on the next steps.

EXECUTE Erase the Flash partition dedicated to DTB storage (you can find the memory mapping on Linux4sam
http://www.at91.com/linux4sam/bin/view/Linux4SAM/GettingStarted

```
U-Boot> nand erase 0x180000 0x80000
```

RESULT

```
NAND erase: device 0 offset 0x180000, size 0x80000
Erasing at 0x1e0000 -- 100% complete.
OK
```

- Write device tree binary in NAND Flash memory. The syntax of the command is `nand write <src addr> <dst addr> <size>`

EXECUTE

```
U-Boot> nand write 0x21000000 0x180000 0x5279
```

RESULT

```
NAND write: device 0 offset 0x180000, size 0x5279
 20663 bytes written: OK
```

- Update bootcmd since the DT size may have changed

EXECUTE

```
U-Boot> setenv bootcmd 'nand read 0x21000000 0x00180000 0x00005279; nand read
0x22000000 0x00200000 0x00347ad8; bootz 0x22000000 - 0x21000000'
U-Boot> saveenv
```

- Reset the system and check that LED3 is blinking

RESULT The DTB has been successfully programmed in SAMA5D3 Xplained NAND Flash.

6. Conclusion

In this hands-on we covered the syntax of the Device Tree data structure. We learnt in quite detail how the SAMA5D3 Device Tree is implemented. We also saw some examples of how to access the Device Tree from Linux User-space on the SAMA5D3 Xplained. The DTC was introduced and we learnt how .dtb is generated by the DTC. In this hands-on, we have seen how to compile a DT and how to dump its content that can be useful when debugging to be sure to use the proper device tree. Then we used U-Boot to flash the new DTB file.

7. Revision History

Doc. Rev.	Date	Comments
Rev0.1	04/2014	Initial document release
Rev1.0	05/2014	Update of the document

Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370

SAMA5D3-Xplained: Atmel Qt SDK installation and Qt Creator configuration

AN-8513

Prerequisites

- **Hardware Prerequisites**
  - Atmel® SAMA5D3 Xplained XSTK
  - micro USB to USB-A cables
  - Ethernet cable
  - USB serial TTL adapter (optional)
    - FTDI TTL-232R-3V3 USB to TTL serial cable
- **Software Prerequisites**
  - Linux PC running Ubuntu 12.04 LTS
- **Estimated completion time**: 45 min

Introduction

The goal of this hands-on is to install easily the Atmel Qt SDK and Qt Creator tool that allows you to build easily a embedded UI application. You will program SAM5D3-Xplained connected to 4.3” PDA touch screen with the Atmel Smart Fridge Demo. By completing this hands-on you will

- Learn how to setup cross-development environment for your Qt software
- Gain operational knowledge on Qt Creator

Table of Contents

Icon Key Identifiers

INFO	Delivers contextual information about a specific topic
TIPS	Highlights useful tips and techniques
TO DO	Highlights objectives to be completed on the Linux computer
RESULT	Highlights the expected result of an assignment step
WARNING	Indicates important information
EXECUTE	Highlights actions to be executed out of the SAMA5D3 Xplained board

1. Assignment 1: Install SAM-BA on Ubuntu 12.04

1.1 Serial device access Setup

With Ubuntu distributions, a user has to be member of the `dialout` group to access serial devices like RS232 serial port, usb serial, serial modem, etc.

As SAM-BA uses the USB serial class driver to connect to the SAMA5D3 Xplained board, it is important to enable serial devices access to the Ubuntu user.

TO DO Enable access to serial devices

- Add current "user" to `dialout` group:

```
# cd ~
# sudo adduser at91sam dialout
# cat /etc/group | grep dialout
# dialout:x:20:at91sam
```

- **Log out from Ubuntu** and log in again to make the changes effective and access the serial port.

WARNING Logging out from Ubuntu is mandatory to have the user added to the `dialout` group.

RESULT Current user has now access to serial devices including the USB serial port.

1.2 Install SAM-BA on Ubuntu 12.04

TO DO Download SAM-BA for linux archives.
SAM-BA is also available on atmel.com:
http://www.atmel.com/tools/ATMELSAM-BAIN-SYSTEMPROGRAMMER.aspx

```
# cd ~/at91data
# wget ftp://ftp.linux4sam.org/pub/sam-ba/sam-ba_2.14_cdc_linux.zip
# cd ~
```

WARNING For Ubuntu 64 bit only. SAM-BA requires 32 bit libraries

```
# uname –-machine
x86_64 → "64" means you are using a 64 bit distribution
# sudo apt-get install ia32-libs
[sudo] password for at91sam: → type your password (sam)
```

INFO User is `at91sam` for that hands-on with `sam` as password. Use your login/password whenever required if different.

TO DO Uncompress SAM-BA archive.

```
# cd /opt
# sudo unzip ~/at91data/sam-ba_2.14_cdc_linux.zip
```

- Add SAM-BA directory to PATH variable:

```
# export PATH=$PATH:/opt/sam-ba_cdc_linux
```

- Edit `.bashrc` file to permanently add SAM-BA directory to the PATH variable:

```
# gedit ~/.bashrc
```

- Add at the end of the file the following line to update the PATH:

```
# export PATH=$PATH:/opt/sam-ba_cdc_linux
```

- Activate the `.bashrc` file with the updated PATH using the source command:

```
# source ~/.bashrc
```

RESULT SAM-BA for Linux has been successfully installed.

2. Assignment 2: Download and Install Atmel Qt SDK

INFO Atmel provides an Yocto Project meta-layers optimized to produce BSP and SDK for SAMA5D3 reference boards: https://github.com/linux4sam/meta-atmel
One of the main outputs from Yocto build system is the QT-SDK (meta-toolchain-qte) which when installed on a development host PC provides all necessary tools to develop C/C++ and Qt software.

TO DO Open a terminal: Press “CRTL+ATL+T and create a working folder in your home directory:

```
# mkdir ~/at91data
```

TO DO Download and install Atmel Yocto QTE SDK

WARNING For Ubuntu 64 bit installation

```
# cd ~/at91data
# wget ftp://ftp.linux4sam.org/pub/demo/yocto-qte-sdk/poky-eglibc-x86_64-meta-toolchain-qte-cortexa5hf-vfp-toolchain-qte-1.5.1.sh
# chmod a+x ./poky-eglibc-x86_64-meta-toolchain-qte-cortexa5hf-vfp-toolchain-qte-1.5.1.sh
# ./poky-eglibc-x86_64-meta-toolchain-qte-cortexa5hf-vfp-toolchain-qte-1.5.1.sh
Enter target directory for SDK (default: /opt/poky/1.5.1):
You are about to install the SDK to "/opt/poky/1.5.1". Proceed[Y/n]?
[sudo] password for at91sam:
SDK has been successfully set up and is ready to be used.
```

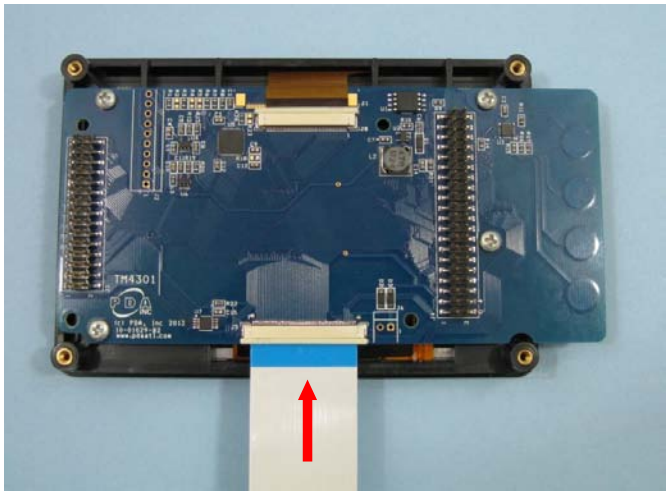
WARNING For Ubuntu 32 bit installation

```
# cd ~/at91data
# wget ftp://ftp.linux4sam.org/pub/demo/yocto-qte-sdk/poky-eglibc-i686-meta-toolchain-qte-cortexa5hf-vfp-toolchain-qte-1.5.1.sh
# chmod a+x ./poky-eglibc-i686-meta-toolchain-qte-cortexa5hf-vfp-toolchain-qte-1.5.1.sh
# ./ poky-eglibc-i686-meta-toolchain-qte-cortexa5hf-vfp-toolchain-qte-1.5.1.sh
Enter target directory for SDK (default: /opt/poky/1.5.1):
You are about to install the SDK to "/opt/poky/1.5.1". Proceed[Y/n]?
[sudo] password for at91sam:
SDK has been successfully set up and is ready to be used.
```

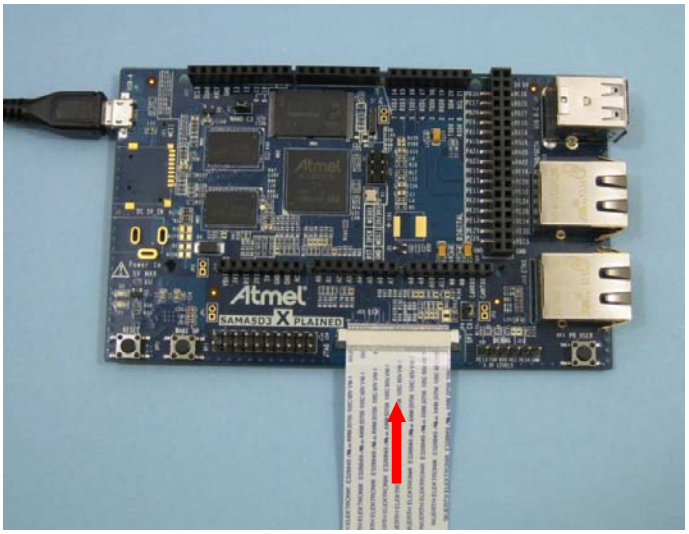
3. Assignment 3: Install “atmel-qte-demo_sama5d3_xplained”

3.1 Connect the 4.3” PDA to the SAMA5D3 Xplained

EXECUTE Power OFF the board and Connect the flex cable to the PDA Touch Module Contact on the flex face down toward PCB



EXECUTE Connect flex cable to SAMA5D3 Xplained LCD connector (J22) Contact on flex up away from PCB



EXECUTE Re-position the setup with the Touch module oriented as shown below and power ON the board:

3.2 Install on SAMA5D3_Xplained board prebuilt demo image

INFO This image is based on Yocto 1.5.1 plus meta-atmel layers
Prebuilt demo binaries are available on Linux4SAM ftp server:
ftp://ftp.linux4sam.org/pub/demo/

TO DO Open a terminal (CRTL+ATL+T) and download demo archive

```
# cd ~/at91data
# wget ftp://ftp.linux4sam.org/pub/demo/linux4sam_4.4/linux4sam-poky-qte-sama5d3_xplained-4.4.zip
# unzip linux4sam-poky-qte-sama5d3_xplained-4.4.zip
#
```

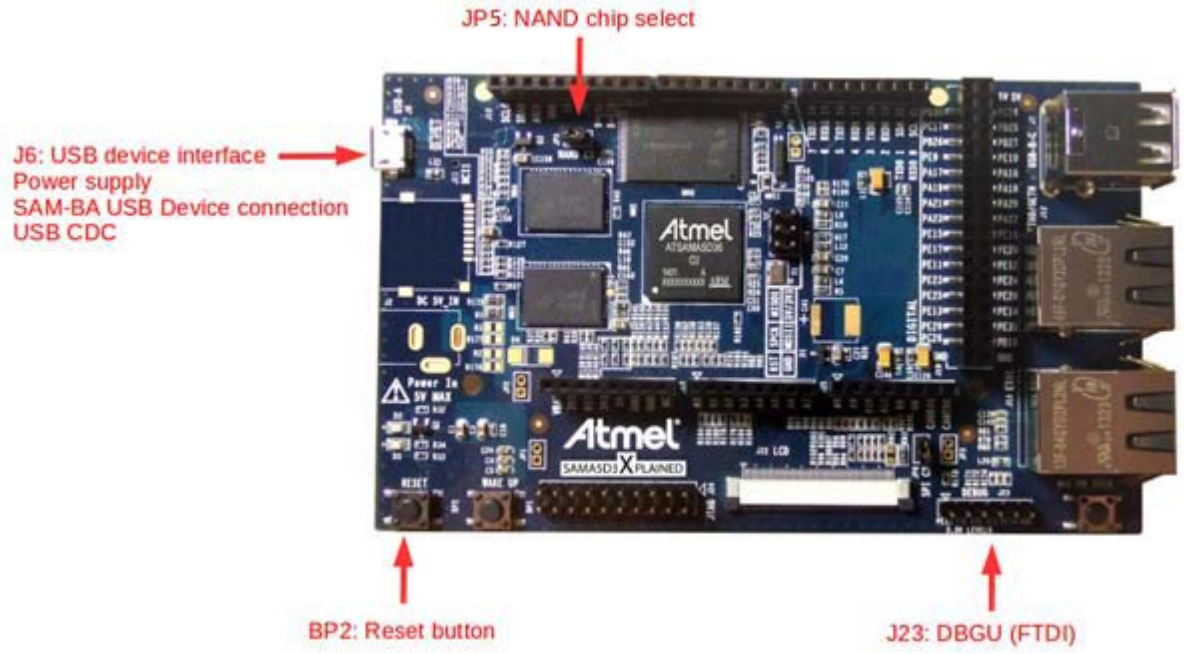
TO DO Inspect demo contents

```
# cd ~/at91data/linux4sam-poky-qte-sama5d3_xplained-4.4
# ls –lh
at91-sama5d3_xplained.dtb                    → regular xplained dtb file (no LCD)
at91-sama5d3_xplained_pda4.dtb               → xplained + LCD pda 4.3” dtb file
at91-sama5d3_xplained_pda7.dtb               → xplained + LCD pda 7” dtb file
zImage-sama5d3_xplained.bin                  → kernel zImage (Linux 3.10-at91)
u-boot-sama5d3_xplained-v2013.07-at91-r2.bin    → u-boot 2013.07
sama5d3_xplained-nandflashboot-uboot-3.6.2.bin  → at91bootstrap (nandflash boot)
atmel-qt4e-demo-image-sama5d3_xplained.ubi      → root file system (UBIFS)

demo_linux_nandflash.bat                     → Windows SAM-BA scripts
demo_linux_nandflash_pda4.bat                  One for each board variant
demo_linux_nandflash_pda7.bat

demo_linux_nandflash.sh                      → Linux SAM-BA scripts
demo_linux_nandflash_pda4.sh                   One for each board variant
demo_linux_nandflash_pda7.sh

demo_linux_nandflash.tcl                     → TCL SAM-BA scripts
demo_linux_nandflash_pda4.tcl                  One for each board variant
demo_linux_nandflash_pda7.tcl
demo_script_linux_nandflash.tcl
```

3.3 Flash the demo

EXECUTE

Open JP5 to disable NAND Flash memory access
Press BP2 reset button to boot from on-chip Boot ROM
Close JP5 to enable NAND Flash memory access



TO DO

Identify the USB connection by monitoring the last lines of "*dmesg*"command.
The *"/dev/ttyACMx"* number will be used to configure the terminal emulator

```
# dmesg
usb 2-6: new high-speed USB device number 60 using ehci_hcd
cdc_acm 2-6:1.0: This device cannot do calls on its own. It is not a modem.
cdc_acm 2-6:1.0: ttyACM0: USB ACM device
```

TIPS

"lsusb" command may also help to identify right ttyACM number.
Monitor USB device number.

```
# lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
[…]
Bus 002 Device 060: ID 03eb:6124 Atmel Corp. at91sam SAMBA bootloader
```

WARNING If the /dev/ttyACMx that appears is different from /dev/ttyACM0, edit the demo_linux_nandflash_pda4.sh file and modify /dev/ttyACMx device number.

```
# cd ~/at91data/linux4sam-poky-qte-sama5d3_xplained-4.4
# vi demo_linux_nandflash_pda4.sh
```

TIPS Below are some useful commands for vi editor:

<Esc> key -> to enter a new command or escape
<Esc> i -> insertion mode
<Esc>:wq -> save and exit the text
<Esc>:q -> exit the text without saving.
<Esc> /foo -> Search for “foo” string in the current files

Or if you are uncomfortable with vi, use gedit

```
# cd ~/at91data/linux4sam-poky-qte-sama5d3_xplained-4.4
# gedit demo_linux_nandflash_pda4.sh &
```

TO DO Flash the demo

```
# cd ~/at91data/linux4sam-poky-qte-sama5d3_xplained-4.4
# chmod a+x ./ demo_linux_nandflash_pda4.sh
#./demo_linux_nandflash_pda4.sh
```

TO DO When the logfile.log appears on the terminal (this will take a few minutes), check that = Done. = is written at the end of the file

```
[…]
-I- Complete 99%
-I-     Writing: 0x20000 bytes at 0xEC40000 (buffer addr : 0x20010BA8)
-I-     0x20000 bytes written by applet
-I- === DONE. ===
```

TIPS During FLASH operation, you can monitor SAM-BA logfile.
Open a new terminal: Press “CRTL+ATL+T”

```
# cd ~/at91data/linux4sam-poky-qte-sama5d3_xplained-4.4
# tail –f logfile.log
```

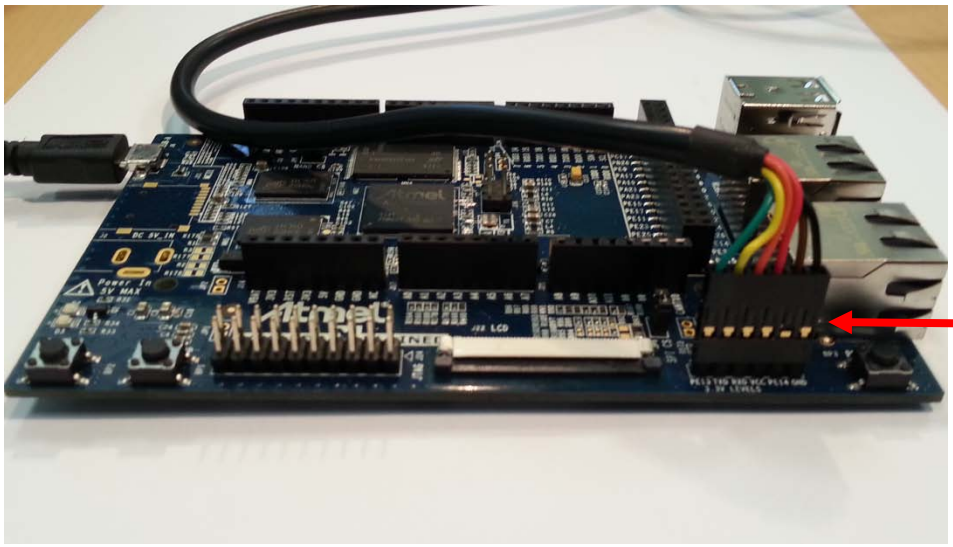
EXECUTE Press BP2 reset button to boot on NAND Flash memory and start the demo

4. Assignment 4: Board bring up

4.1 The DBGU interface

The Debug Unit (DBGU) provides a two-pin UART that can be used to control a Linux-based system on Atmel SoCs. All boot logs are sent to the DBGU and ways to interact with the bootloaders and the Linux system are provided through this essential serial interface.

On the board, the DBGU pins are accessible on the 6-pin male header J23 "DEBUG" connector.



WARNING The signal level must be 3.3V and the pin-out is displayed on the silkscreen and explained in the "SAMA5D3 Xplained User Guide" document.

INFO You can note that this connector's pin-out is the one used by the popular FTDI TTL to USB converter reference design also referenced as FTDI TTL-232R-3V3 TTL to USB serial cable.

4.2 Installing USB/Serial cable drivers

TO DO Take the TTL to USB serial cable provided for the training session and plug it to your host computer.

INFO On a Windows system you would need to install a driver that you can find on the FTDI website (http://www.ftdichip.com/Drivers/VCP.htm).

As we run our development host on Linux, the USB-serial driver is already installed and should appear as a "/dev/USBx" device when plugged

TO DO Now you must identify the serial USB connection by monitoring the last lines of dmesg command. The **/dev/ttyUSBx** number will be used to configure the terminal emulator. **On the host**, in a shell window (CTRL-ALT-T), run the following command:

```
# dmesg | tail
usb 1-1.1.2: new full-speed USB device number 17 using ehci-pci
usb 1-1.1.2: New USB device found, idVendor=0403, idProduct=6001
usb 1-1.1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 1-1.1.2: Product: TTL232R-3V3
usb 1-1.1.2: Manufacturer: FTDI
usb 1-1.1.2: SerialNumber: FTGNVZ04
ftdi_sio 1-1.1.2:1.0: FTDI USB Serial Device converter detected
usb 1-1.1.2: Detected FT232RL
[…]
usb 1-1.1.2: FTDI USB Serial Device converter now attached to ttyUSB0
```

4.3 Full access to the system

TO DO Now, in another window (hit CTRL-ALT-T), open your favorite serial terminal emulator with appropriate settings: **115200 bauds, 8-N-1**.

```
# picocom -b 115200 /dev/ttyUSB0
```

WARNING You can now plug the 6-pin female connector of your cable on the J23 male headers paying attention to the insertion way: the ground pin indicated on the silkscreen "GND" have to be connected with the black wire of the FTDI cable.

EXECUTE Hit the enter key and login as `root`: you have now access to a Linux shell.

```
# sama5d3_xplained login: root
```

INFO If you reboot the board, you will see that each and every step of the system boot process is displayed.

```
root@sama5d3_xplained:~# reboot
```

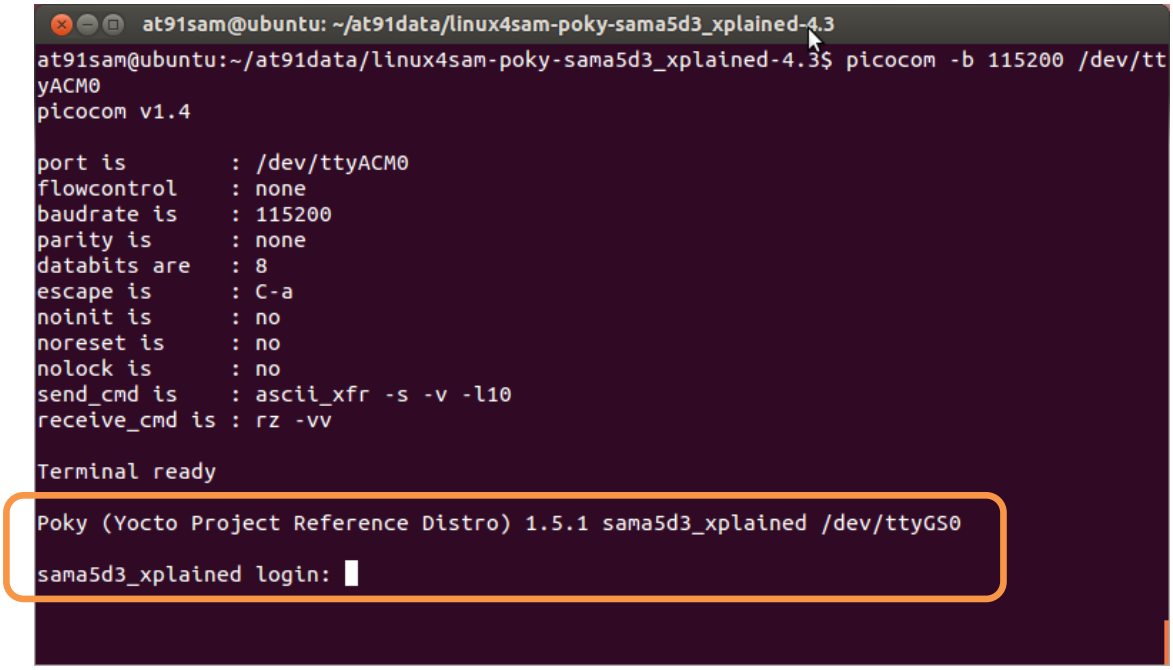
TIPS If you do not have any serial DBGU debug connection (using for example a USB serial TTL adapter), you can get Linux Kernel debug information by opening a serial terminal on the USB connection (ttyACMx):

- Open a serial terminal using `picocom` program with the required settings (115200 bauds, 8-N-1 with no hardware flow control):

```
# picocom -b 115200 /dev/ttyACM0
```

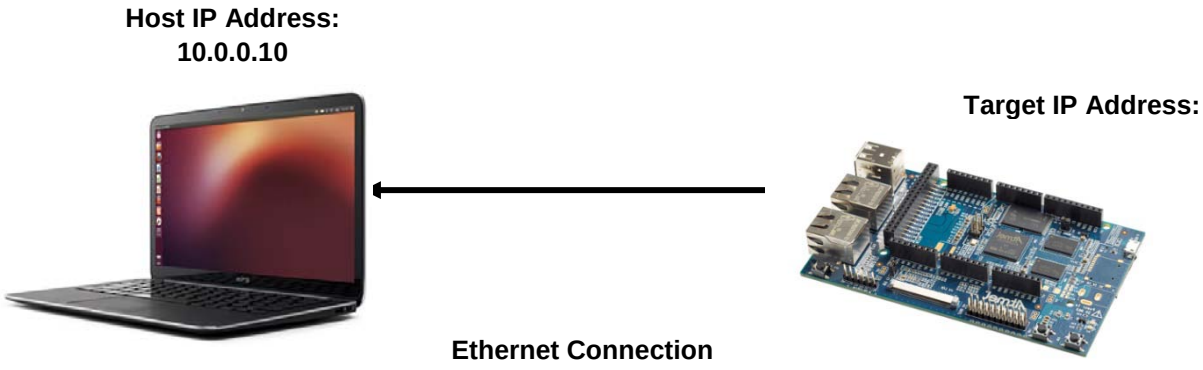
RESULT The demo has booted successfully.

You should see the Linux `sama5d3_xplained_login` prompt:



4.4 Configure the network

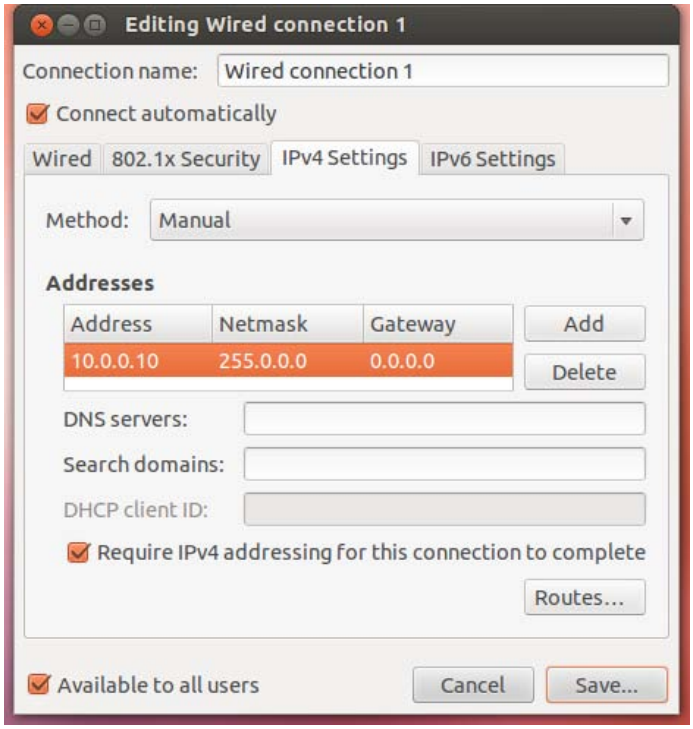
TO DO Establish an Ethernet connection with the crossed cable provided. The following picture shows the Ethernet connection and IP addresses of the PC and the board. Use the Ethernet connector marked `ETH0/GETH` on the SAMA5D3 Xplained.



TO DO Set up laptop's IP Address by using the drop-down menu as shown on thesam-screenshot below:

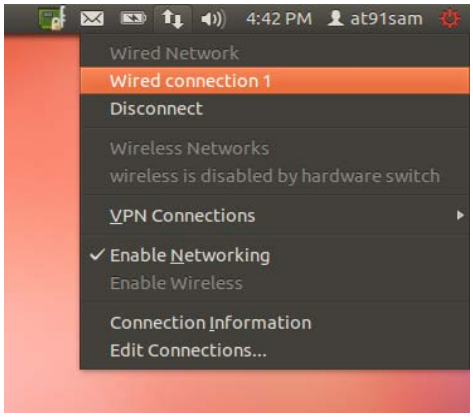
TO DO

Choose “Wired connection 1” and click “Edit…”.
Choose the “IPv4 Settings” tab, and select Method as “Manual”. Click “Add”, and set the IP address, the Netmask and Gateway as follow. Finally, click the “Save…” button



TO DO

Click on “Wired connection 1” to activate this network interface



EXECUTE Configure the network on Target.
Edit "/etc/network/interfaces" to setup default Ethernet configuration by adding these lines:

```
# root@sama5d3_xplained:~# vi /etc/network/interfaces
[…]
auto eth0
iface eth0 inet static
        address 10.0.0.20
        netmask 255.0.0.0
        gateway 10.0.0.10
[…]
```

TIPS Below are some useful commands for vi editor:

<Esc> key -> to enter a new command or escape
<Esc> i -> insertion mode
<Esc>:wq -> save and exit the text
<Esc>:q -> exit the text without saving.
<Esc> /foo -> Search for "foo" string in the current files

TIPS Please check that only one eth0 is defined into /etc/network/interfaces
Do not forget that file system use cache. So if you plan to reboot immediately with the reset button you have to be sure that you data are effectively stored.
Use the sync command (for more information use "man sync")

```
root@sama5d3_xplained:~# sync
```

EXECUTE Restart network on target with new configuration

```
root@sama5d3_xplained:~# /etc/init.d/networking stop
root@sama5d3_xplained:~# /etc/init.d/networking start
Configuring network interfaces... IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
done.
macb f0028000.ethernet eth0: link up (100/Full)
IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
```

INFO You can also use the `restart` command

```
root@sama5d3_xplained:~# /etc/init.d/networking restart
```

EXECUTE Test network on the target – ping command

```
root@sama5d3_xplained:~# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data.
64 bytes from 10.0.0.10: icmp_seq=1 ttl=64 time=0.266 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=64 time=0.252 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=64 time=0.244 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=64 time=0.214 ms
```

TO DO Test network on the host – ping command

```
# ping 10.0.0.20
PING 10.0.0.20 (10.0.0.20) 56(84) bytes of data.
64 bytes from 10.0.0.20: icmp_req=1 ttl=64 time=0.458 ms
64 bytes from 10.0.0.20: icmp_req=2 ttl=64 time=0.195 ms
64 bytes from 10.0.0.20: icmp_req=3 ttl=64 time=0.158 ms
65 bytes from 10.0.0.20: icmp_req=4 ttl=64 time=0.173 ms
```

TIPS As SAMA5D3 Xplained embeds a ssh server, you can use “ssh” to login to the target with ssh

```
# ssh root@10.0.0.20
The authenticity of host '10.0.0.20 (10.0.0.20)' can't be established.
ECDSA key fingerprint is 50:24:4e:16:0a:54:cc:5e:0d:41:51:2c:cc:d0:bd:ae.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.0.20' (ECDSA) to the list of known hosts.
root@sama5d3_xplained:~#
```

5. Assignment 4: Download, Install and configure QtCreator

5.1 Donwload and Install

Right now you need to download the Qt Creator tool as all the required settings and tools are available thanks to the Atmel Qt SDK.

TO DO Qt Creator download and installation
Open a terminal: Press “CRTL+ATL+T”

INFO Installer has to be executed with super user rights.
We will install qt-creator in “/opt” directory

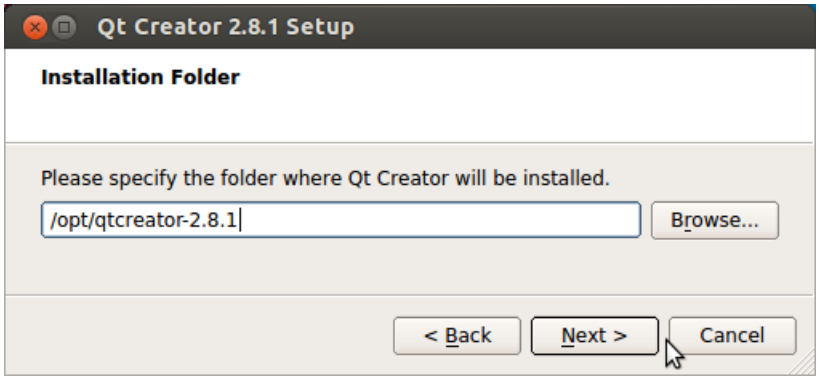
WARNING For Ubuntu 64 bit installation

```
# cd ~/at91data
# wget http://download.qt-project.org/official_releases/qtcreator/2.8/2.8.1/qt-creator-linux-x86_64-opensource-2.8.1.run
# chmod a+x ./qt-creator-linux-x86_64-opensource-2.8.1.run
# sudo ./qt-creator-linux-x86_64-opensource-2.8.1.run
```

WARNING For Ubuntu 32 bit installation

```
# cd ~/at91data
# wget http://download.qt-project.org/official_releases/qtcreator/2.8/2.8.1/qt-creator-linux-x86-opensource-2.8.1.run
# chmod a+x ./qt-creator-linux-x86-opensource-2.8.1.run
# sudo ./qt-creator-linux-x86-opensource-2.8.1.run
```

TO DO Set the installation folder to `:/opt/qtcreator-2.8.1/`



WARNING **DO NOT launch Qt Creator at the end of the installation!**

5.2 Qt Creator Configuration

Qt Creator now will be configured to work with the Atmel Qt SDK. So we need to define the links for all the Qt environment parameters.

TO DO Set Qt-SDK Environment variables with qtcreator

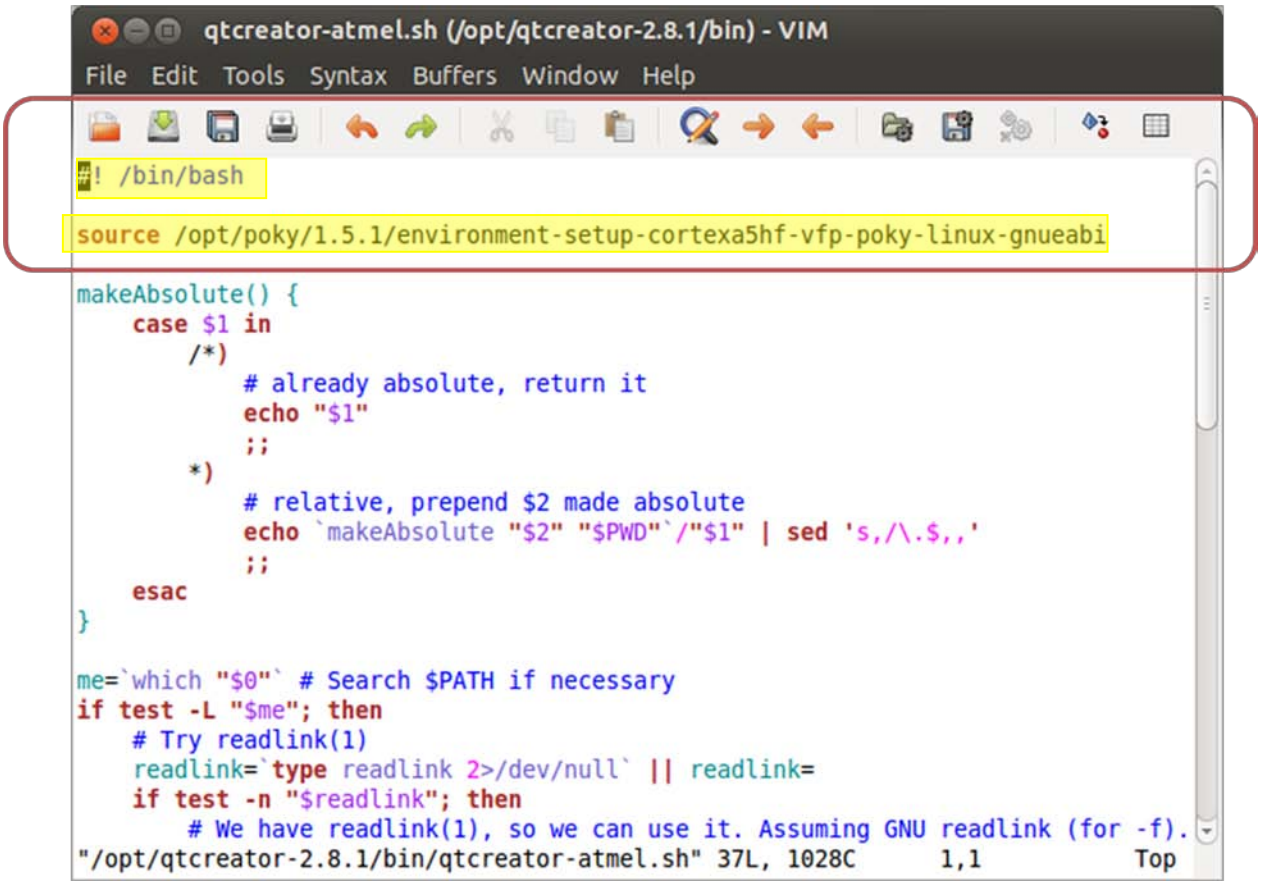
- We will first copy the default qtCreator launch script to create one using Atmel Qt SDK.

In a terminal window type the command:

```
# sudo cp   /opt/qtcreator-2.8.1/bin/qtcreator.sh \
            /opt/qtcreator-2.8.1/bin/qtcreator-atmel.sh
```

- We simply source Atmel QT SDK environment variable before to launch qt-creator. We have also to change the default shell used, as by default Ubuntu shell is dash, we need bash! So edit the file and add these **two** lines in the top (yellow underlined).

```
# sudo vim /opt/qtcreator-2.8.1/bin/qtcreator-atmel.sh
```



TIPS Below are some useful commands for vi /vim editor:

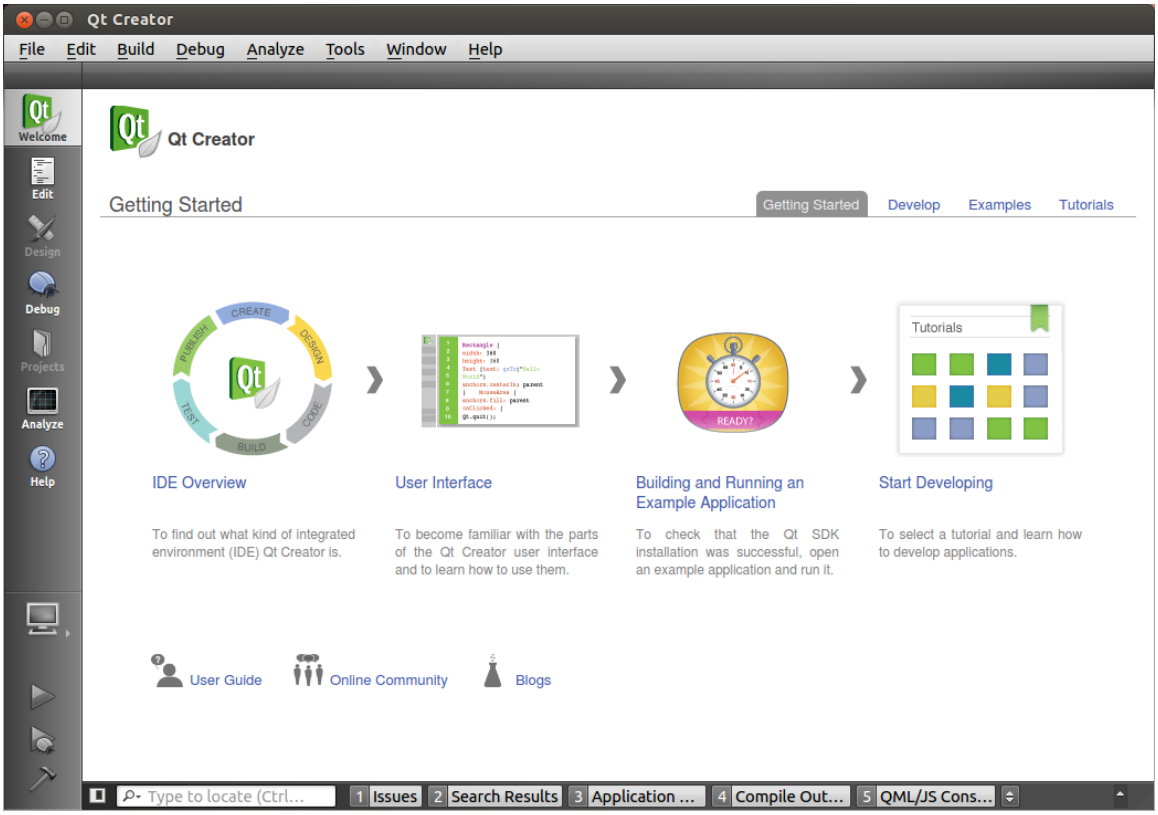
<Esc> key -> to enter a new command or escape
<Esc> i -> insertion mode
<Esc>:wq -> save and exit the text
<Esc>:q -> exit the text without saving.
<Esc> /foo -> Search for “foo” string in the current files

TIPS Inspect the poky environment set up file. By sourcing this file, all environment variables needed to cross compile applications are set.
For example PATH, CC, GDB but also QT environments variables are set.
Please note default ARM GCC compiler options are set to match SAMA5D3 architecture: armv7-a, hard floating point, cortex-a5, etc.

```
# cat /opt/poky/1.5.1/environment-setup-cortexa5hf-vfp-poky-linux-gnueabi
export PATH=/opt/poky/1.5.1/sysroots/x86_64-pokysdk-linux/usr/bin:[…]:$PATH
[…]
export CC="arm-poky-linux-gnueabi-gcc -march=armv7-a -marm -mthumb-interwork -
   mfloat-abi=hard -mtune=cortex-a5 --sysroot=/opt/[…]/cortexa5hf-vfp-[…]"
[…]
export QMAKESPEC=/opt/poky/1.5.1/sysroots/cortexa5hf-vfp-poky-linux-
   gnueabi//usr/share/qtopia/mkspecs/linux-g++
[…]
export OE_QMAKE_LIBDIR_QT=/opt/poky/1.5.1/sysroots/cortexa5hf-vfp-poky-linux-
gnueabi//usr/lib
export OE_QMAKE_INCDIR_QT=/opt/poky/1.5.1/sysroots/cortexa5hf-vfp-poky-linux-
   gnueabi//usr/include/qtopia
```

TO DO Launch qt-creator

```
# /opt/qtcreator-2.8.1/bin/qtcreator-atmel.sh
```

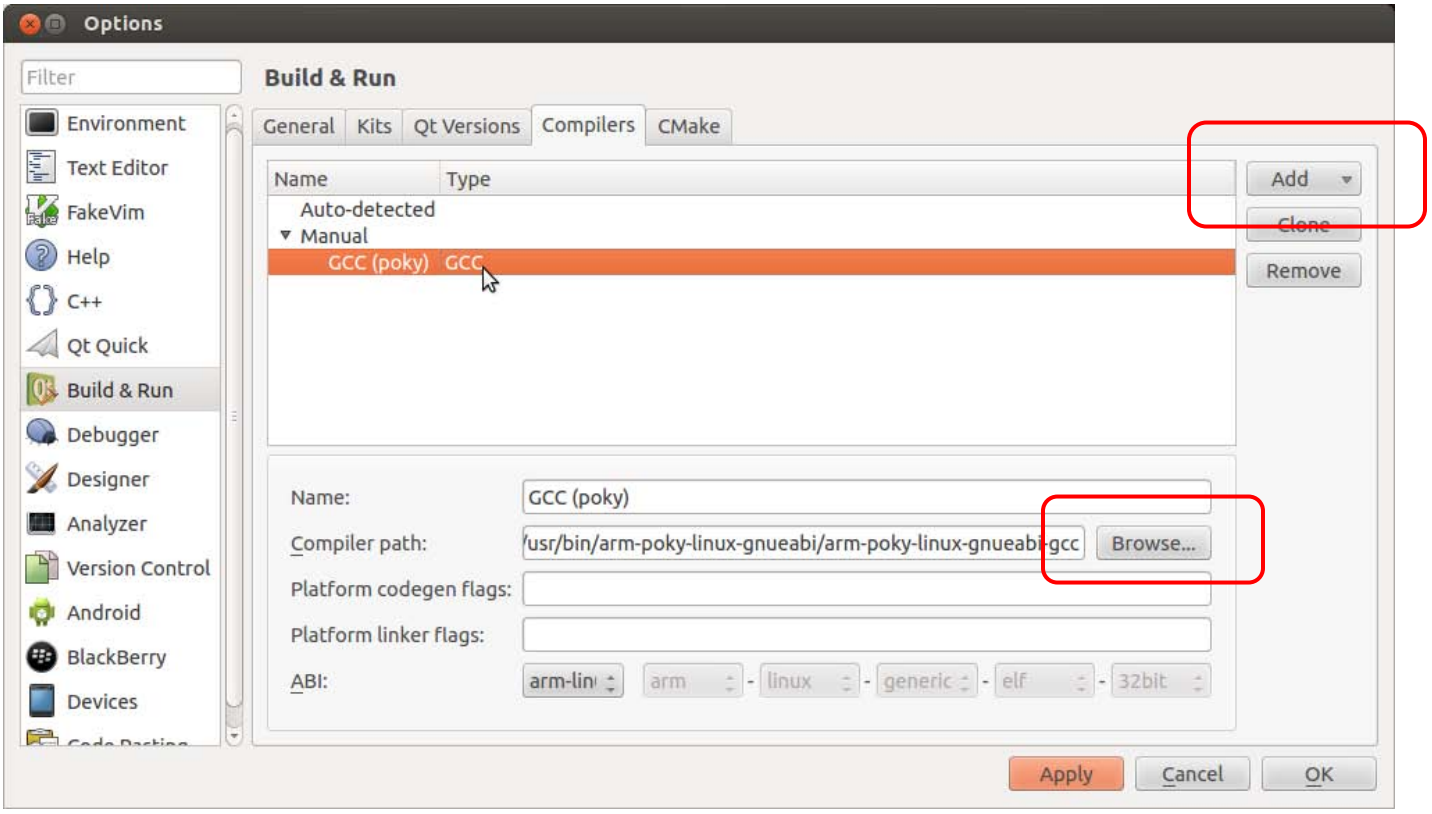


TO DO Qt-SDK cross compile environment set-up.
As we will develop a Qt application for The SAMA5D3 (ARM Cortex-A5 hard float), we need a cross-compiler tool as the target is not the one of the native Linux machine. Here we will use the Yocto (poky) Qt-Embedded SDK.

- You need to go through Qt Creator menus: Menu → Tools → Options. Then:
  1) Build&Run → 2) Compilers tab → 3) Add button → 4) choose GCC as compiler type
- Rename the compiler added as : GCC(poky)

WARNING Compiler path for 32-bit Linux machine:
/opt/poky/1.5.1/sysroots/i686-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-gcc

WARNING Compiler path for 64-bit Linux machine: :
/opt/poky/1.5.1/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-gcc

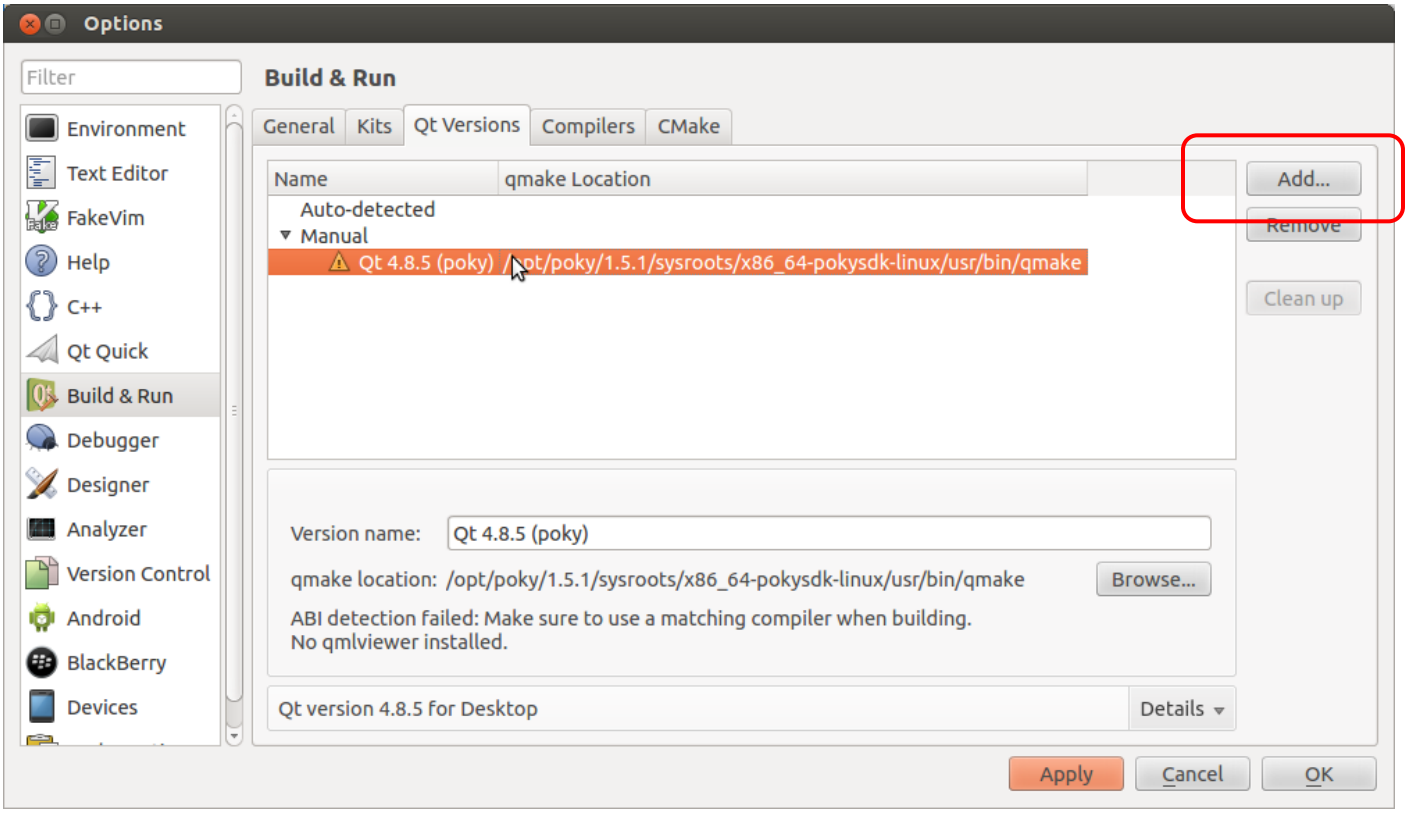


TO DO Qt qmake tool set-up.
This tool generates a Makefile based on the information in a project file.
It automates the generation of Makefiles for a QT project.

- You need to go through Qt Creator menus: Menu → Tools → Options
  1) Build&Run → 2) Qt Versions Tab → 3) Add button

WARNING qmake path for 32-bit Linux machine:
/opt/poky/1.5.1/sysroots/i686-pokysdk-linux/usr/bin/qmake

WARNING qmake path for 64-bit Linux machine
/opt/poky/1.5.1/sysroots/x86_64-pokysdk-linux/usr/bin/qmake

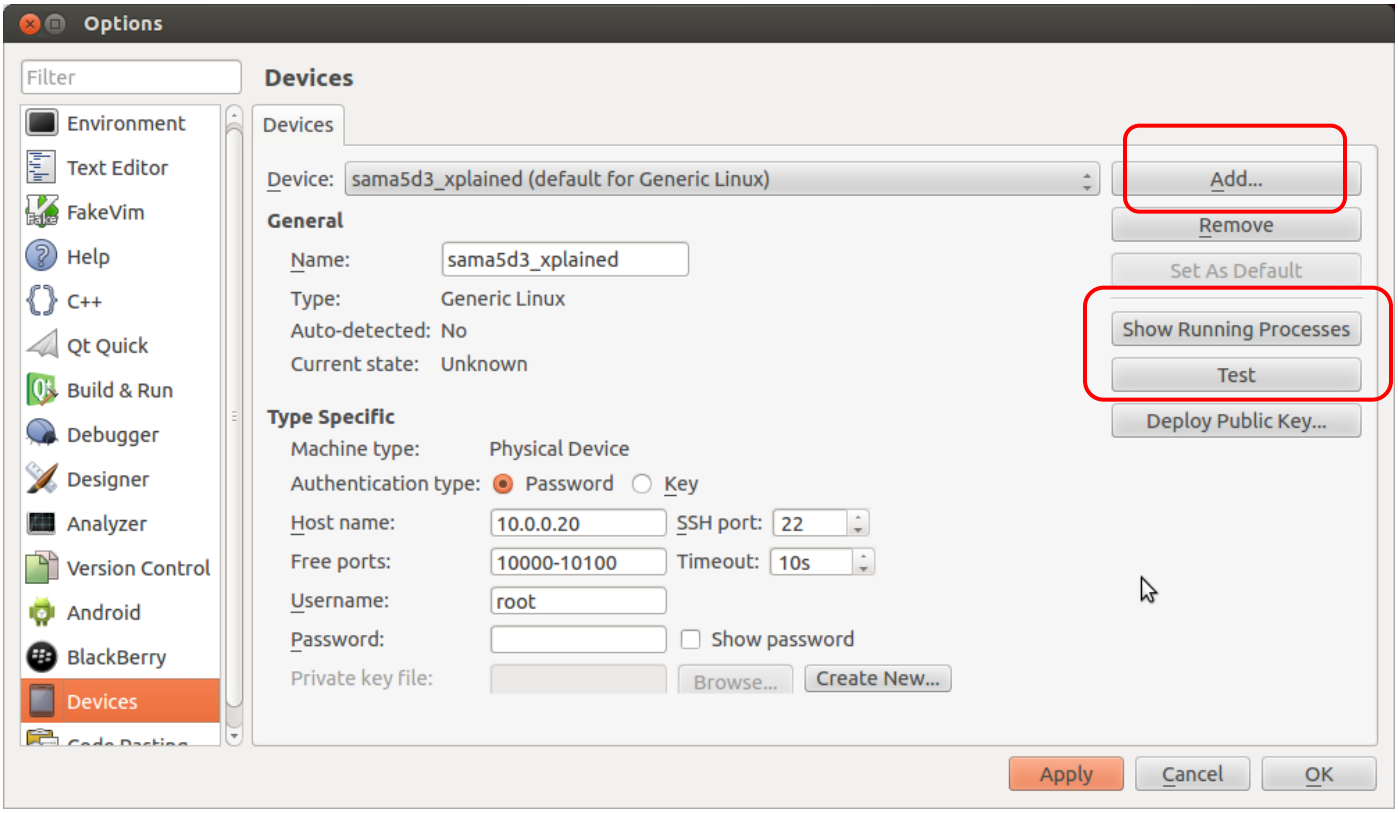


INFO The warning is not a problem for this setup step.

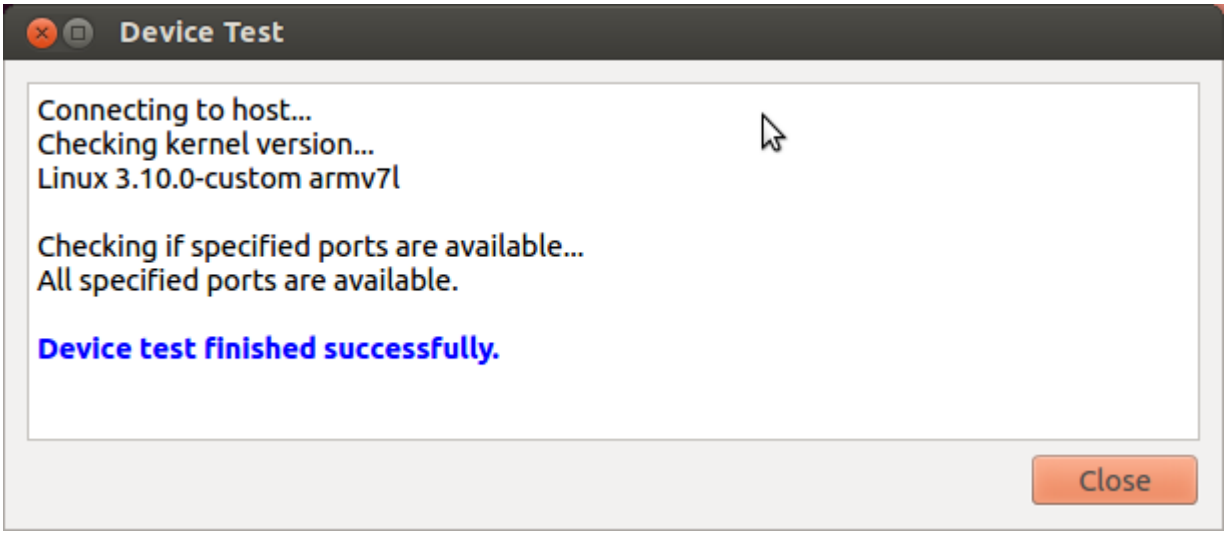
TO DO Create our own device: “sama5d3_xplained”

- You need to go through Qt Creator menus: Menu → Tools → Options. Then:
  1) Devices → 2) Add button

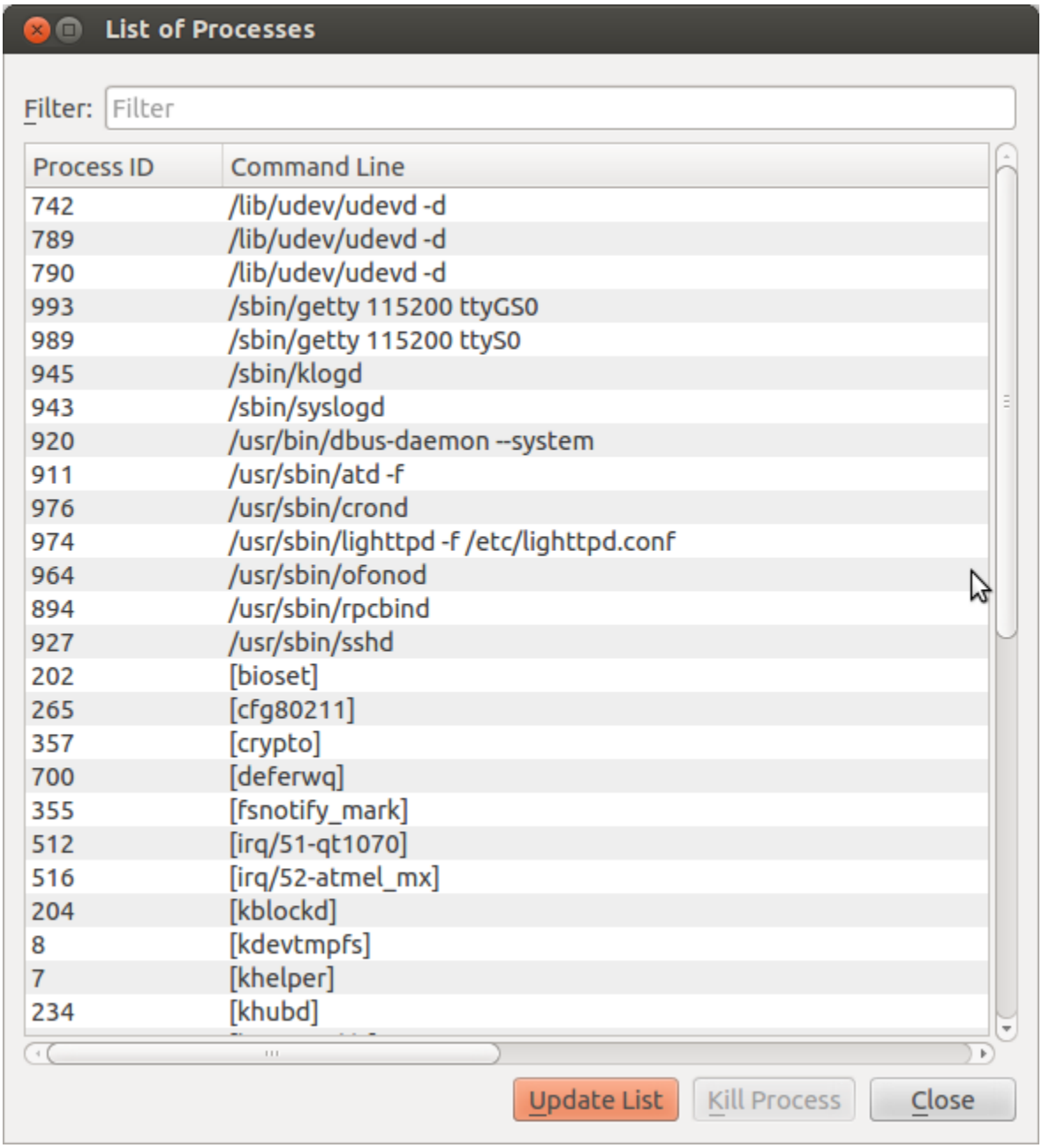
WARNING SAMA5D3_Xplained network has to be configured



- Now you can test connection with “Test” button.



- You can also look at all the processes running on your sama5d3_xplained with ‘Show Running Processes” button

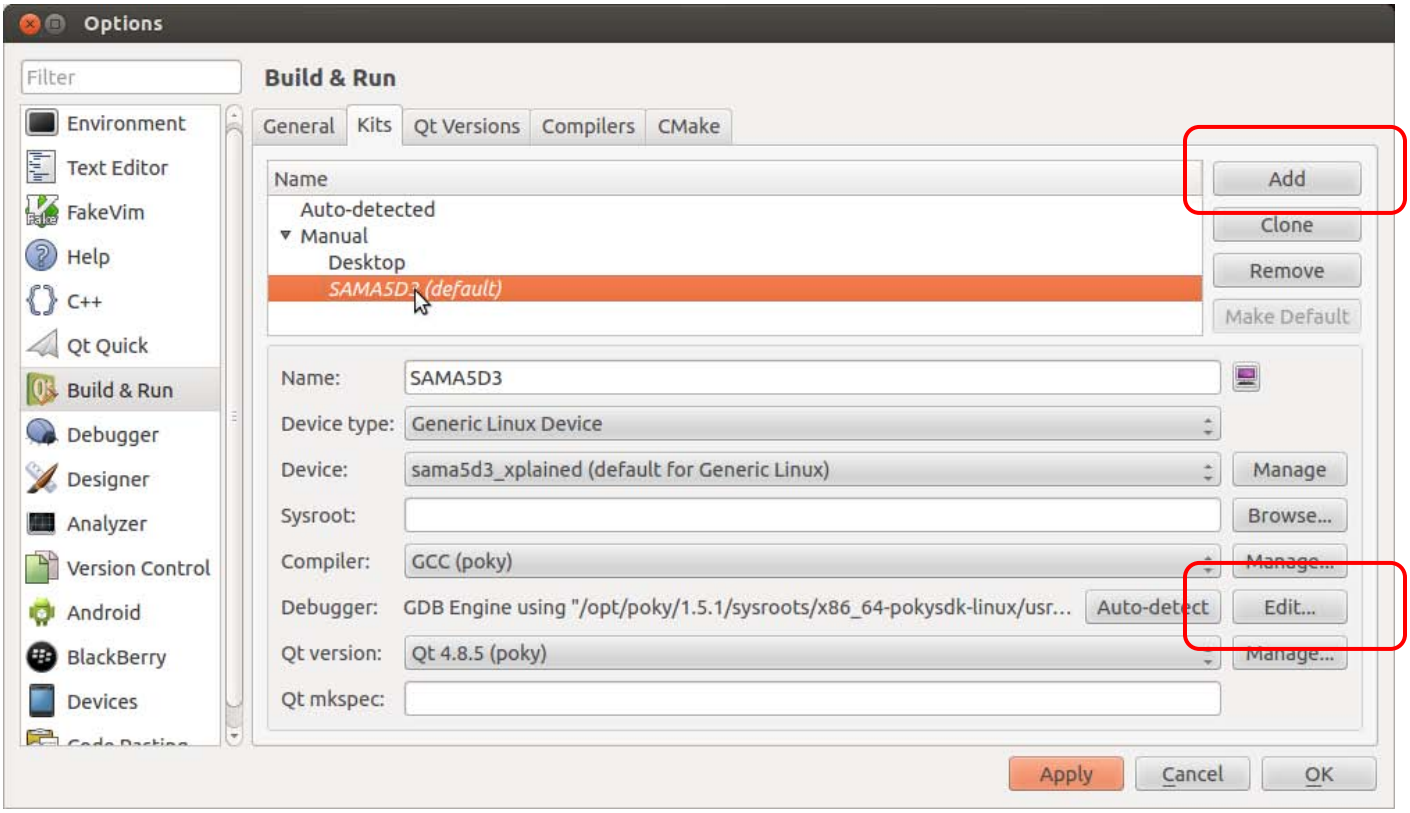


TO DO Link cross compilation information together under a “kit”

- You need to go through Qt Creator menus: Menu → Tools → Options. Then:
  1) Build&Run → 2) Kits tab → 3) Add button
- To add Debugger press “Edit” button and enter cross gdb path

WARNING Debugger path for 32-bit Linux machine:
/opt/poky/1.5.1/sysroots/i686-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-gdb

WARNING Debugger path for 64-bit Linux machine: :
/opt/poky/1.5.1/sysroots/x86_64-pokysdk-linux//usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-gdb



6. Revision History

Doc. Rev.	Date	Comments
1.0.0	07/04//2014	Initial document release
1.1	07/04/2014	Standalone document on Qt set-up

Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370

SAMA5D3-Xplained:
How to develop your Qt based application

AN-8518

Prerequisites

- **Hardware Prerequisites**
  - Atmel® SAMA5D3 Xplained XSTK
  - micro USB to USB-A cables
  - Ethernet cable
  - USB serial TTL adapter (optional)
    - FTDI TTL-232R-3V3 USB to TTL serial cable
- **Software Prerequisites**
  - Linux PC running Ubuntu 12.04 LTS
  - Qt Creator installed
- **Estimated completion time**: 120 min

Introduction

The goal of this hands-on is to create a Qt based application and deploy it on the SAMA5D3-Xplained connected to 4.3” PDA touch screen. By completing this hands-on you will

- Gain operational knowledge on Qt Creator
- Discover basics of Qt C++ development
  - Main Widgets
  - Interactions
- Debug your application on host and target
- Learn about Custom Widget library from Atmel
  - How to integrate custom widget in your code

Table of Contents

Icon Key Identifiers

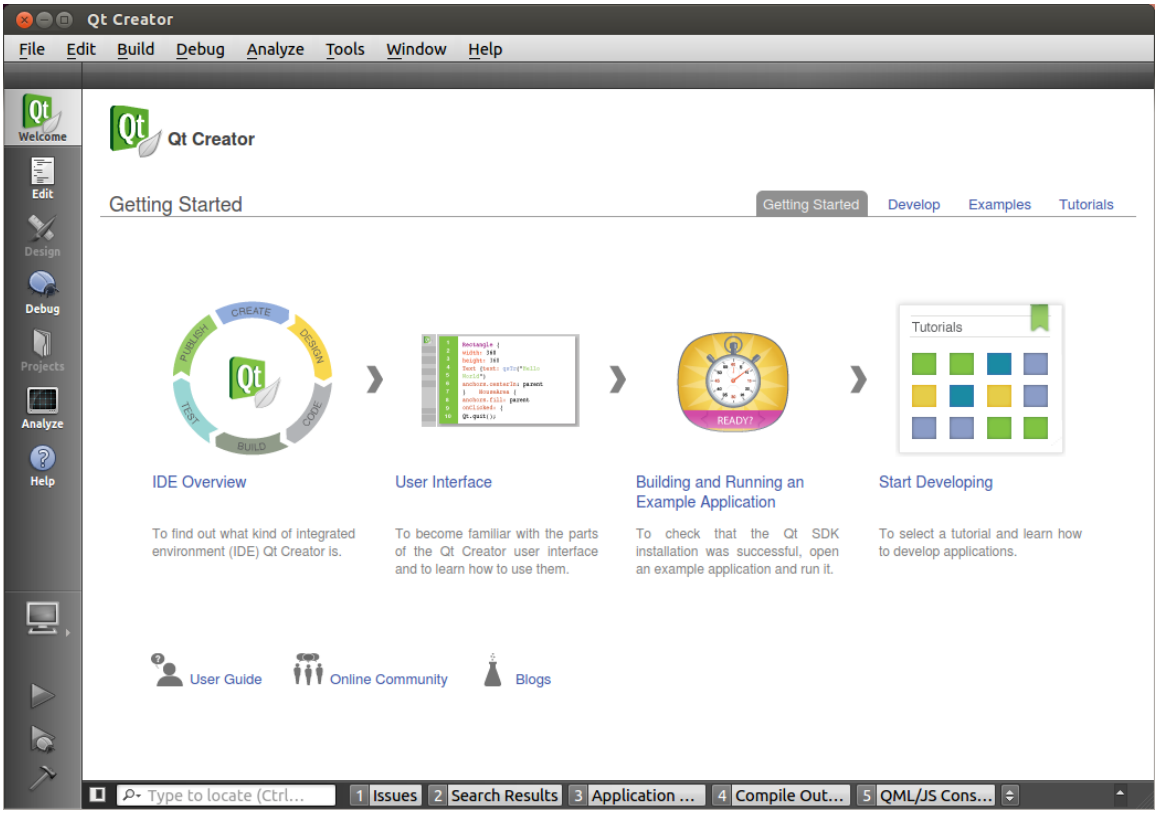
INFO	Delivers contextual information about a specific topic
TIPS	Highlights useful tips and techniques
TO DO	Highlights objectives to be completed on the Linux computer
RESULT	Highlights the expected result of an assignment step
WARNING	Indicates important information
EXECUTE	Highlights actions to be executed out of the SAMA5D3 Xplained board

1. Assignment 1: First Qt application

As Qt Creator is installed on your laptop you will now build your first Qt basic application. A step by step procedure is fully detailed and you will build a simple UI based on Qt widget and deploy it on the SAMA5D3 Xplained.
You will create a simple Qt based application using the standard Qt widget library.

TO DO Launch qt-creator

```
# /opt/qtcreator-2.8.1/bin/qtcreator-atmel.sh
```



TO DO Create your working folder "`qtlabs`"

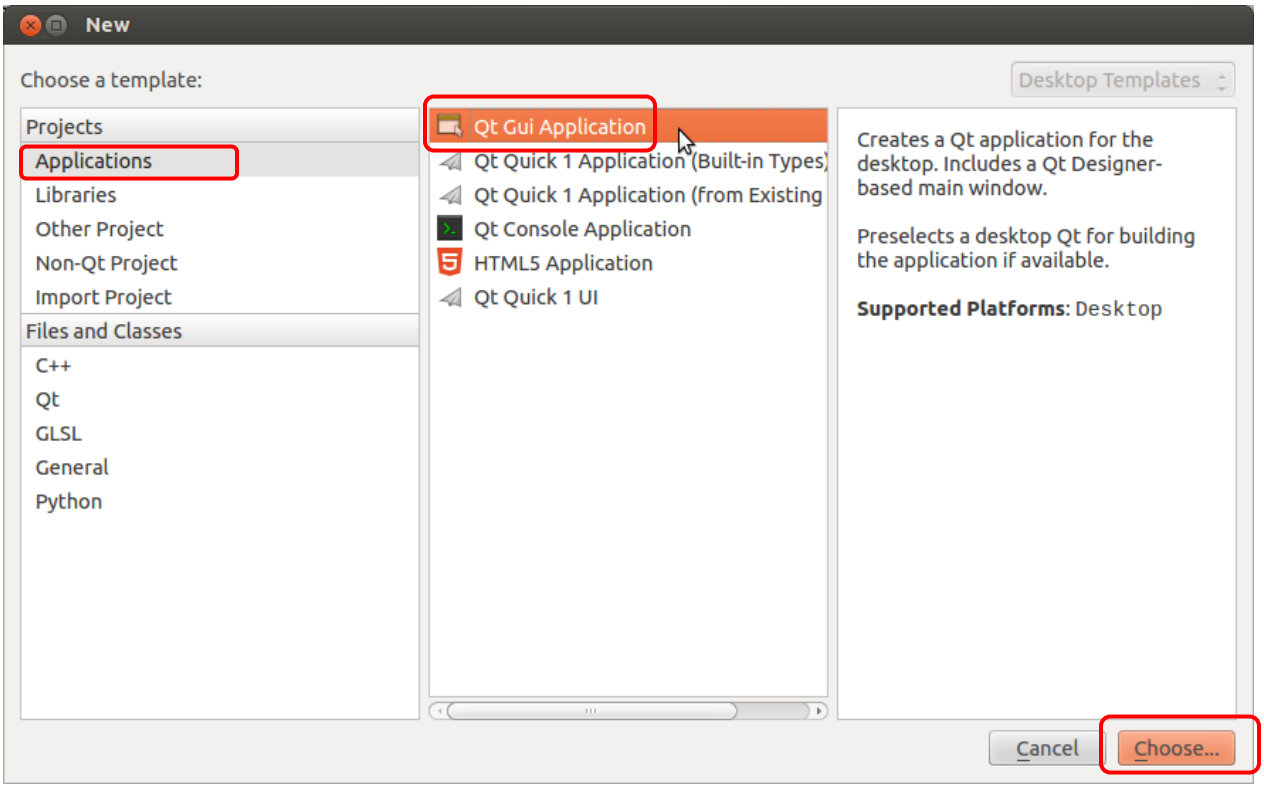
```
# cd ~/at91data
# mkdir qtlabs
```

RESULT Now you have all the setup to build your first Qt based application.

1.1 Create a simple Qt application using qt-creator provided wizards

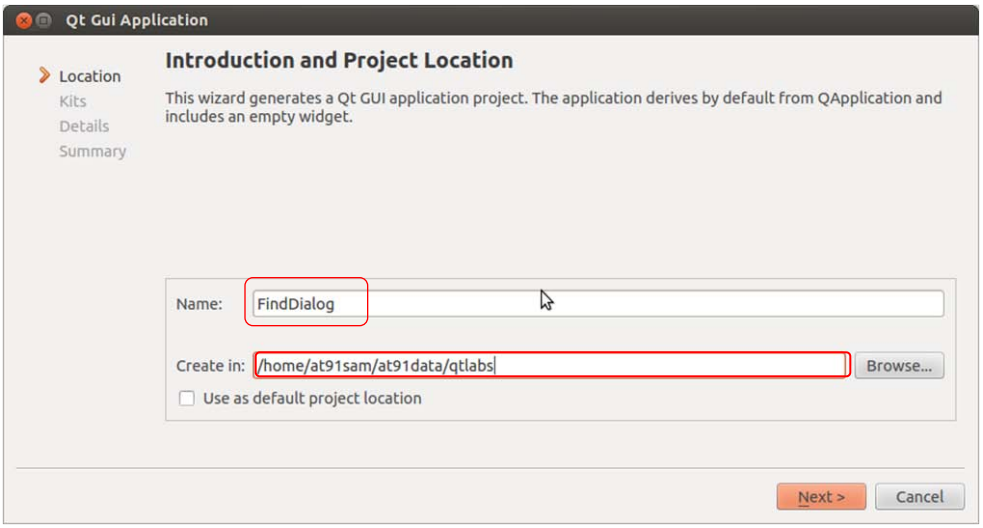
TO DO

You need to go through Qt Creator menus:
File → New File or Project: Then choose “Qt Gui Application”
Select: **Applications → Qt Gui Application**



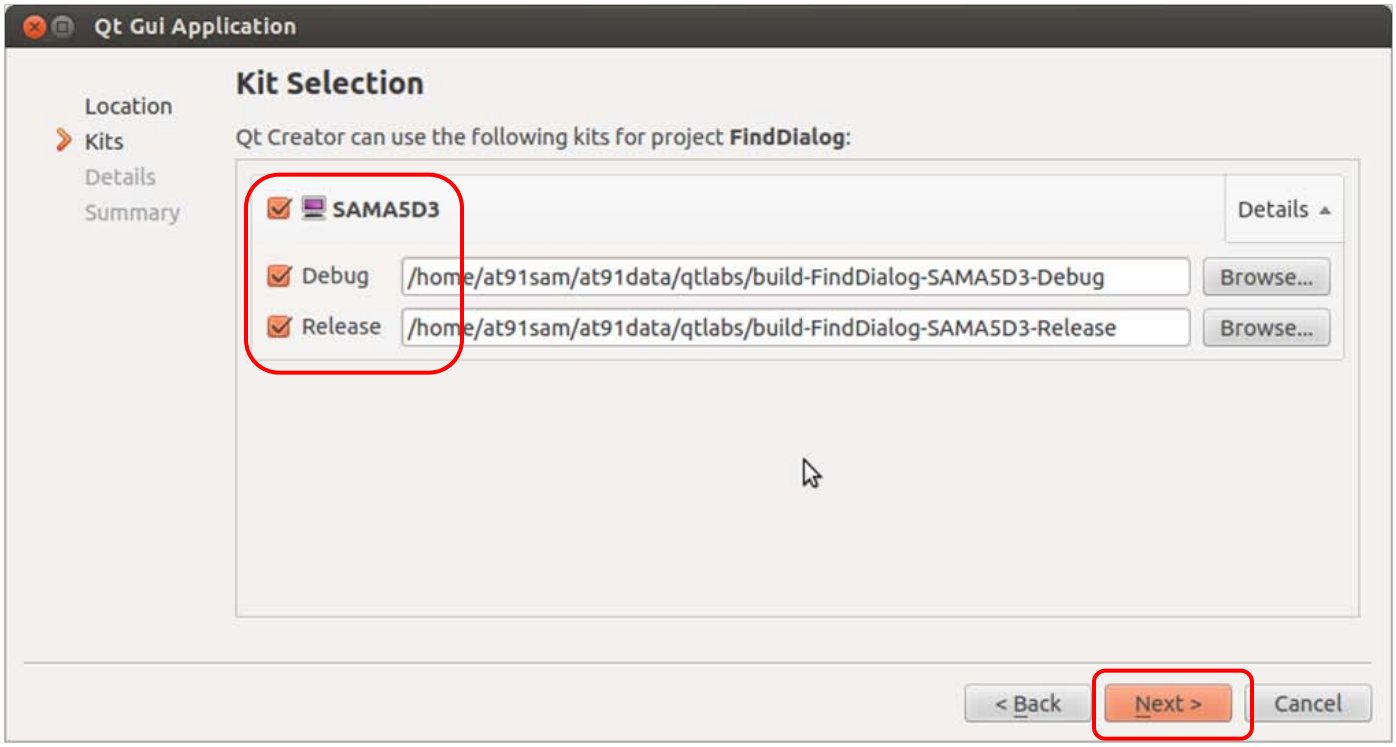
TO DO

Specify your application name and make sure you create new project in your LAB area



TO DO

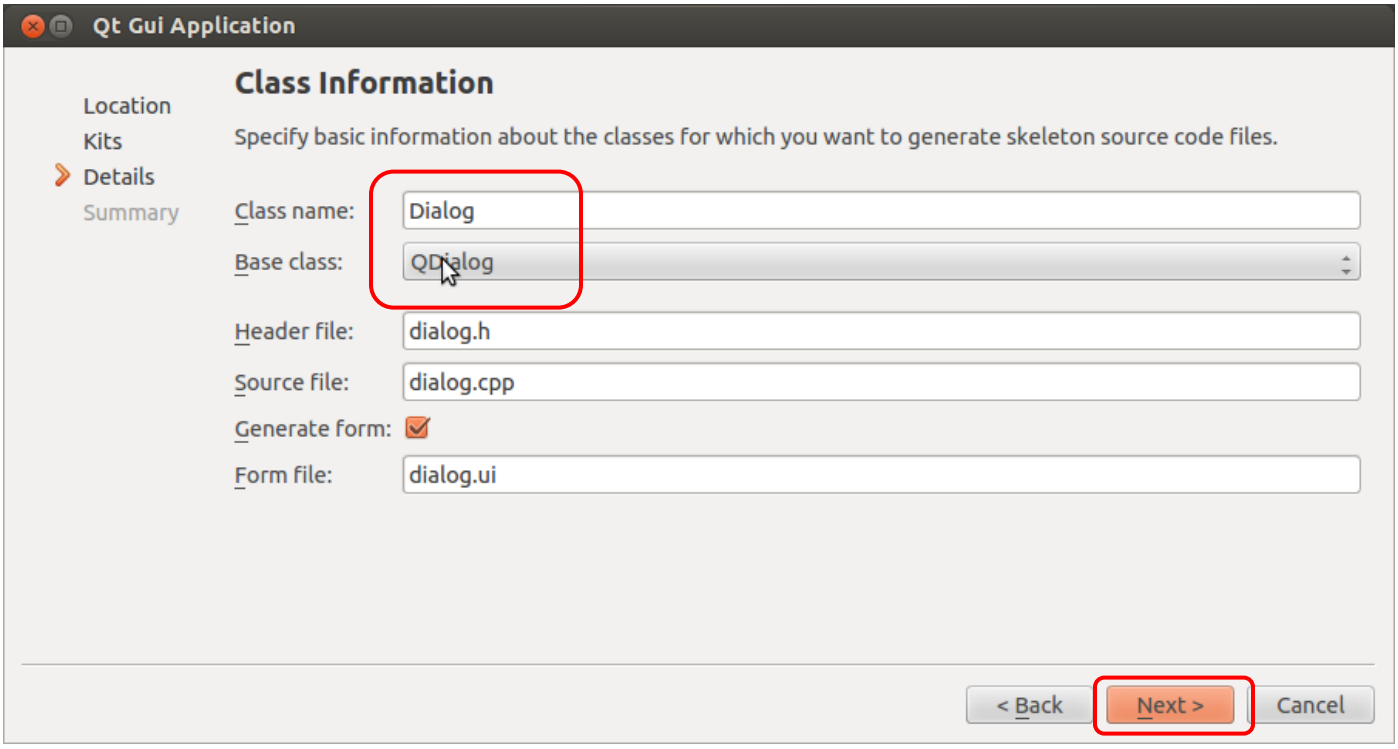
You can now select which Kit should be enabled for this new project. Make sure you select your new Kit which enables cross-compilation for the Atmel sama5d3_xplained target.



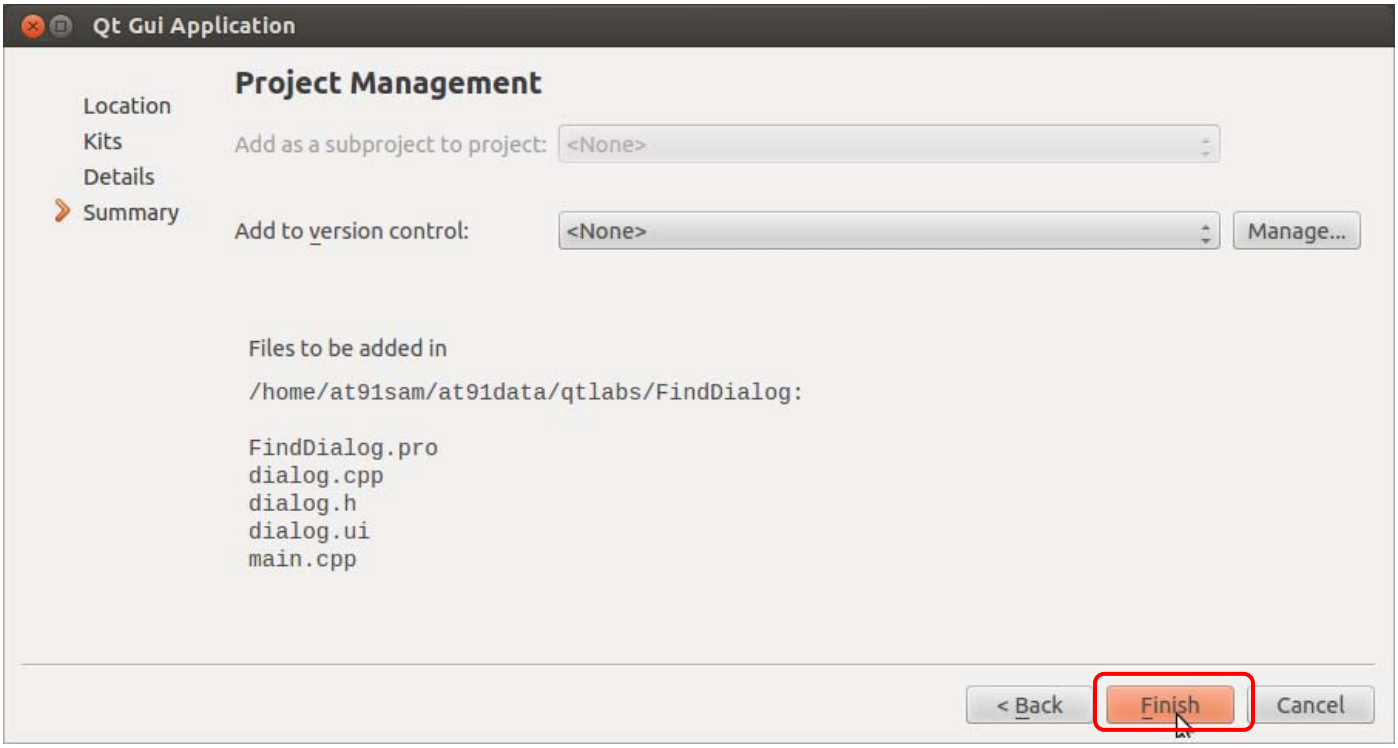
INFO

Differences between Qwidget , QMainWindow and QDialog ?
More information: http://qt-project.org/doc/qt-4.8/qtgui.html
QWidget is a base class for all other GUI elements in QtWidgets module. It can constitute a window by itself, or be part of a QLayout, or just a member of parent-child hierarchy
QDialog is usually used to – surprise! – display a temporary dialog when user input is required
QMainWindow is a convenience class that can be used as the main window of your application. It has some nice features built-in: a status bar, tool bars and a menu bar.

TO DO You can select a window decoration for your application. Let's switch to QDialog as a Base class. Finish project setup. The new project will open up in QtCreators' Editor and project browser.



TO DO You just have to click on the "Finish" button.

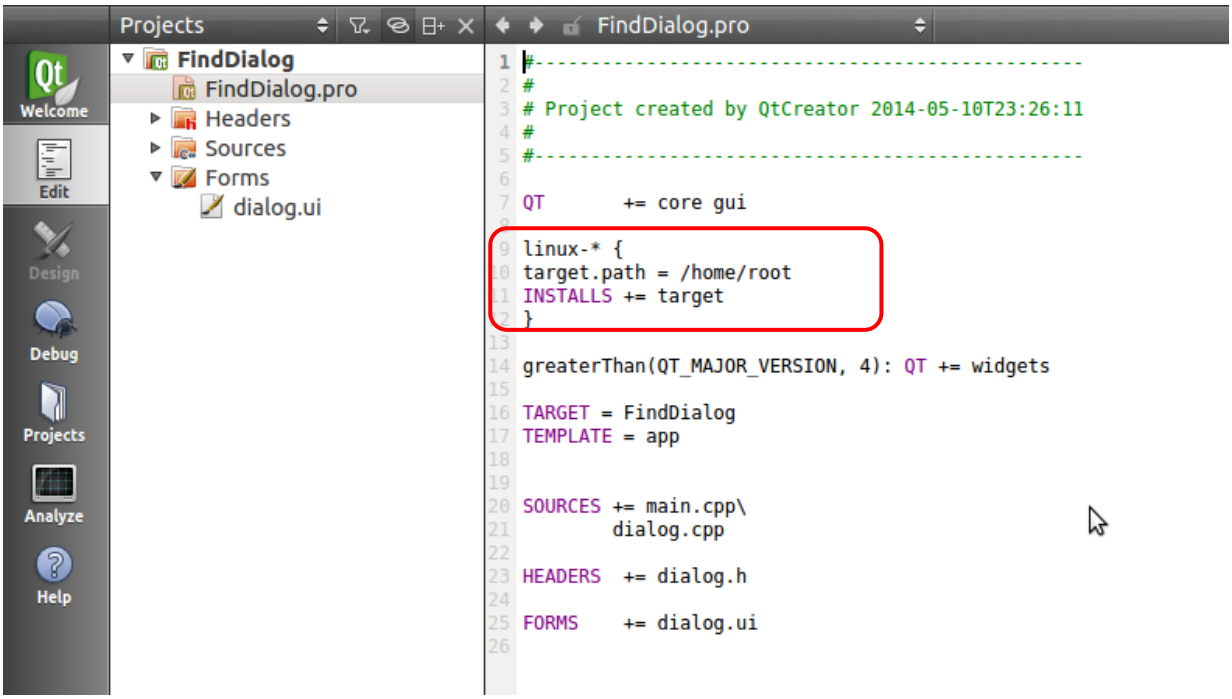


1.2 Build deploy and debug the application on the target.

TO DO

To enable deployment on the remote target the following lines have to be added `FindDialog.pro` file.

```
linux-* {
       target.path =/home/root
       INSTALLS += target
}
```

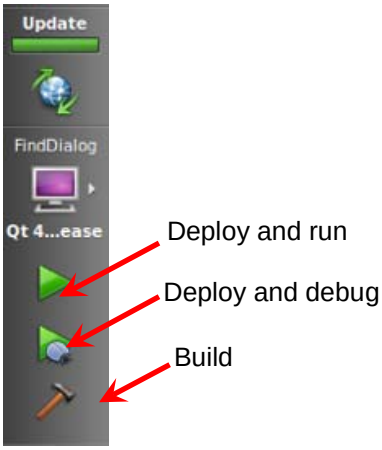


INFO

This “qmake” variable contains a list of resources that will be installed when make install or a similar installation procedure is executed. Each item in the list is typically defined with attributes that provide information about where it will be installed.
http://qt-project.org/doc/qt-4.8/qmake-variable-reference.html#installs

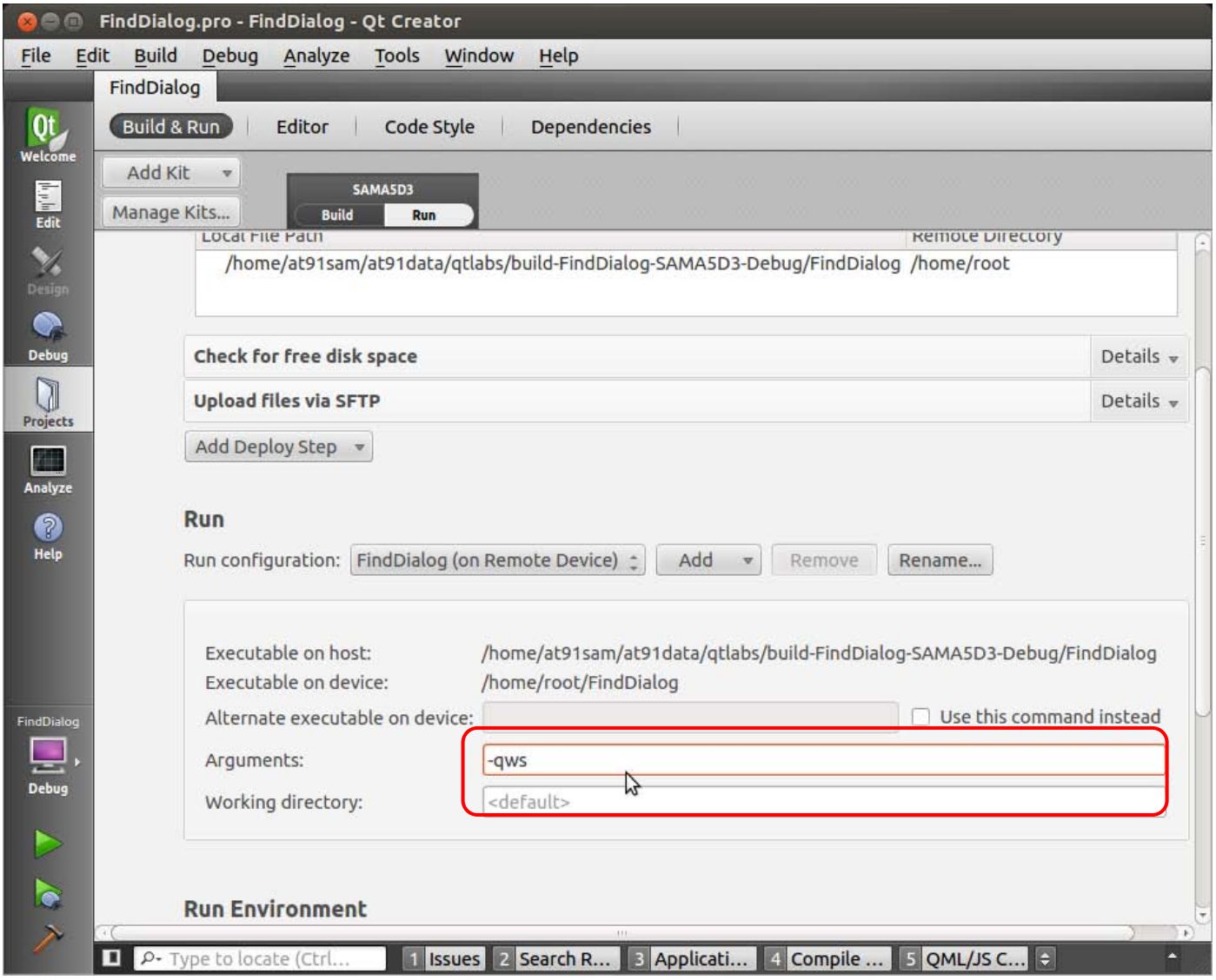
TO DO Save the project

RESULT Now “Deploy” and “Debug buttons” are active.

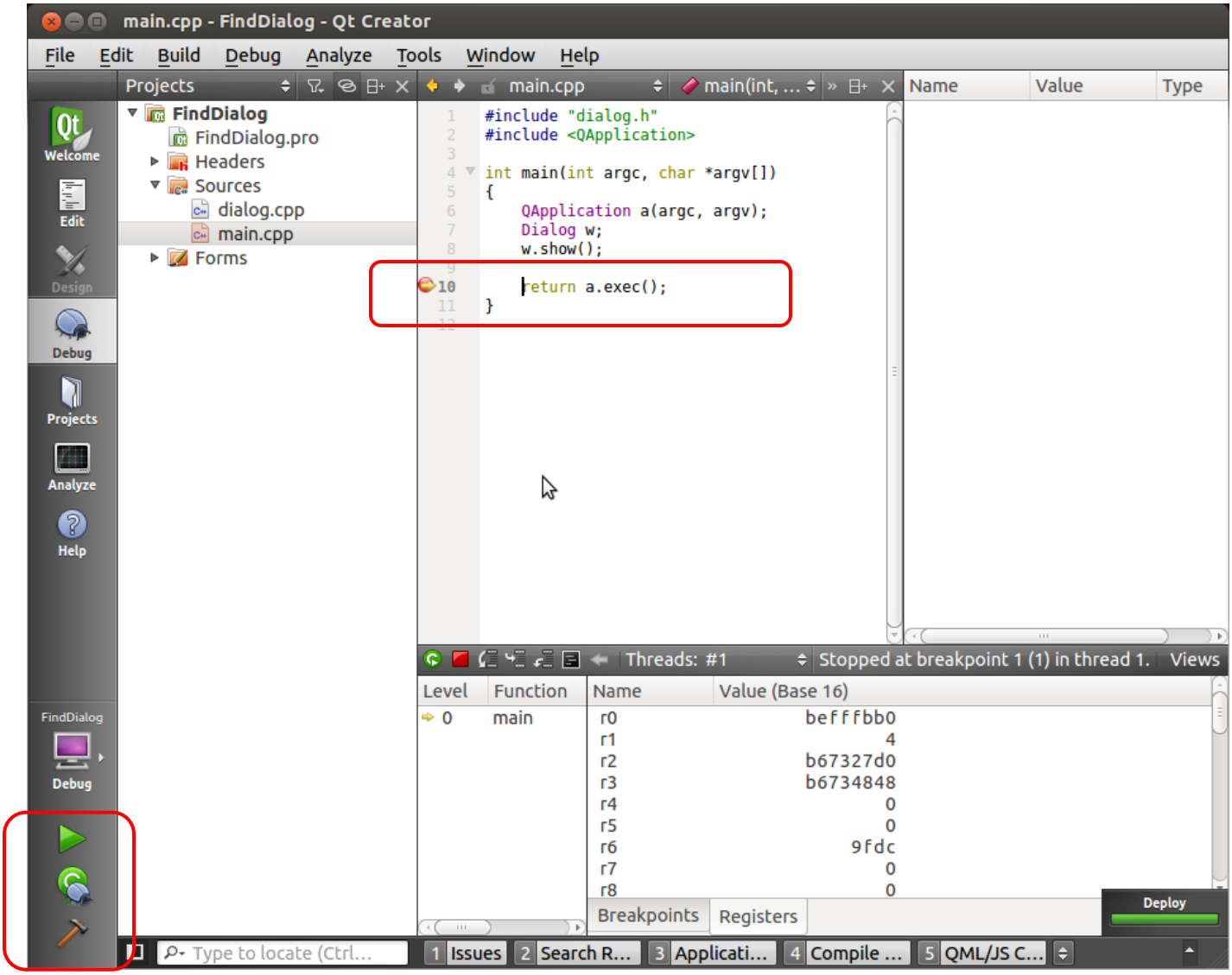


INFO A Qt for Embedded Linux application requires a server application to be running, or to be the server application itself. Any Qt for Embedded Linux application can be the server application by constructing the QApplication object with the QApplication::GuiServer type, or by running the application with the -qws command line option.
More information: http://qt-project.org/doc/qt-4.8/qt-embedded-running.html
Previously the Qt server was created by the running Qt demo on the target.

TO DO Add “-qws” option argument to our FindDialog application Select “Project View” → select run tab” then add “-qws” option in “Arguments field.
This option allows to run the UI interface on remote target.


FindDialog.pro - FindDialog - Qt Creator
File Edit Build Debug Analyze Tools Window Help
FindDialog
Build & Run
Editor
Code Style
Dependencies
Add Kit
Manage Kits...
SAMA5D3
Build
Run
/home/at91sam/at91data/qtlabs/build-FindDialog-SAMA5D3-Debug/FindDialog
/home/root
Check for free disk space
Details
Upload files via SFTP
Details
Add Deploy Step
Run
Run configuration:
FindDialog (on Remote Device)
Add
Remove
Rename...
Executable on host:
/home/at91sam/at91data/qtlabs/build-FindDialog-SAMA5D3-Debug/FindDialog
Executable on device:
/home/root/FindDialog
Alternate executable on device:
Use this command instead
Arguments:
-qws
Working directory:
<default>
Run Environment
Welcome
Edit
Design
Debug
Projects
Analyze
Help
FindDialog
Debug
Type to locate (Ctrl...
1 Issues
2 Search R...
3 Applicati...
4 Compile ...
5 QML/JS C...

TO DO You can now build, deploy and debug on target.
Before to launch the debugger you can set a breakpoint in main function.



RESULT You have deploy your first UI application on the SAMA5D3 Xplained.

2. Assignment 2: Layouts and Widget placement

Now that our project is created and can be deploy and debug on SAMA5D3 Xplained, we can start the development of our application. In this assignment we will learn how to instantiate widgets and how to use layouts to place widgets in appropriate locations on the canvas. Layouts allow applications to dynamically adjust to different screen resolutions and sizes

2.1 Free Nand Flash space on the target

Before starting the development, we have to ensure that there is enough space in Linux File system to deploy a graphical application

TO DO Now, in another window (hit CTRL-ALT-T), open your favorite serial terminal emulator with appropriate settings: **115200 bauds, 8-N-1**.

```
# picocom -b 115200 /dev/ttyUSB0
```

EXECUTE You can check the available disk space on sama5d3_xplained with df command

```
# root@sama5d3_xplained:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs          216M  210M  6.2M  98% /
ubi0:rootfs     216M  210M  6.2M  98% /
devtmpfs        124M     0  124M   0% /dev
tmpfs           124M  128K  124M   1% /run
tmpfs           124M  160K  124M   1% /var/volatile
```

TO DO We need space disk to be able to deploy large applications. We will remove unnecessary packages on sama5d3_xpained thanks to the usage of opkg command More information: http://en.wikipedia.org/wiki/Opkg

EXECUTE Execute on the sama5d5_xplained the following command

```
# root@sama5d3_xplained:~# opkg remove --force-remove homeautomation atmel-qt-
demo-init samegame smartrefrigerator minehunt qmlbrowser applicationlauncher
```

```
# root@sama5d3_xplained:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs          216M  191M   25M  89% /
ubi0:rootfs     216M  191M   25M  89% /
devtmpfs        124M     0  124M   0% /dev
tmpfs           124M  128K  124M   1% /run
tmpfs           124M  160K  124M   1% /var/volatile
```

TIPS You can blank the screen with the following command:

```
# root@sama5d3_xplained:~# cat /dev/zero > /dev/fb0
cat:write error: No space left on device
# root@sama5d3_xplained:~#
```

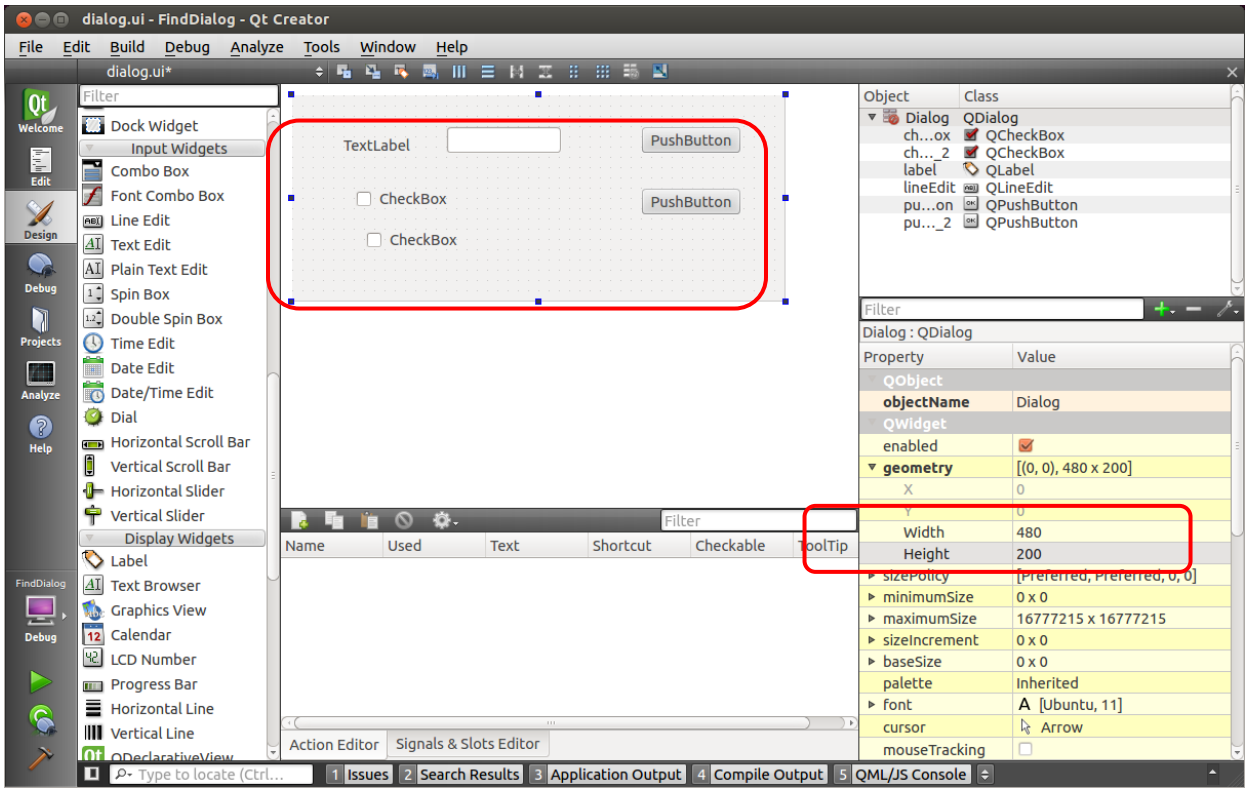
WARNING The error "`cat:write error: No space left on device`" is because we fill the frame buffer fb0 with " zero" until there is no anymore space left.

2.2 Become familiar with QtCreator's QtDesigner for C++ programming

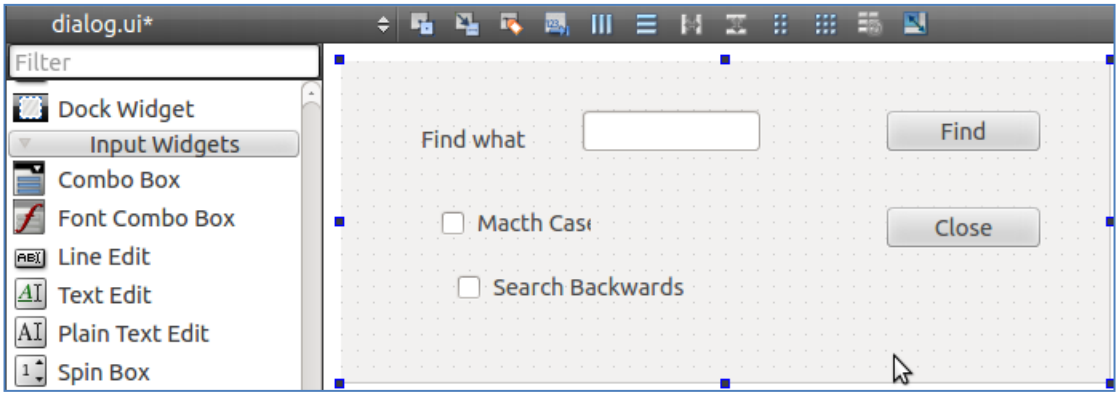
Now that we have enough space on our target , we can start our development and learn how to rough out the look of our application. For this purpose, we will instantiate widgets and use layouts to place them in appropriate locations on the application canvas. By doing so, we will create the graphical user interface of the application. You will learn how to instantiate widgets and how to use layouts to place widgets in appropriate locations on the canvas. The layouts allow applications to dynamically adjust to different screen resolutions and sizes. In this assignment we will enhance our application from the previous exercise by adding widgets.

TO DO Create your UI

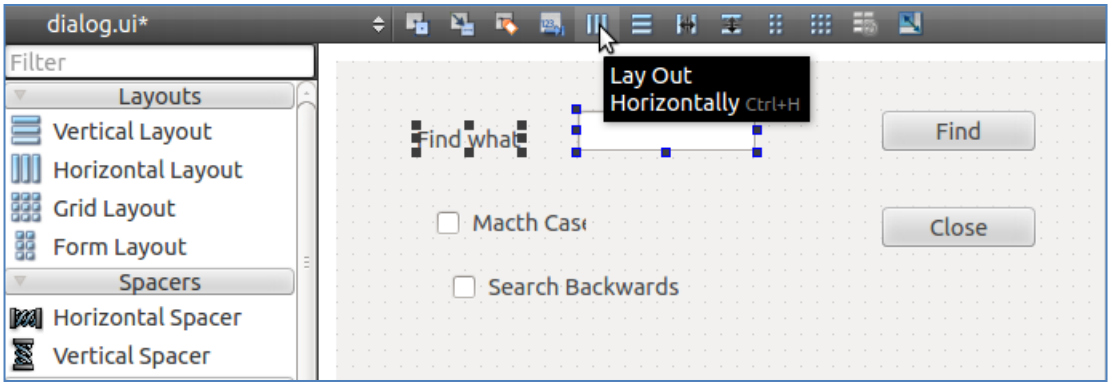
- Open the designer view by double clicking on "`dialog.ui`" file located under "`Forms`" folder.
  Drag and Drop widgets onto the canvas. Remember in this step we are roughing out the look of our application.
  Feel free to drop the widgets in desired positions but **do not spend time aligning them**.
  Adjust the size of your dialog application (Become familiar with QtCreator's QtDesigner for C++ programming)



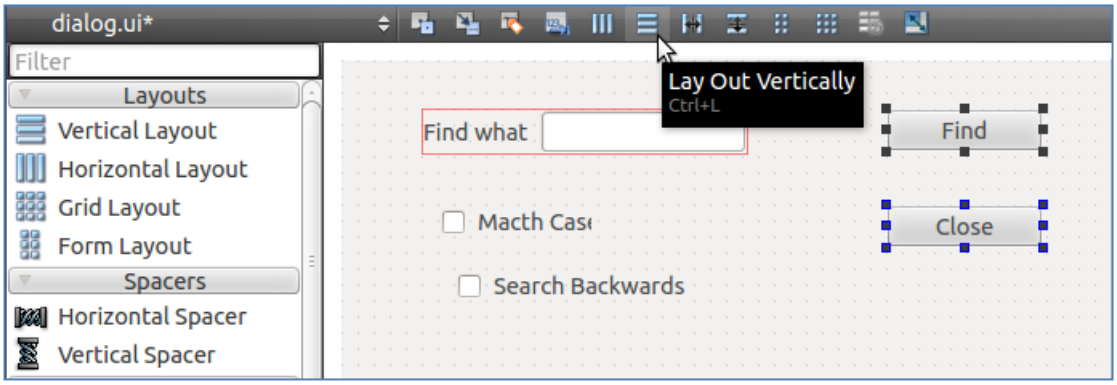
TO DO Edit labels of each objects



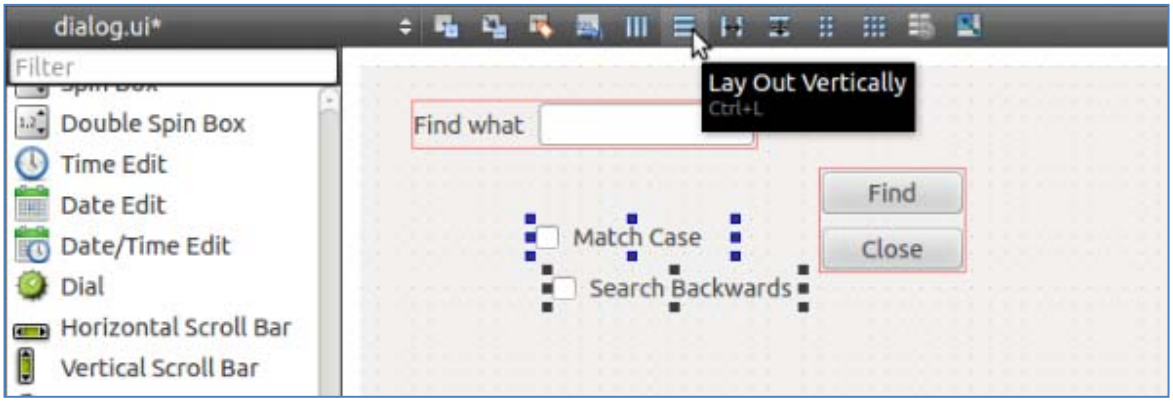
TO DO Now we can align our widgets:
Select Label “Find What” and “LineEdit”. Apply a Horizontal Layout



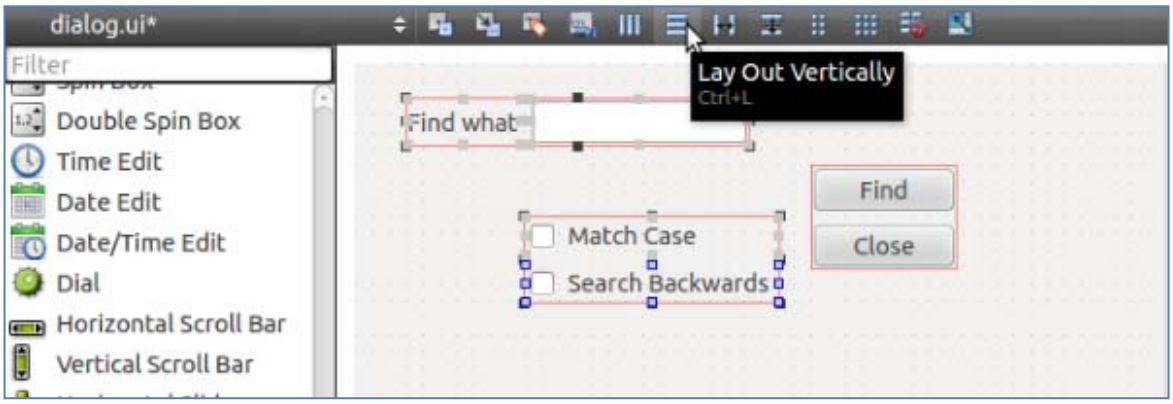
TO DO Select the two buttons and apply a Vertical Layout



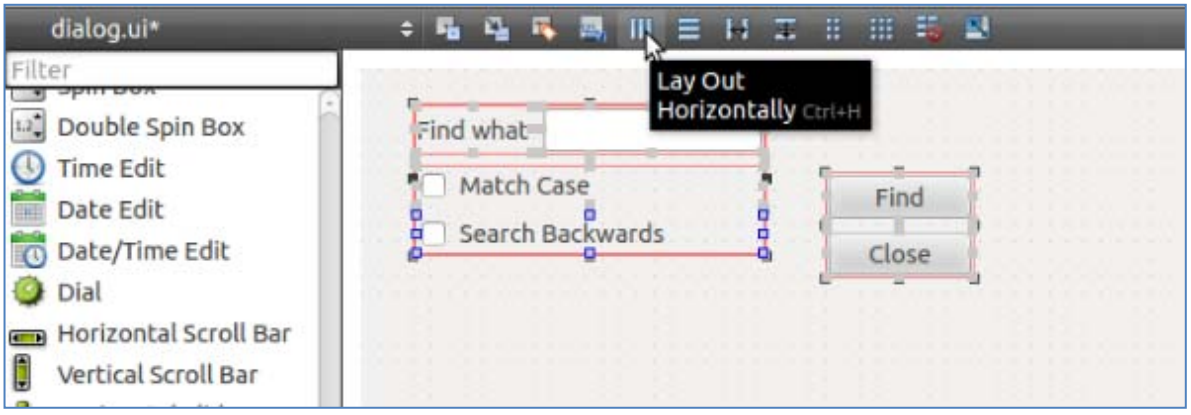
TO DO Select the 2 check boxes and apply Vertical Layout



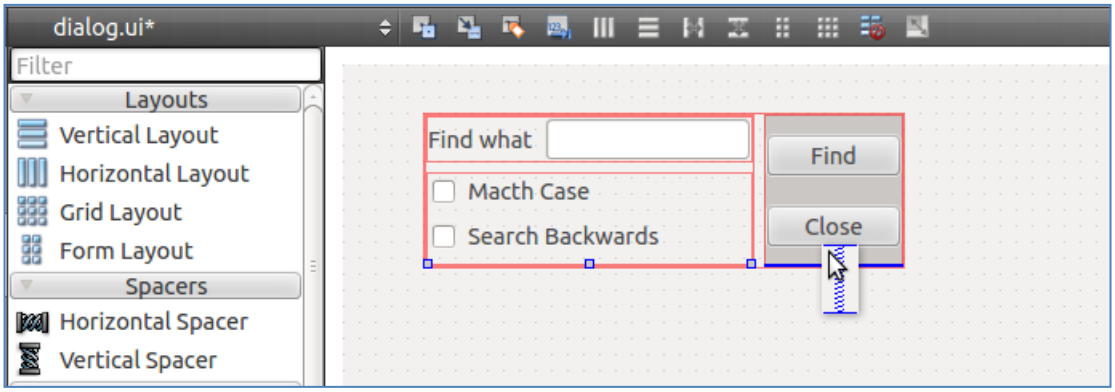
TO DO Select the two Layouts “Find what” and “Checkboxes” and apply Vertical Layout



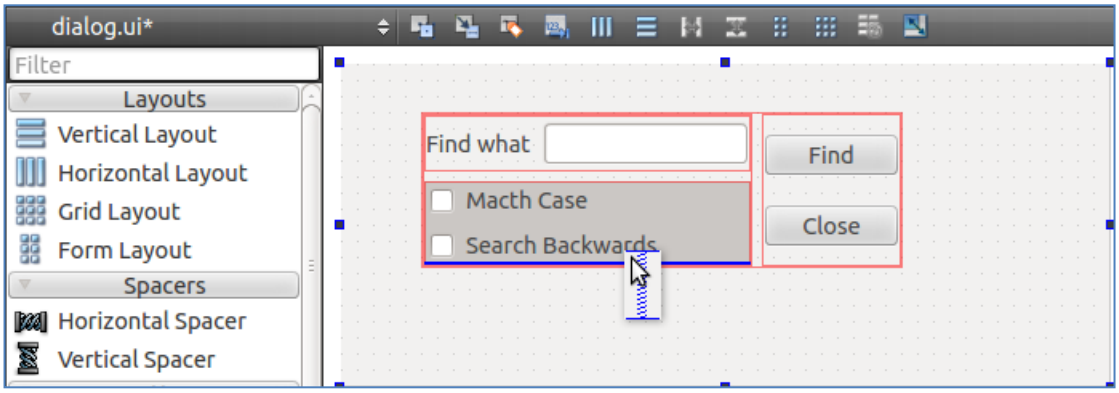
TO DO Select all Layouts and apply Horizontal Layout



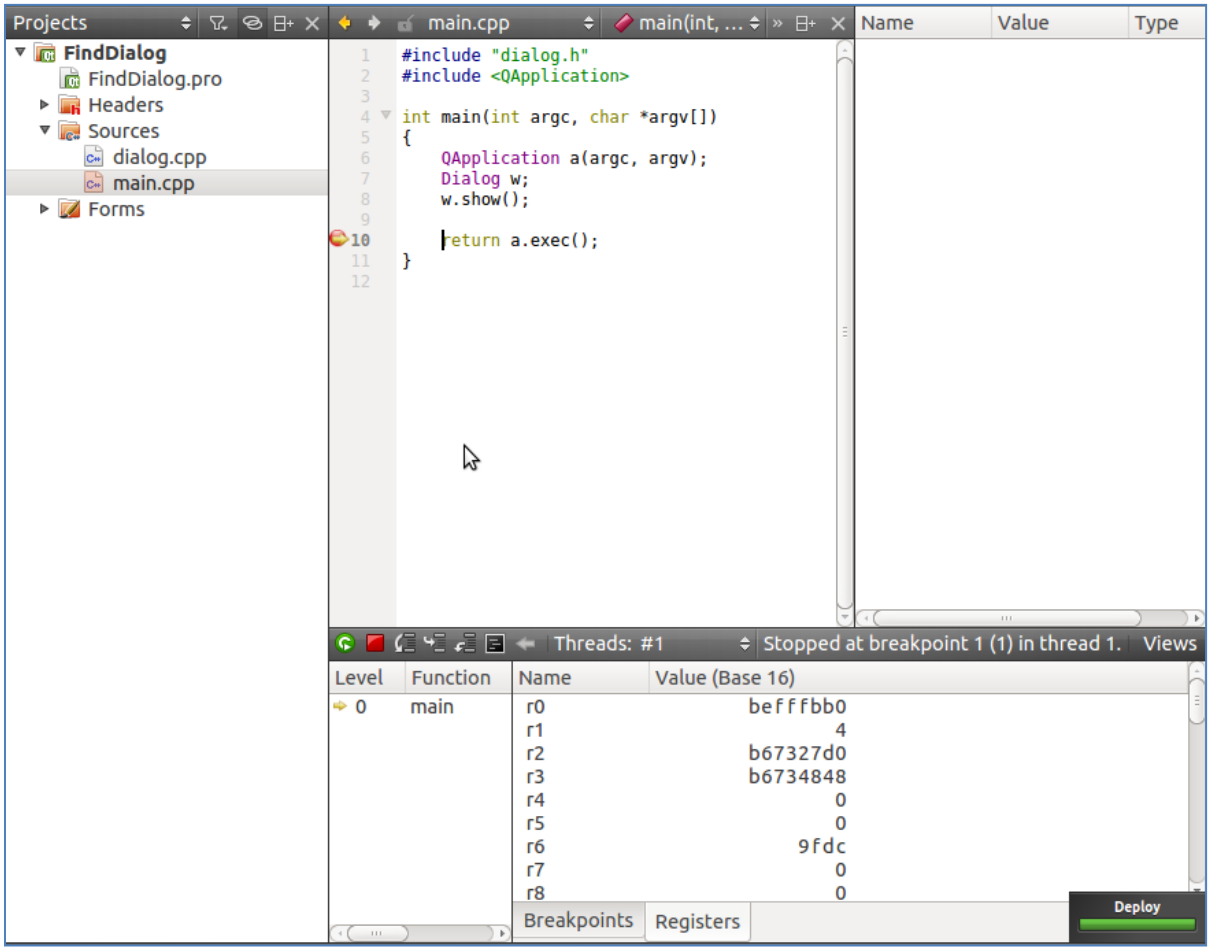
TO DO Add a spacer widget under the “Close” button and checkboxes for better alignment



TO DO Add a spacer widget under checkboxes for better alignment



TO DO At this point feel free to compile and run your code on host and target.

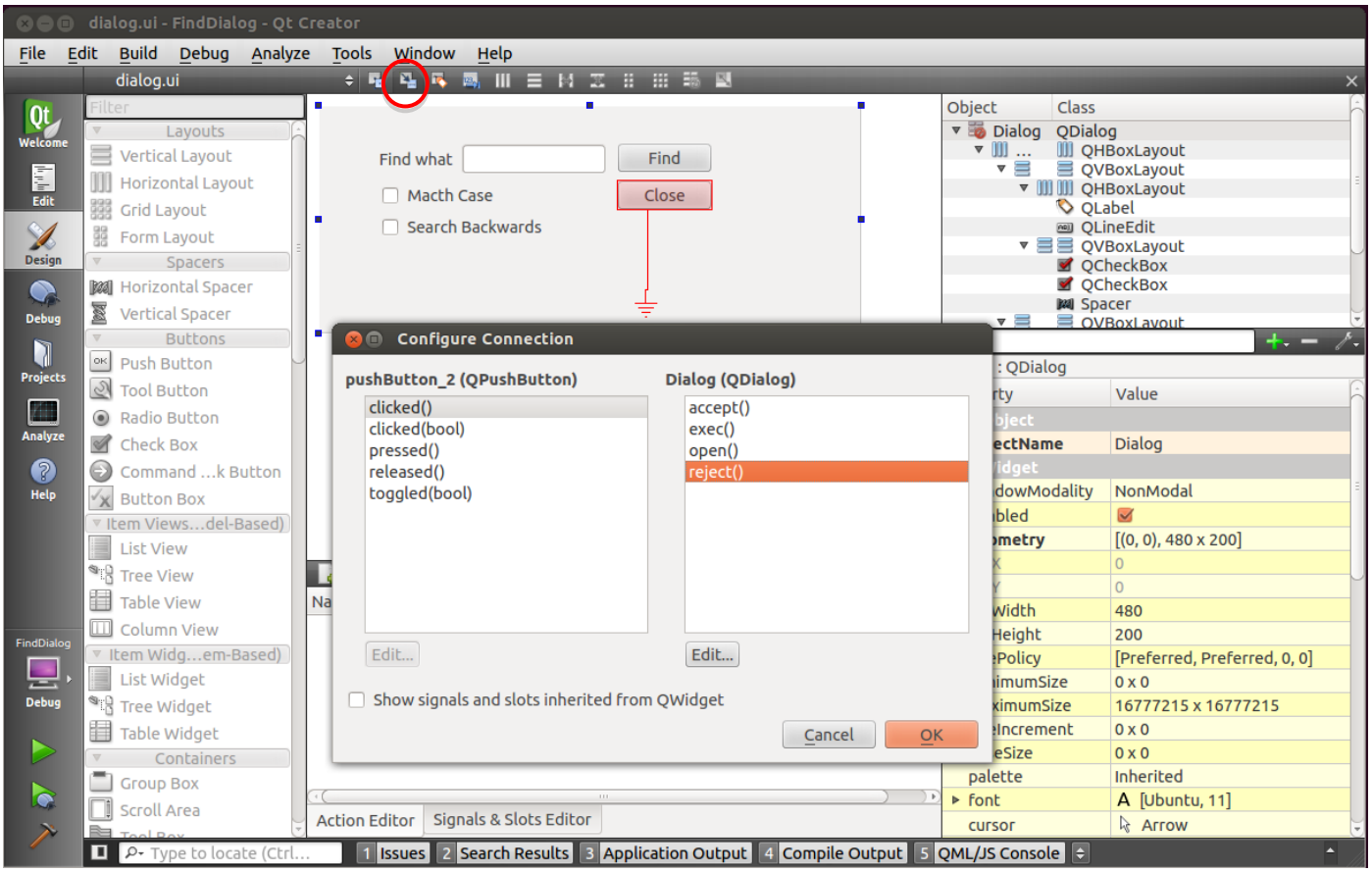


3. Assignment 3: Widget communication via Signals and Slots

Interactions between widgets are implemented through the signals/slots mechanism.
This assignment will introduce you to some various ways signal and slots can be added to a Qt application.

TO DO Switch to Signals/Slots editor on the Toolbar:

TO DO On the canvas, click and hold on the close button and then release the mouse on the main window area. This brings up the Connection Configuration dialog where you can select which signal will be connected to which slot. Please select **clicked()** signal and **reject()** slot.

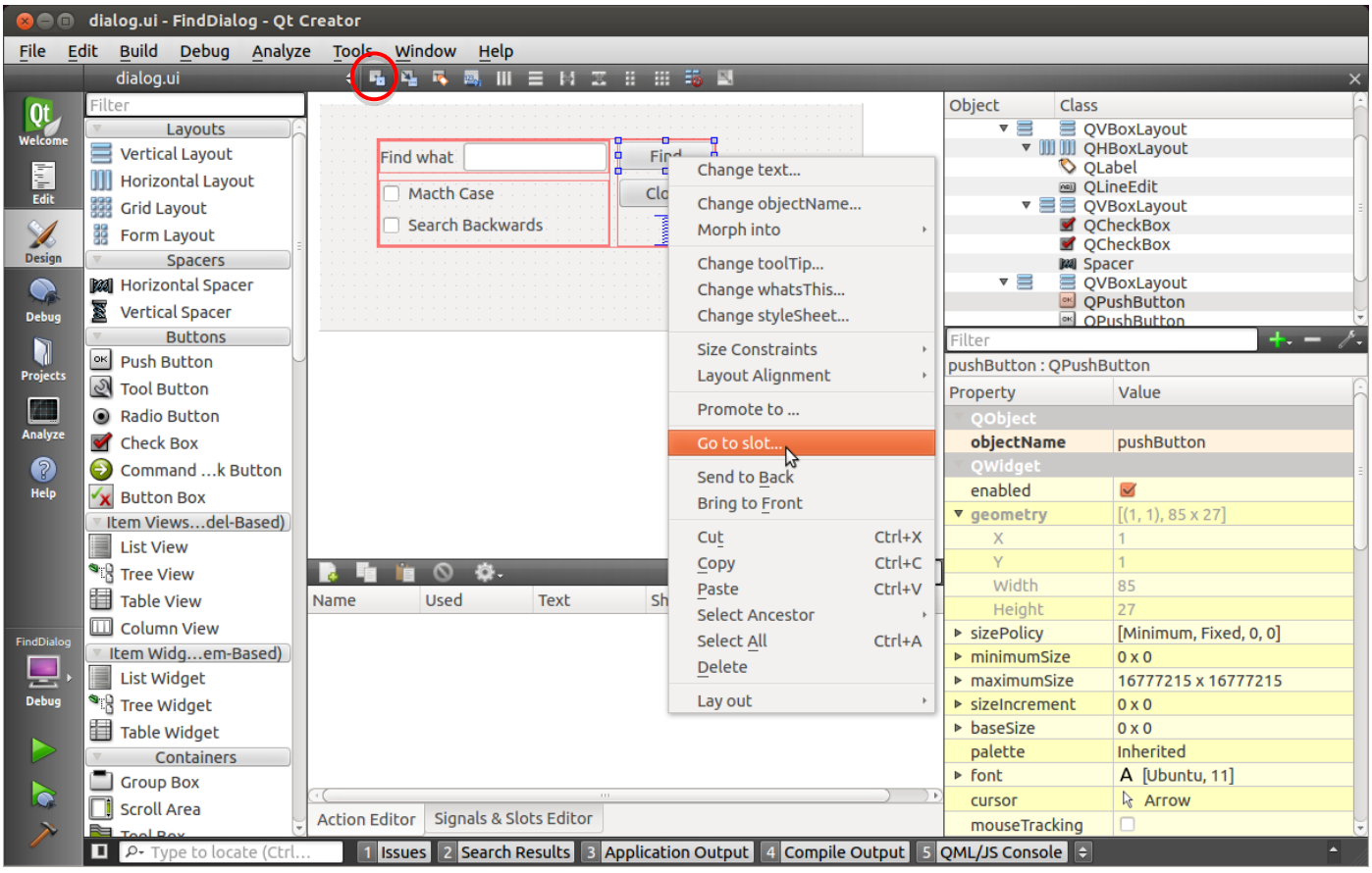


TO DO Compile and run the code, press **Close Button**. What is the behavior?

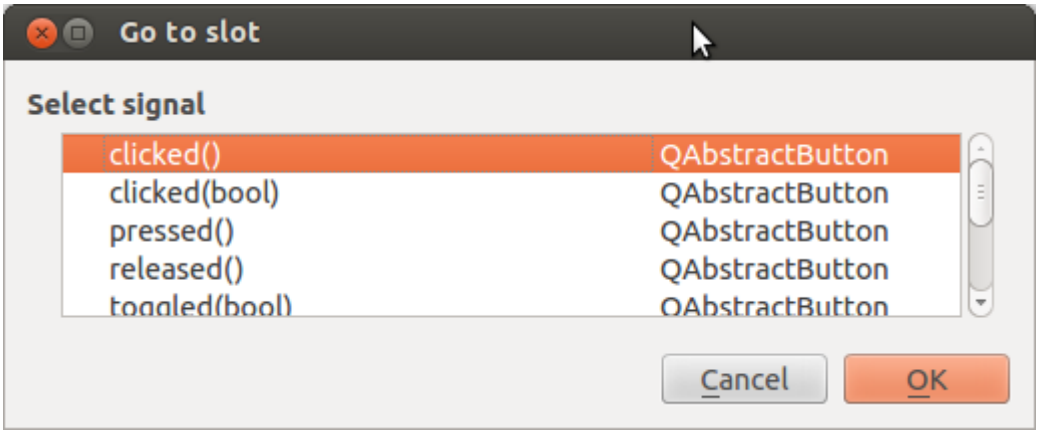
3.1 Generate code stubs for generated clicked() signals

In this part of this exercise, we will generate code stubs for generated **clicked**() signals. This way we can trigger certain code when we press a button.

TO DO Switch back to Widget Editing view and right mouse click on a Find button.
Select the "**Go to slot…**" option. Choose "**clicked()**" signal.
This will generate a code stub and QtCreator will switch to Editor View.

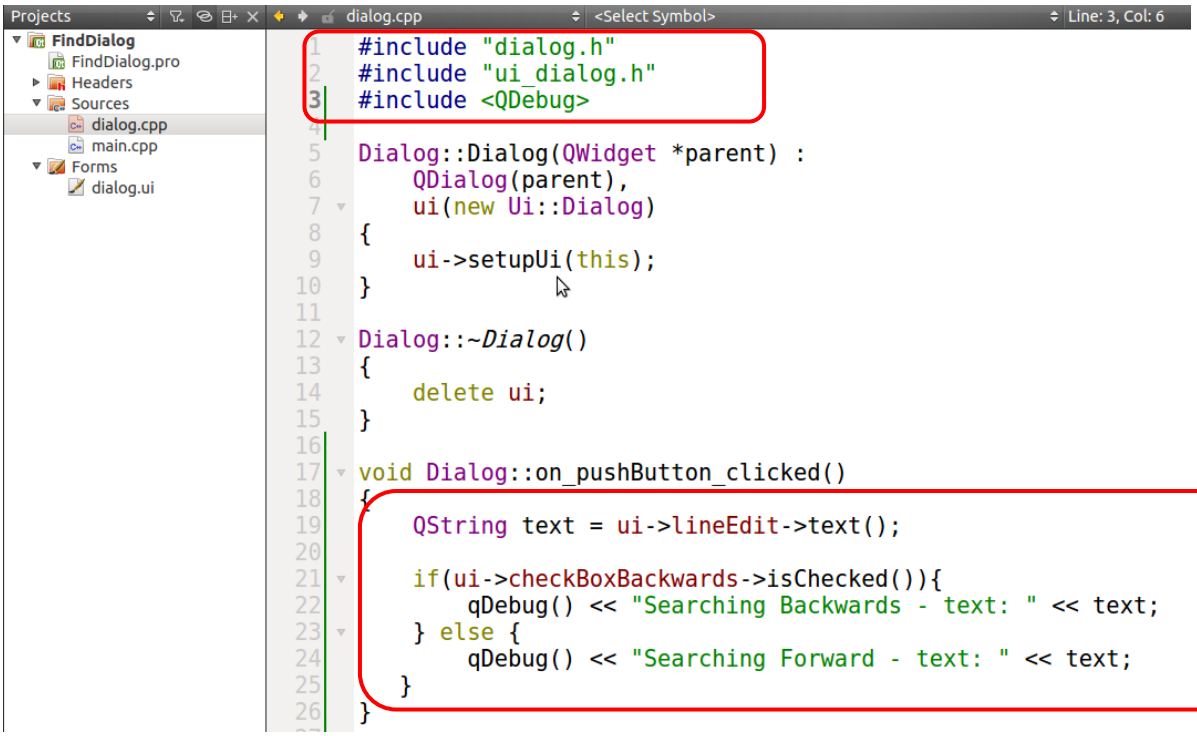


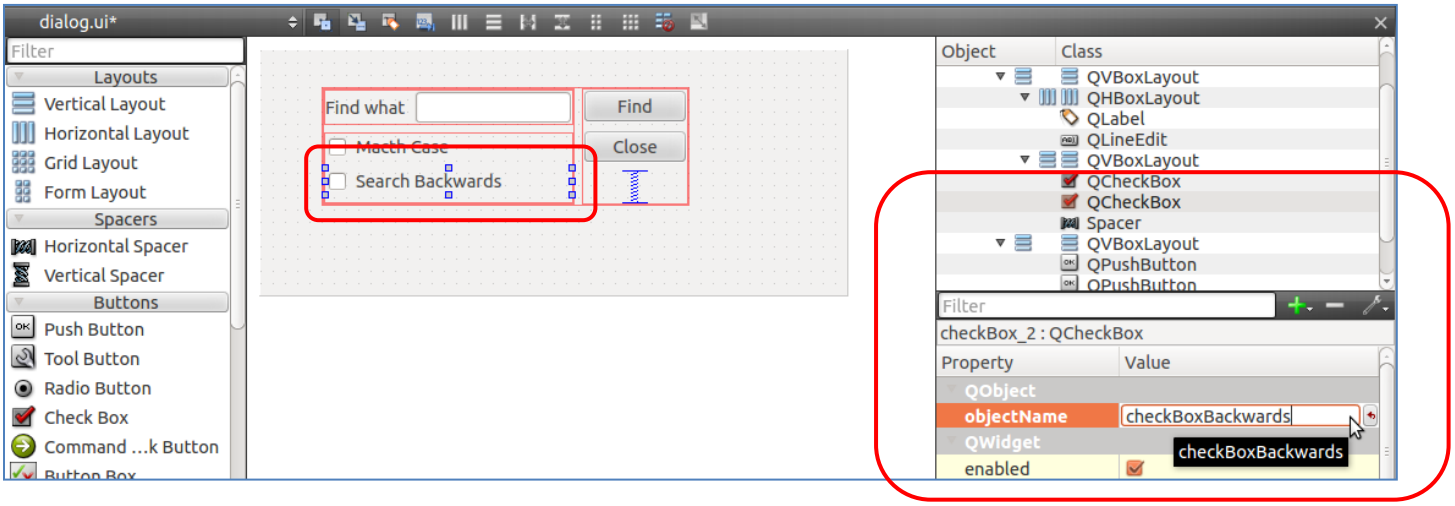
TO DO Choose "**clicked()**" signal



WARNING The below code assumes that objects have been renamed. In particular the name corresponding to **“Backwards Checkbox”** is “**checkBoxBackwards**”.
You can rename any object with the object explorer.

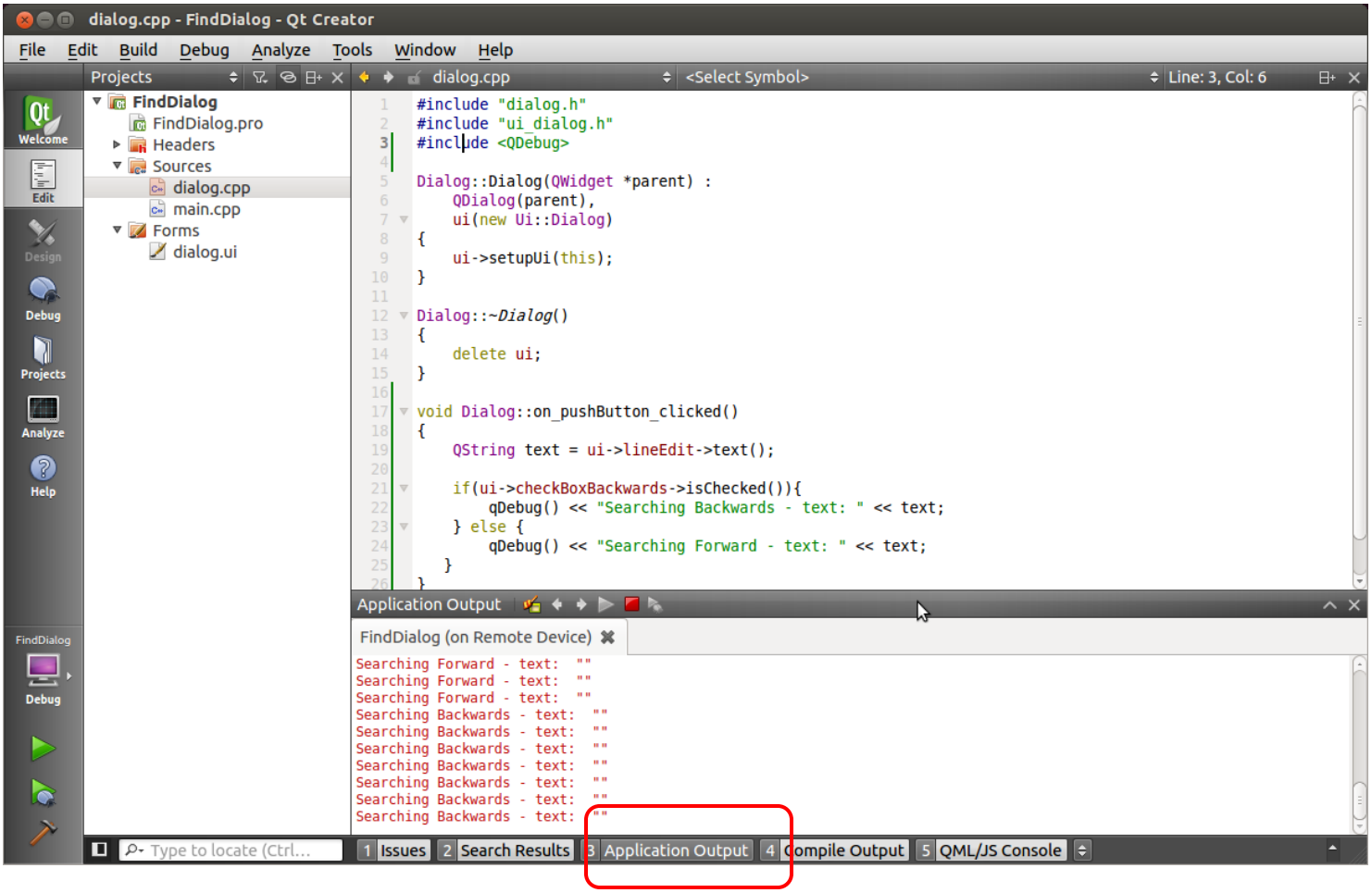
TO DO Add the following stub code to your program:





TO DO

Execute your program, open **Application Output** tab and click the Find button. Can you see output in the application's console window?



4. Assignment 4 – Using Atmel Widget Library

Atmel provides a library of free widgets that can be used by Atmel customers in their own applications. In this assignment you will learn how to add an existing widget to your own software. Those widgets are located as source files in Atmel Qt demo archives under directory “CustomWidgets”.
All archives are available on Linux4SAM FTP server: ftp://ftp.linux4sam.org/pub/demo/qtdemo/

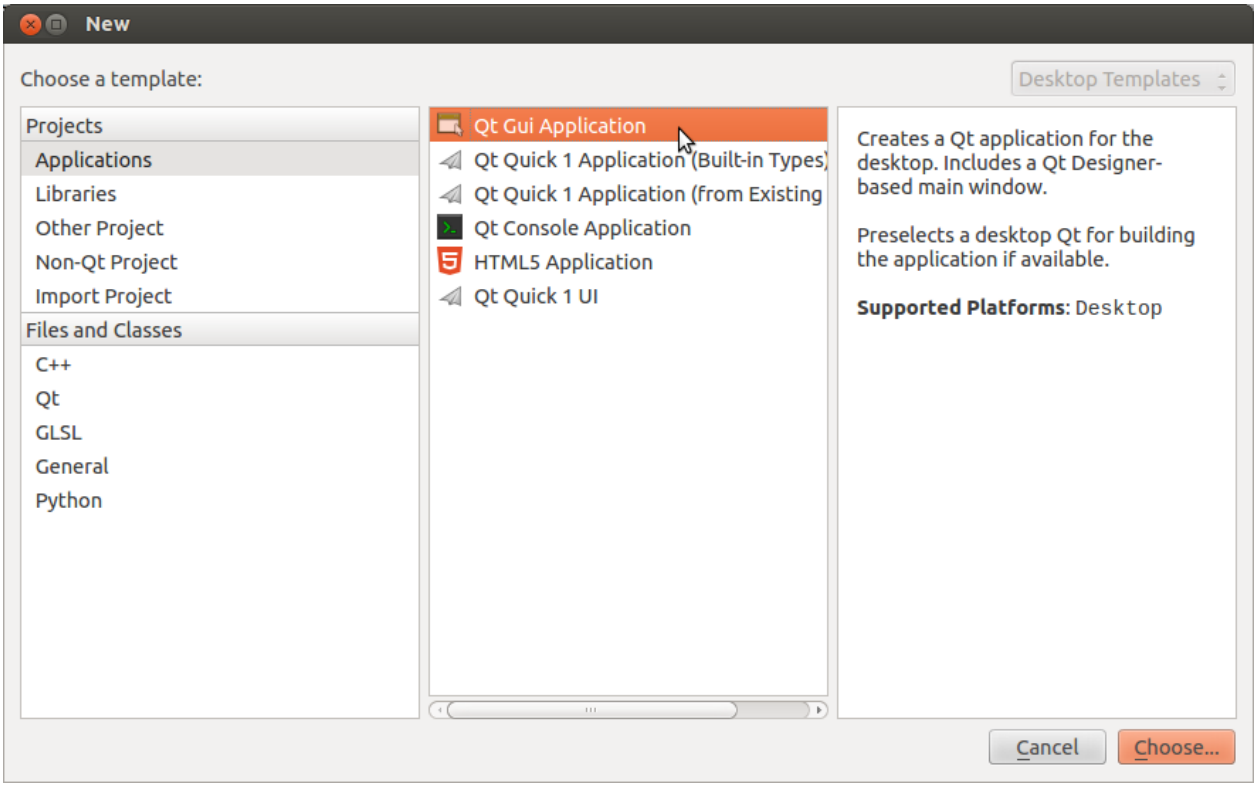
4.1 Downl-oad and extract “Atmel Home Automation” source archive.

TO DO Open a terminal: Press “CRTL+ATL+T and execute the following commands

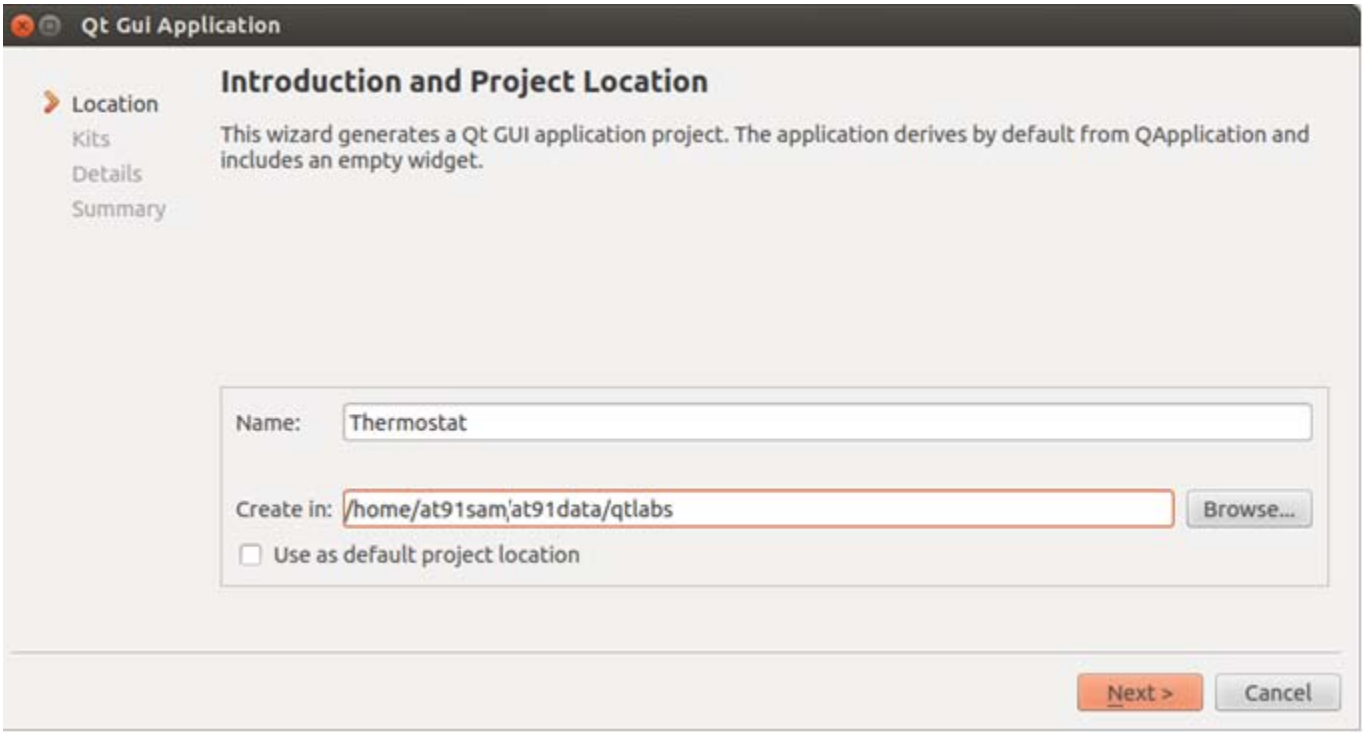
```
# cd ~/at91data/qtlabs/
# wget ftp://ftp.linux4sam.org/pub/demo/qtdemo/home-automation-1.5.tar.gz
# tar zxvf home-automation-1.5.tar.gz
```

4.2 Create a new Qt application using qt-creator provided wizards.

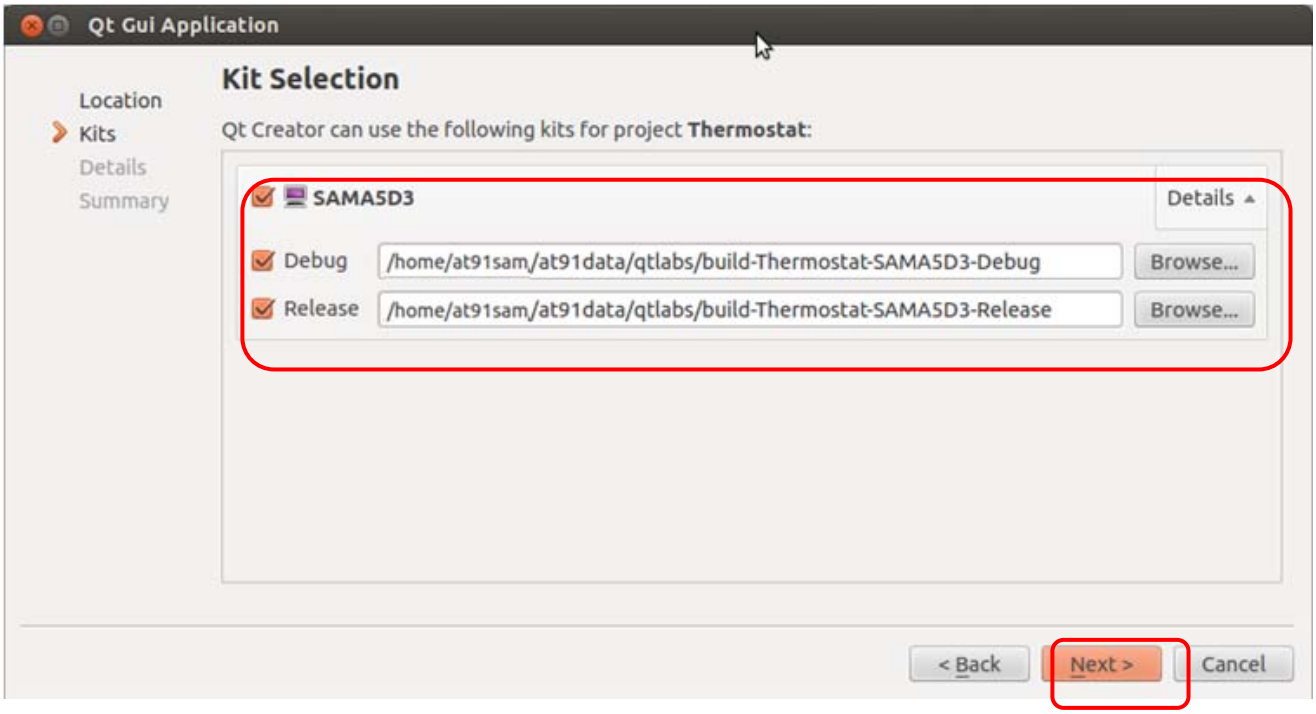
TO DO You need to go through Qt Creator menus:
File → New File or Project: Then choose “Qt Gui Application”
Select: **Applications → Qt Gui Application**



TO DO Specify your application name and make sure you create new project in your LAB area



TO DO You can now select which Kit should be enabled for this new project. Make sure you select your new Kit which enables cross-compilation for the Atmel sama5d3_xplained target.

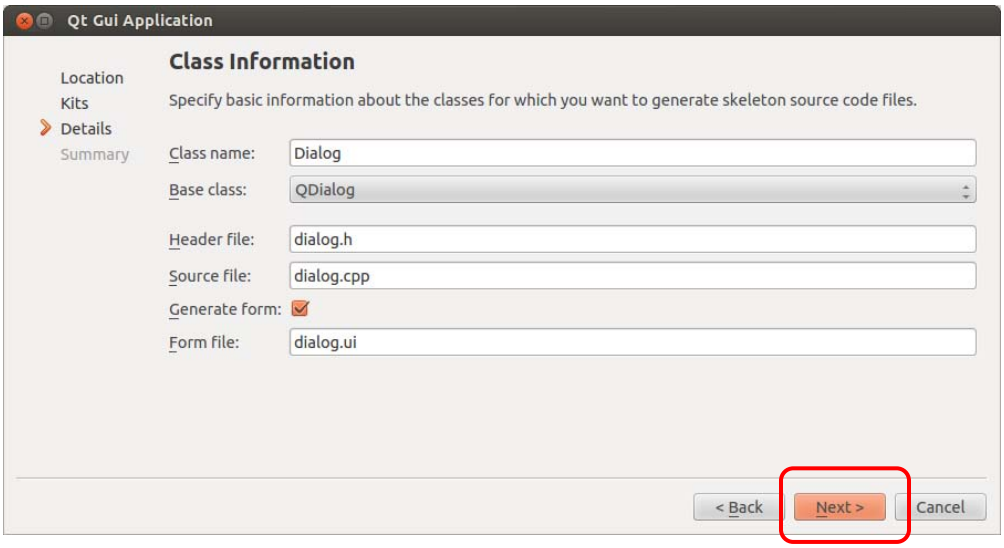


INFO

Differences between Qwidget , QMainWindow and QDialog ?
More information: http://qt-project.org/doc/qt-4.8/qtgui.html
QWidget is a base class for all other GUI elements in QtWidgets module. It can constitute a window by itself, or be part of a QLayout, or just a member of parent-child hierarchy
QDialog is usually used to – surprise! – display a temporary dialog when user input is required
QMainWindow is a convenience class that can be used as the main window of your application. It has some nice features built-in: a status bar, tool bars and a menu bar.

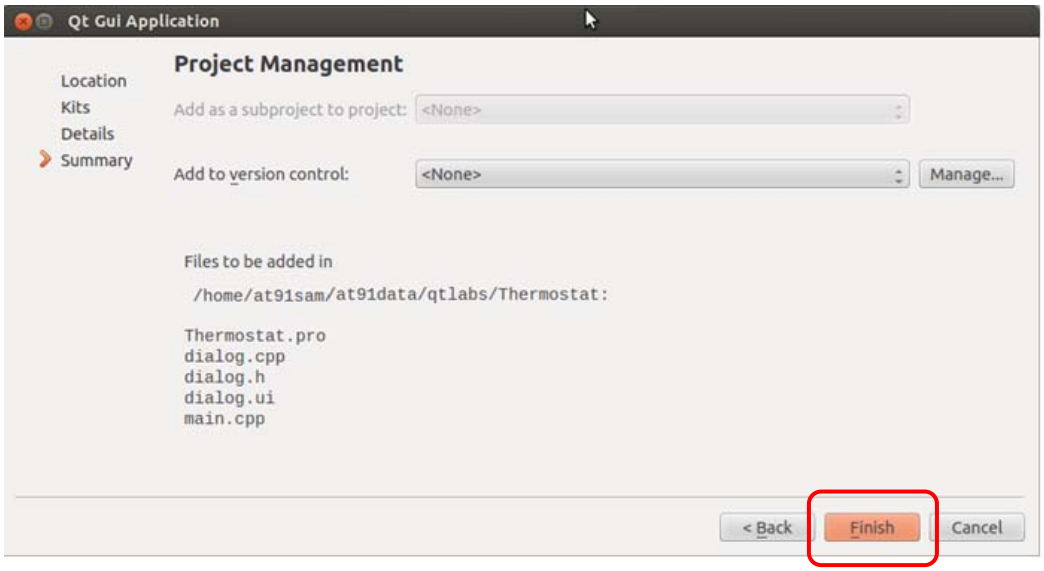
TO DO

You can select a window decoration for your application. Let's switch to QDialog as a Base class. Finish project setup. The new project will open up in QtCreators' Editor and project browser.



TO DO

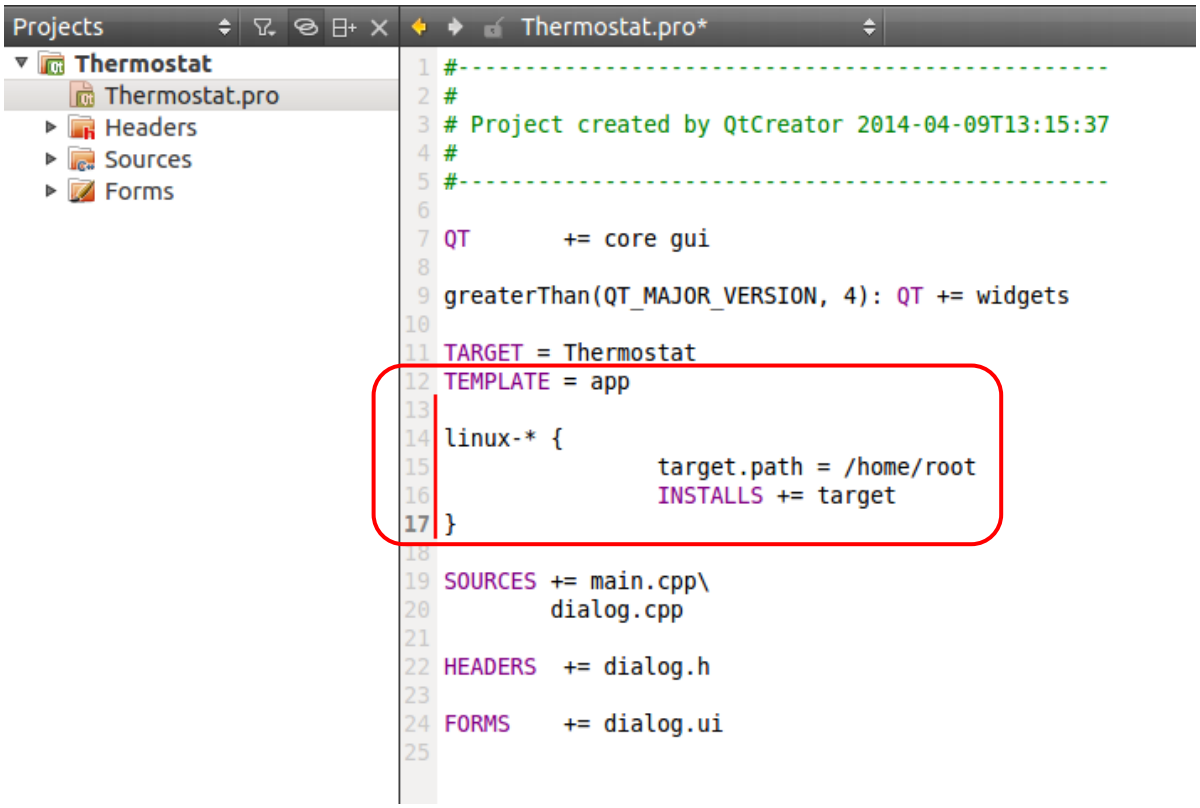
You have just to click "Finish" button.



4.3 Build deploy and debug the application on the target.

TO DO To enable deployment on remote target the following lines have to be added Thermostat.pro file.

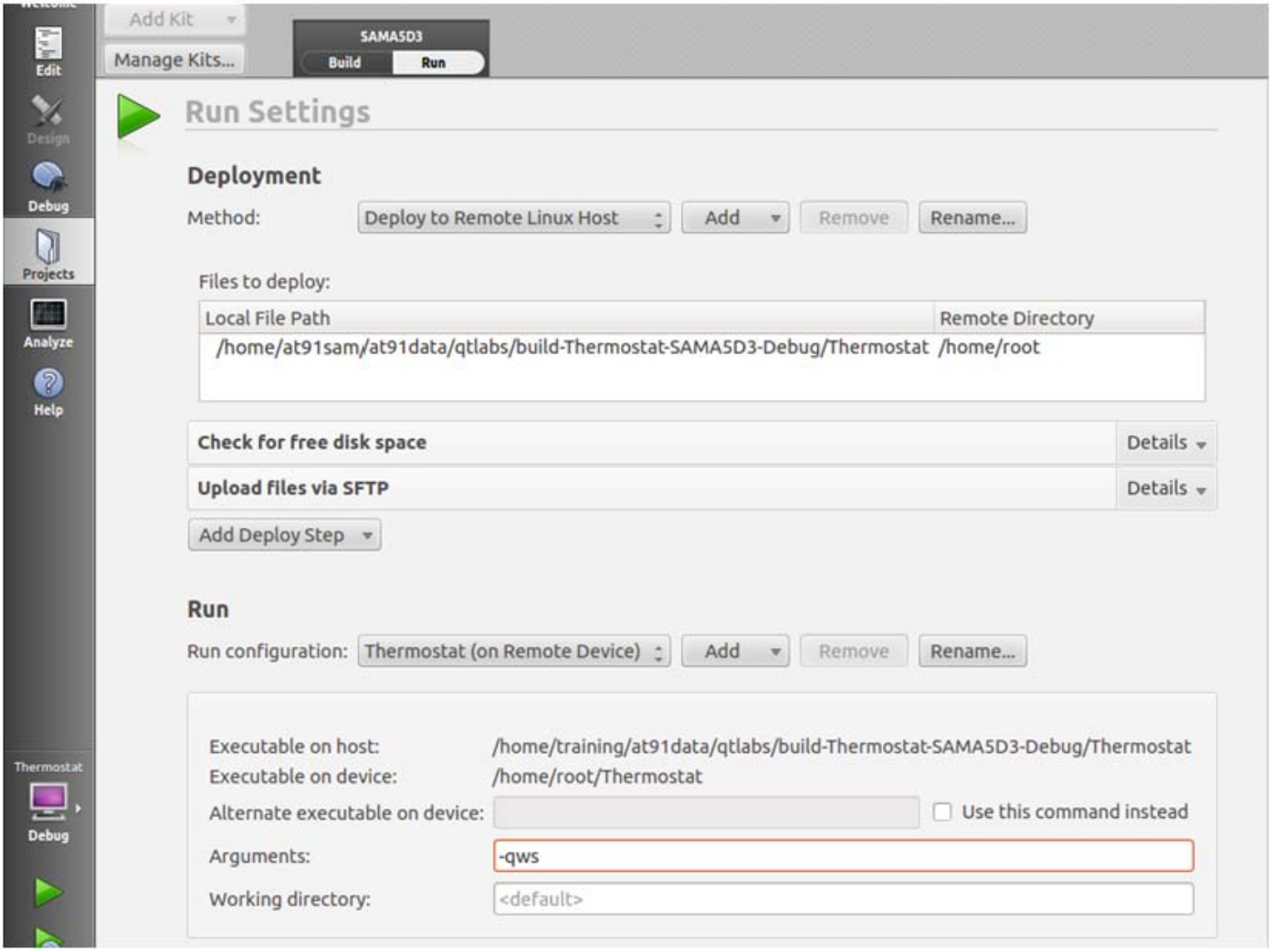
```
linux-* {
       target.path = /home/root
       INSTALLS += target
}
```



INFO This “qmake” variable contains a list of resources that will be installed when make install or a similar installation procedure is executed. Each item in the list is typically defined with attributes that provide information about where it will be installed. http://qt-project.org/doc/qt-4.8/qmake-variable-reference.html#installs

INFO Now “Deploy” and “Debug buttons” are active.

TO DO Add “-qws” option argument to our Thermostat application



INFO At this point feel free to compile, deploy, run and debug your code on target.

4.4 Import “TemperatureControl ATMEL Widget” in your application.

TO DO

First we need to copy the “CustomWidgets” directory from home-automation demo to current directory.
Open a new terminal if needed window (hit CTRL-ALT-T) and execute the following commands

```
# cd ~/at91data/qtlabs
# cp -rf ../home-automation-1.5/CustomWidgets/  Thermostat/
```

RESULT

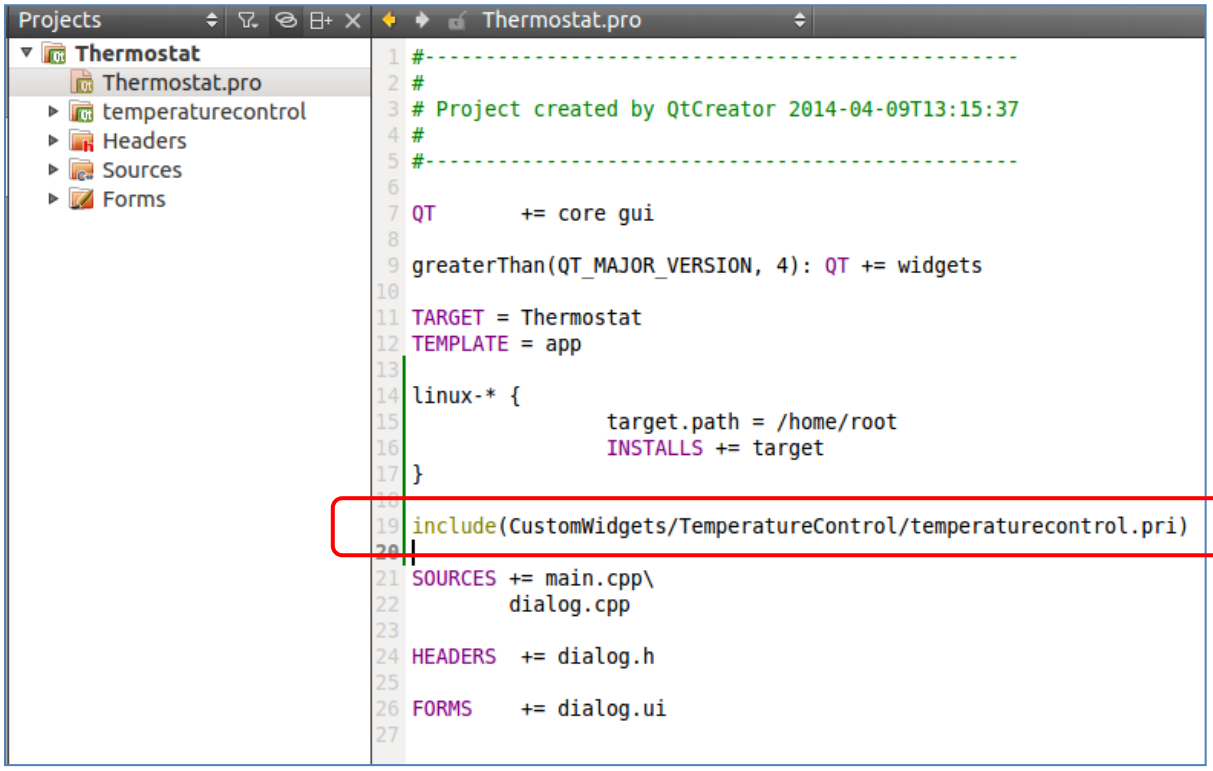
The home-automation library is available into your project and your are able to include the temperaturecontrol widget into your UI. You can check easily the path into yourUI project source code:

```
# ls –al Thermostat/CustomWidgets/TemperatureControl
# drwxr-xr-x 3 training training  4096 May 11 01:34 .
# drwxr-xr-x 9 training training  4096 May 11 00:22 ..
# drwxr-xr-x 2 training training  4096 May 11 00:22 resources
# -rw-r--r-- 1 training training 20325 May 11 01:21 temperaturecontrol.cpp
# -rw-r--r-- 1 training training 11741 May 11 01:34 temperaturecontrol.h
# -rw-r--r-- 1 training training   509 May 11 00:22 temperaturecontrol.pri
# -rw-r--r-- 1 training training  3453 May 11 00:22 temperaturecontrol.ui
# -rw-r--r-- 1 training training   581 May 11 00:22 tempresources.qrc#
```

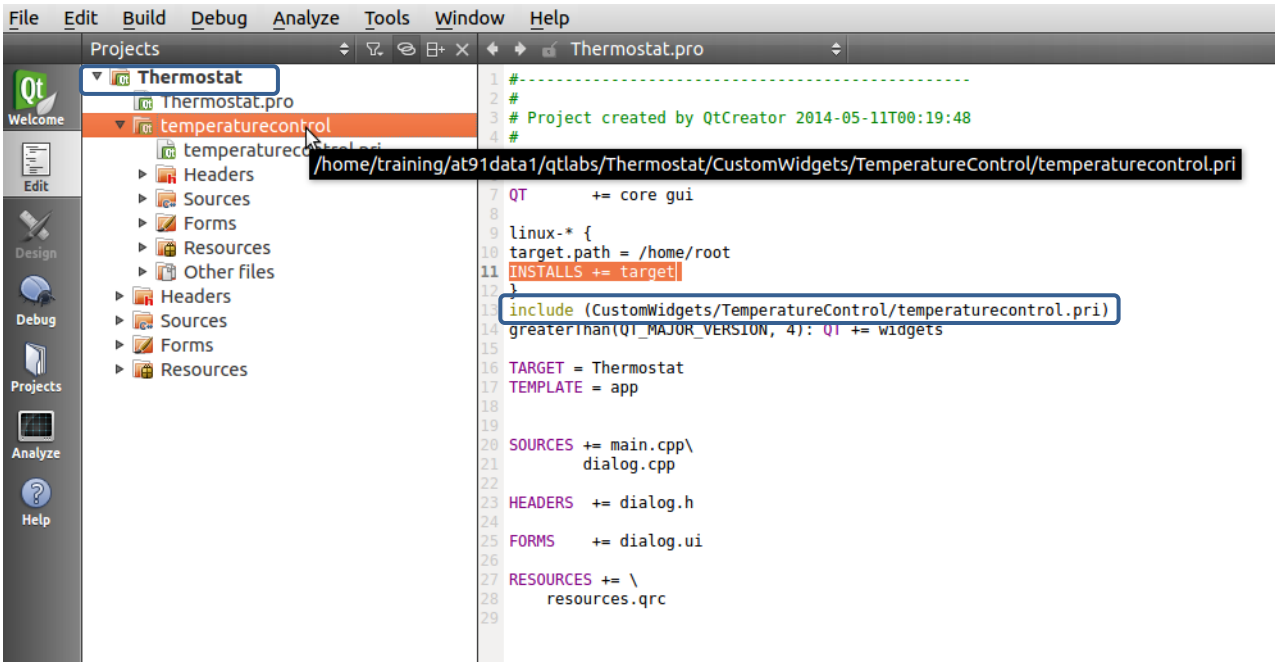
TO DO

To enable the temperaturecontrol Widget into your project you need to add the following lines in Thermostat.pro file (your main project file).

```
include(CustomWidgets/TemperatureControl/temperaturecontrol.pri)
```

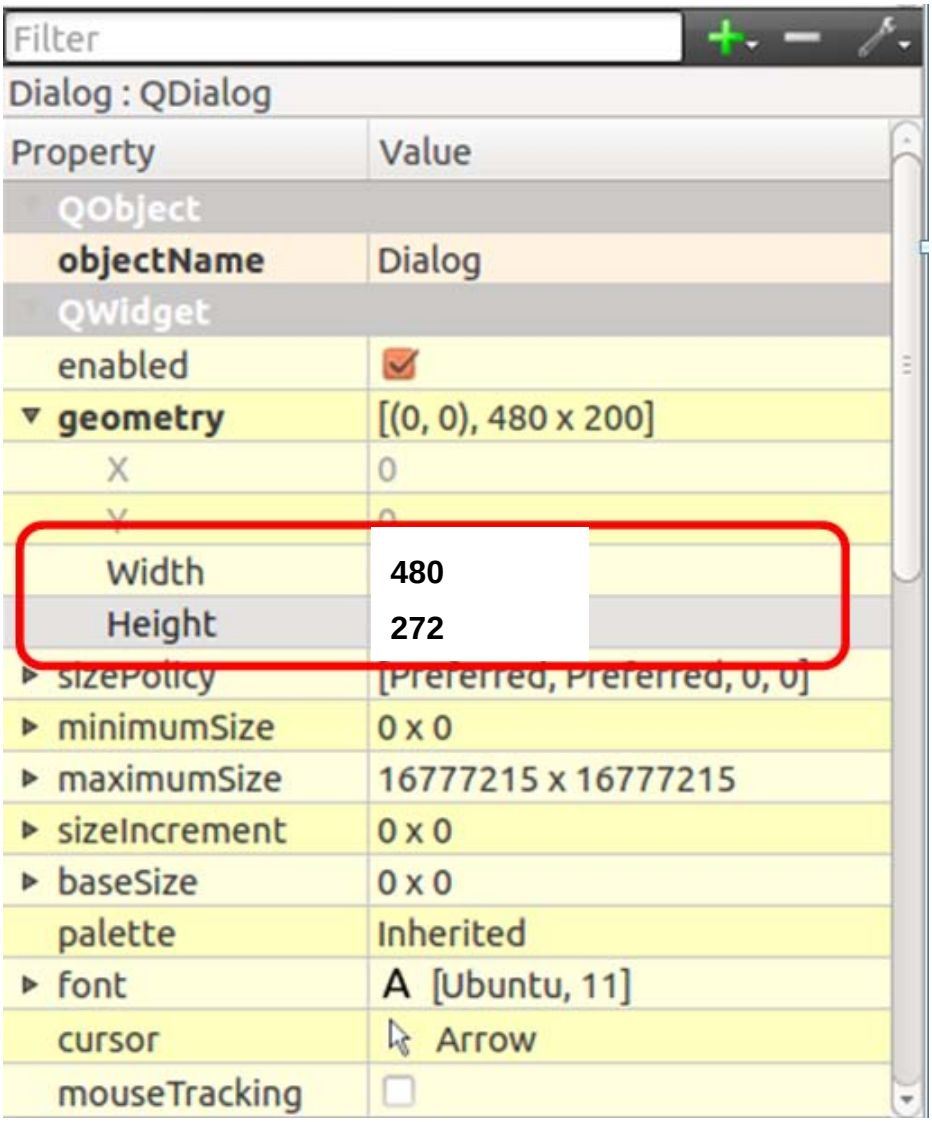


INFO Now The Qt Widget "**temperaturecontrol**" should appear in your project



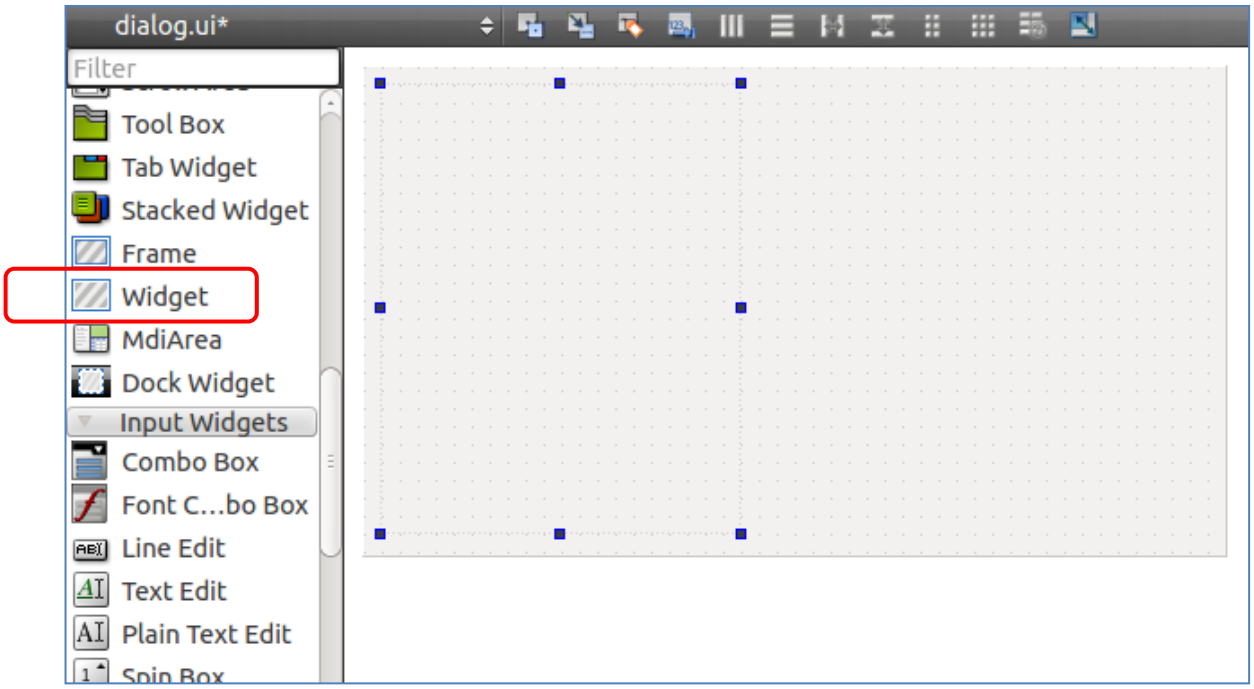
TO DO Resize our application

- Select the QDialog Object and set the size to **480x272 (WQVGA)**
  We will do a full screen application without the window bar.

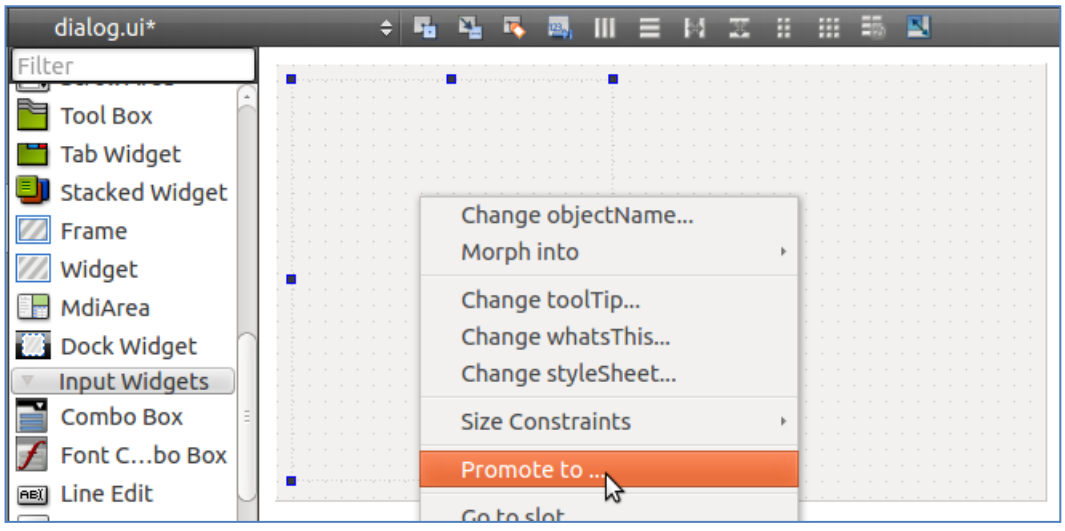


4.5 Use Atmel TemperatureControl Widget

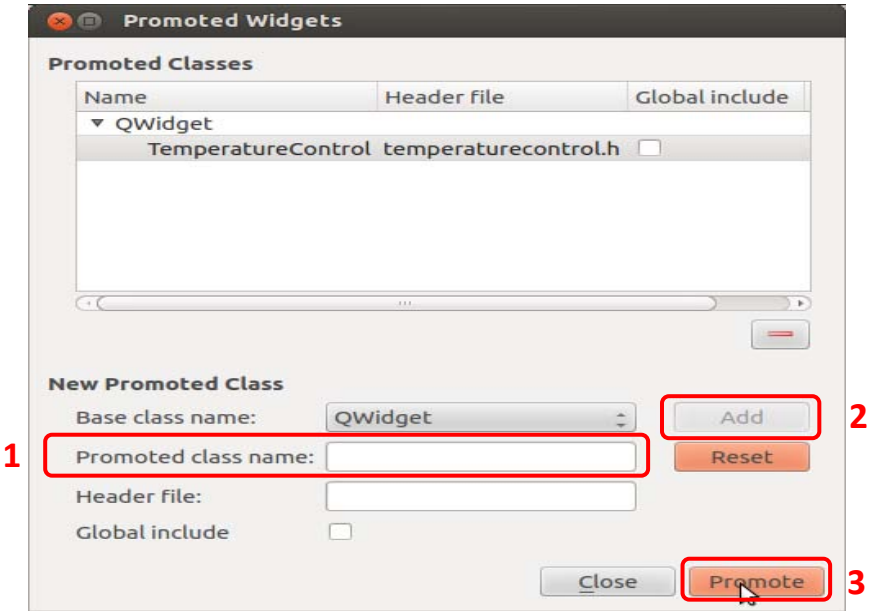
TO DO Drag and drop a “Widget” object and resize it.



TO DO Right click on your widget and select “Promote”

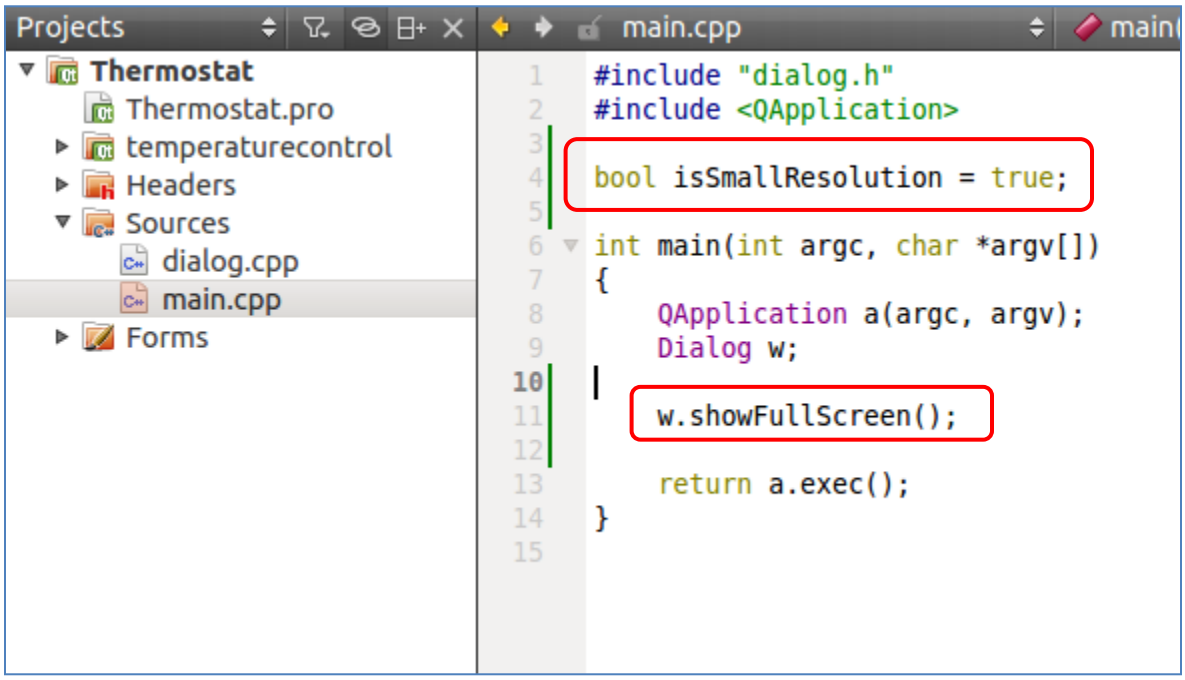


TO DO In field “Promoted class name” type: “**TemperatureControl**” then press “**Add**” button and finally press “**Promote**” button.



4.6 Make application full screen (now window bar)

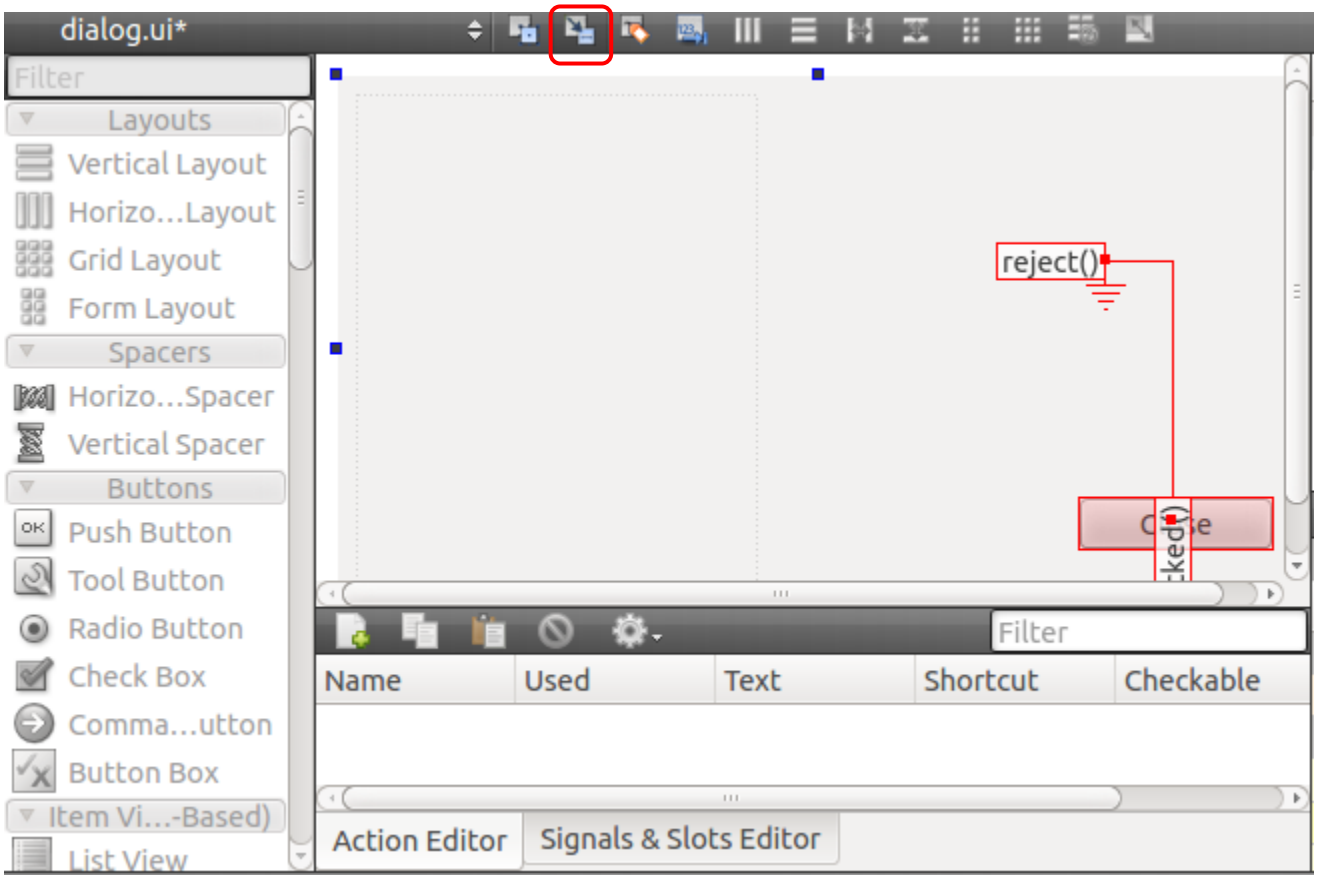
TO DO Open main.cpp file and replace “**w.show**()” by ” **w.showFullScreen**
Add **isSmallResolution** boolean variable. It is needed by **TemperatureControl** widget to adapt itself to small resolution.



4.7 Add a close button

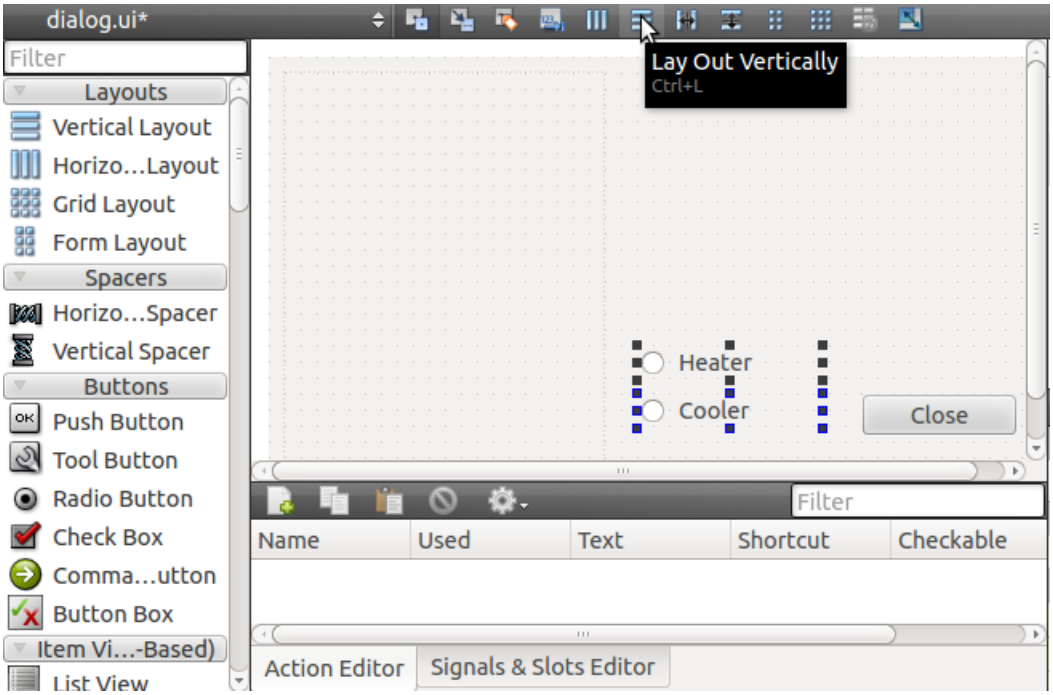
TO DO

Open dialog.ui (double click on it) in designer view and add a push button named Close.
On the canvas, select signal view click and hold on the close button and then release the mouse on the main window area.
This brings up the Connection Configuration dialog where you can select which signal will be connected to which slot. Please select **clicked()** signal and **reject()** slot

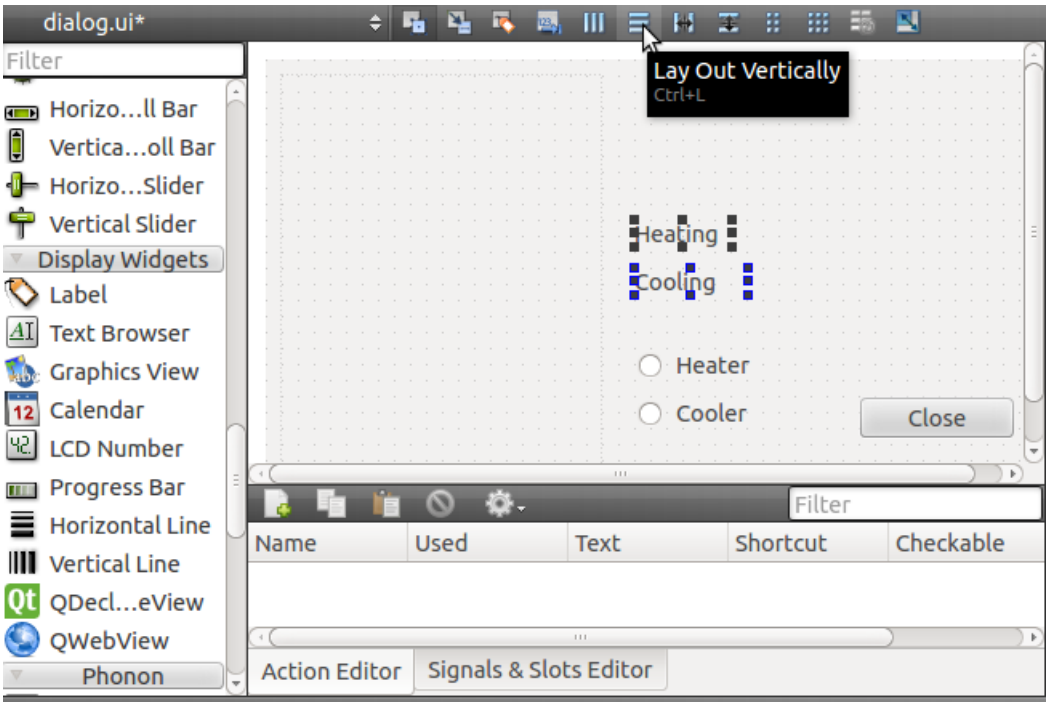


4.8 Let's design – Add objects and layouts

TO DO Add two radio button (Heater & Cooler), select them and apply a vertical layout

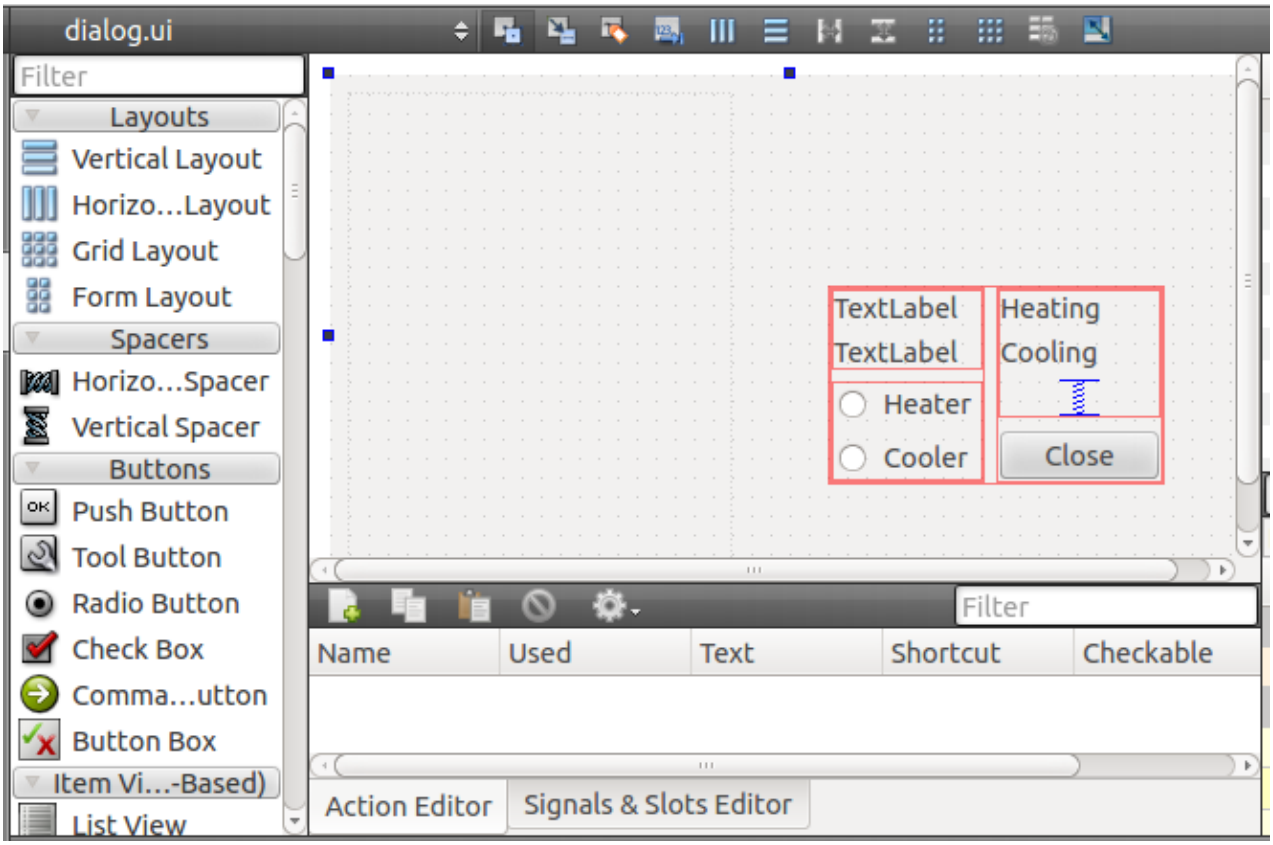


TO DO Add two text label (Heating & Cooling) , select them and apply a vertical layout

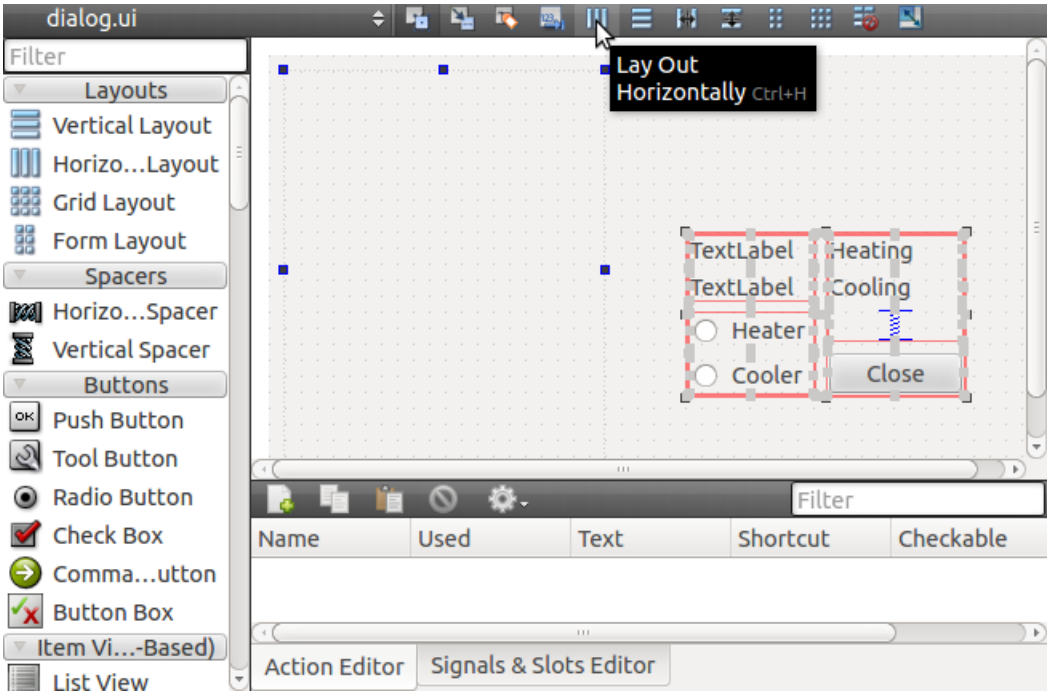


TO DO

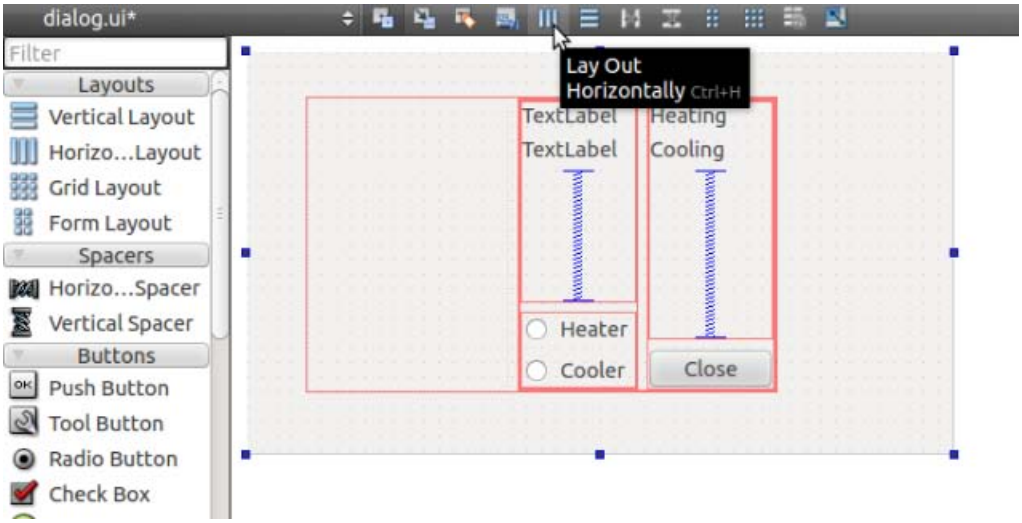
Be autonomous now!
Add two other text label (no name needed), select them and apply a vertical layout
Apply a vertical layout between “no name Text labels” and Radio buttons”
Apply a vertical layout between “Heating & Cooling Text labels” and Close buttons
Add a vertical spacer between Heating & Cooling and Close button
Select the two group (Labels + Radio) and (Heating/Cooling + Close Button) and apply a horizontal layout



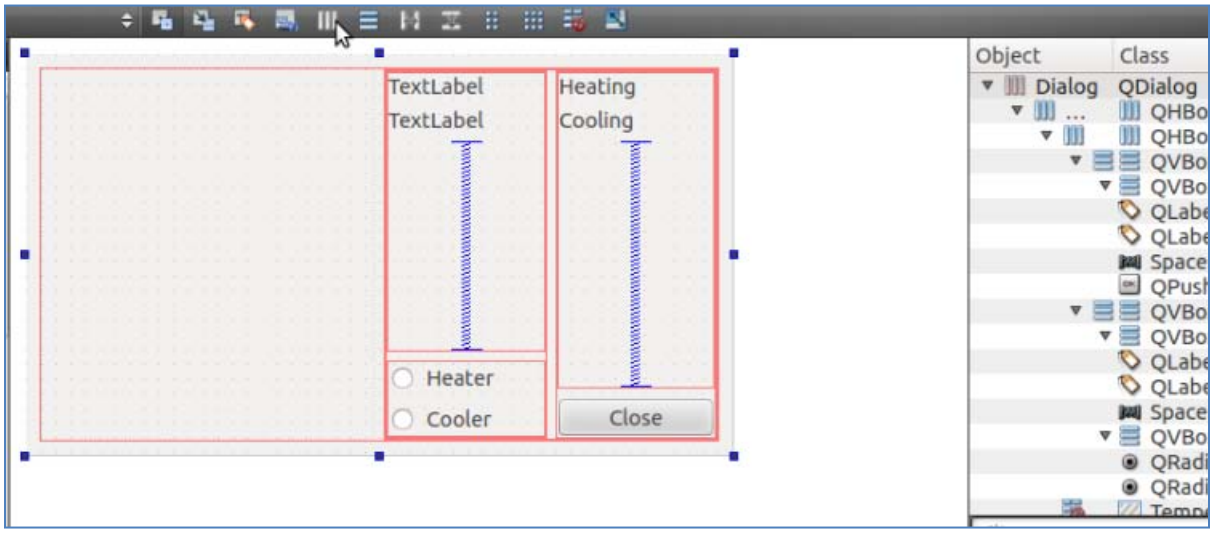
TO DO Select the widget and the group of (text labels, radio and close button) and apply a horizontal layout.



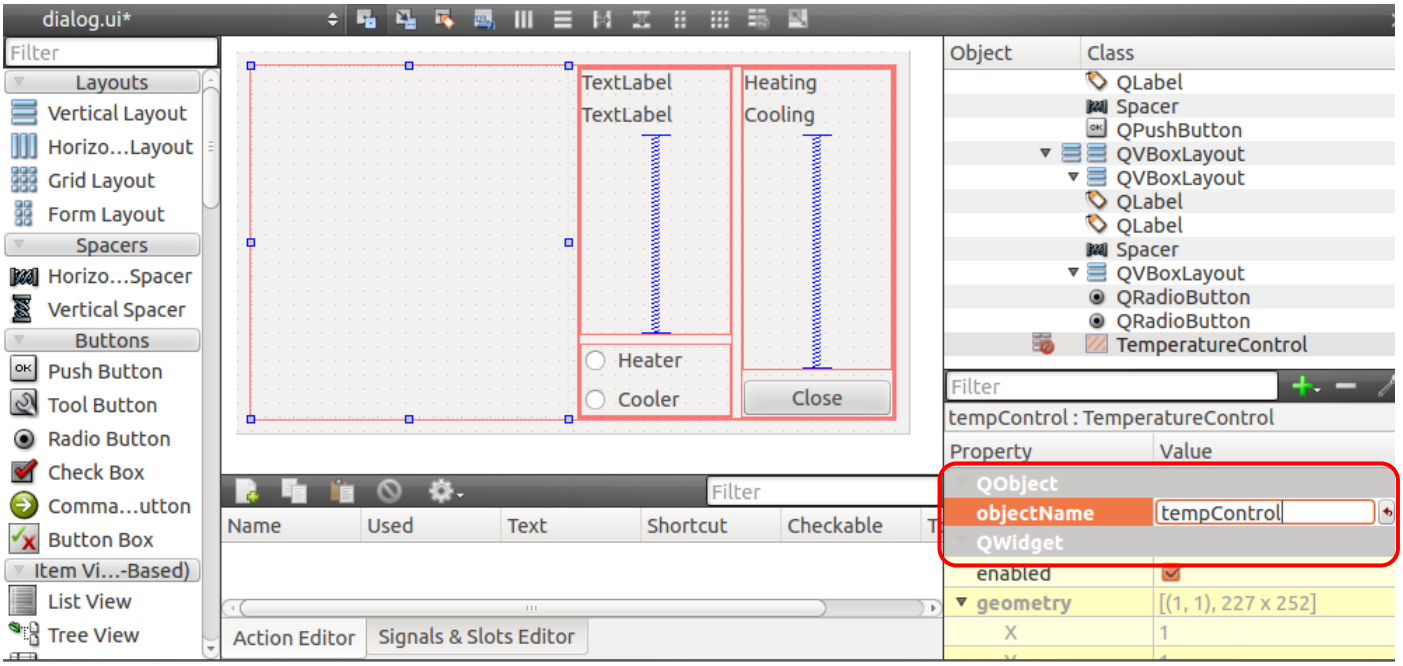
TO DO Add a vertical spacer between no name Text Label and Radio Button



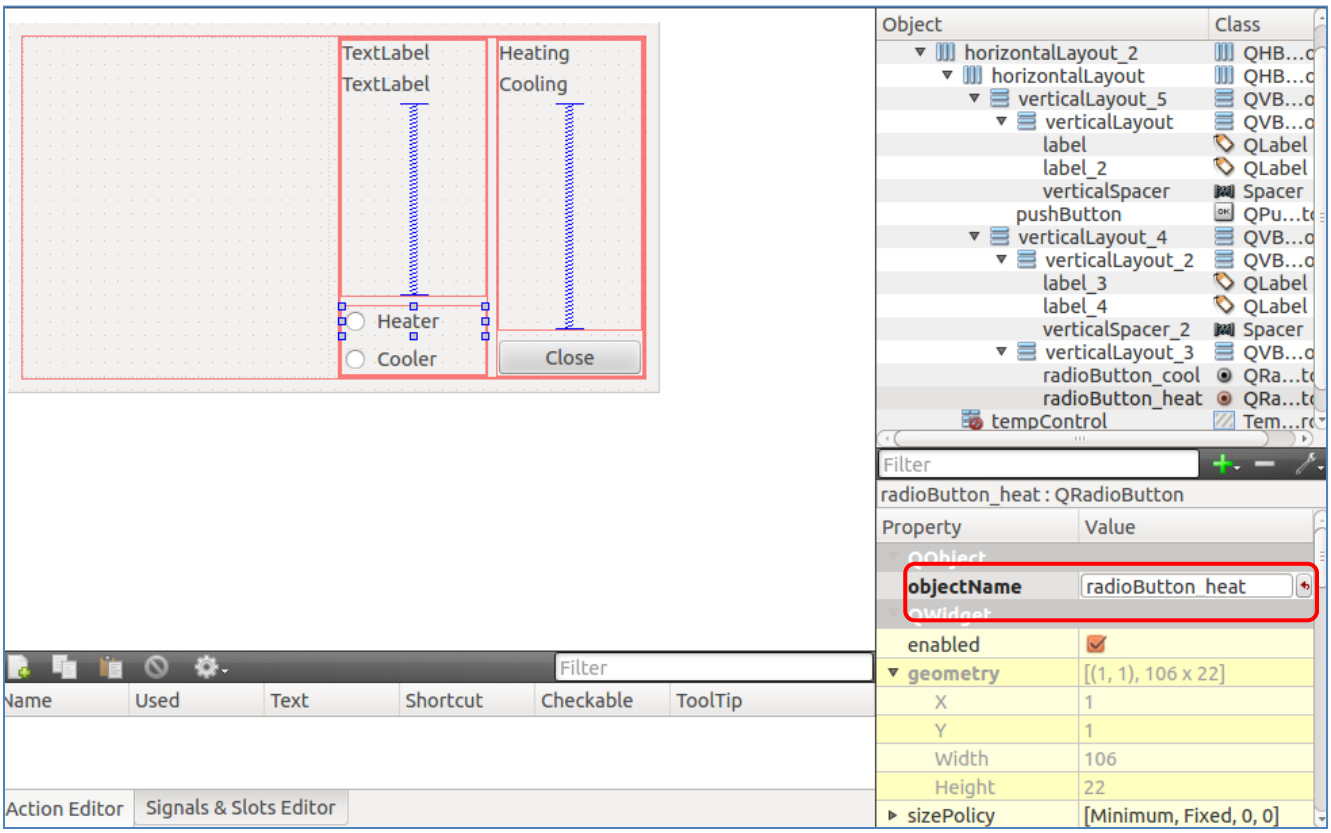
TO DO finally select the **Dialog Object** with the **Object Explorer** and apply a horizontal layout

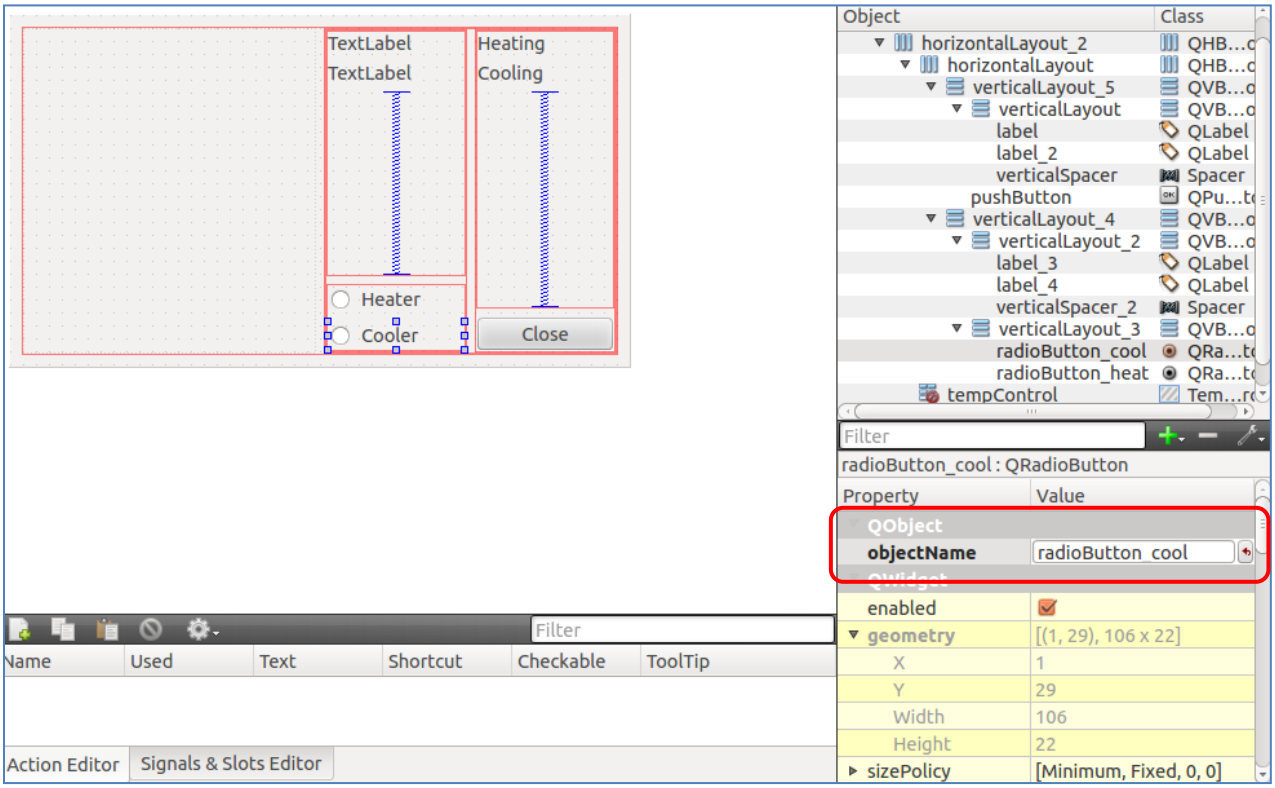


TO DO Before to start to code we have to give name to our ojects
Rename **TemperatureControl** Widget → **tempControl**

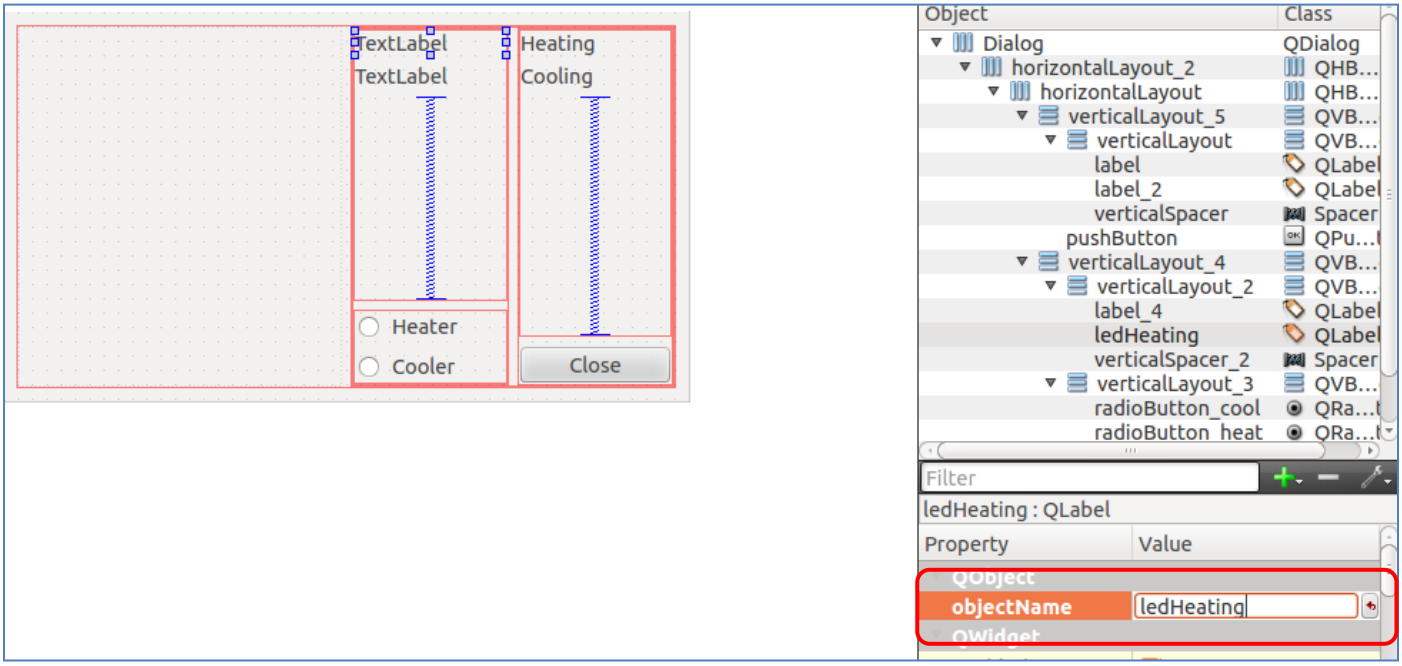


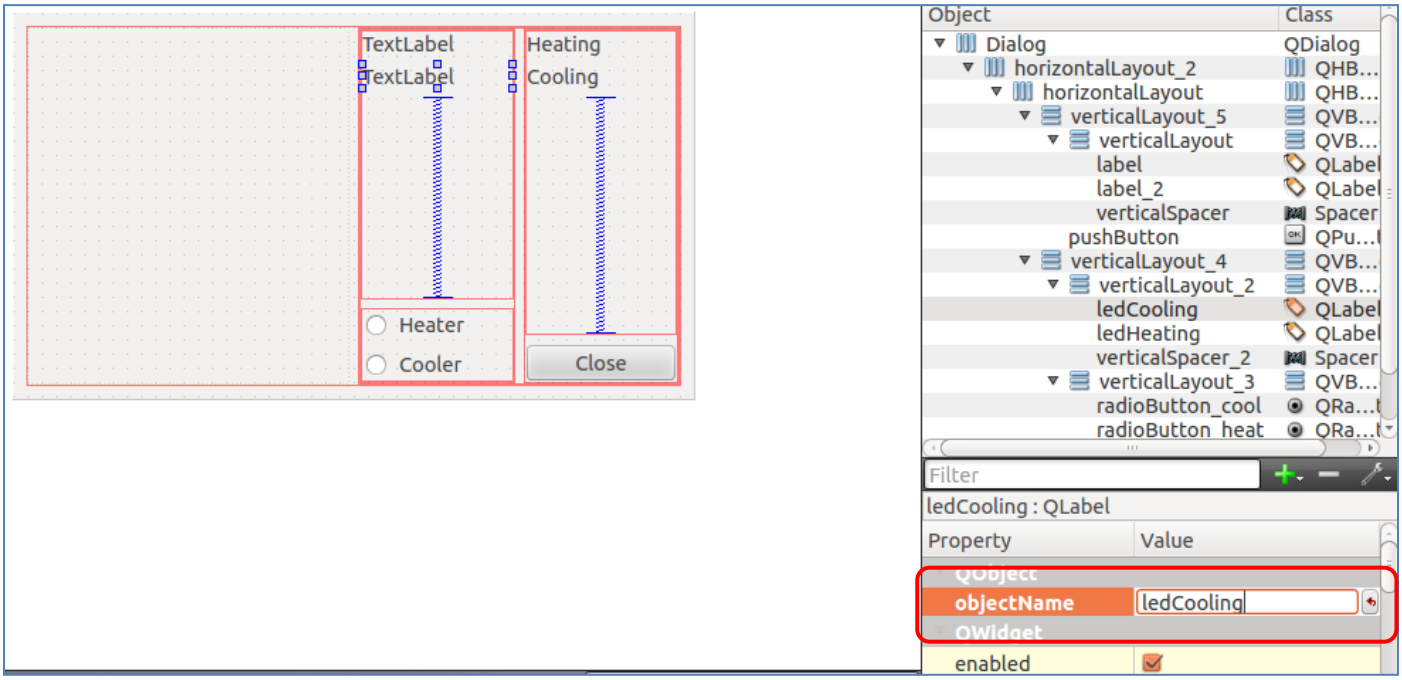
TO DO Rename **Radio Buttons → radioButton_heat** & **radioButton_cool**





TO DO Rename **Text Labels → ledHeating** & **→ ledCooling**



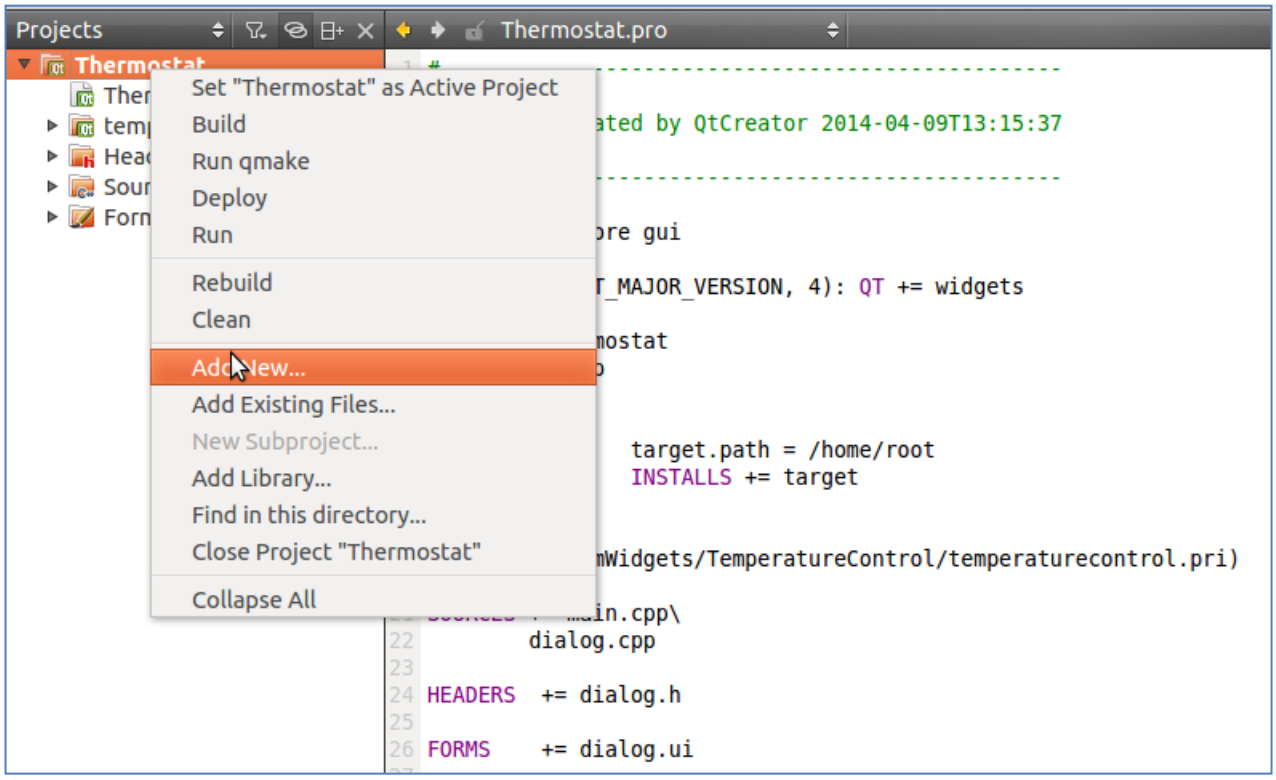


4.9 Add Resources to our application (png files).

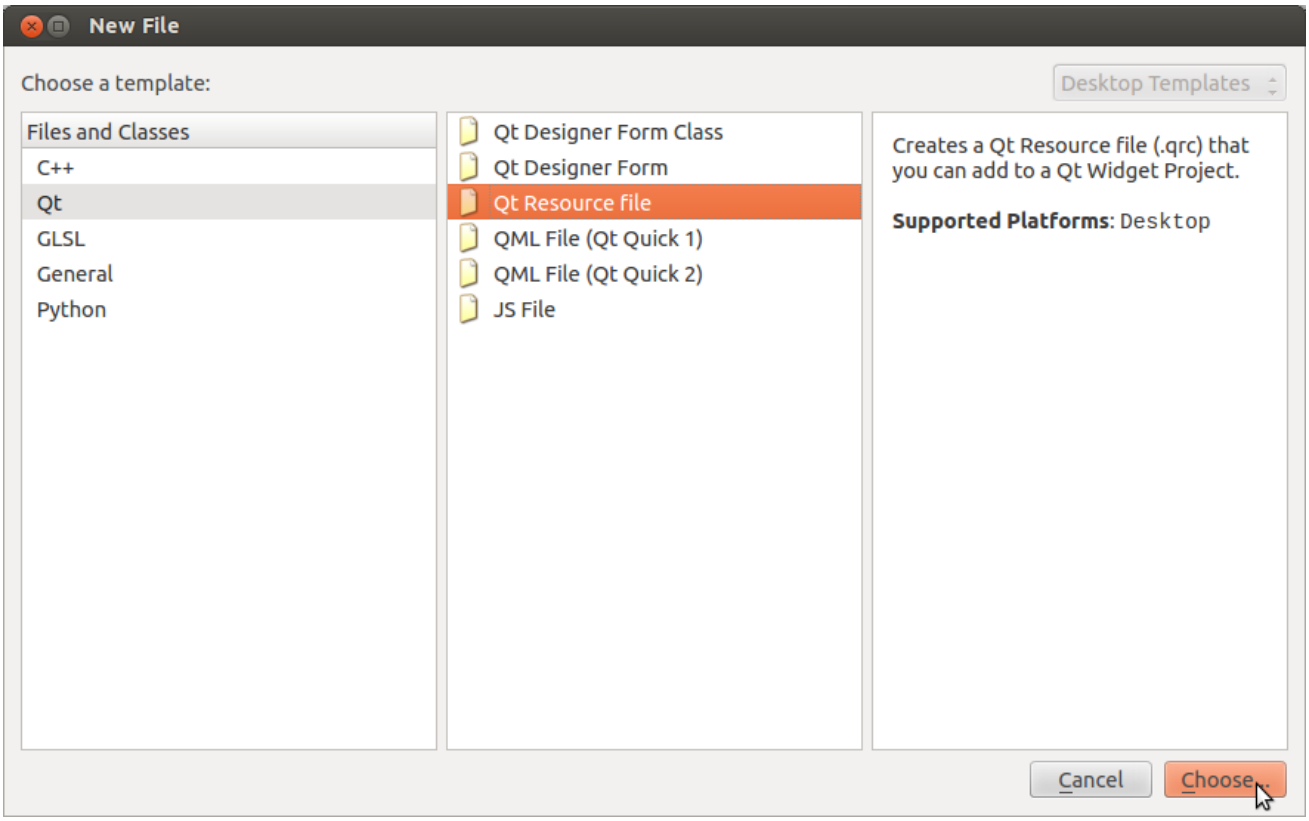
TO DO Open a terminal and execute following commands

```
# cd ~/at91data/qtlabs/Thermostat
# cp ~/at91data/greenLED.png .
# cp ~/at91data/redLED.png .
```

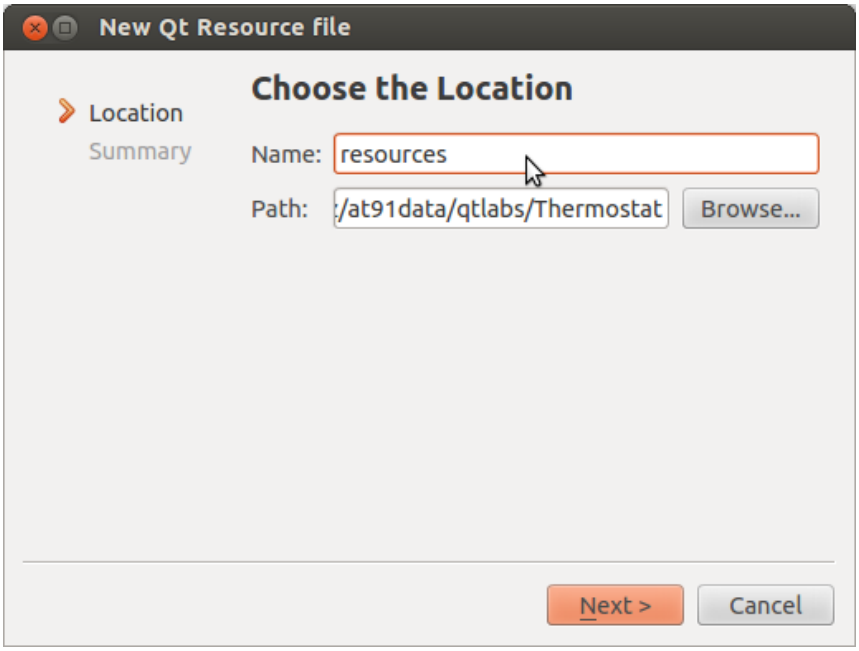
TO DO Select Thermostat project, mouse right click and selec **"Add New…"** menu entry



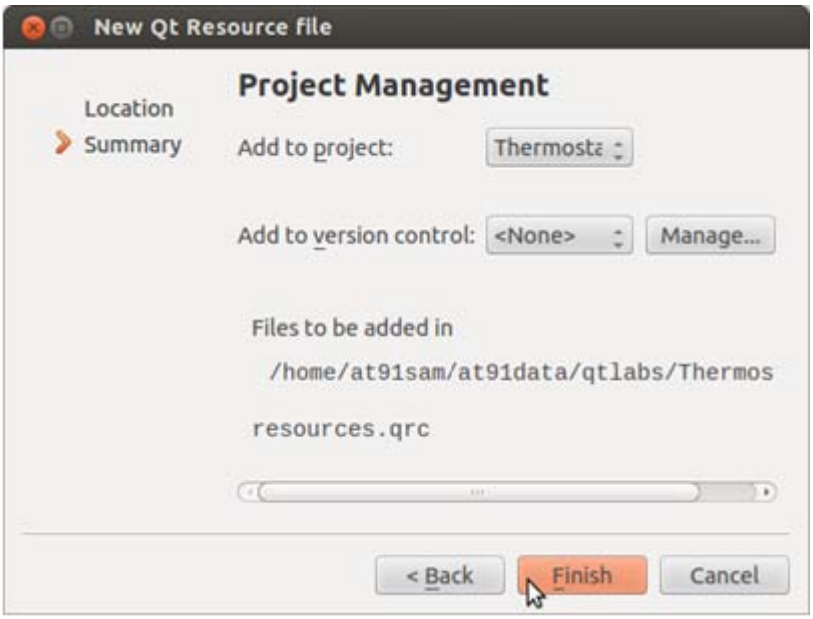
TO DO Select: Qt → Qt Resource file, and click choose.



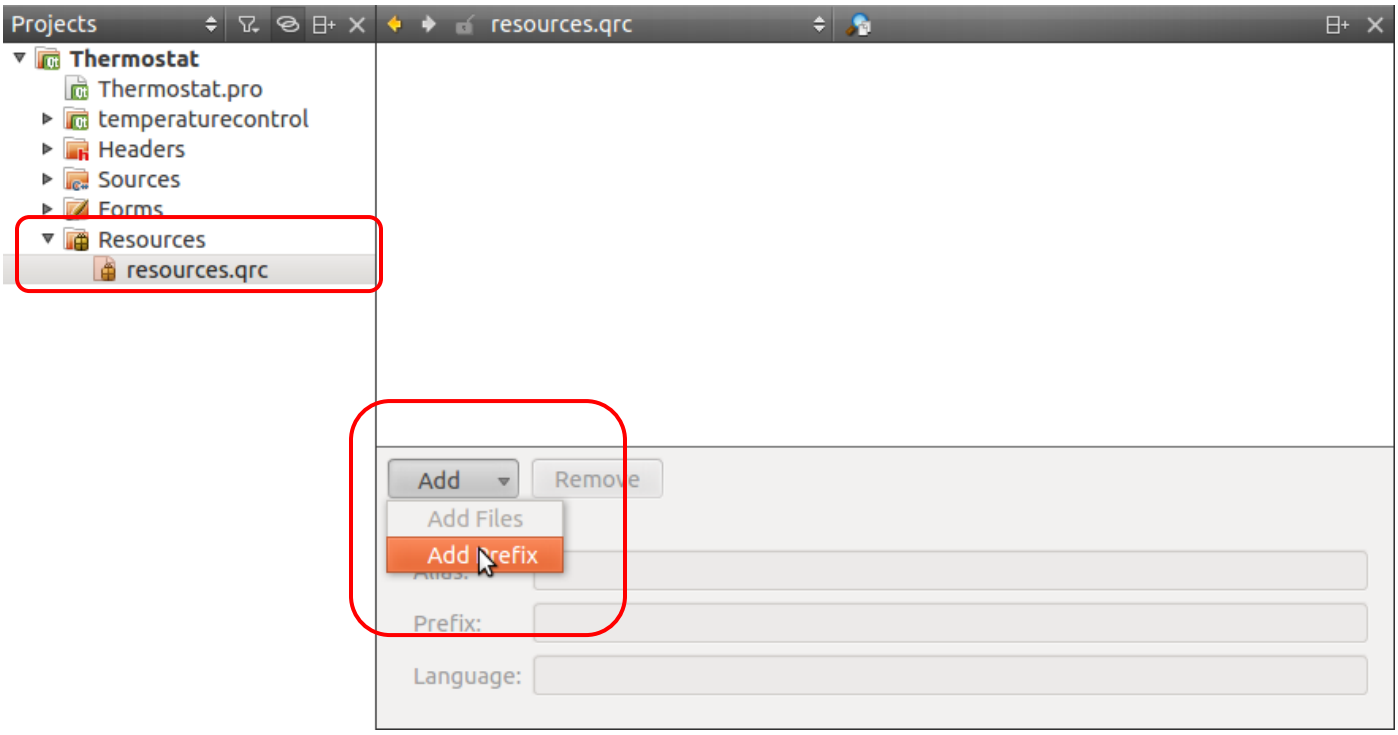
- Choose the Location: ~/at91data/qtlabs/Thermostat



TO DO And then click Finish

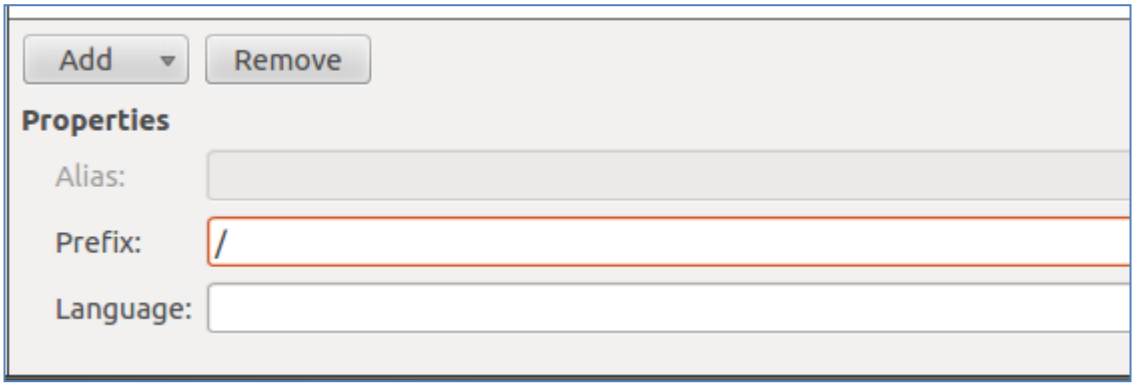


TO DO Select **resources.qrc** file, click **Add** button then select **Add Prefix**



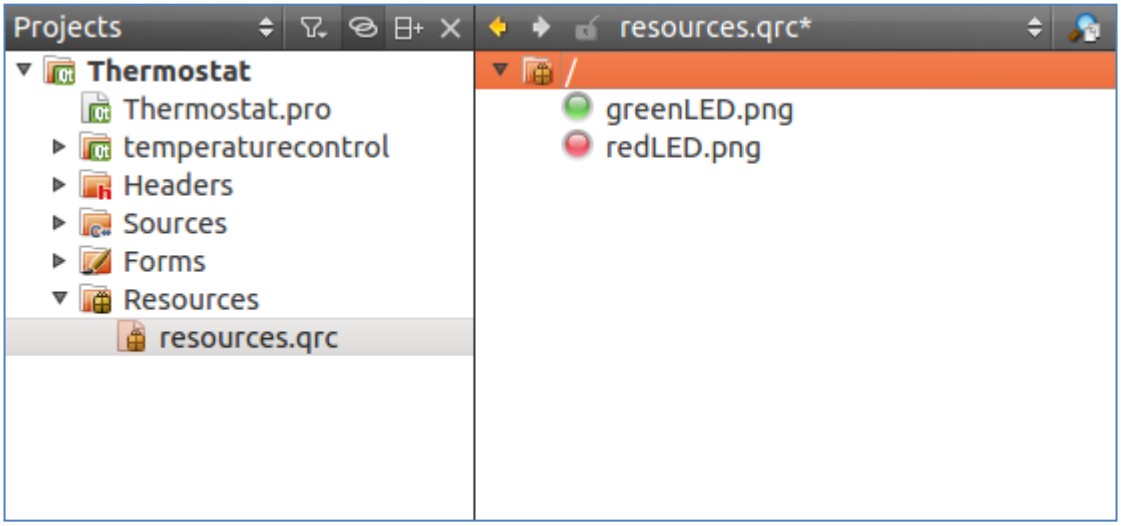
TO DO

Enter “**/**” as **Prefix**
Press **Add** button and select ~/at91data/qtlabs/**redLED.png** & **greenLED.png** files
If Qtcreator ask you to copy it into project directory → say **yes**



TO DO

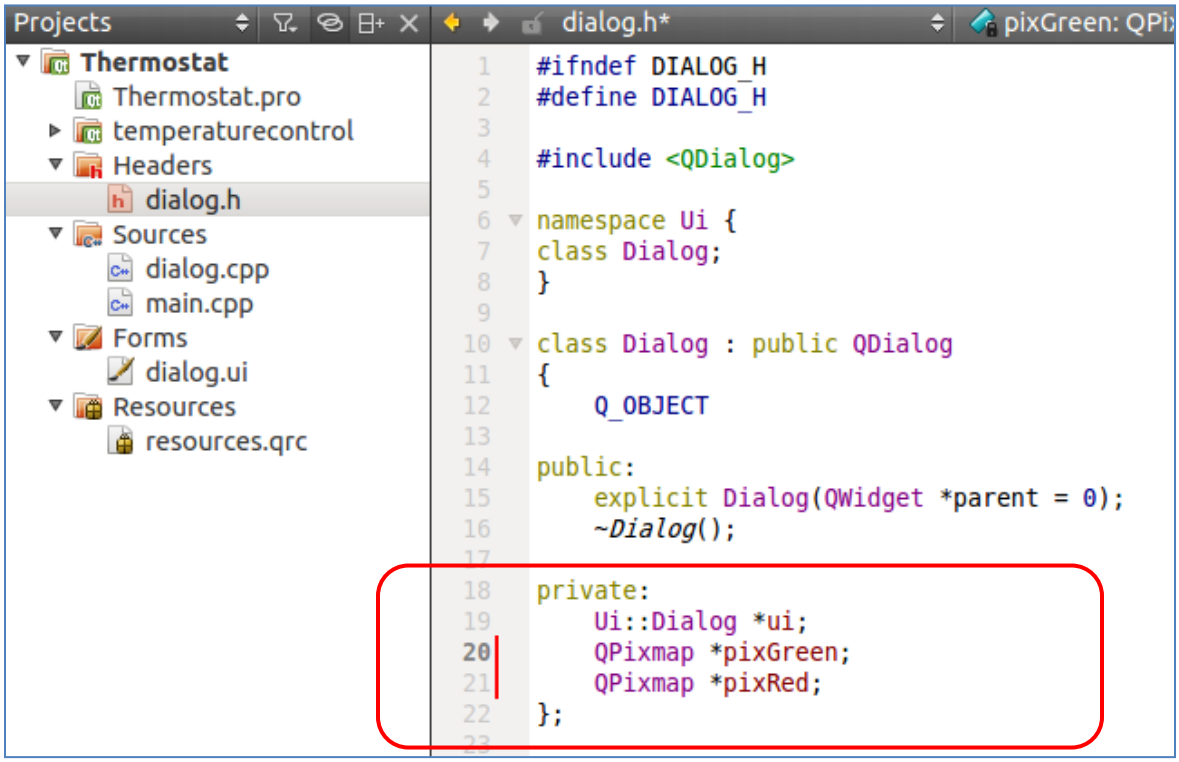
Inspect your resources.qrc file. You can see two png files.



4.10 Pixmap for Text Label.

Go back to code now. We will set Pixmap of our two no name labels. Text will be replaced by our png files. Feel Free to build and deploy at any step.

TO DO Edit dialog.h and add “QPixmap” objects (**pixGreen** and **pixRed**) **to Dialog Class**.

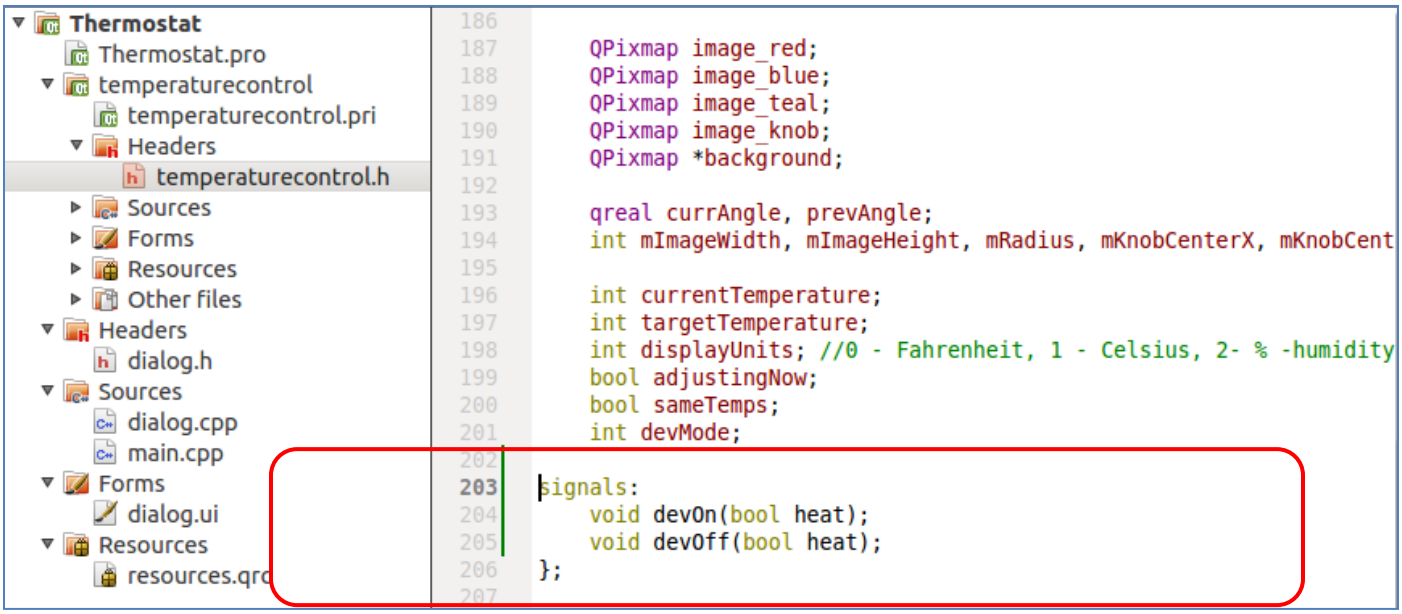


TO DO Edit dialog.cpp and add “setPixmap” code section.

4.11 Add signals to TemperatureControl Widget

We will use these signals to control the colour of our virtual LEDS:
Green Cooler / Heater are on, Red off

TO DO Add signals to **TemperatureControl** class. Open temperaturecontrol.h file and **devOn** and **devOff** signals.



INFO

Now we have to implement “emissions”.
In **TemperatureControl** class there is tow functions to control **Heater** and **Cooler** devices: **deviceOn** and **deviceOff.**

TO DO

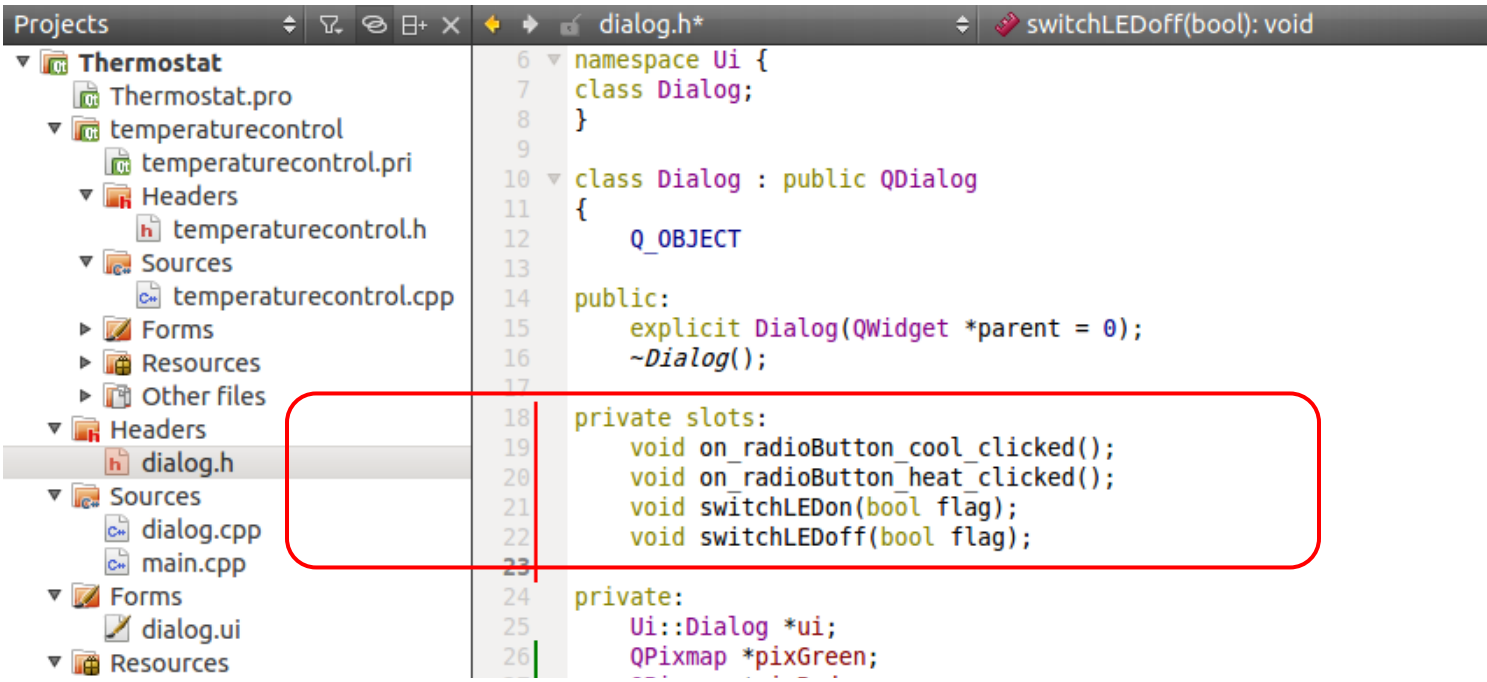
Open **temperaturecontrol.cpp** file, find **deviceOn** function and add emissions of devOn signal

TO DO Open **temperaturecontrol.cpp** file, find **deviceOff** function and add emissions of devOff signal

4.12 Adding Slots to Thermostat Application

We have **signals** now we need slots. We will create slots for radio buttons and also to control virtual LEDS.

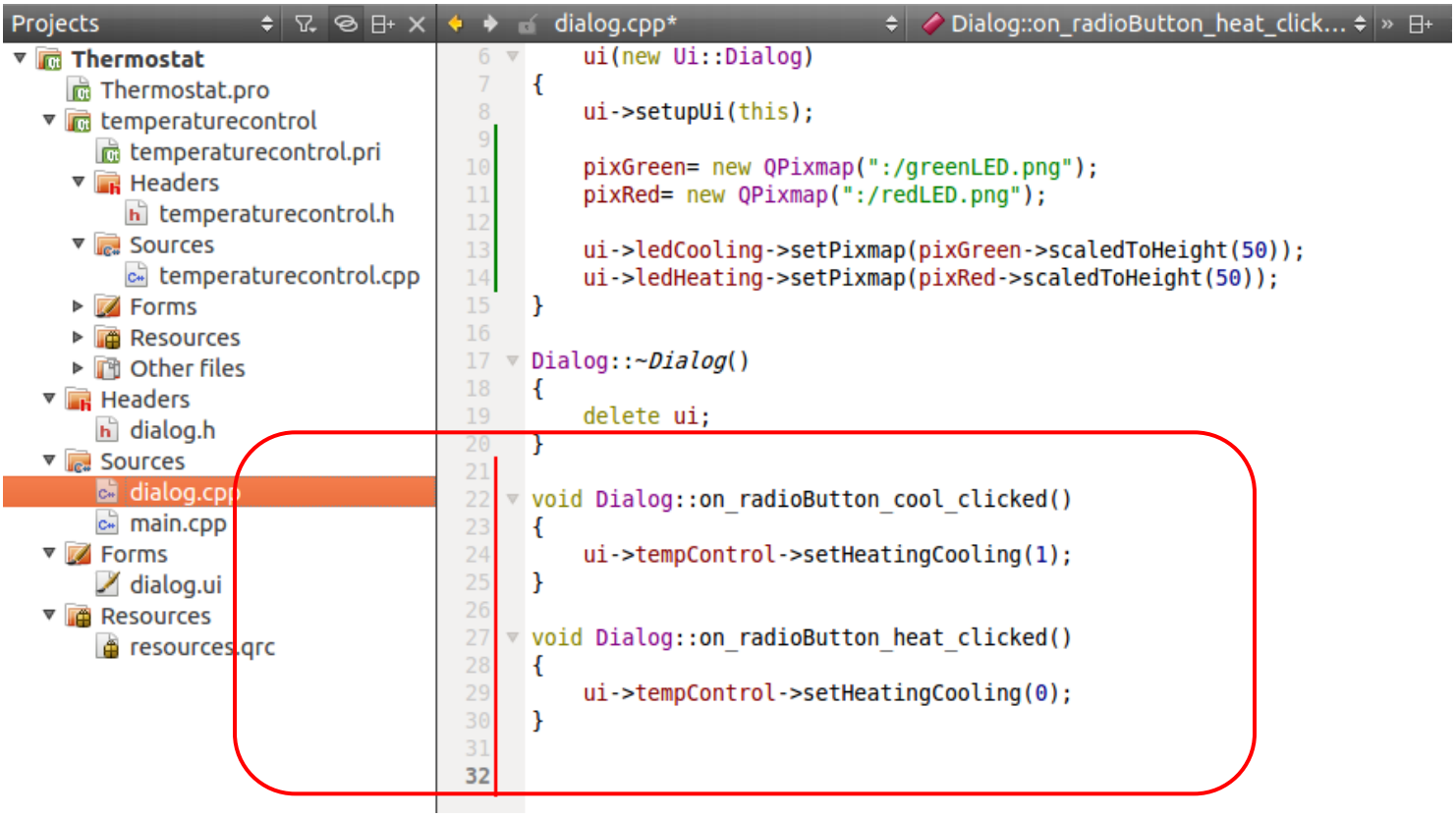
TO DO Open dialog.h file, and add slots: on_radio_button_cool_clicked, on_radio_button_heat_clicked, switchLEDOn and switchLEDOff



TO DO

Connect Radio button signals and slots.
The temperature control widget can control a Heater (Red Circle) and a Cooler (Blue Circle). The function to control the switch between the two modes is **TemperatureControl::SetHeatingCooling**

- Open **dialog.cpp** file, and add on_radio_button_cool_clicked, on_radio_button_heat_clicked functions



TO DO Open **dialog.cpp** file, and add switchLEDOn and switchLEDOff functions

TO DO Open **dialog.cpp** file, and connect devOn/Off signals and switchLEDOn/Off slots.

5. Revision History

Doc. Rev.	Date	Comments
1.0.0	07/05//2014	Initial document release

Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370

Prerequisites

- **Hardware Prerequisites**
  - Atmel® SAMA5D3 Xplained XSTK
  - micro USB to USB-A cables
  - Ethernet cable
  - USB serial TTL adapter (optional)
    - FTDI TTL-232R-3V3 USB to TTL serial cable
- **Software Prerequisites**
  - Linux PC running Ubuntu 12.04 LTS
  - Qt Creator installed

- **Estimated completion time**: 30 min

Introduction

The goal of this hands-on is to create a QML based application and deploy it on the SAMA5D3-Xplained connected to 4.3” PDA touch screen. By completing this hands-on you will

- Basics QML elements
- Using Atmel QML elements
- Gain operational knowledge on Qt Quick and QML elements

Table of Contents

Icon Key Identifiers

INFO	Delivers contextual information about a specific topic
TIPS	Highlights useful tips and techniques
TO DO	Highlights objectives to be completed on the Linux computer
RESULT	Highlights the expected result of an assignment step
WARNING	Indicates important information
EXECUTE	Highlights actions to be executed out of the SAMA5D3 Xplained board

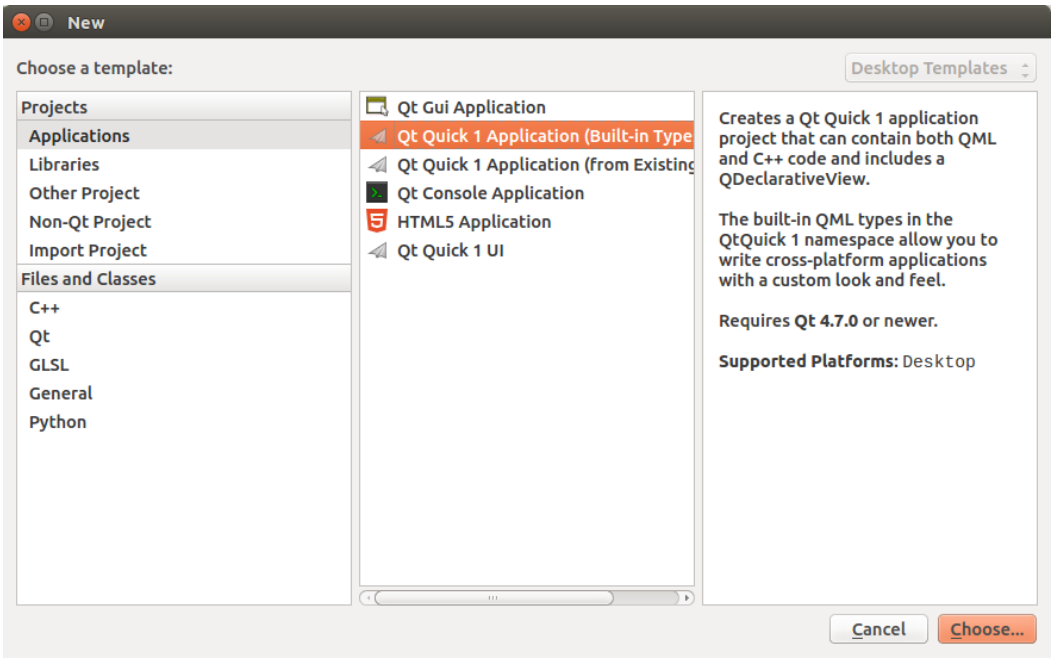
1. Assignment 1 – Basic QML Application

QML is a declarative language that allows creation of elements by specifying shapes, placements, states, transitions and animations. This goal is to create your first QML application where two circles move in a loop on the screen

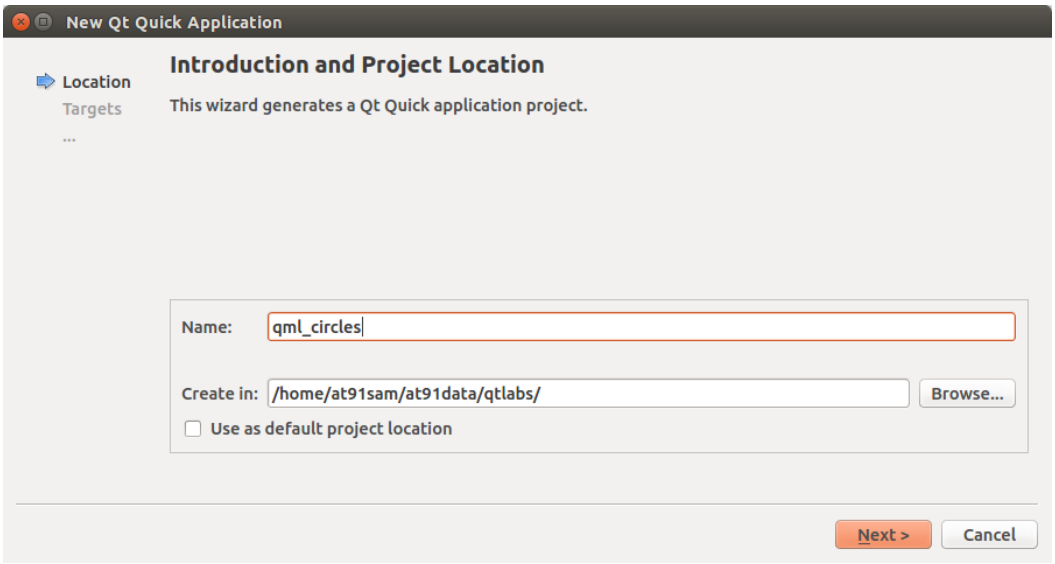
1.1 Create a new Qt QML application using qt-creator provided wizards.

TO DO

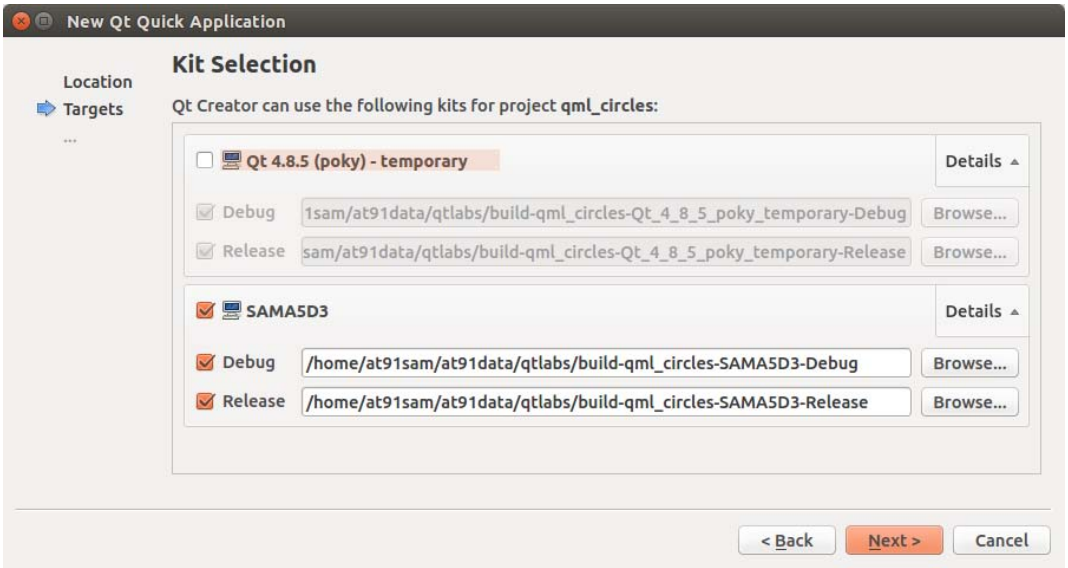
You need to go through Qt Creator menus:
File →New File or Project:
Select: Applications → Qt Quick 1 Application (Built-in Type)



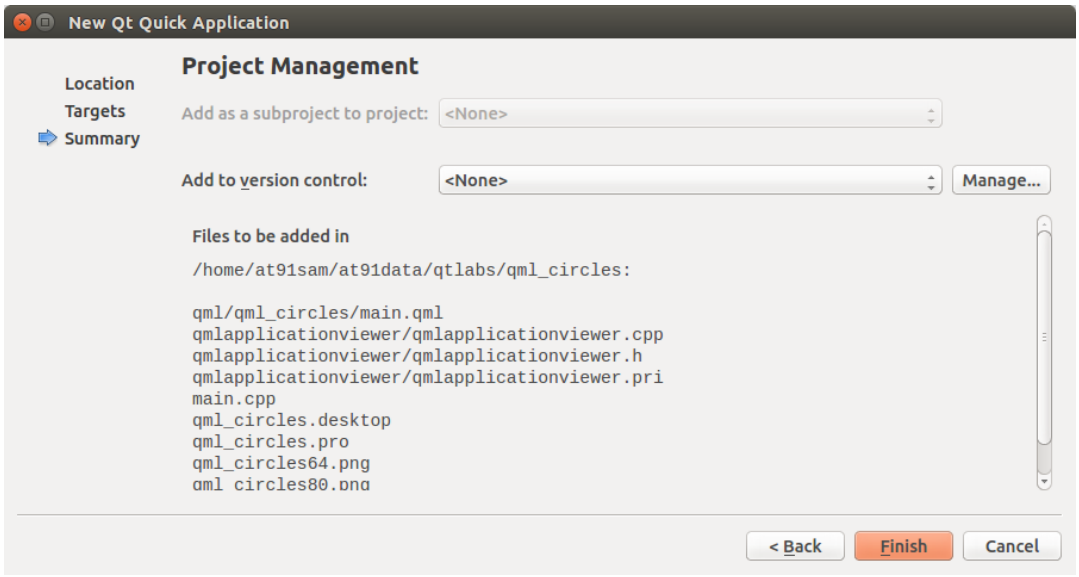
TO DO Specify your application name and make sure you create new project in your LAB area



TO DO You can now select which Kit should be enabled for this new project. Make sure you select your new Kit which enables cross-compilation for the Atmel sama5d3_xplained target.



TO DO Click “Finish” button.



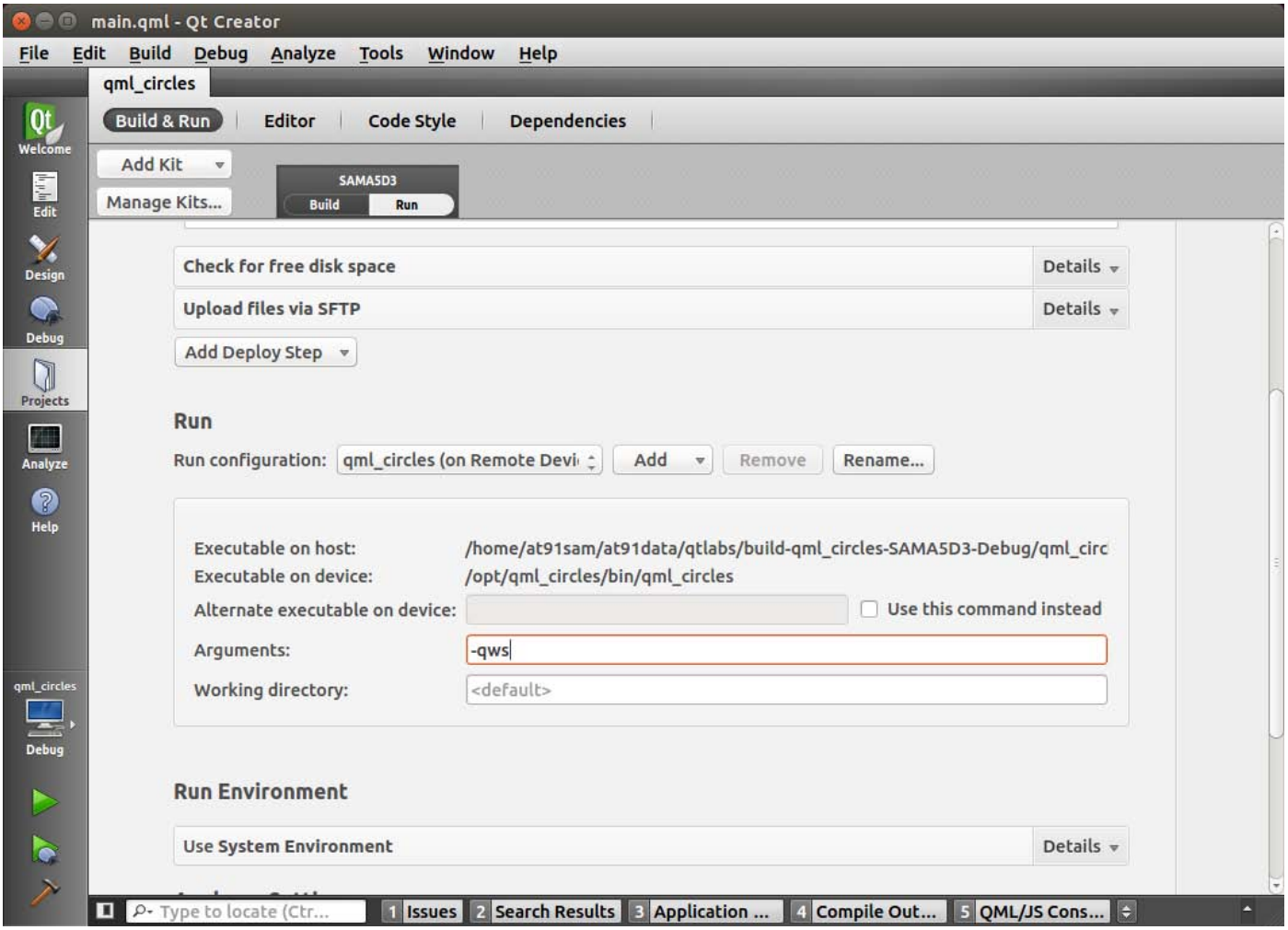
1.2 Basic Shape creation

INFO

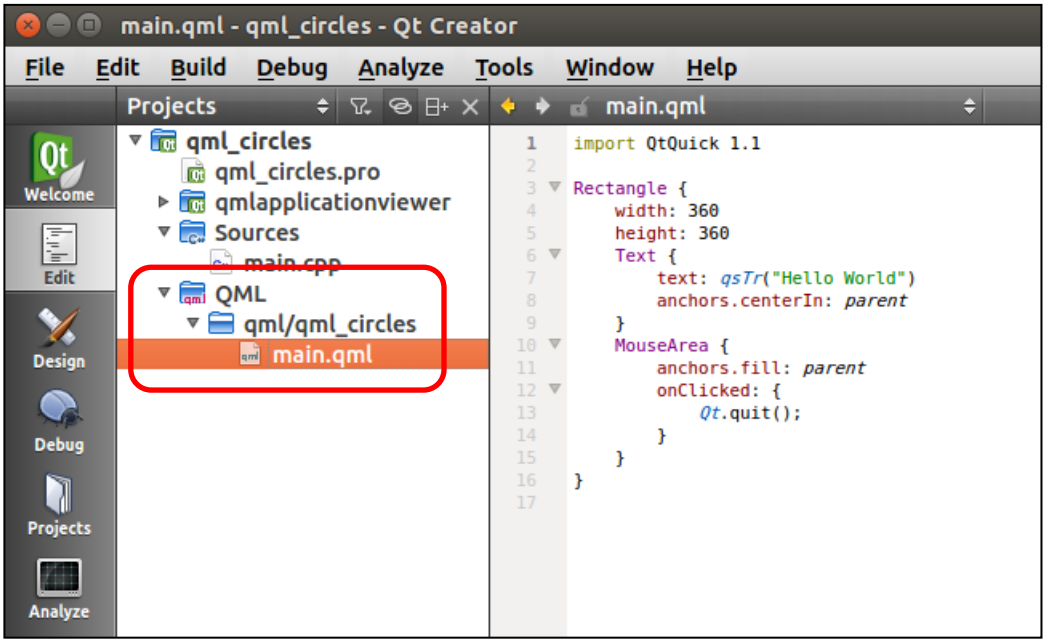
A Qt for Embedded Linux application requires a server application to be running, or to be the server application itself. Any Qt for Embedded Linux application can be the server application by constructing the QApplication object with the QApplication::GuiServer type, or by running the application with the -qws command line option.
More information: http://qt-project.org/doc/qt-4.8/qt-embedded-running.html

TO DO

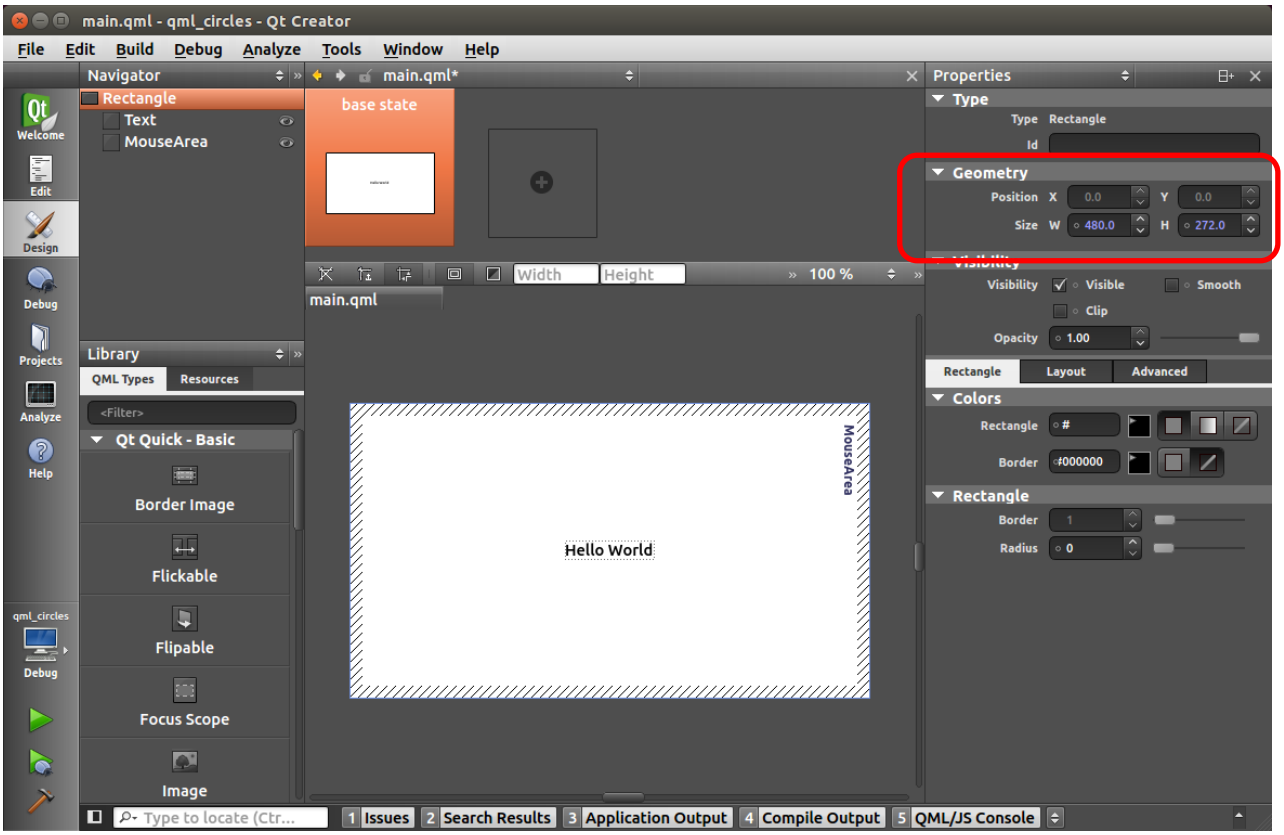
Add “-qws” option argument to qml_circles application Select “Project View”☐ select run tab” then add “-qws” option in “Arguments field.



TO DO Open the designer view by selecting on “main.qml” file and then by clicking “Design” sidebar button.



TO DO Adjust the screen size of your application (480x272)

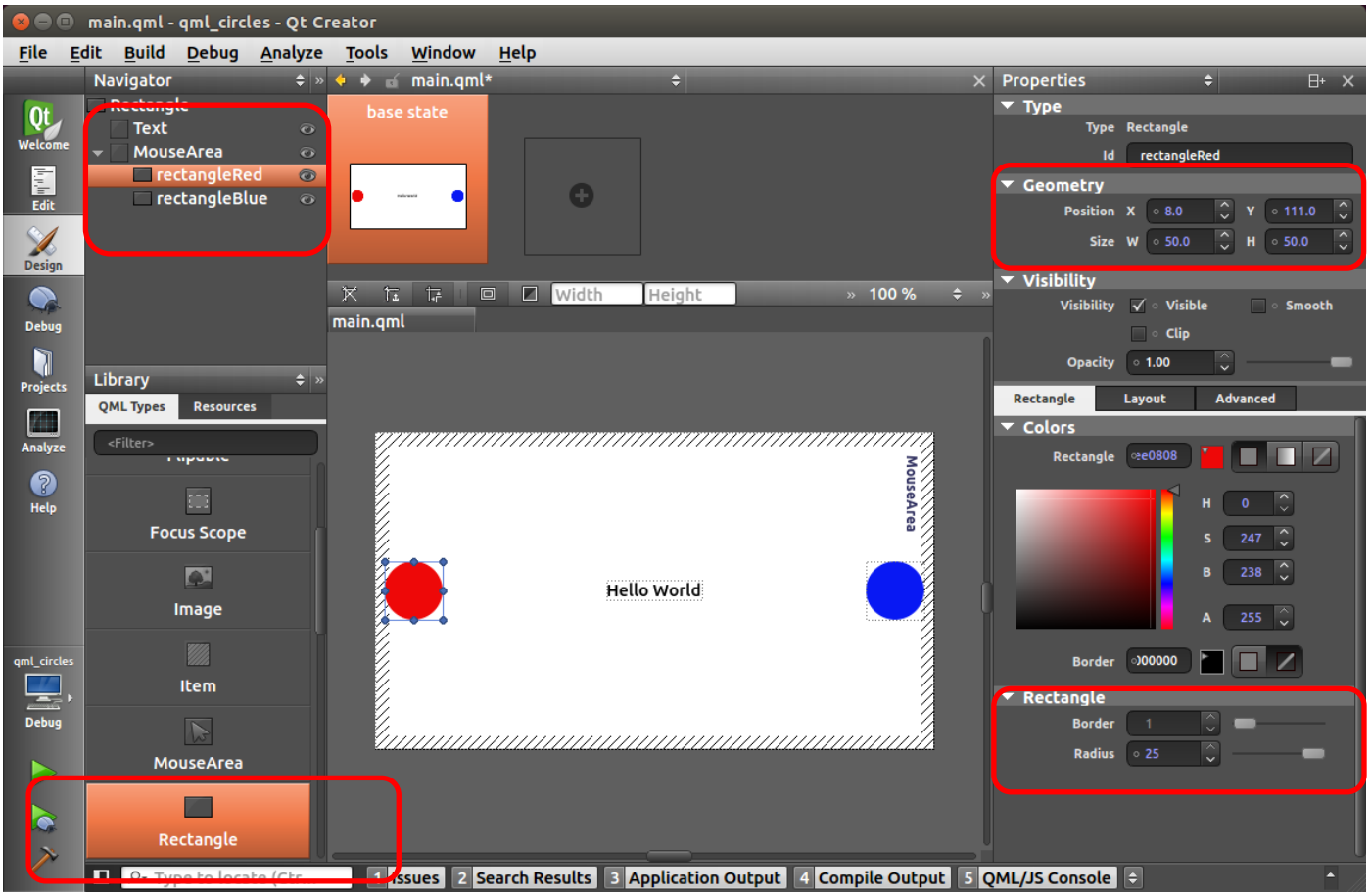


TO DO

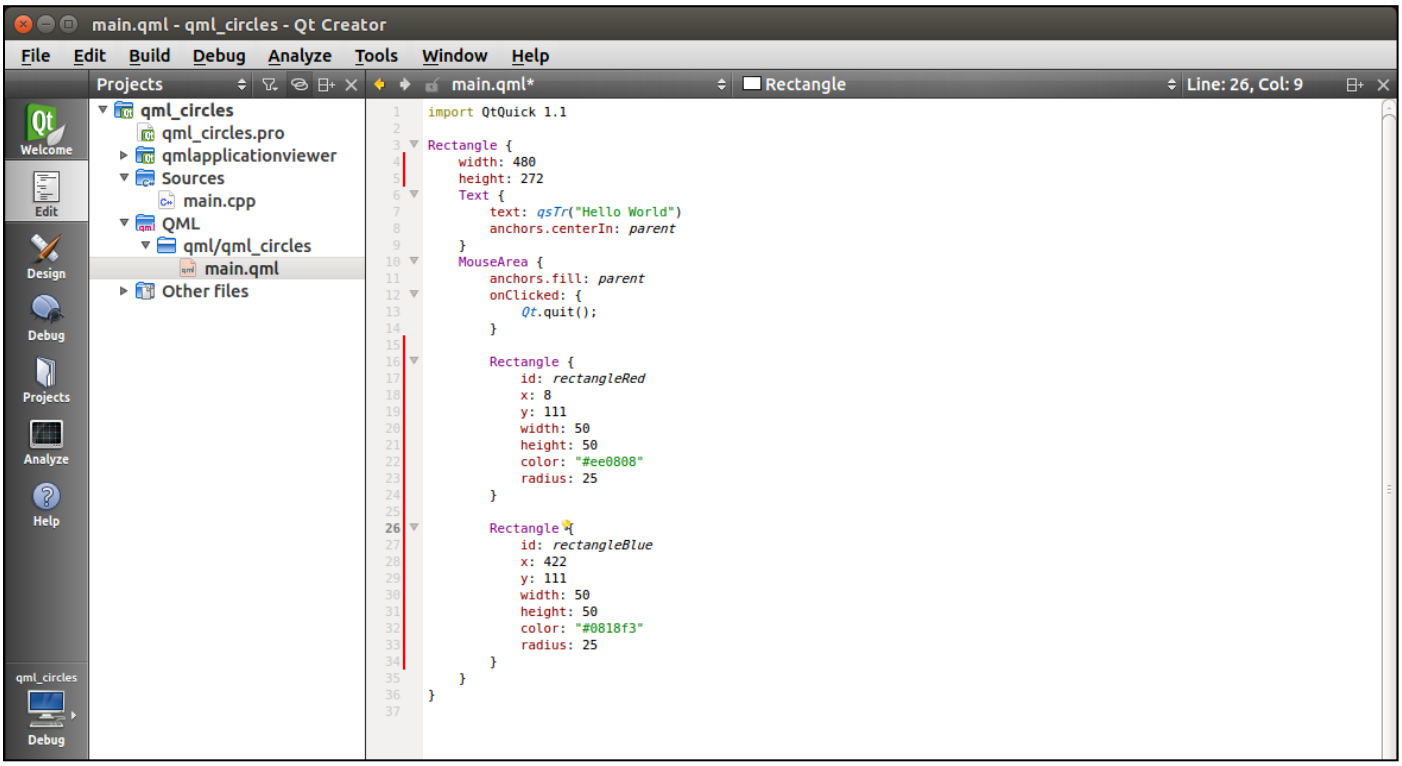
Create two circles inside the Rectangle. Make them of the size 50.
Assign blue and red colors to the two circles. e.g. color: “blue” – you can use HEX values if you’d like.

TIPS

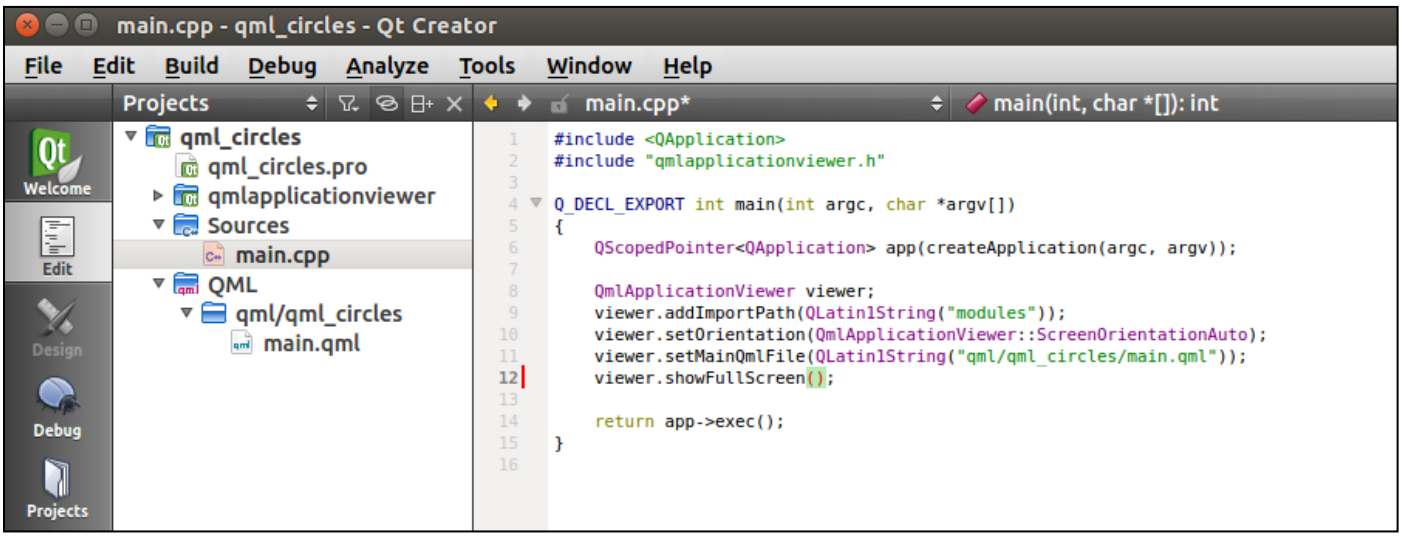
To create a circle we start with a square and we use a radius property that should be set to half the size of the square’s width. Remember, square is a rectangle with equal width and height.



TO DO Open main.qml and check modifications. Run your program, what do you see?



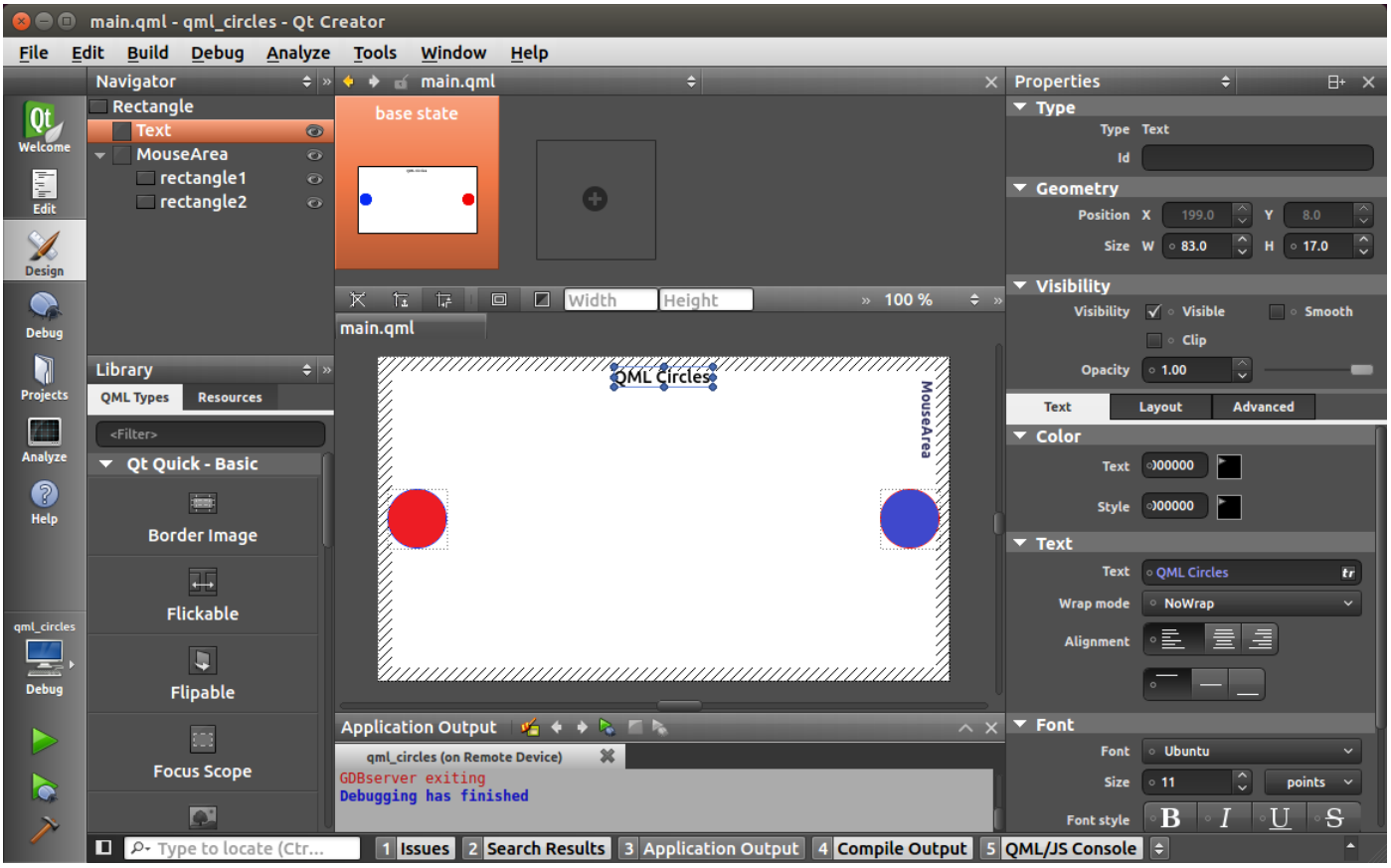
TO DO Open main.cpp file and replace “viewer.show()” by ” viewer.showFullScreen. This will remove the main window bar and will display your application in full screen mode.



EXECUTE Now you can deploy and run your application on the target.

1.3 Animation

TO DO Modify text and position of “Text” element.



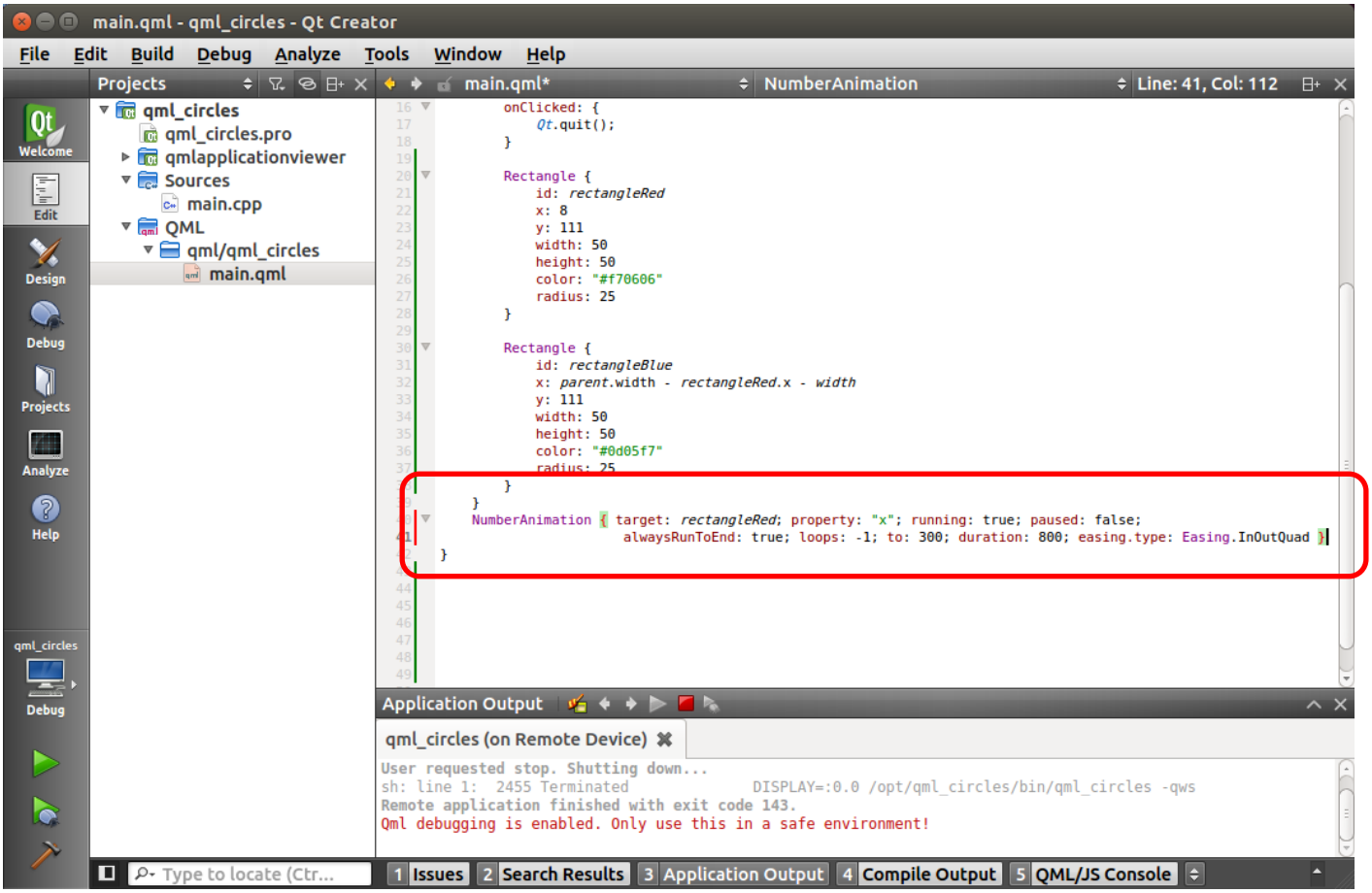
TO DO Set position of the blue circle relative to the position of the red one:

```
[…]
    x: parent.width – rectangleRed.x – width
[…]
```

TO DO

The two circles' positions are now bound by the relation we have defined earlier for x property. In order to animate both circles, we can focus on animating one: red circle's x property.
Animation is done typically via NumberAnimation. Add the following line to your code at the bottom:

```
NumberAnimation {
          target: rectangleRed; property: "x"; running: true; paused: false;
          alwaysRunToEnd: true; loops: -1; to: 300; duration: 800;
          easing.type: Easing.InOutQuad
}
```

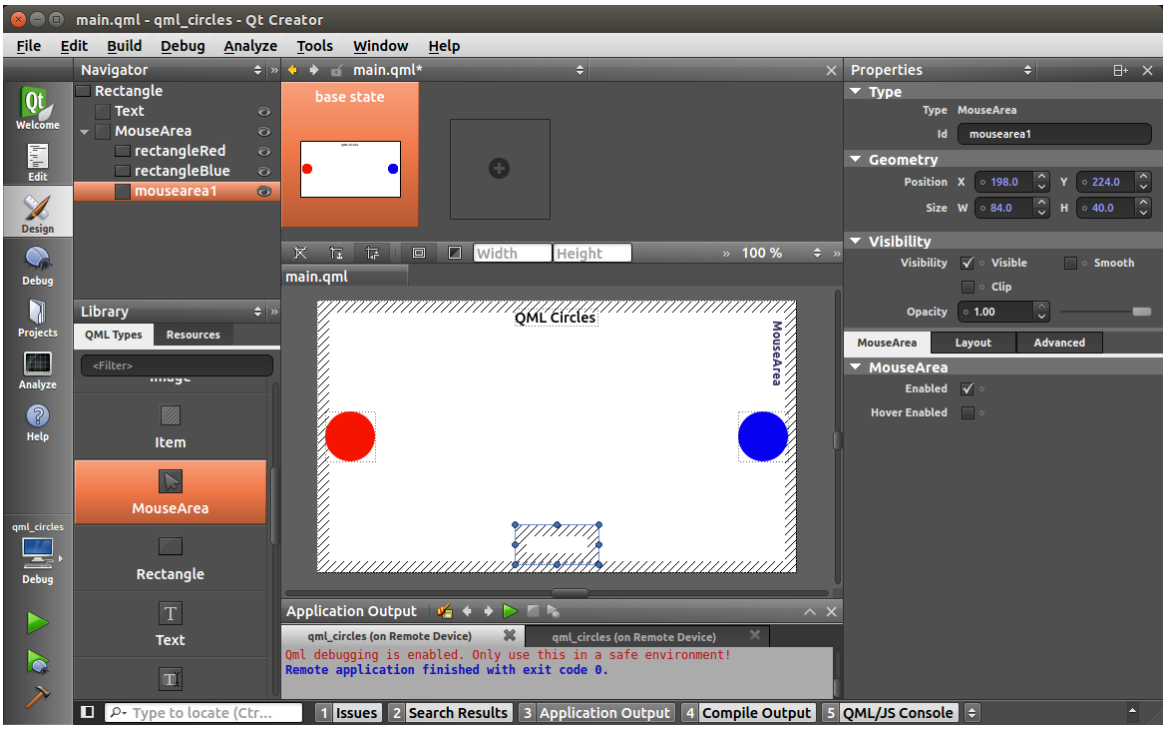


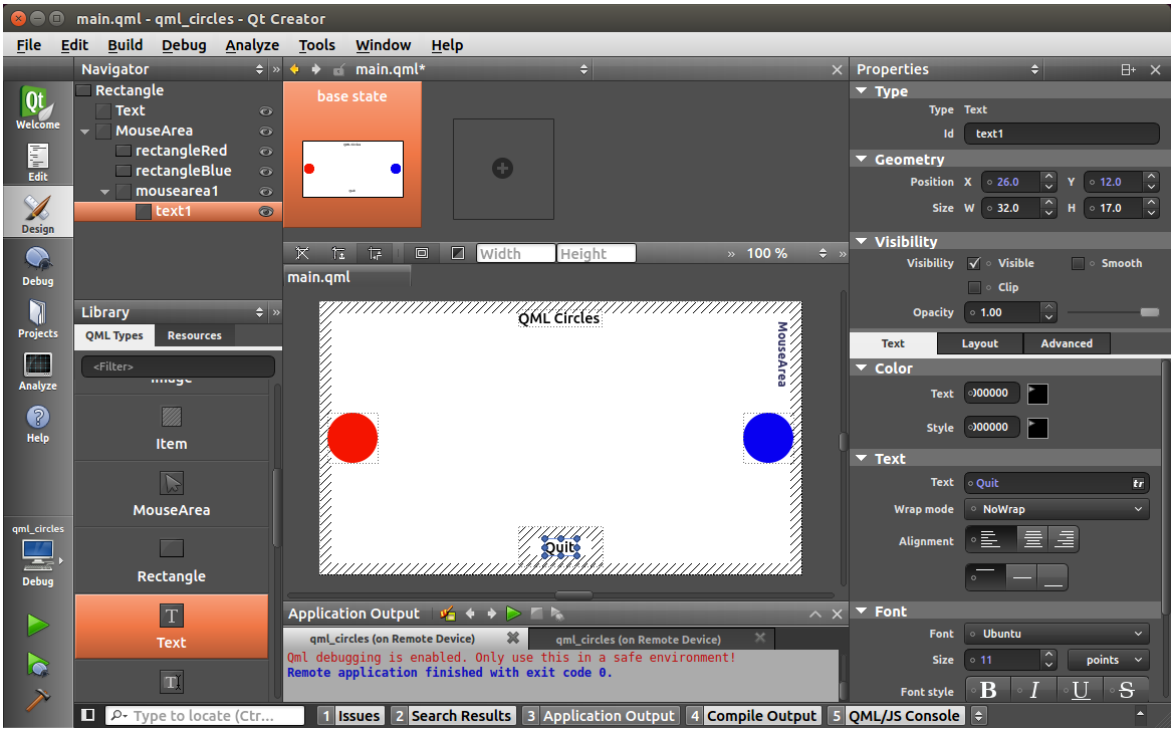
1.4 User interactions through “States”

Often times we would prefer to trigger animations when we clock on something. This can be accomplished through a combination of MouseArea elements, States and Transitions that implement animations.

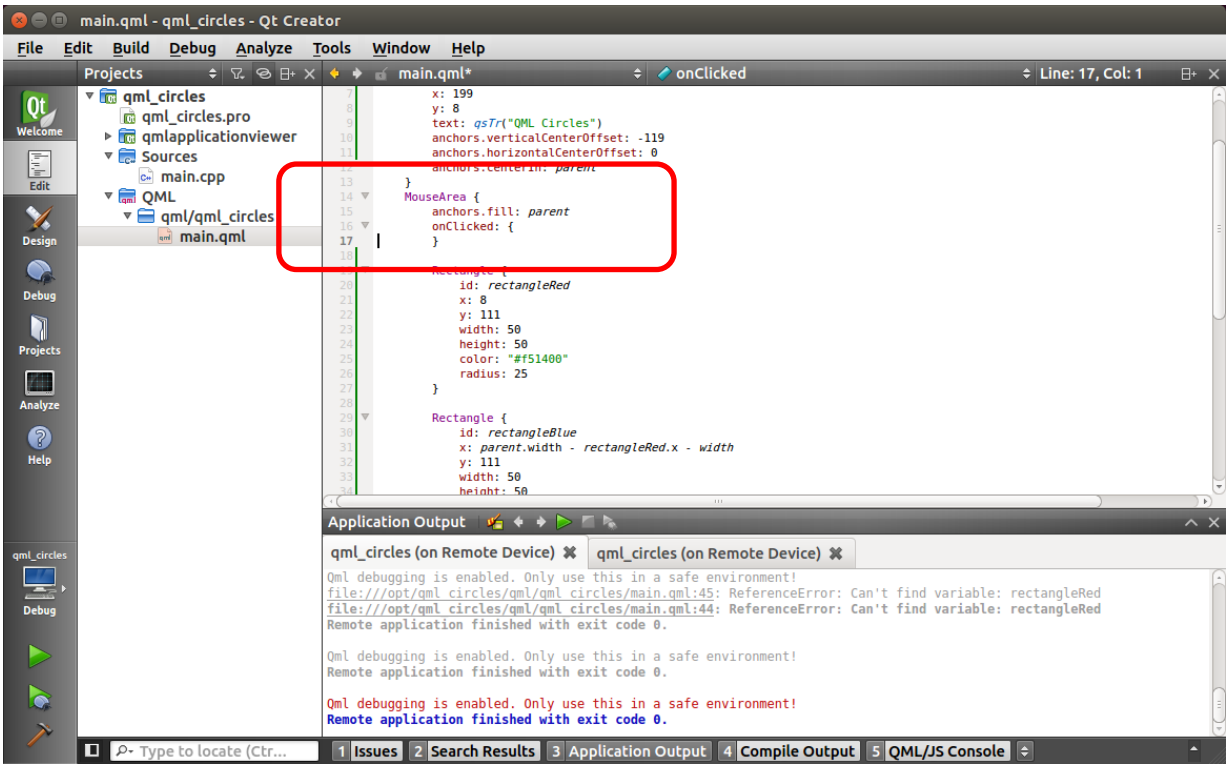
TO DO

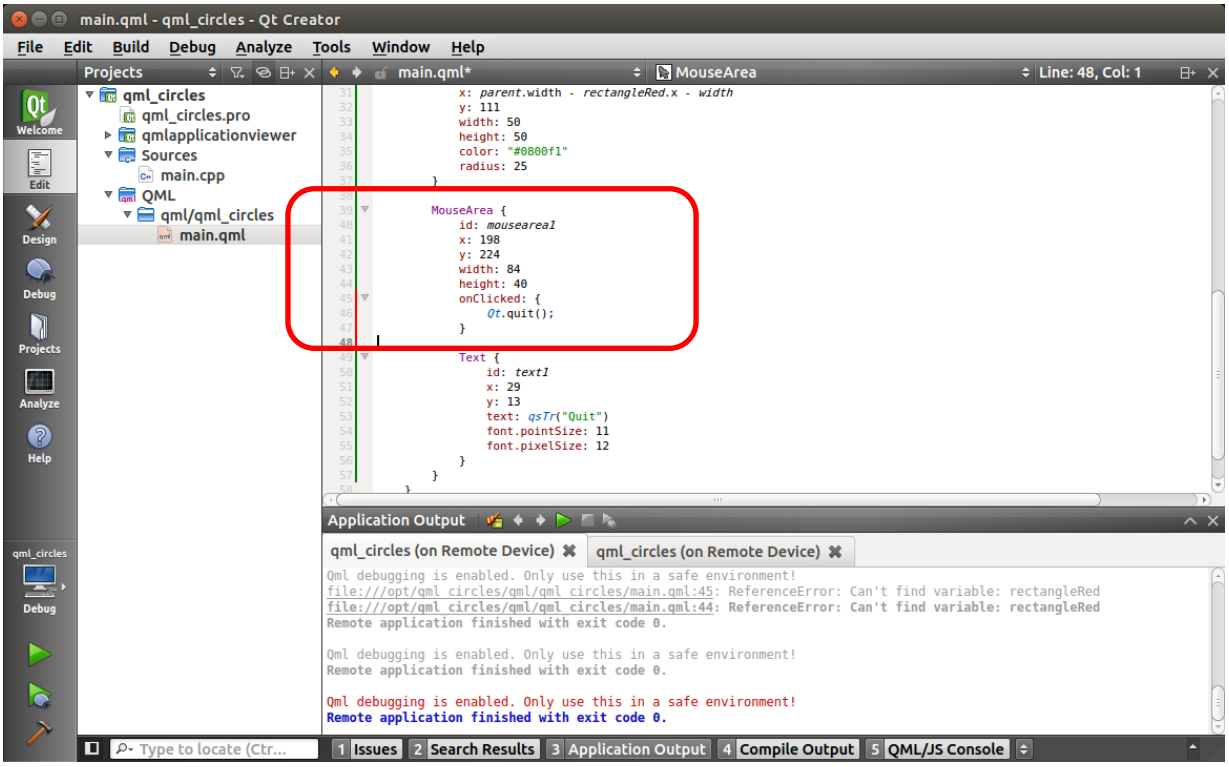
First, let’s comment out the animations from previous assignment and create a new mouse area to quit the application.
Add a new mouse area and a text label (Quit)



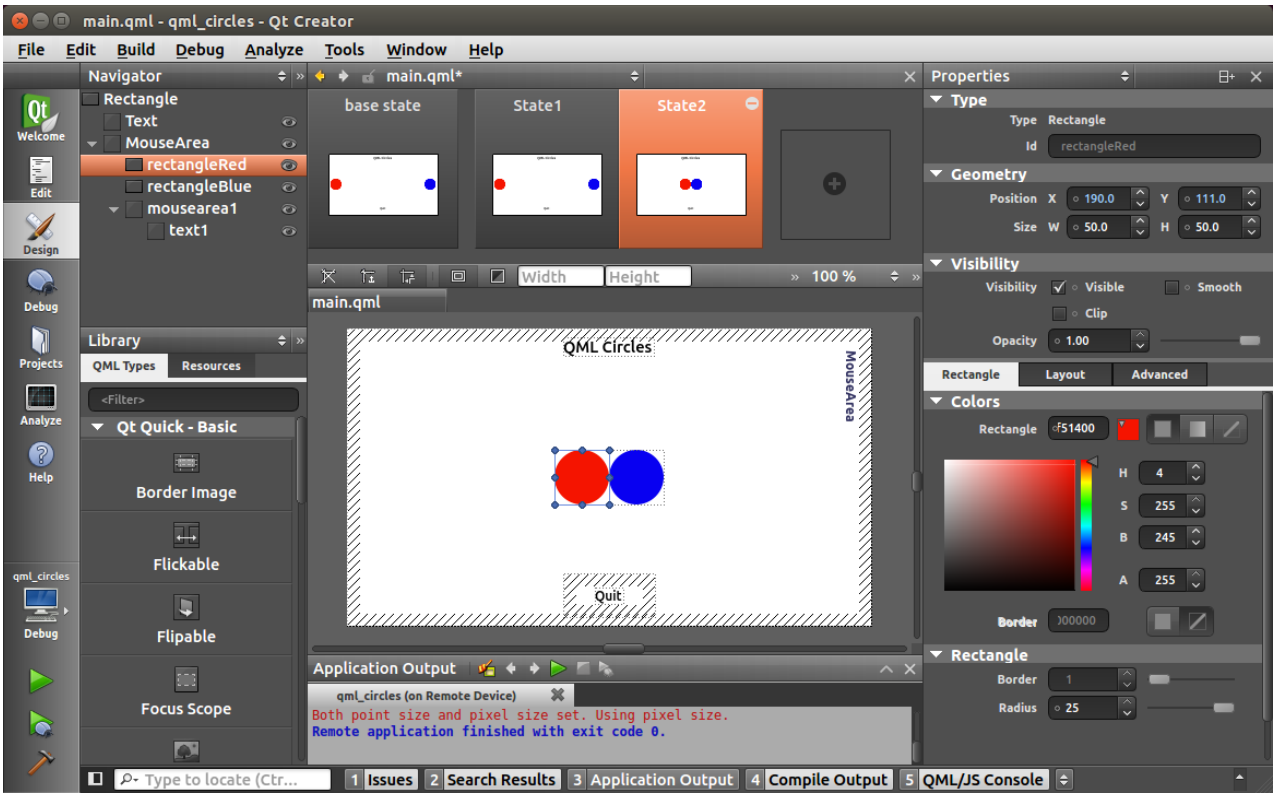


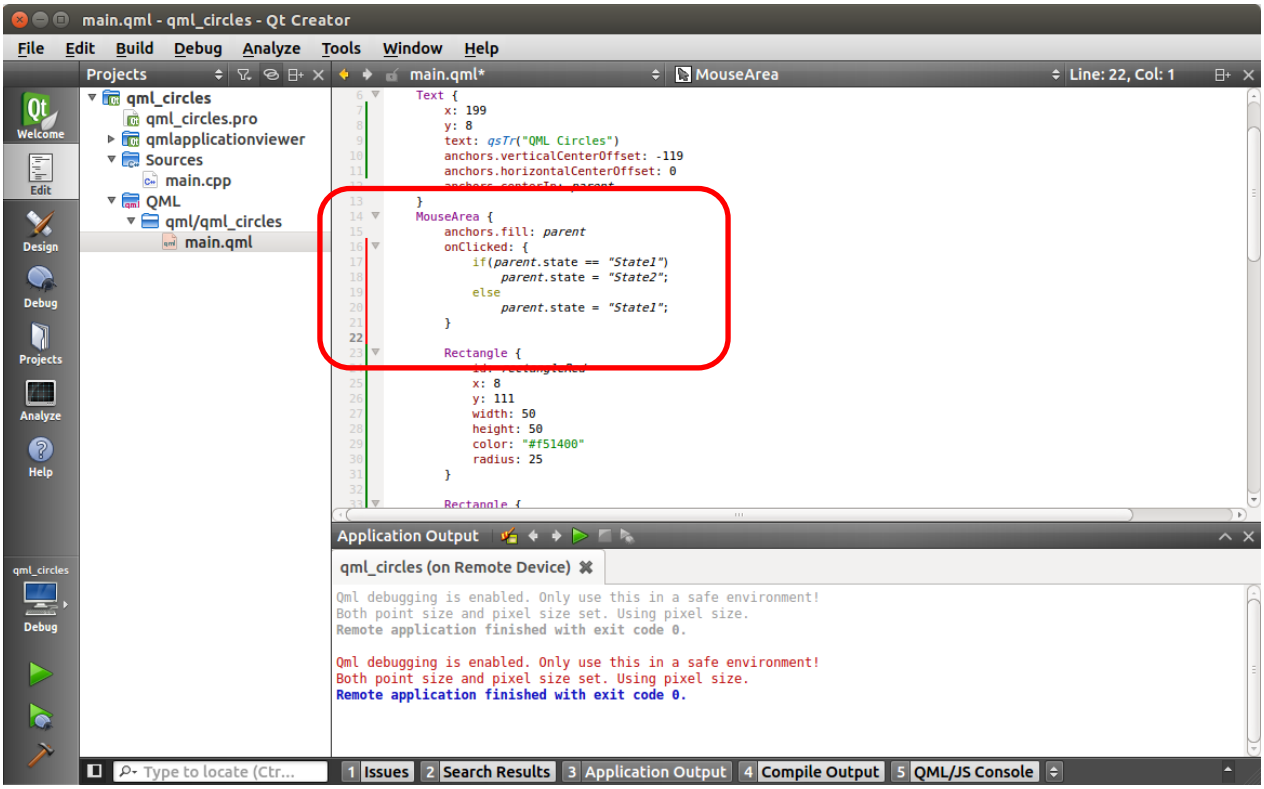
TO DO Move “Qt.quit()" from “mousearea.onClicked()” to new mouse area. Build and deploy.





TO DO Add two states and use main mouse click event as a trigger.
Build and deploy

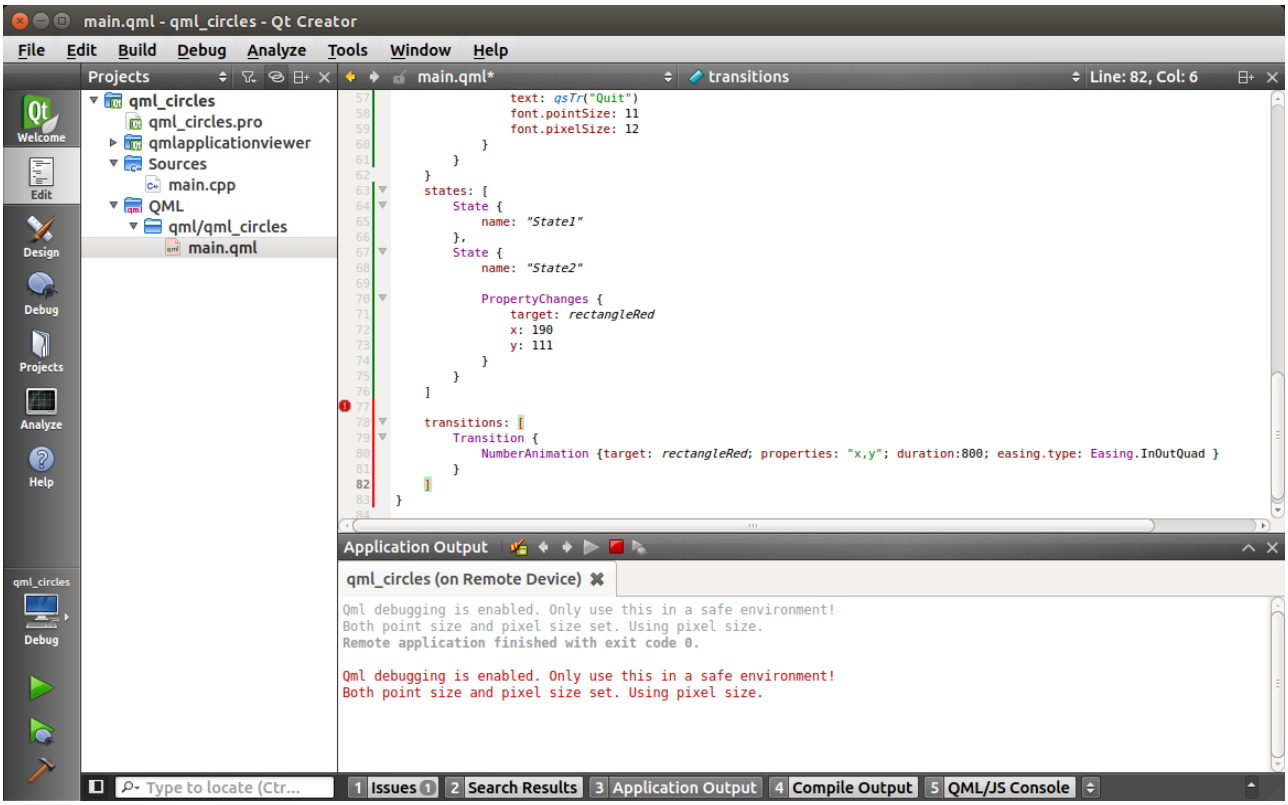




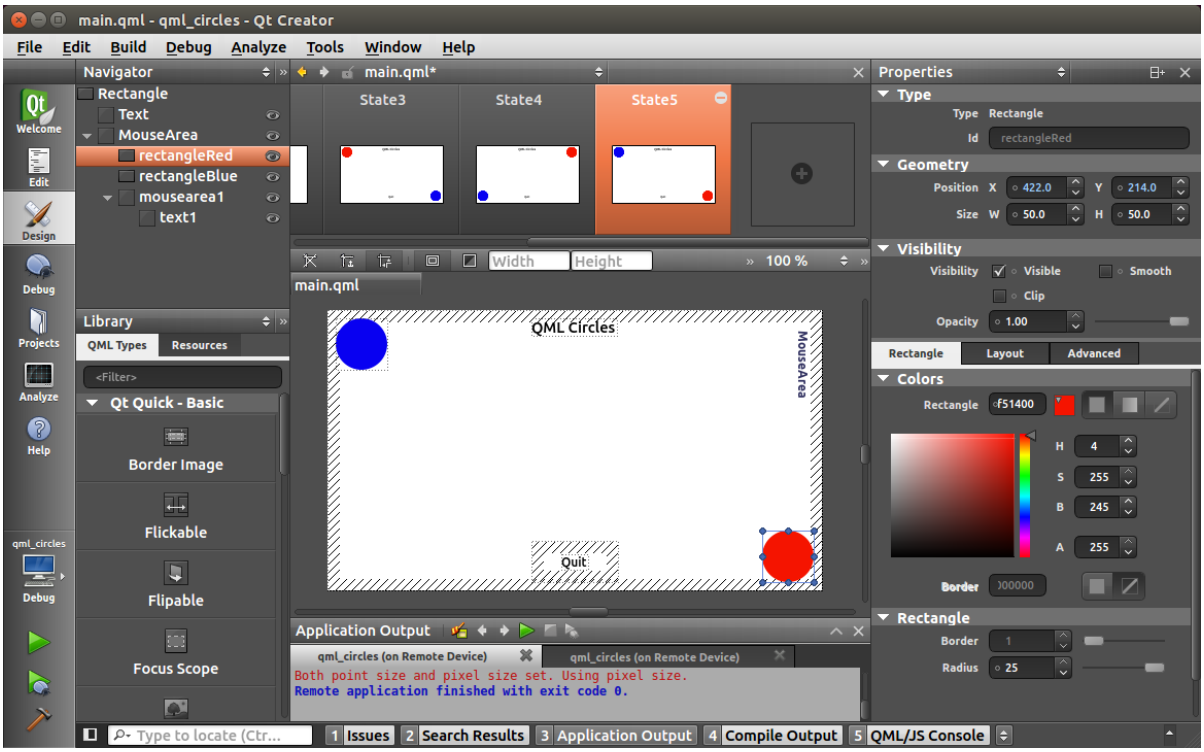
TO DO

Add transition rule between states.
This transition will create an animation where x and y properties of rectangleRed will change between states in 800ms. The easing curve describes how the value is changed
More information http://qt-project.org/doc/qt-4.8/qeasingcurve.html

```
transitions: [
    Transition {
        NumberAnimation {target: rectangleRed; properties: "x,y"; duration:800;
                         easing.type: Easing.InOutQuad }
    }
]
```



TO DO Now you can add more states



TO DO Add a transition rule for rectangleBlue (x,y) properties and modify trigger code.

```
transitions: [
    Transition {
        NumberAnimation {target: rectangleRed; properties: "x,y"; duration:800;
                         easing.type: Easing.InOutQuad }
        NumberAnimation {target: rectangleBlue; properties: "x,y"; duration:800;
                         easing.type: Easing.InOutQuad }
    }
]
```

```
MouseArea {
        anchors.fill: parent
        onClicked: {
            if(parent.state == "State1")
                parent.state = "State2";
            else if(parent.state == "State2")
                parent.state = "State3";
            else if(parent.state == "State3")
                parent.state = "State4";
            else if(parent.state == "State4")
                parent.state = "State5";
            else
                parent.state = "State1";
        }
```

2. Revision History

Doc. Rev.	Date	Comments
xxxxxA	xx/2014	Initial document release

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370